EK-CI780-TD-001

CI780 Hardware Technical Description

Prepared by Educational Services
of
Digital Equipment Corporation

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

Printed in U.S.A.

The manuscript for this book was created on a DIGITAL Word Processing System and, via a translation program, was automatically typeset on DIGITAL's DECset Integrated Publishing System. Book production was done by Educational Services Development and Publishing in South Lawrence, MA.

CONTENTS

CHAPTER 5 DATA PATH MODULE

CHAPTER 6 SBI MODULE

FIGURES

TABLES

CHAPTER 1
INTRODUCTION

1.1 MANUAL SCOPE

This document provides a technical description of the CI780 computer interconnect hardware. It does not treat the CI780 port architecture or other software applications such as the CI780 port driver, command queues, or the VAX/VMS operating system.

A basic description of the CI780 computer interconnect is given in this chapter. Chapters 2, 3, 5 and 6 provide a detailed description of each of the four CI780 modules. Chapter 4 describes the microcode control store and associated control logic. By describing the control store, its addressing logic, and its branching logic in a separate chapter we can treat it as a single cohesive function although the hardware is distributed over two modules (packet buffer and data path).

Three appendixes supplement the information contained in this manual. Appendix A defines the mnemonics found within this document. Appendix B explains the symbology used in the flow diagrams. Appendix C is a description of hardware registers used for maintenance purposes.

1.2 THE COMPUTER INTERCONNECT (CI)

The computer interconnect (CI) (Figure 1-1) is a high-speed, serial data bus that is used to link computer subsystems (nodes) to form a CI cluster. Typically, the cluster is confined to a computer room environment. Nodes may consist of CPUs and memory. Nodes may also include intelligent mass storage, communication, or data acquisition subsystems.

Features of the CI include:

- Dual signal paths capable of simultaneous operation
- 70-megabit-per-second bandwidth and transfer rate
- 32-bit CRC generation and checking
- Low error rate
- Packet-oriented data transfers
- Immediate acknowledgement of the reception of a packet
- Contention arbitration at light loading and round-robin arbitration at heavy loading
- Internal and external data looping for diagnostic purposes.

Each node within a cluster connects to the computer interconnect via a CI780 interface that provides two separate signal paths. Dual paths provide a high degree of data availability between nodes. One pair of nodes can communicate over one path (path A) while another pair of nodes communicates over the second path (path B).

Each path contains a central star coupler (SC008) that receives the data transmitted by a node and distributes it to the other nodes within the cluster. A single CI path consists of a pair of bus cables (one for transmit, one for receive). These cables provide the connection between a node and the signal distribution coupler (star coupler) for that path.

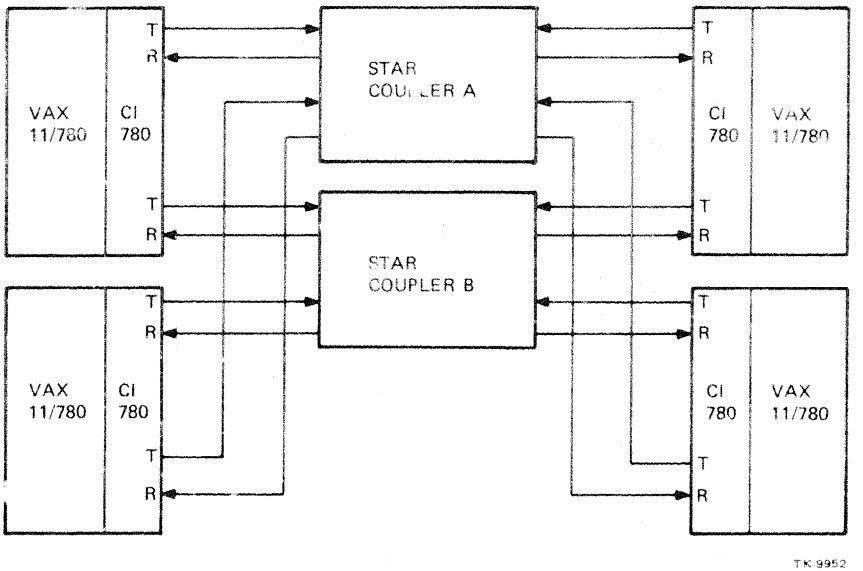


Figure 1-1 Four-Node CI Cluster

1.3 RELATED DOCUMENTS

Table 1-1 is a list of documents providing additional information related to the CI780.

Table 1-1 CI780 Related Documents

Item	Title	Document Number	Contents
1	*CI780 User's Guide*	EK-CI780-UG	Contains instructions for unpacking, installing, and acceptance testing the CI780. A physical description of the CI780 is also provided. Information is also provided on the CI780 backplane jumpers.
2	*SC008 Star Coupler User's Guide*	EK-SC008-UG	Contains a description of the SC008 Star Coupler. Also provides instructions for unpacking and installing the various Star Coupler configurations.
3	*VAX-11/780 Power System Technical Description*	EK-PS780-TD	Contains a technical description (physical and functional) of the H7100 option power supply and the H7101, −5.2 V regulator.
4	*VAX Architecture Handbook*	EB-19580-20	Contains a description of the VAX family architecture, including data representations, instructions, registers, and operational modes.
5	*VAX Hardware Handbook*	EB-21710-20	Provides a hardware overview of the VAX family. Hardware descriptions include the 11/780, the 11/750, and 11/730.

1.4 THE CI780 INTERFACE

The CI780 is the interface used to connect a VAX-11/780 system to the CI cluster. It connects between the synchronous backplane interconnect (SBI) of the host system and the CI cluster. Figure 1-2 illustrates the CI780 connection.

The CI780 is an intelligent interface that performs the function of a buffered communications port. It utilizes the queue structure provided under the VAX/VMS operating system to transfer messages and blocks of data between the host's memory system and other nodes within the CI cluster. By providing data buffering, address translation, and serial encoding and decoding, the CI780 reduces the amount of overhead software processing required to complete high-level intercomputer communications.

The CI780 may be installed in any four-inch option slot in either the H9602-H SBI expansion cabinet or the standard CPU cabinet of the host system. The CI780 contains the following four modules functionally connected as shown in Figure 1-3:

1. Link Interface Module (ILI) L0100
2. Packet Buffer Module (IPB) L0101
3. Data Path Module (IDP) L0102
4. SBI Interface Module (ISI) L0104

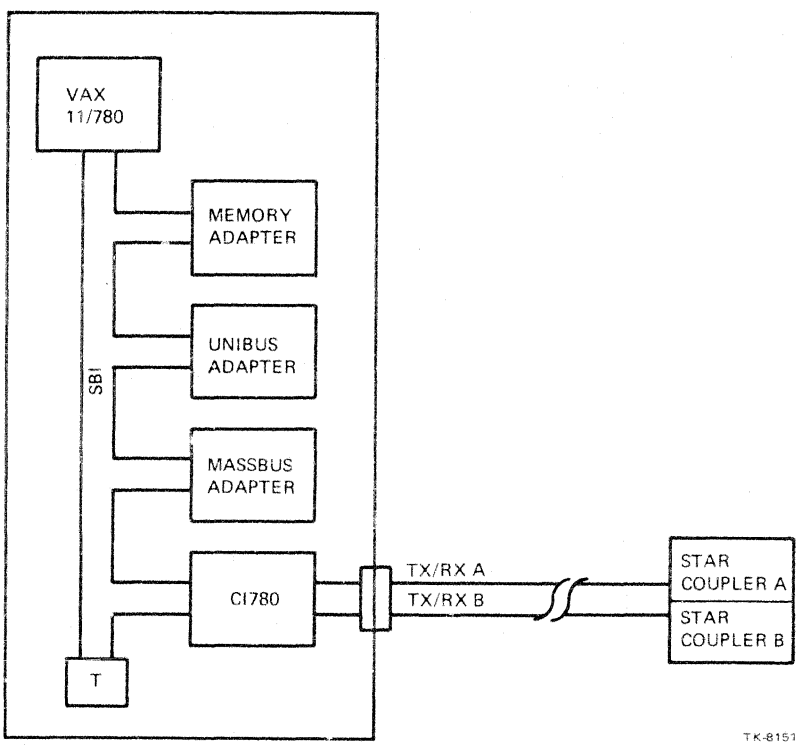
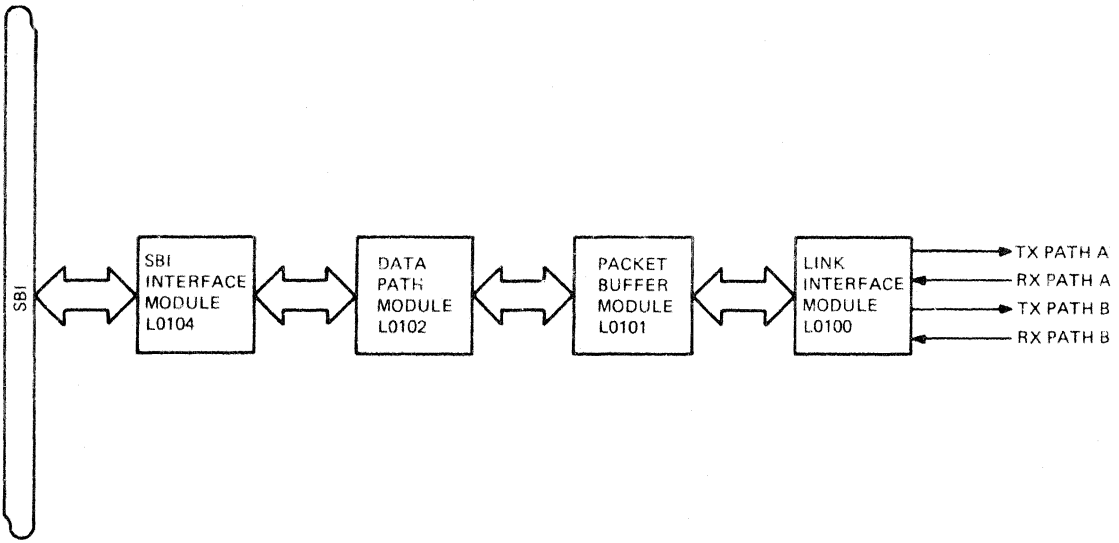


Figure 1-2 CI780 Connection



TK-8150

Figure 1-3 CI780 Modules

Figure 1-4 is a block diagram of the CI780. It should be used with the following discussion of the CI780 modules.

1.4.1 Link Module

The link module provides the interface to the CI bus and has the capability of servicing both CI paths. The module is functionally divided into a transmit path and a receive path with a Cyclic Redundancy Check (CRC) function shared between the two channels. The link can transmit or receive over only one CI path at a time due to the common CRC logic being used by both channels.

Data packets are received from the packet buffer (PB) module over the XMIT DATA BUS, and are appended with header information and a trailer. The header functions to identify the source and destination of the packet. Node address switches provide the node with an address on the CI cluster. The packet header contains this address as a source identification. The trailer serves to keep the node receiver locked up while the last data bytes in the packet are being processed.

The CRC logic uses the packet data bytes to generate four CRC check bytes that are appended to the data packet. The CRC bytes are unique for the specific data bytes in the packet. The bytes are used for error checking at the packet destination.

The link transmitter converts the data packet from a byte format to a 70-megabit-per-second serial format and then applies it to a Manchester encoder.

The Manchester encoder combines the serial data with the bit rate clock to produce a modulated (phase encoded) carrier for the CI bus.

The path selection logic selects the CI path (A or B) for the transmission. The path selection is under microcode control.

Carrier detection logic monitors the two CI paths and connects the receiver channel to whichever path is active.

The serial data from the CI is applied to a Manchester decoder which separates the signal into its clock and data components. The clock and data signal components are applied to the link receiver.

The link receiver converts the packet data from a 70 megabit-per-second serial format to a byte format.

The link receiver then supplies the packet data to the CRC logic. The CRC logic validates the packet by checking the packet data against the packet CRC bytes. If a CRC error is detected, no response (ACK or NACK) is returned to the transmitting node.

If there is no CRC error, the packet is sent to the PB module over the RCVR DATA bus. If the PB module can accept the packet, the link returns a positive acknowledgement (ACK) to the transmitting node. If the buffers on the PB module are full and cannot accept the packet, the link returns a NACK to the transmitting node which will then retransmit the packet.

1.4.2 Packet Buffer Module (PB)

The PB module provides buffering for the data packets transferring through the CI780. Two transmit and two receive buffers (A and B) are provided. Each buffer has a capacity of 1K. When data packets are being transmitted, transmit buffer A is filled from the data path (DP) module over the PORT DATA bus. The next data packet is loaded into buffer B while the link is unloading the data from buffer A.

Likewise, received data packets are loaded into receive buffer A from the link module over the RCVR DATA bus. The following data packet is loaded into receive buffer B while the DP is unloading the data from receive buffer A.

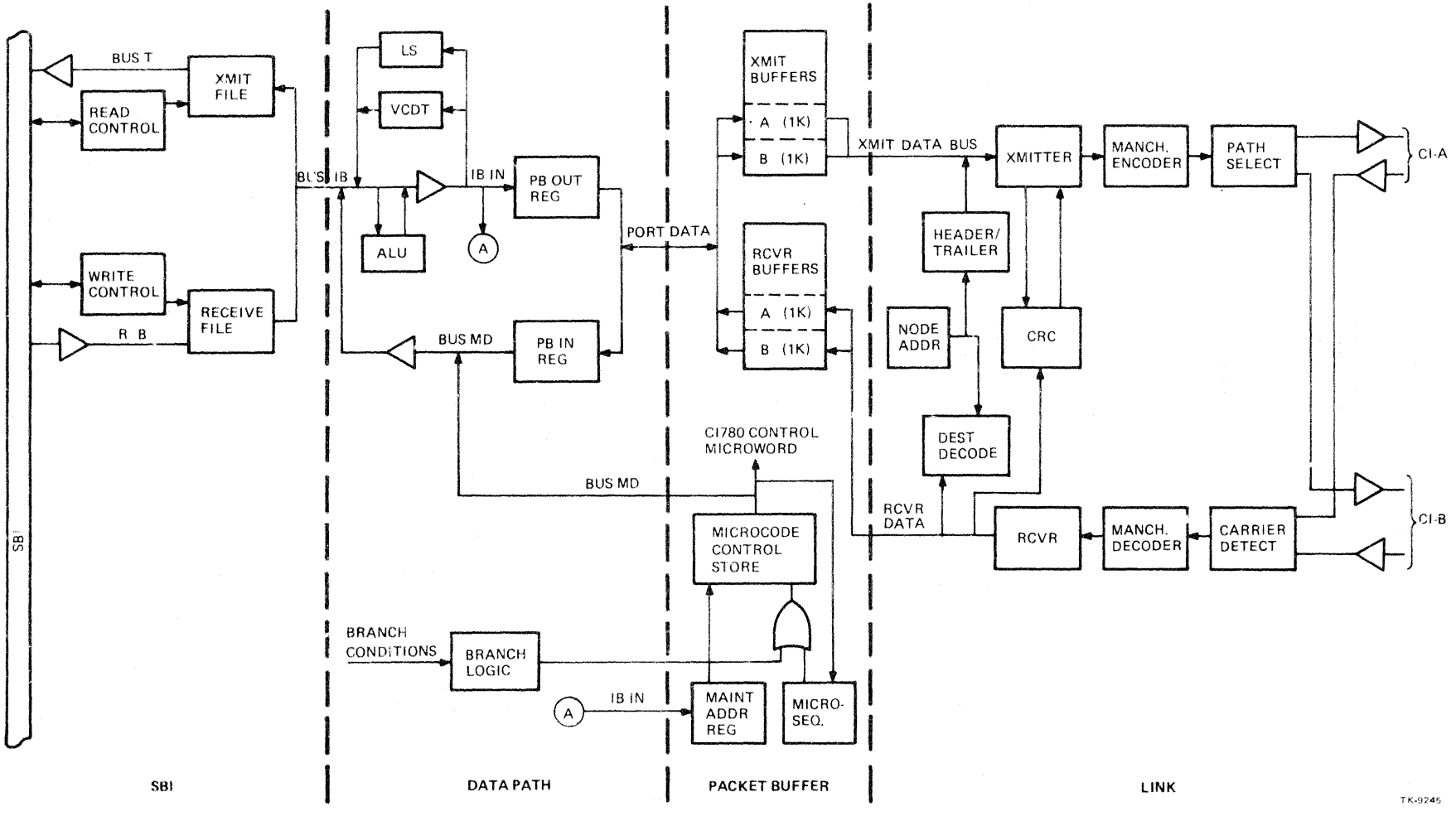


Figure 1-4 CI780 Block Diagram

The CI780 microcode resides in a 3K RAM/PROM control store located on the PB module. The control store RAM/PROM outputs a 47-bit microword that controls and regulates operations throughout the CI780. Stepping of the microcode is controlled by a microsequencer which samples the next address field of the microword. The microcode is also subject to branching conditions via branching logic located in the DP module. The branching logic tests various conditions throughout the CI780. The test results are ORed with the microsequencer output to provide branching of the microcode sequences.

The CI780 control microword can be read by the host system via the BUS MD in the DP module.

Under certain conditions (system initialization or detection of an error) the host system can force a routine by inputting the starting address via the DP IB IN bus and the maintenance address register.

1.4.3 Data Path Module (DP)

Conversion of data packets from longword to byte format and from byte to longword format is accomplished on the DP module. Data to be transmitted is input into a 32-bit PB OUT register from the IB IN bus (internal bus), in longword format. The PB OUT register is unloaded onto the PORT DATA bus a byte at a time.

Likewise, received data bytes from the PB module are input into a 32-bit PB IN register from the PORT DATA bus. When four bytes have been loaded into the register, the register outputs the 32-bit longword onto the MD (miscellaneous data) bus.

The DP module contains LS (local store) and VCDT (virtual circuit descriptor table) storage areas. The LS is a 256 × 32 RAM area used to store software status blocks and software registers associated with the CI780 port architecture. The VCDT is a 256 × 16 RAM area used to store CI node parameters.

Also contained on the DP module is a 2901A ALU used to perform general purpose arithmetic and logical operations.

The data interface between the DP and the SBI module is the BUS IB bus. Data input to the DP on the BUS IB, has several possible destinations. It may be applied to the LS or the VCDT for storage, to the ALU for processing, or to the PB OUT register for transfer to the PB module.

Output data from the DP on the BUS IB, may be obtained from several sources. It may be data read from the LS or the VCDT area, it may be data obtained from the ALU after processing, or it may be data obtained from the BUS MD. Data from the BUS MD may be from the PB IN register or it may be the CI780 control microword read from the control store in the PB module.

The destination and source of the BUS IB data is selected by the CI780 microcode.

1.4.4 The Synchronous Backplane Interconnect Module (SBI)

The basic function of the SBI module is to interface with the SBI. All SBI protocol and timing must be observed while transferring data to and from the SBI. A transmit and a receive file act as isolation buffers. The SBI side of the files are loaded and unloaded under SBI timing and control while the DP side of the files are loaded and unloaded under CI780 microcode control.

Data received from the DP over the BUS IB is loaded into the transmit file by the microcode. Up to two extended write transfers of data can be stored in the file. The microcode informs the read control that data is available in the file. The read control arbitrates for the SBI, receives and supplies information fields as required by SBI protocol, and unloads (reads) the data in the transmit file out to the SBI over the BUS T bus.

Data received from the SBI is loaded into the receive file by the write control. Up to two extended read transfers of data can be stored in the file. The write control receives and supplies the information fields required by SBI protocol and then loads (writes) the SBI data into the file over the RB bus. The write control informs the CI780 microcode that data is available in the file. The microcode unloads the data from the receive file onto the BUS IB for transfer to the DP module.

The SBI module provides for CPU access of CI780 registers via unsolicited SBI requests. Both reads and writes of the registers can be performed.

The SBI module also requests interrupts of the CPU when service routines are run on the CI780.

1.4.5 CI780 Power

Power is supplied to the CI780 from an H7100 option power supply with an H7101, −5.2 V regulator. The supply receives 120 V, 60 Hz from a switched outlet on the 869 power controller located in the cabinet. The supply provides +5.0 V to the four CI780 modules, and −5.2 V to the link and SBI modules. The supply also provides ACLO and DCLO to the SBI module.

Power signals and voltages pass from the power supply to the CI780 modules via the CI780 backplane. Figure 1-5 illustrates the routing of the power signals and voltages.

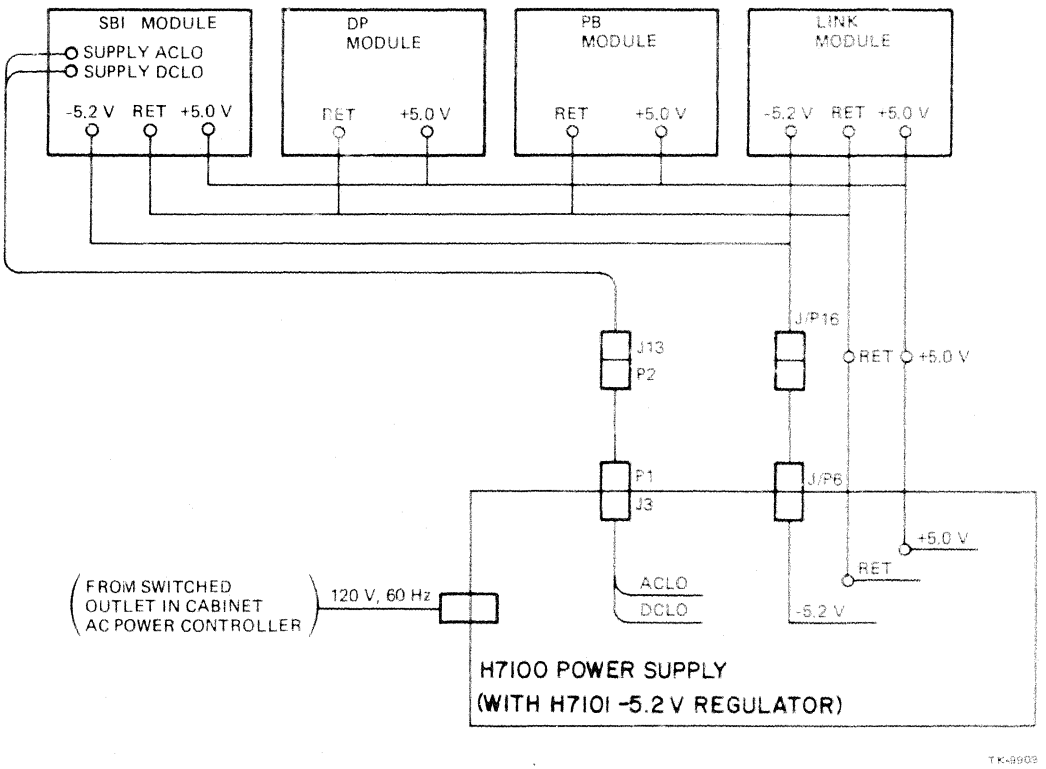


Figure 1-5 CI780 Power

CHAPTER 2
LINK MODULE

NOTE

The functional block diagrams in Chapter 2 use logical AND and OR symbols. It does not necessarily follow that a corresponding gate exists on the link logic prints. The assertion of inputs A and B causing the assertion of output C may be represented on a block diagram by a single AND gate, yet the engineering drawing may show that several circuit stages are involved in the ANDing operation.

The functional block diagrams in this chapter are keyed to the link engineering circuit schematics (CS prints) by letter designations in parentheses. The letters specify the link CS sheet that contains the detailed logic associated with the functional blocks in the diagram.

The signal names used in the functional block diagrams are the names used on the engineering CS prints. Where other signal names or notes are used, they are enclosed in parentheses.

2.1 PACKET FORMATS

Formats of the two types of packets, information and ACK/NACK (acknowledge/negative acknowledge), are described below.

2.1.1 Information Packet

Figure 2-1A illustrates the format of an information packet. The information packet is used to transmit both messages and data across the CI. Parts of the packet are generated by the link and inserted into the packet as it passes through the link to be transmitted.

2.1.1.1 Bit Synchronization – The first five bytes of the packet are for bit synchronization within the link. The bytes are 55 hexadecimal which is an alternating pattern of 1's and 0's used to turn on the carrier detect circuits and to synchronize the Manchester decoder prior to the receipt of useful data. The link inserts the bit sync bytes into the packet.

2.1.1.2 Character Synchronization – The character synchronization byte (96 hexadecimal) is used to indicate the start of useful data in the packet. When the sync character is recognized during packet reception, it starts the framing of the serial data into eight-bit bytes. The link inserts the sync character into the packet.

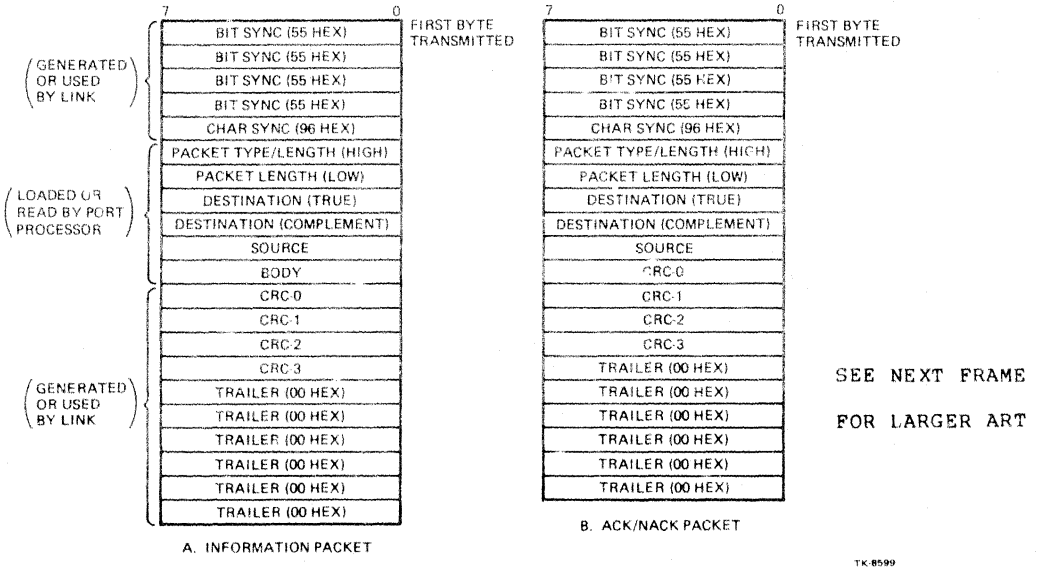


Figure 2-1 Packet Formats

2.1.1.3 Packet Type/Length (High) – The packet type and length (high) byte specifies the type of packet (information or acknowledge) and contains the upper four bits of a 12-bit packet length word. Bits ⟨7:4⟩ are the packet type bits. For an information packet bit 7 is a 0 (1 for an ACK/NACK packet) and bits ⟨6:4⟩ are 0's.

Bits ⟨3:0⟩ are the upper four bits of the 12-bit word that specifies the packet length. Information packets are of variable length in one-byte increments up to 1K bytes* with the minimum packet length being seven bytes. The packet length specified by the 12-bit packet length word includes all data from the packet type and length (high) byte up to and including the last byte of the body.

The port processor supplies the packet type and length (high) byte as part of the packet.

2.1.1.4 Packet Length (Low) – This byte contains the low eight bits of the 12-bit packet length word. The port processor supplies this byte as part of the packet.

* Limited by the capacity of the buffers in the PB. The link is capable of processing packets up to 4K bytes.

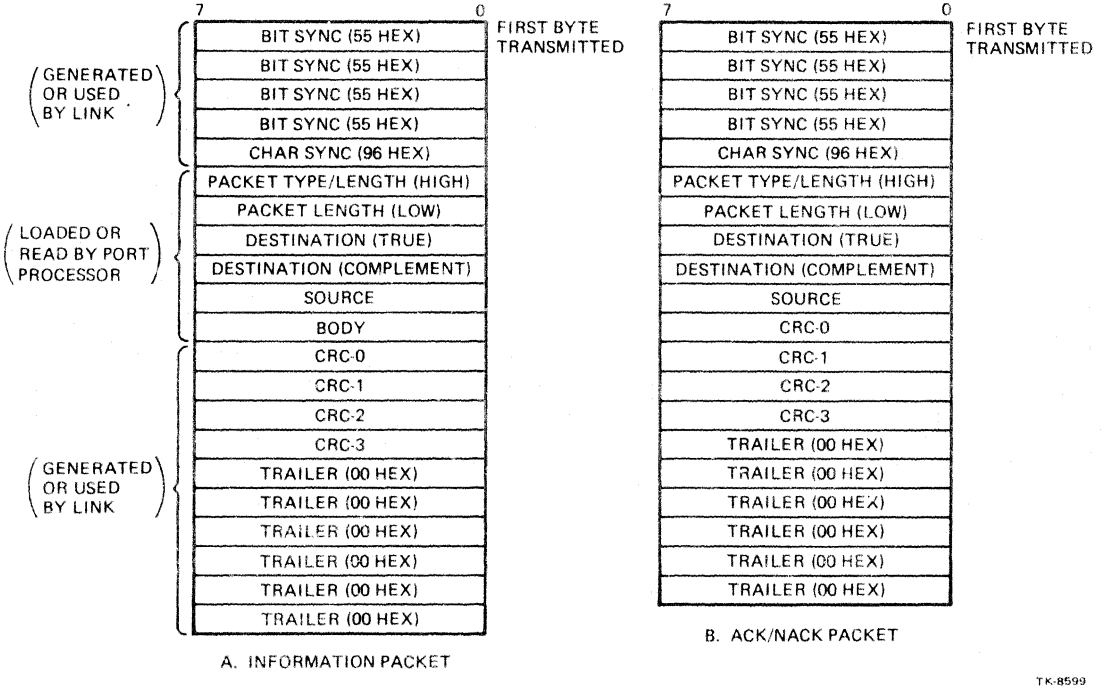


Figure 2-1 Packet Formats

2.1.1.5 Destination (True and Complement) – The destination is the eight-bit address of the CI node to which the packet is transmitted. There are two destination bytes; the first being the true node address value and the second being the complement of the true value. The port processor supplies the destination bytes as part of the packet.

Redundant destination addresses are used to preclude a single logic failure bringing down both paths on the CI bus. With a single address decode circuit, a failure which caused a node to decode another node's address might result in both nodes transmitting an acknowledge packet at the same time. This would result in a collision on the CI bus and would be seen as a "no response" by the transmitting node.

2.1.1.6 Source – The source is the eight-bit address of the sending node and is provided by the port processor as part of the packet.

2.1.1.7 Body – The body contains the data and port-processed protocol information. The body is supplied by the port as part of the packet.

2.1.1.8 Cyclical Redundancy Check (CRC) Bytes – Following the body are four CRC bytes generated by the CRC logic in the link. During a packet transmission, the packet [starting with the packet type and length (high) byte], is input into the CRC logic which generates the coefficients of a CRC polynomial. The coefficients are expressed as a 32-bit longword that is a function of the packet data. Each CRC word is unique for the specific packet that generated it.

During packet reception, the CRC longword is regenerated and compared to the four CRC bytes generated during the transmission. An error-free packet results in a match between the two longwords.

2.1.1.9 Trailer – The trailer consists of six bytes of all 0's. It is used to insure that all bits of a received packet have been shifted through the link front end before the carrier detect logic senses the end of packet reception. The link inserts the trailer into the packet.

2.1.2 Acknowledge/Negative Acknowledge (ACK/NACK) Packet

Figure 2-1B illustrates the format of ACK and NACK packets. ACK and NACK packets are sent by the receiving node to inform the transmitting node that the packet arrived without data loss or bus collision (CRC checked OK).

If the receiving node successfully accepted the packet into the buffers on the PB, an ACK packet is returned indicating a successful bus transaction and storage in the PB. If the receiver buffers in the PB were full and, therefore, unable to accept the packet, a negative acknowledge (NACK) packet is sent to inform the transmitting node that the packet was successfully received but could not be accepted. The transmitting node must then retransmit the packet.

The entire ACK (or NACK) packet is generated and transmitted by the link.

An ACK/NACK packet differs from an information packet in the following three ways:

A. It has no body. An ACK/NACK packet only acknowledges reception of an information packet. It does not transfer messages or data as such.

B. It has no packet length word. All ACK and NACK packets are the same length. Consequently bits (3:0) of the packet type and length (high) byte are 0's and there is no packet length (low) byte.

C. The packet type bits (bits (7:4)) of the packet type and length (high) byte specifies the type of packet as follows:

   Bit 7 = 1 indicating an ACK/NACK packet (0 for an information packet)

   Bit 6 = 1 for an ACK packet
   0 for an NACK packet.

2.2 LINK OVERVIEW

The link (Figure 2-2) is functionally divided into a receive channel and a transmit channel with a CRC function shared between the two. The overview briefly describes the following four link operations with the transmit and receive channels functioning as they would for the specific type of packet being processed. The operations are described as they would occur with B following A and D following C.

A. The reception of an information packet
B. The transmission of an ACK/NACK packet
C. The transmission of an information packet
D. The reception of an ACK/NACK packet

Link control logic receives commands from the port to select and start link operations, and senses signal conditions to control the transfer of data packets through the link. A receive clock (RCVR CLK) and a transmit clock (XMIT CLK) are generated on the link to time operations in their respective channels.

2.2.1 Information Packet Reception

Data packets on the CI bus are in serial format at a serial bit rate of 70 MHz. The data is Manchester encoded (phase encoded) wherein the clock is incorporated into the modulated signal.

CI paths A and B are input to a RCVR select multiplexer (mux) in the link front end. Carrier detect logic monitors both CI paths. When the logic senses the initial presence of a carrier on one of the paths and if that path has been enabled by the port, it switches the mux to the active path, selecting CIA RCVR or CIB RCVR for the Manchester decoder. The port may also select the internal loop path wherein the mux selects the output from the transmit channel and loops it back into the port. This feature is used for maintenance operations.

The mux output is applied to a Manchester decoder where the signal clock is extracted from the modulated signal. The Manchester decoder outputs the data (RCVR SERIAL DATA) and the clock (MDECODER CLOCK) to the byte framer. The byte framer contains the sync character detector.

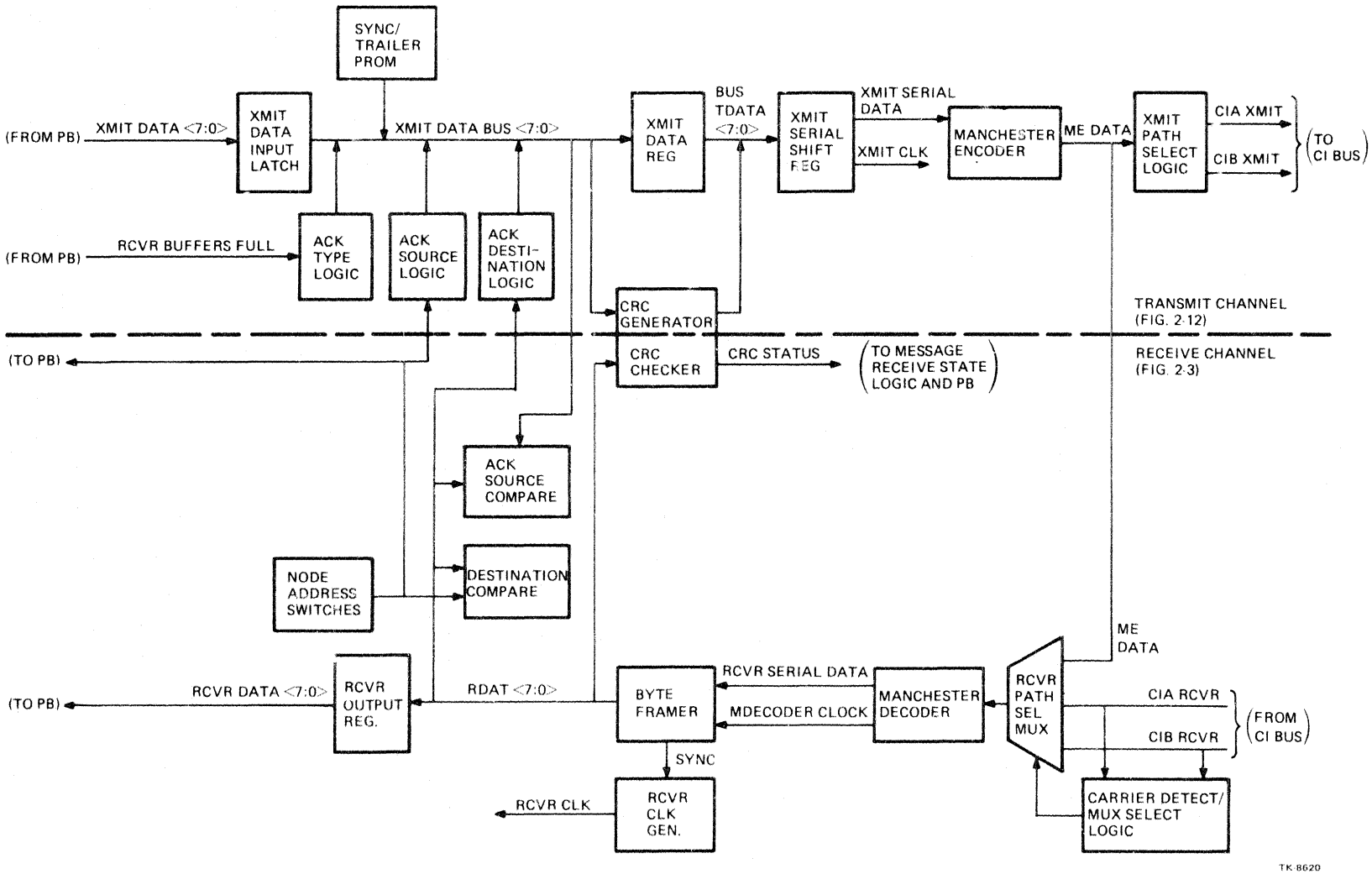


Figure 2-2 Link Simplified Block Diagram

The byte framer performs serial to parallel conversion of the signal data. The framer is enabled by the sync character detector which activates the framer when it recognizes the sync character. When enabled, the byte framer ouputs a data byte (RDAT ⟨7:0⟩) for every eight serial bits received from the Manchester decoder. A RCVR CLK generator develops RCVR CLK which times the transfer of data through the link receive channel. SYNC from the byte framer synchronizes RCVR CLK with the data bytes so as to occur approximately centered on the asserted time period of RDAT ⟨7:0⟩.

The RDAT ⟨7:0⟩ data bytes are coupled to the RCVR output register and then to the PB as RCVR DATA ⟨7:0⟩.

The link verifies that the packet is meant for this node by comparing the packet destination bytes to the node address set into the node address switches. The comparision is made in the destination compare logic. If a match is not obtained, the receiver is cleared and reception is terminated.

The packet source byte is extracted from the incoming packet and placed into the ACK destination register. When the link transmits an ACK packet in response to the information packet now being received, it will use the address in the register (the source of the information packet) as the ACK destination.

The packet bytes extending from the packet type and length (high) byte up to and including the last byte of the body, are applied to the CRC checker. The bytes are acted on by the CRC algorithm which generates the 32-bit CRC longword. The four CRC bytes in the packet are compared to the generated longword and if the packet is free of error, CRC STATUS is asserted to message receive logic.

After the packet trailer has passed through the link front end, the carrier detect logic senses the end of the packet and informs the ACK transmit logic. The ACK transmit logic then initiates the transmission of an ACK packet.

2.2.2 ACK/NACK Packet Transmission

An ACK/NACK packet is generated and transmitted entirely by the link. No packet data is received from the PB as XMIT DATA ⟨7:0⟩.

The link ACK transmit logic initiates the transmit operation by enabling the sync/trailer PROM which outputs five bit-sync bytes and a sync character byte onto the XMIT DATA bus (XMIT DATA BUS ⟨7:0⟩).

The ACK type logic is then enabled and outputs the packet type byte onto the XMIT DATA bus. The logic sampled the state of PB signal RCVR BUFFERS FULL at the start of the information packet reception. If RCVR BUFFERS FULL was true, the PB was not able to accept the information packet just received. In this case, the ACK type logic outputs the code for a NACK packet. If RCVR BUFFERS FULL was false, the logic outputs the code for an ACK type packet.

The link control logic then enables the output of the ACK destination register which outputs the two destination bytes onto the XMIT DATA bus. The destination value used is the source address taken from the information packet just received.

The ACK source logic is then enabled and transfers the node address from the node address switches to the XMIT DATA bus as the source byte.

The ACK/NACK packet is transferred to the BUS TDATA bus via the XMIT data register. The packet, starting with the packet type byte, has also been input into the CRC generator where a 32-bit CRC longword is generated. After the source byte has been input to the CRC generator, the link control gates the CRC longword onto the BUS TDATA bus a byte at a time. The four CRC bytes are thus inserted into the ACK/NACK packet.

Finally, the ACK transmit logic re-enables the sync/trailer PROM which outputs six trailer bytes onto the XMIT DATA bus to complete the ACK/NACK packet.

The ACK/NACK packet on the BUS TDATA bus is applied to the XMIT serial shift register which performs parallel to serial conversion of the signal data. Data bytes are input to the register and then shifted out serially to the Manchester encoder as XMIT SERIAL DATA. The bit rate of the serial data is 70 MHz. The register logic also generates XMIT CLK which times the transfer of data through the link transmit channel. XMIT CLK is synchronized with the serial data within the shift register.

The XMIT SERIAL DATA is applied to the Manchester encoder where the bit rate clock is combined with the serial data to produce a phase-encoded carrier. The Manchester encoder outputs the modulated carrier (ME DATA) to the CI bus. The ACK transmit logic selects the same CI path used by the information packet just received. The ME DATA can also follow an internal loop path into the receive channel if the link is in internal loop mode and the receiver inputs from the CI bus are disabled. This feature is used for maintenance testing.

2.2.3 Information Packet Transmission

An information packet is mostly generated by the port and input to the link transmit channel from the PB. The information packet bytes that are inserted by the link are:

A. The five-bit sync bytes
B. The character sync byte
C. The four CRC bytes
D. The six trailer bytes.

Transfer of an information packet utilizes only some of the functions described in Paragraph 2.2.2. The functions that are used operate as previously described.

The port initiates the transmit operation via the message transmit logic. When the transmit operation is initiated, the link enables the sync/trailer PROM which outputs five bit sync bytes and a sync character byte onto the XMIT DATA bus.

The packet type and length (high) byte and the packet length (low) byte are provided by the port.

The destination bytes are also provided by the port. When the destination bytes are on the XMIT DATA bus the link enters the destination address into the ACK source compare logic. When the ACK/NACK response packet is received from the target node, the packet source byte is compared with the contents of the compare logic. If the correct node responded, a match will be obtained.

The source byte is inserted by the PB, not by the link. The address source is the link node switches which output the node address to the PB. The source byte, then, is an input to the XMIT DATA bus from the PB.

The CRC generator functions to produce the four CRC bytes just as for an ACK/ NACK transmission. However, the information packet has a body which is also input to the CRC generator and contributes to the generation of the CRC longword.

Finally, the link message transmit logic re-enables the sync/trailer PROM which outputs the six trailer bytes onto the XMIT DATA bus to complete the information packet.

2.2.4 ACK/NACK Packet Reception

Transfer of an ACK/NACK through the receive channel utilizes most of the functions described in Paragraph 2.2.1, Information Packet Reception. The functions also operate as previously described.

With regard to the link receive channel, the basic difference between the reception of an ACK/NACK packet and an information packet is in the handling of the packet source byte. The source byte is not entered into the ACK destination register but is applied to the ACK/NACK source compare logic. The source compare logic presently contains the destination address of the information packet just transmitted. The source byte is compared to the destination address. The address will match if the correct nodes are involved in the data transfer.

2.3 LINK OPERATING STATES

Paragraphs 2.4 and 2.5 provide a detailed description of the operation of the receive channel and transmit channel hardware. Control of the hardware is a function of commands from the port, the type of operation being executed, and conditions sensed by the logic (e.g. errors) during the operation. Hardware control is implemented via programmable array logic (PALs) which define various hardware states during each operation. The states are represented in four diagrams contained in the engineering drawing set. The operations described by the diagrams are shown in Table 2-1 and described in Paragraph 2.10.

Table 2-1 Link State Diagrams

Operation	Number of States
Information Packet Reception	13
ACK Packet Transmission	8
Information Packet Transmission	13
ACK Packet Reception	8

2.4 RECEIVE CHANNEL

Figure 2-3 is a block diagram of the receive channel and should be referred to throughout Section 2.4.

The receive channel hardware contains both transistor-transistor (TTL) logic and open collector emitter coupled logic (ECL). The carrier detection/path selection logic, Manchester decoder, byte framer, and sync character detector all use ECL logic. ECL has an active high and non-active low state on common lines resulting in a different interpretation of circuit logic than with TTL. A description of the receive path select mux is given Paragraph 2.4.1.2 as an example for those unfamiliar with ECL logic.

2.4.1 CI Carrier Detection and Path Selection

The carrier detect and path select logic monitors activity on the CI bus and, when activity is detected, selects the active path as an input to the link receive channel. The port uses port and link control PALs to specify which receive channel(s) are allowed to receive signal inputs from the CI bus. The PALs enable the receive channel(s) by asserting RCVR A ENABLE or RCVR B ENABLE.

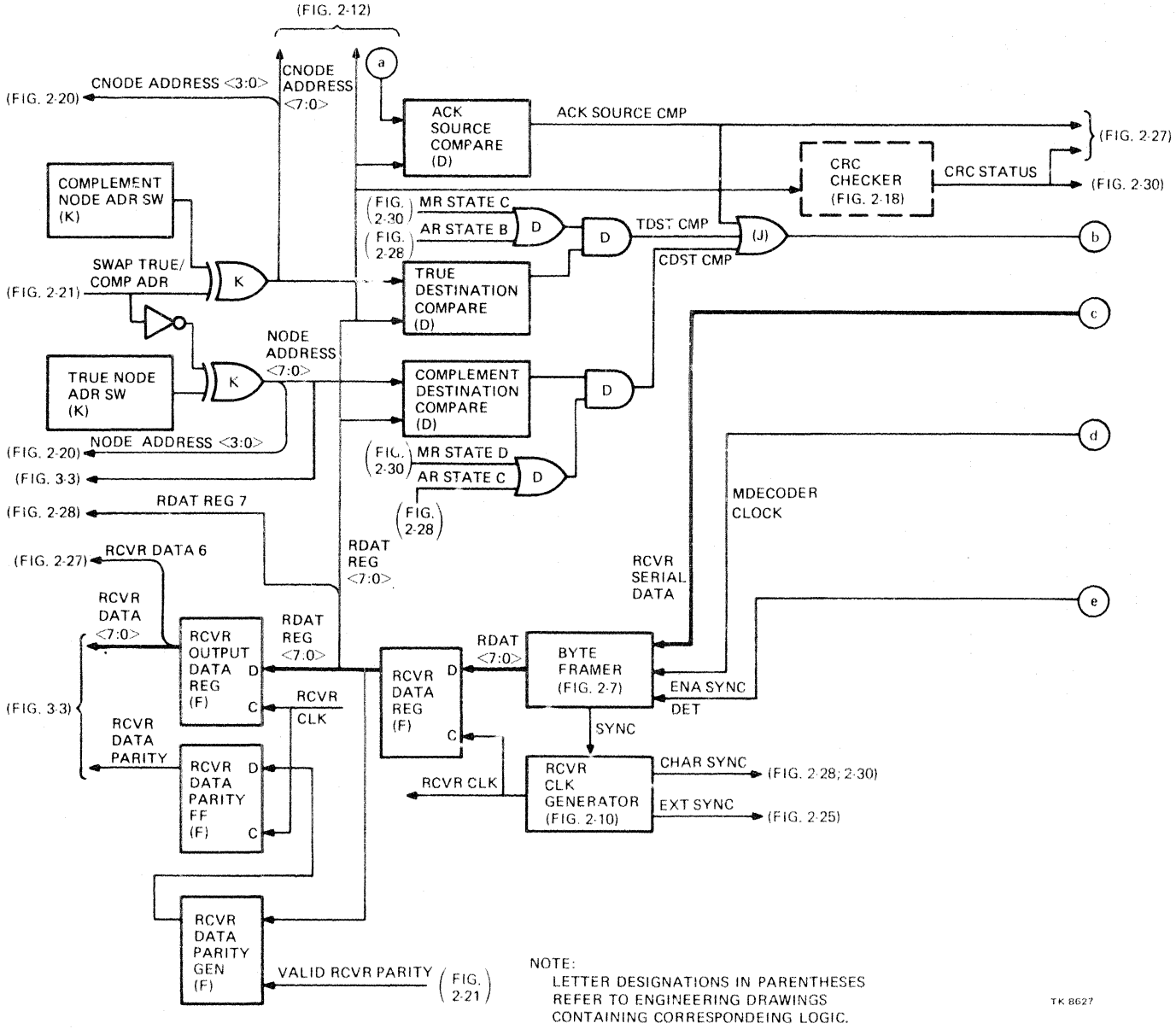


Figure 2-3 Receive Channel Block Diagram (a)

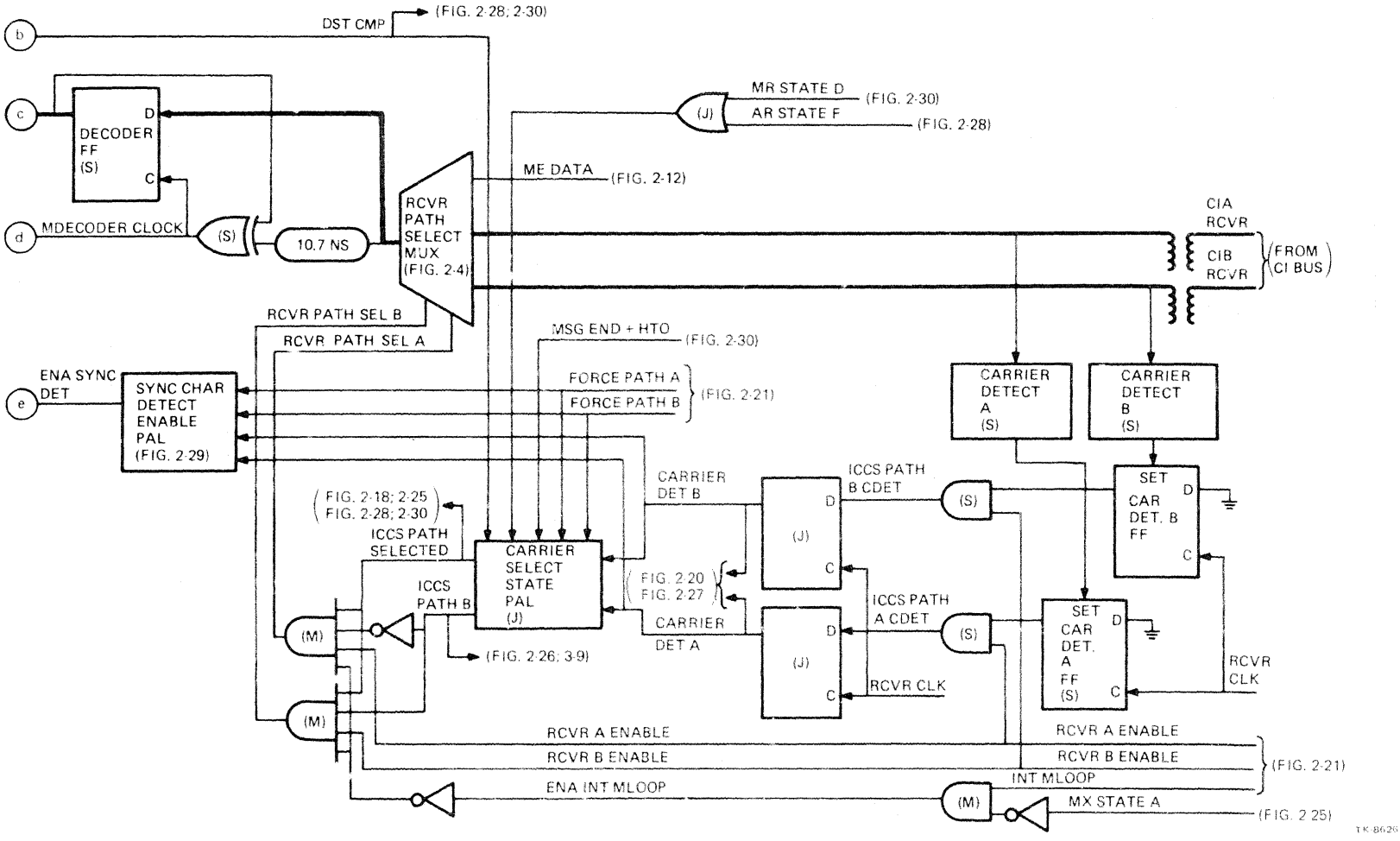


Figure 2-3 Receive Channel Block Diagram (b)

2.4.1.1 Carrier Detect Logic – Identical and parallel logic monitors paths A and B. If a carrier is present on CI path A, the carrier detect A logic sets the carrier detect A flip-flop. If the port has enabled channel A (RCVR A ENABLE true), ICCS PATH A CDET asserts and causes CARRIER DET A to be asserted by a flip-flop on the next RCVR CLK. The flip-flop outputs CARRIER DET A to the carrier select state PAL. If the existing state of the port is such that a receive channel may be opened, the carrier state select PAL outputs an asserted ICCS PATH SELECTED and a negated ICCS PATH B. RCVR PATH SEL A asserts to the receive path select mux to select CI path A for the mux input.

Note that the receiver carrier detect flip-flop is clocked by RCVR CLK which resets the flip-flop as soon as the carrier detect A output negates. Thus, the CI input path to the receive channel is closed once the carrier presence is no longer sensed.

Had activity been sensed on CI path B, similar logic would have selected CI path B for the mux input.

FORCE PATH A and FORCE PATH B from the link control logic force a corresponding path selection from the carrier select state PAL. When the port commands a message transmission, the path selected for the transmission is reserved in the receive channel in preparation to receive the ACK response.

The port and link control PALs can also select the internal maintenance loop (INT MLOOP) wherein ME DATA from the transmit channel is selected for the mux input. The true state of INT MLOOP inhibits both RCVR PATH A and RCVR PATH B which causes the mux to select the ME DATA input signal.

2.4.1.2 Receive Path Select Mux – ECL Logic – The receive path select mux is on sheet S of the engineering drawing set. The detailed operation of circuit logic is not usually described in a functional description manual, however, the operation of the mux is described here as an example of the ECL logic referred to in Paragraph 2.4.

Refer to Figure 2-4. If RCVR PATH SEL A is true, the output of OR gate A can follow the CIA RCVR signal input. The signal RCVR PATH B is false which holds the output of OR gate B low. In ECL logic, a signal low is the non-active state and a high is the active state. Any gate connected to a common line can pull the line up to the active state. Thus, OR gate B is held inactive (low) while OR gate A transfers the CIA RCVR signal to the Manchester decoder. The true state of RCVR PATH SEL A also holds the LOOP OR gate in the inactive state.

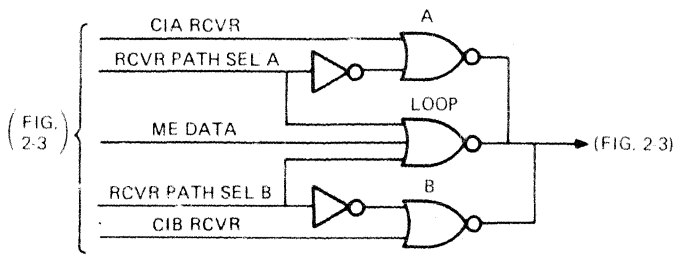


Figure 2-4 Receive Path Select Mux-ECL Logic

If RCVR PATH SEL B were true (RCVR PATH SEL A false), OR gate A and the LOOP OR gate would be held inactive and OR gate B would function to transfer CIB RCVR to the Manchester decoder.

If the internal maintenance loop is selected, both RCVR PATH SEL signals are false holding OR gates A and B in the inactive state. However, the LOOP OR gate is now active and transfers ME DATA to the Manchester decoder.

2.4.2 Manchester Decoder

2.4.2.1 Phase Encoding – Phase encoding (Figure 2-5) is a modulation technique in which a signal phase reversal occurs for each bit of information. A "1" is defined as a positive level followed by a negative transition, while a "0" is defined as a negative level followed by a positive transition. Phase reversals are at the data rate or at twice the data rate. Consecutive 1's or consecutive 0's will cause phase reversals to occur at twice the data rate (Figure 2-5A). Alternate 1's and 0's cause flux reversals to occur at the data rate (Figure 2-5B).

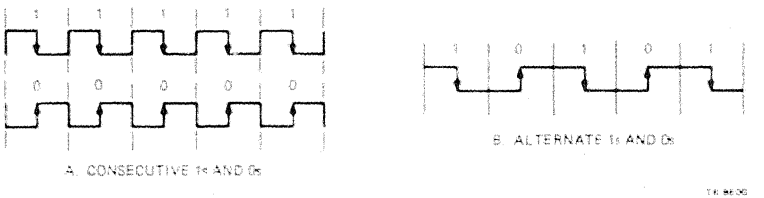


Figure 2-5 PE (Phase Encoded) Data

2.4.2.2 Decoder Logic – The Manchester decoder decodes the encoded signal data by separating out the 70 MHz bit rate clock (MDECODER CLOCK) leaving the serial data (RCVR SERIAL DATA). The decoder consists of a flip-flop with the signal data from the receive path select mux as the D input. The flip-flop clock input is derived from XORing the delayed output of the receive path select mux (delayed 10.7 ns) with the output of the decoder flip-flop.

Figure 2-6 illustrates the action of the decoder logic. The signal data from the receive path select mux is shown with 1 or 0 transitions at the center of each bit cell. With a 70 MHz bit rate, the width of the bit cells is 14.28 ns. The output of the delay line is seen as the signal data delayed 10.7 ns. XORing the delayed data with the flip-flop output (RCVR SERIAL DATA) generates the MDECODER CLOCK waveform. Note that in the case of alternating 1's and 0's, the width of the MDECODER CLOCK pulse is the set and reset times of the decoder flip-flop. In the case of consecutive 1's or 0's, the clock is identical to the inverse of the delayed data.

The MDECODER CLOCK is at 70 MHz with a 14.28 ns period. The XOR action serves to generate the clock's rising edge ¼ into each bit cell. This centers the rising edge in the valid strobe area (first half of the bit cell).

SEE NEXT FRAME

FOR LARGER ART

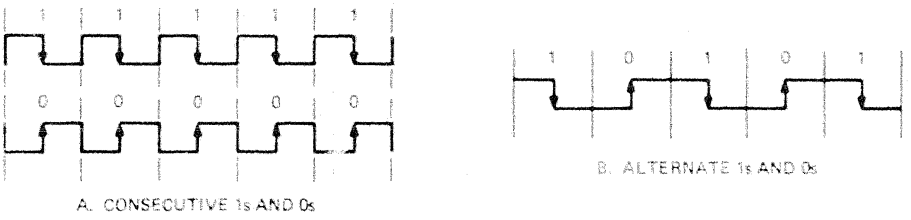


Figure 2-5 PE (Phase Encoded) Data

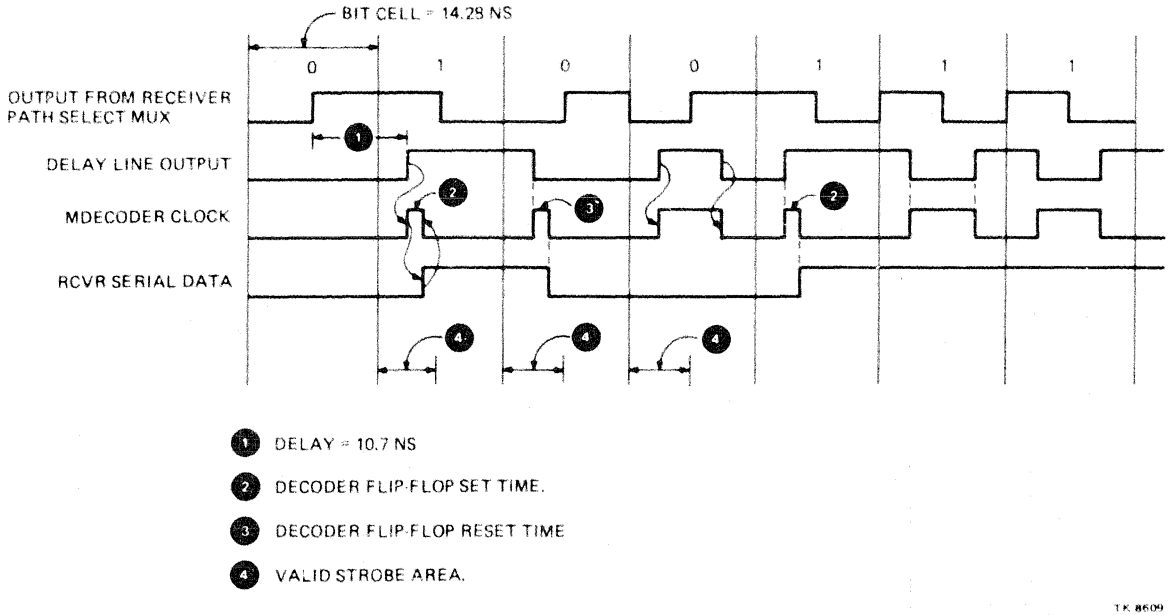


Figure 2-6 Manchester Decoder Timing Diagram

2.4.3 Sync Character Detect Enable PAL

The purpose of the sync character detect enable PAL is to assert ENA SYNC DET to the byte framer when a packet is expected. The PAL monitors CARRIER DET A and CARRIER DET B and asserts ENA SYNC DET when it senses that a signal carrier is being received. The PAL negates ENA SYNC DET during node transmissions (FORCE PATH A, FORCE PATH B) so the link will not respond to its own transmissions. The PAL asserts ENA SYNC DET immediately after information packet transmissions in anticipation of the ACK (or NACK) response.

The byte framer contains a sync detector which is enabled by ENA SYNC DET. The sync detector looks for the packet sync character as a means of recognizing that a packet is being received. When the detector recognizes the sync character, it enables the byte framer to start processing the packet bytes. By keeping the detector disabled except when a packet is expected, the sync character detect PAL prevents the detector from erroneously recognizing noise as a sync character.

The sync character detect enable PAL is discussed in more detail in Paragraph 2.10.2.2.

2.4.4 Byte Framer

The byte framer is enabled when it receives the sync character byte. Once the framer recognizes the sync character, it then functions to convert the serial signal data from the Manchester decoder into eight-bit data bytes for the RDAT bus.

As shown in Figure 2-7, RCVR SERIAL DATA is input to the RCVR serial shift register. The register is held in the load state by the negated state of E197-R2 (Figure 2-8), thus no data is shifted into the register. When a carrier presence is sensed at the front end of the receive channel, the sync character detect enable PAL also senses the carrier presence. If the PAL deems that this is a valid time to receive a packet,it asserts ENA SYNC DET to the SYNC ENA flip-flop. On the next RCVR CLK, the flip-flop outputs SYNC ENA to another flip-flop which asserts E197-R2 to the RCVR serial shift register. The true state of E197-R2 enables the register by changing its state from load to shift. RCVR SERIAL DATA is now shifted into the register at the 70 MHz bit rate by MDECODER CLOCK. Figure 2-8 illustrates the timing of the enabling of the RCVR serial shift register.

The RCVR serial shift register outputs eight-bit bytes onto a data bus. The data bytes are then applied to the RCVR input register. The sync detector monitors the data on the bus looking for the sync character byte. When the detector recognizes the sync character, it asserts E198-3 to the sync flip-flop. The next MDECODER CLOCK sets the flip-flop and asserts SYNC to the external data framer.

Note that only seven of the eight bits on the data bus are fed into the sync detector. The eighth bit is taken from the RCVR SERIAL DATA being fed into the RCVR serial shift register. Thus, the sync detector recognizes the sync character before the last character bit is shifted into the shift register. The next MDECODER CLOCK that clocks the last bit into the register, also sets the sync flip-flop. Hence, SYNC asserts when the sync character is in the shift register and not one clock pulse later (Figure 2-9).

When SYNC asserts, the external framer shift register functions to switch the RCVR input register from the hold state to the load state (for one clock pulse) every eight MDECODER CLOCK pulses. RCVR SERIAL DATA continues to be shifted into the RCVR serial shift register. Every eight clock pulses a data byte is present in the shift register and on the data bus. At this time the external framer shift register switches the RCVR input register from hold to load. The next MDECODER CLOCK pulse then loads the data byte into the register.

The D7 input to the external framer shift register is tied high. Before the assertion of SYNC, the framer register is in the load state, hence the R7 output is true. The true state of the R7 output keeps the RCVR input register in the load state. When SYNC asserts, the framer shift register starts to shift. The 1 at R7 is shifted in and through the framer shift register.

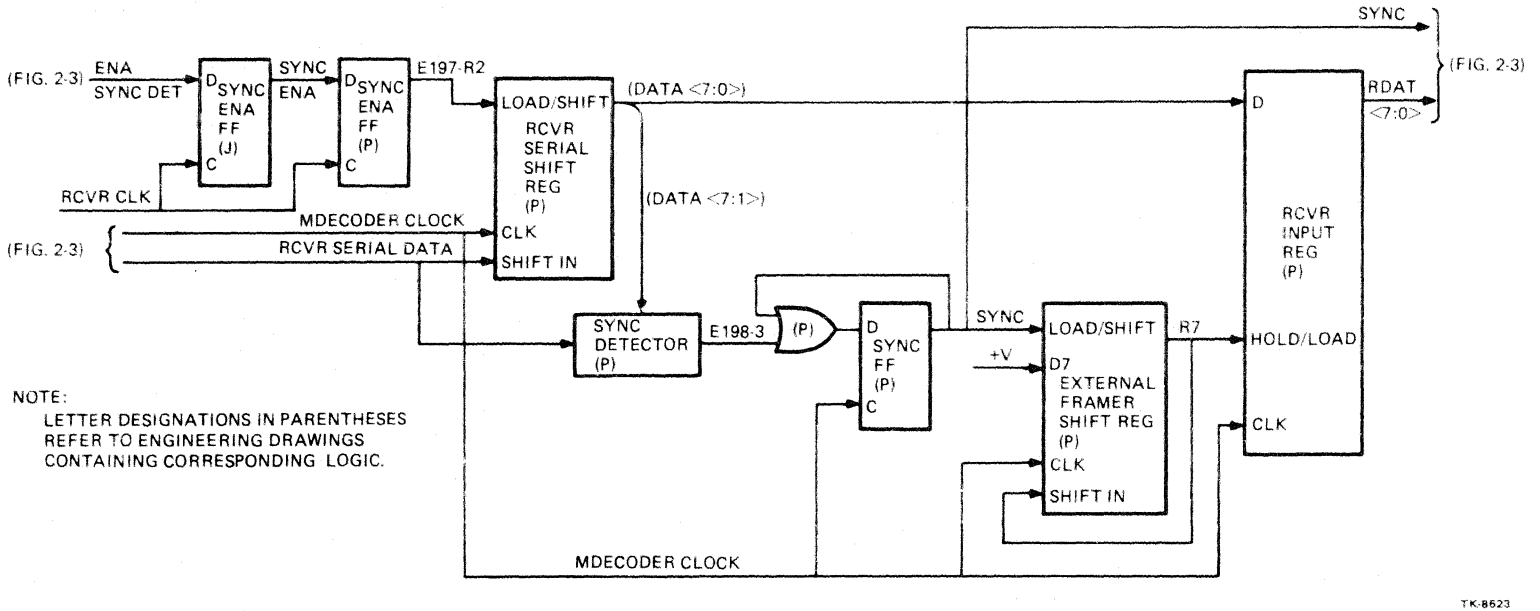


Figure 2-7 Byte Framer Block Diagram

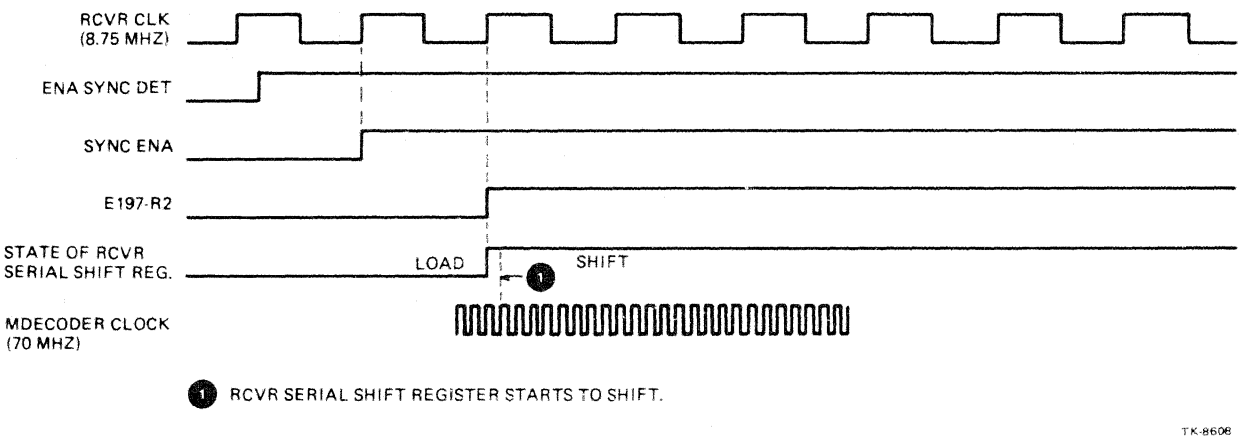
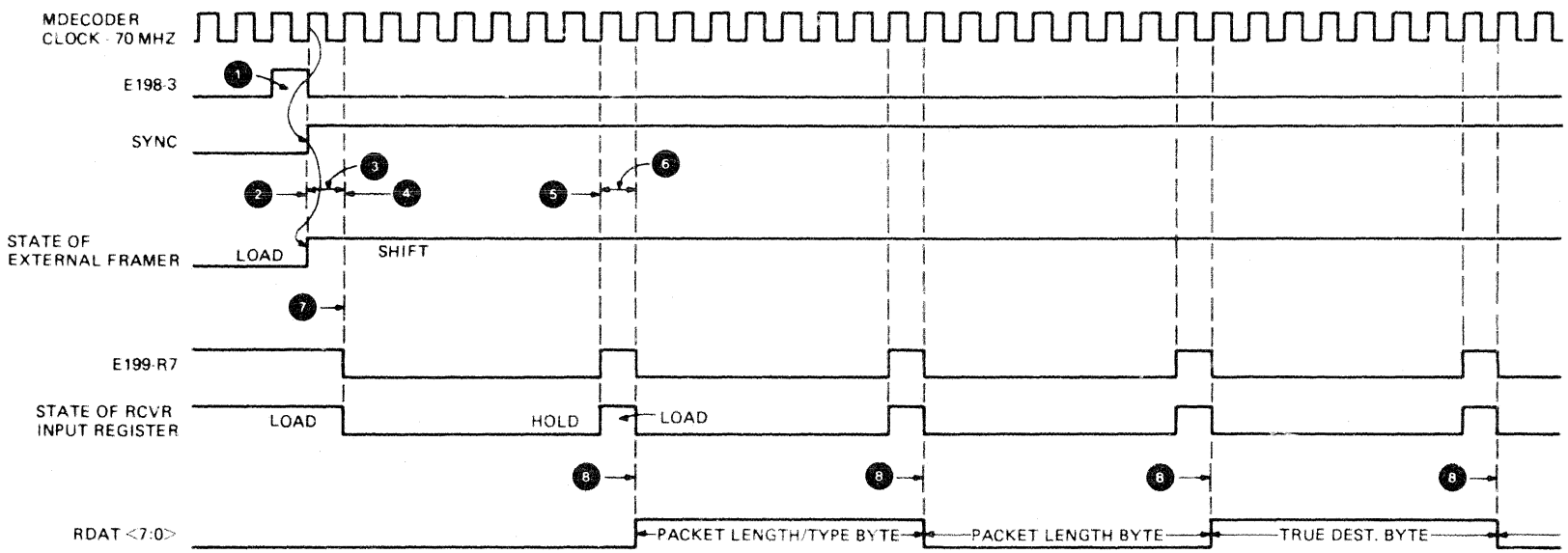


Figure 2-8 Enabling the RCVR Serial Shift Register



1. SYNC CHARACTER BYTE IN SYNC DETECTOR.
2. LAST BIT OF SYNC CHARACTER BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.
3. SYNC CHARACTER BYTE IN RCVR SERIAL SHIFT REGISTER.
4. FIRST BIT OF PACKET LENGTH/TYPE BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.
5. LAST BIT OF PACKET LENGTH/TYPE BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.
6. PACKET LENGTH/TYPE BYTE IN RCVR SERIAL SHIFT REGISTER.
7. EXTERNAL FRAMER SHIFT REGISTER STARTS TO SHIFT.
8. (DATA <7:0>) CLOCKED INTO RCVR INPUT REGISTER.

TK-8612

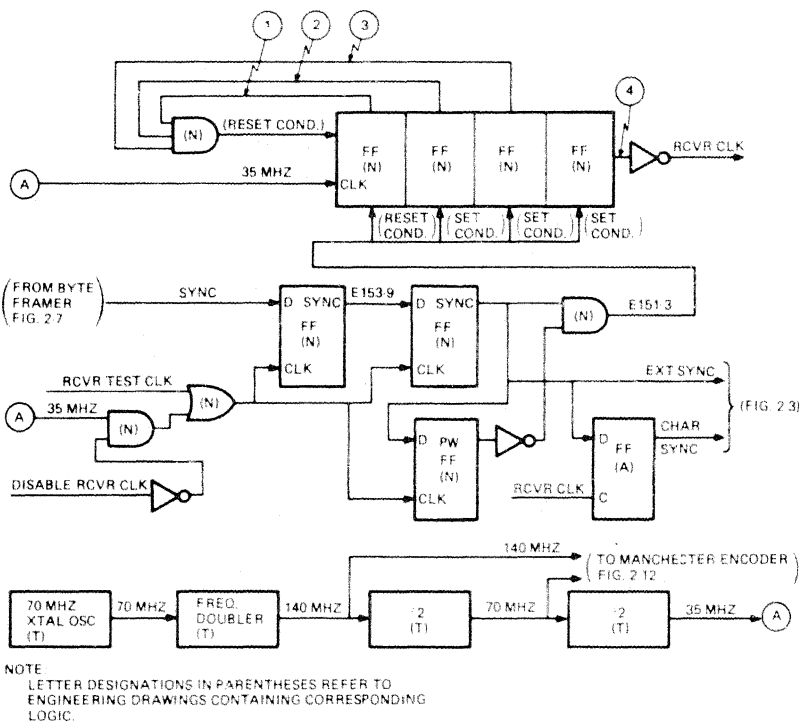
Figure 2-9 Byte Framer Timing Diagram

Every eight MDECODER CLOCK pulses, the 1 is shifted through to the R7 output, switching the RCVR input register to the load state for one clock pulse. As seen in Figure 2-9, the timing is such that a data byte is on the data bus when the RCVR input register is loaded. The timing for the first three bytes of a packet is shown in Figure 2-9.

2.4.5 RCVR CLK Generator

Figure 2-10 is a block diagram of the RCVR CLK generator. The RCVR CLK is derived from a crystal-controlled 70 MHz oscillator. The RCVR CLK pulses function to time and control the operation of the receive channel logic. When a signal packet is received, the RCVR CLK is synchronized to the packet bytes by SYNC received from the byte framer.

The output from the 70 MHz crystal-controlled oscillator is doubled to 140 MHz by a frequency doubler. (The 140 MHz is used in the Manchester encoder in the transmit channel.) The 140 MHz is divided down to 35 MHz and then applied to a shift register consisting of four flip-flops. The shift register divides the 35 MHz by four, outputing RCVR CLK at a frequency of 8.75 MHz (period = 114.28 ns).



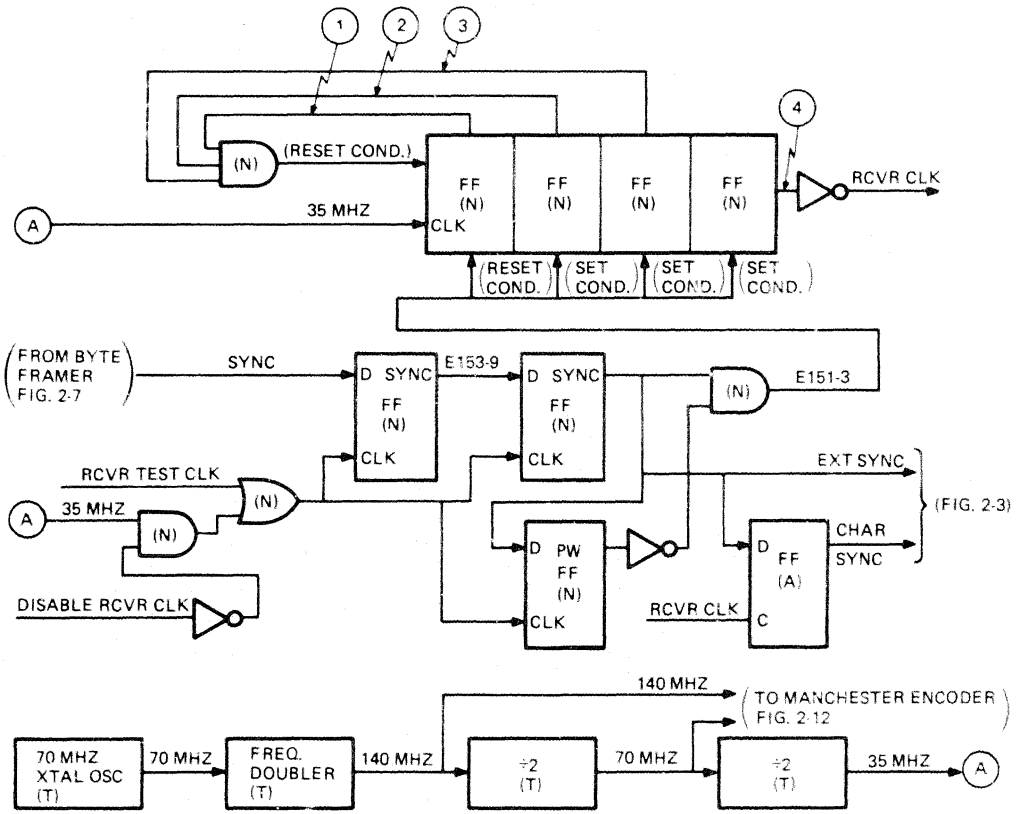
NOTE
LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

TK-8622

Figure 2-10 RCVR CLK Generator

SEE NEXT FRAME

FOR LARGER ART



NOTE:
LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

TK-8622

Figure 2-10 RCVR CLK Generator

Table 2-2 list the frequency and period of the link clocks. The XMIT CLK (discussed in Paragraph 2.5.7) is included in the table.

The register functions to shift a logic low through the flip-flop chain. When the low is in the rightmost flip-flop, the other three flip-flops are set. Outputs from the three set flip-flops are ANDed together to condition the first flip-flop to reset on the next clock pulse thus re-inserting the low into the flip-flop chain. The cycle is then repeated.

Table 2-2 Link Clocks

Frequency (MHz)	Period (ns)	Clock
70	14.28	MDECODER CLOCK
35	28.57	—
8.75	114.28	RCVR CLK
8.75	114.28	XMIT CLK

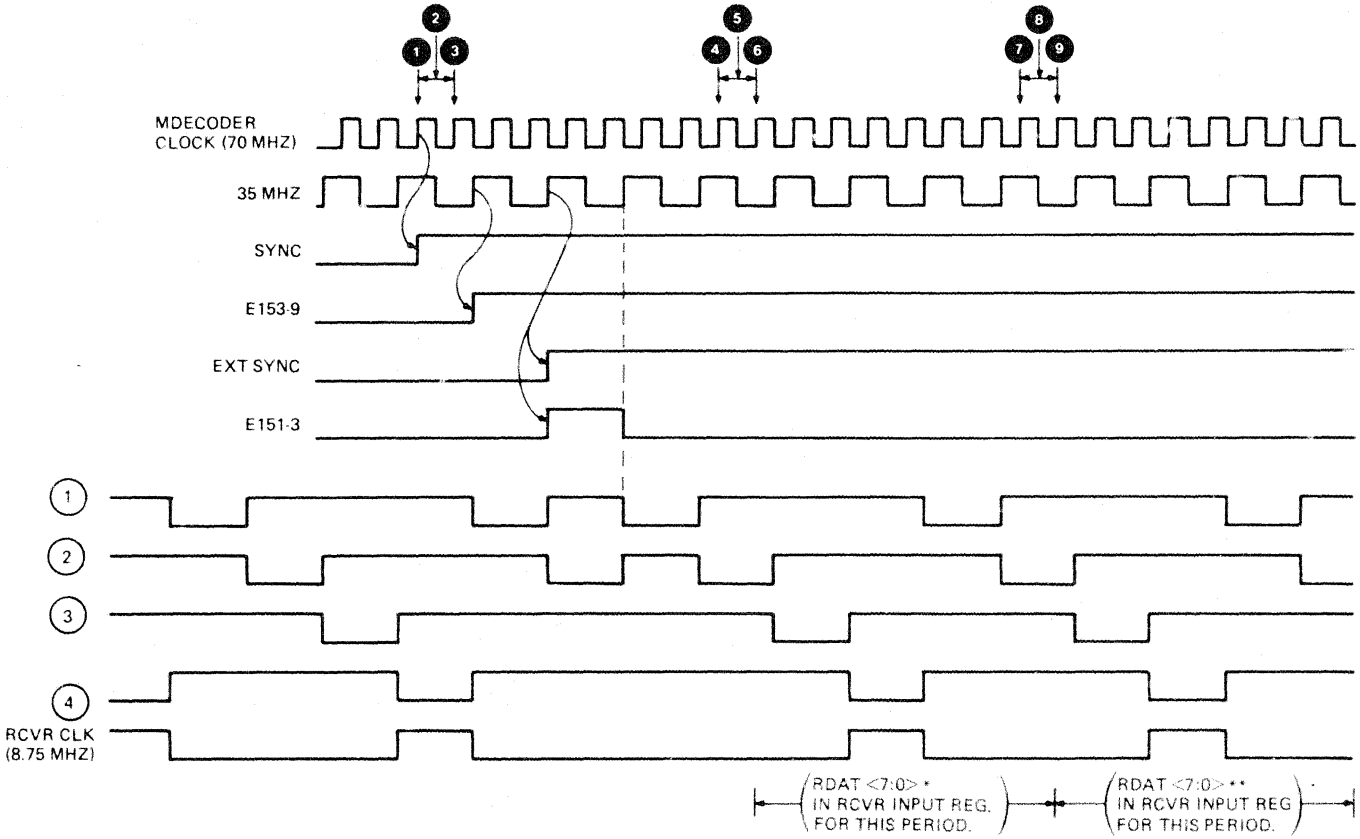
The left and right portions of Figure 2-11 illustrate the operation cycle of the shift register. (The center portion illustrates the synchronization function.) Waveforms 1, 2, 3, and 4 relate to the corresponding points in Figure 2-10. Also shown is the MDECODER CLOCK and SYNC from the byte framer, and the time periods that the RDAT ⟨7:0⟩ bytes are in the RCVR input register. These three signals are time related to each other and are shown as they appear in the byte framer timing diagram (Figure 2-9). The 35 MHz clock and the shift register waveforms are time related to each other but are independent of the byte framer timing. The SYNC signal is used to synchronize the action of the shift register with the data bytes from the byte framer.

As shown in Figure 2-10, when SYNC asserts, two sync flip-flops are set by the 35 MHz clock which in turn assert E151-3. The next 35 MHz clock sets a pulse width (PW) flip-flop which negates E151-3, thus forming an E151-3 pulse to the shift register. The E151-3 pulse synchronizes the register by forcing a reset condition on the first flip-flop and a set condition on the other three flip-flops. The next 35 MHz clock pulse places the register into the conditioned state which is to introduce a logic low into the first flip-flop. Thus, regardless of where the register was in its cycle, it is restarted at the beginning of the cycle.

The assertion of SYNC followed by the assertion of E151-3 is seen in Figure 2-11. Note that the conditions forced onto the shift register by the E151-3 pulse are clocked in by the next 35 MHz clock pulse (the first flip-flop is reset and the other three are set). As seen in Figure 2-11, the logic low had reached the second flip-flop when the register cycle was interrupted and reset back to its starting point. The register cycles from this point on are in synchronization with the byte frame. This results in the generation of RCVR CLK pulses approximately centered in the time period when the packet bytes (RDAT ⟨7:0⟩) are in the RCVR input register.

2.4.6 CRC Check

The packet bytes on the RDAT bus, up to and including the four CRC bytes, are input to the CRC checker. If no errors are detected by the checker, the checker asserts CRC STATUS to the message receive state logic, indicating the reception of a valid, error-free packet.



1. LAST BIT OF SYNC CHARACTER BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.
2. SYNC CHARACTER BYTE IN RCVR SERIAL SHIFT REGISTER.
3. FIRST BIT OF PACKET LENGTH/TYPE BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.
4. LAST BIT OF PACKET LENGTH/TYPE BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.
5. PACKET LENGTH/TYPE BYTE IN RCVR SERIAL SHIFT REGISTER.
6. FIRST BIT OF PACKET LENGTH BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.
7. LAST BIT OF PACKET LENGTH BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.
8. PACKET LENGTH BYTE IN RCVR SERIAL SHIFT REGISTER.
9. FIRST BIT OF TRUE DESTINATION BYTE CLOCKED INTO RCVR SERIAL SHIFT REGISTER.

* PACKET TYPE/LENGTH BYTE
** PACKET LENGTH BYTE

TK 8610

Figure 2-11 RCVR CLK Synchronization

2.4.7 Destination Compare

The node address and the complement of the node address are set into two sets of eight-contact node address switches. The eight-bit output of the complement node address switch is applied to the true destination compare logic as CNODE ADDRESS ⟨7:0⟩. The eight-bit output of the true node address switch is applied to the complement destination compare logic as NODE ADDRESS ⟨7:0⟩.

The true destination byte and complement destination byte are applied from the RDAT bus to both destination compare logic circuits. The state PALs enable the compare logic outputs such that when the true destination byte is on the RDAT bus, the output of the true destination compare logic is enabled. If the true destination byte matches CNODE ADDRESS ⟨7:0⟩ from the complement node address switch, TDST CMP asserts indicating that a true address match was obtained. Likewise, when the complement destination byte is on the RDAT bus, the output of the complement destination compare logic is enabled. If the complement destination byte matches NODE ADDRESS ⟨7:0⟩ from the true node address switch, CDST CMP asserts indicating that a complementary address match was obtained.

True and complement destination matches assert DST CMP to the message receive and ACK receive state logic.

A polarity reversal in the compare logic results in the output of the true node address switch being applied to the complement destination compare logic and the output of the complement node address switch being applied to the true destination compare logic.

The output of the node address switches is coupled to the compare logic via XOR gates. This allows the true address and the complement address to be swapped for maintenance testing.

2.4.8 ACK Source Comparison

The ACK source compare logic is used only during the reception of an ACK packet. The ACK packet was transmitted from its source to acknowledge an information packet that was transmitted from this node. When the information packet was in the transmit channel, the destination address was saved and applied into the ACK source compare logic.

The ACK source compare logic receives inputs from the transmit channel and from the RDAT bus. When the source byte of the ACK packet is on the RDAT bus, the output of the compare logic is sampled. If a match is obtained, ACK SOURCE CMP is asserted indicating that the source address of the ACK packet matches the destination address of the preceding information packet.

2.4.9 Receive Data Parity And Channel Output

Data bytes are transferred from the RDAT bus to the PB via the receiver output data register. The bytes are output from the register as RCVR DATA ⟨7:0⟩.

The data bytes are also applied from the RDAT bus into a receiver data parity generator where odd parity is generated on each byte. A ninth input to the parity generator (VALID RCVR PARITY) provides a means of introducing parity errors for maintenance testing. The output from the parity generator is applied to a parity flip-flop which outputs RCVR DATA PARITY to the PB.

2.5 TRANSMIT CHANNEL

Figure 2-12 is a block diagram of the transmit channel and should be referred to throughout Section 2.5.

2.5.1 Transmit Data Input

Transmit data from the PB (XMIT DATA ⟨7:0⟩) is input into the transmit channel via the XMIT data input latch and then transferred to the XMIT data bus as XMIT DATA BUS ⟨7:0⟩. The input latch is transparent in that the data on the XMIT data bus will follow the XMIT DATA ⟨7:0⟩ input so long as the latch is enabled by ENA XMIT DATA LATCH from the transmit control logic and by the high state of XMIT CLK. When XMIT CLK is low, the latch is disabled (closed).

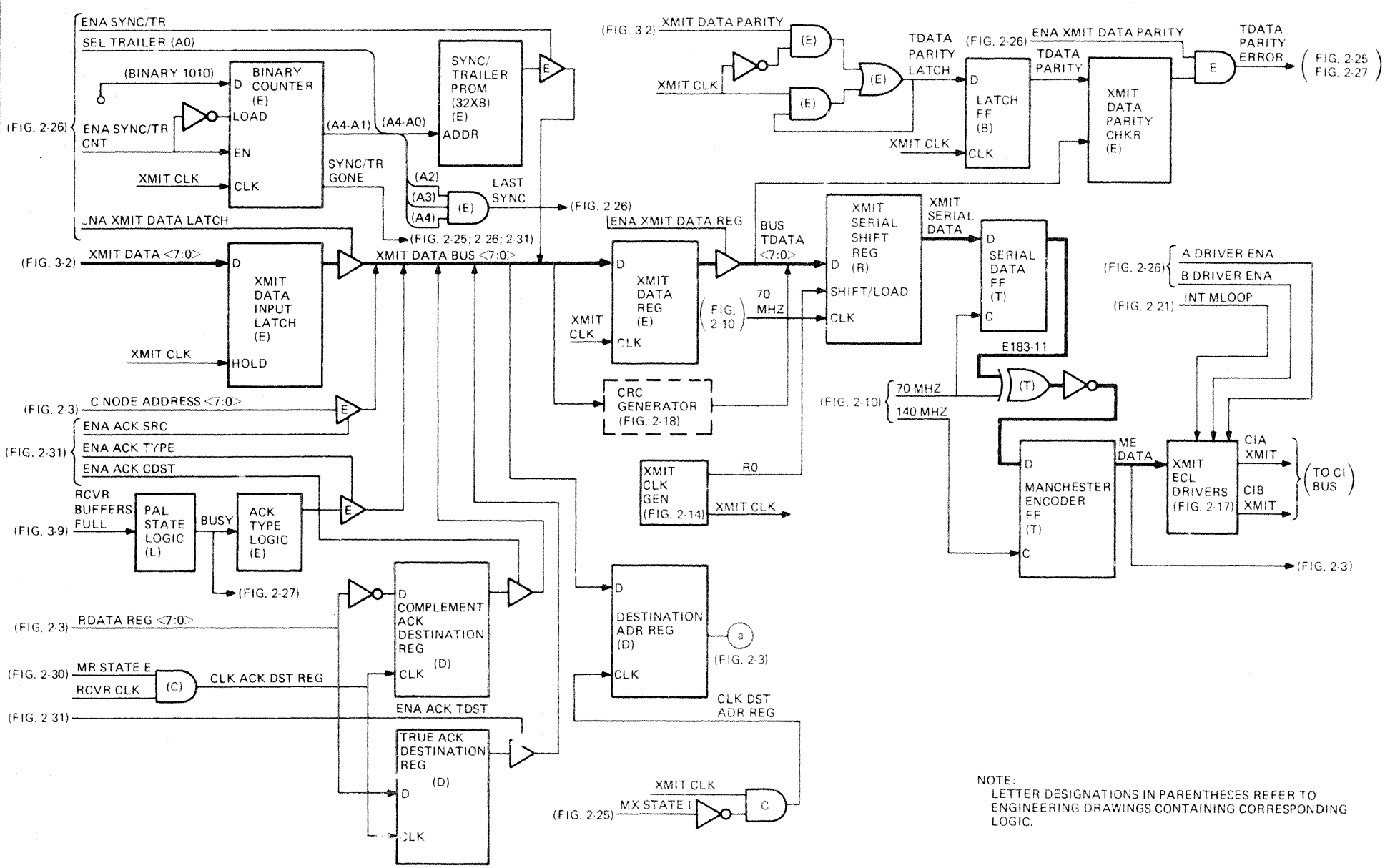


Figure 2-12 Transmit Channel Block Diagram

2.5.2 Bit Sync, Sync Character, and Trailer Bytes

The bit synchronization bytes, the sync character byte, and the trailer bytes reside in a 32 × 8 PROM. The PROM output is enabled by ENA SYNC/TR from the transmit control logic. A five-bit address input to the PROM ((A4:A0)) selects the output bytes which are placed onto the XMIT data bus.

Figure 2-13 illustrates the 32 eight-bit locations in the sync/trailer PROM. The five bit-sync bytes and the sync character byte are located in the upper area of PROM space. They are spaced at every other location starting at address 10101. The six trailer bytes are located in between the sync bytes starting at address 10100. The lower area of the PROM is reserved for possible extension of the header to 16 bytes (15 bytes of bit synchronization and one byte for the sync character).

PROM address bits (A4:A1) are obtained from a binary counter which is enabled by ENA SYNC/TR CNT from the transmit control logic. When ENA SYNC/TR CNT is false, the counter is loaded with starting address 1010. When ENA SYNC/TR CNT asserts, the counter counts up from 1010 addressing every other PROM location. The PROM's least significant address bit (A0) is SEL TRAILER from the transmit control logic. When SET TRAILER is false, the PROM sync bytes are addressed. When SEL TRAILER is true, the PROM trailer bytes are addressed.

Address bits (A4:A2) are monitored and cause LAST SYNC to be asserted to the transmit control logic when all three bits are true. As is shown in Figure 2-13, this occurs when the last sync byte (sync byte 5) is being addressed.

When the binary counter has counted up past the last trailer byte (or past the sync character byte) it overflows and asserts SYNC/TR GONE to the PAL state logic.

2.5.3 ACK Packet Insert

The packet type, source, and destination bytes are inserted into ACK packets by the link. When information packets are being transmitted, these bytes are inserted by the port and do not involve the link hardware.

2.5.3.1 Packet Type Byte – The packet type byte is obtained from the ACK type logic. The logic outputs a 1 in bit position 7 signifying an ACK (or NACK) packet. Bit position 6 is a function of BUSY which is derived from RCVR BUFFERS FULL from the PB. If the receive buffers in the PB are full, the information packet just received could not be accepted by the node causing BUSY to assert. This causes a 1 in bit position 6 signifying that a NACK packet is being transmitted. If BUSY is false, bit position 6 is 0 indicating that an ACK packet is being transmitted.

Bits (5:0) from the ACK type logic are always 0.

2.5.3.2 Source Byte – The ACK source byte is complement node address (CNODE ADDRESS (7:0)) obtained from the complement node address switch. The source byte is gated onto the XMIT data bus by ENA ACK SRC from the PAL state logic.

2.5.3.3 Destination Bytes – The ACK destination bytes are derived from the source byte of the associated information packet. The source byte is taken from the RDAT bus in the receive channel and clocked into the destination address registers by CLK ACK DST REG. RDAT REG (7:0) is entered directly into the true ACK destination register while the inverse (complement) is entered into the complement ACK destination register. The true destination byte and the complement destination byte are gated to the XMIT data bus by ENA ACK TDST and ENA ACK CDST, respectively. The gating signals are asserted by the PAL state logic to insert the bytes into the ACK packet at the appropriate insertion times.

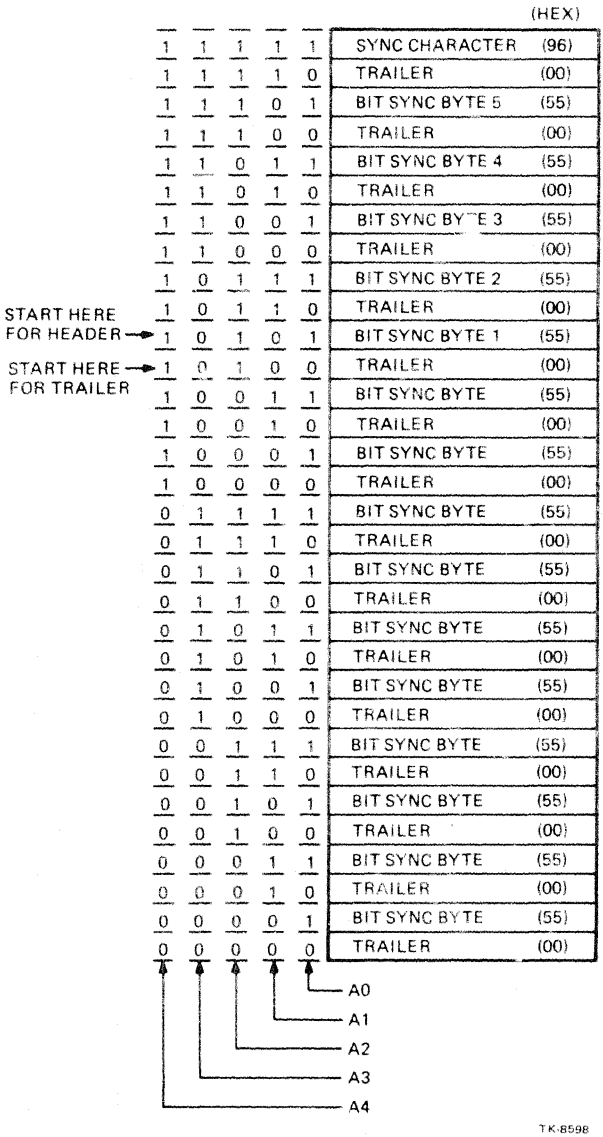


Figure 2-13 Sync/Trailer PROM Space

2.5.4 Destination Address Register

The destination address register saves the destination address of an information packet that is being transmitted. CLK DST ADR REG asserts at the correct time to clock the true destination byte into the register. The destination byte is used when the associated ACK packet is received. It is compared to the source of the ACK packet in the receive channel where a match will be obtained if the correct node responded to the message transmission.

2.5.5 Transmit Data Parity Check

Data on the XMIT data bus is transferred to the BUS TDATA bus via the XMIT data register. The register output is gated to the BUS TDATA bus by ENA XMIT DATA REG from the PAL state logic.

Data from the BUS TDATA bus is applied to the XMIT data parity checker where a parity check is made on the packet bytes. The parity bits (XMIT DATA PARITY) are received from the PB and applied to a latch flip-flop as TDATA PARITY LATCH. An OR feedback network holds TDATA PARITY LATCH true for both alternations of XMIT CLK to allow the latch flip-flop to set (if parity is a 1). The latch flip-flop outputs the parity bit (TDATA PARITY) to the parity checker. Parity is checked when ENA XMIT DATA PARITY asserts and enables the parity checker output. If a parity error occurred, TDATA PARITY ERROR is asserted to the message state logic.

2.5.6 CRC Generation

The packet bytes on the XMIT data bus, starting with the packet type byte and ending with the last byte of the body, are input into the CRC generator. The generator functions to produce a 32-bit CRC longword unique to the packet being transmitted. The longword is inserted into the packet, a byte at a time, after the packet body.

2.5.7 XMIT CLK Generator

Figure 2-14 is a block diagram of the XMIT CLK generator. The transmit clock (XMIT CLK) is derived from a 70 MHz input received from a crystal oscillator network in the RCVR CLK generator. The transmit clock generator functions to produce XMIT CLK pulses at 8.75 MHz (period = 114.28 ns). The generator also outputs an R0 pulse to load the XMIT serial shift register from the TDATA bus.

The XMTR framer shift register is clocked at 70 MHz and has an eight-bit parallel output ((R7:R0)). The inverse of bits (R6:R0) are ANDed such that when all seven bits are false, a 1 is input to the framer shift register. The 1 is clocked up to the R7 output at which time another 1 is generated for the shift register input. This action is illustrated in Figure 2-15.

R6 and R7 from the framer shift register are applied to the D input of the XMIT CLK flip-flop causing the flip-flop to set for two 70 MHz clocks. The output of the flip-flop is XMIT CLK. Figure 2-15 illustrates the time relationship of XMIT CLK relative to the outputs of the framer shift register.

For maintenance testing, the output of the XMIT CLK flip-flop can be disabled and an XMIT TEST CLK substituted.

2.5.8 Parallel to Serial Data Conversion

Eight-bit data bytes from the TDATA bus are input to the XMIT serial shift register. R0 from the XMIT CLK generator asserts every eighth 70 MHz clock to load the shift register with a data byte from the TDATA bus. After being loaded, the register returns to the shift state and shifts out the data byte a bit at a time as XMIT SERIAL DATA. As the last bit is shifted out, R0 asserts again to load the next packet byte into the serial shift register. Figure 2-15 illustrates the load and shift time periods of the serial shift register.

The XMIT SERIAL DATA is applied to a serial data flip-flop clocked by 70 MHz. The flip-flop output (E183-11) is then applied to the Manchester encoder.

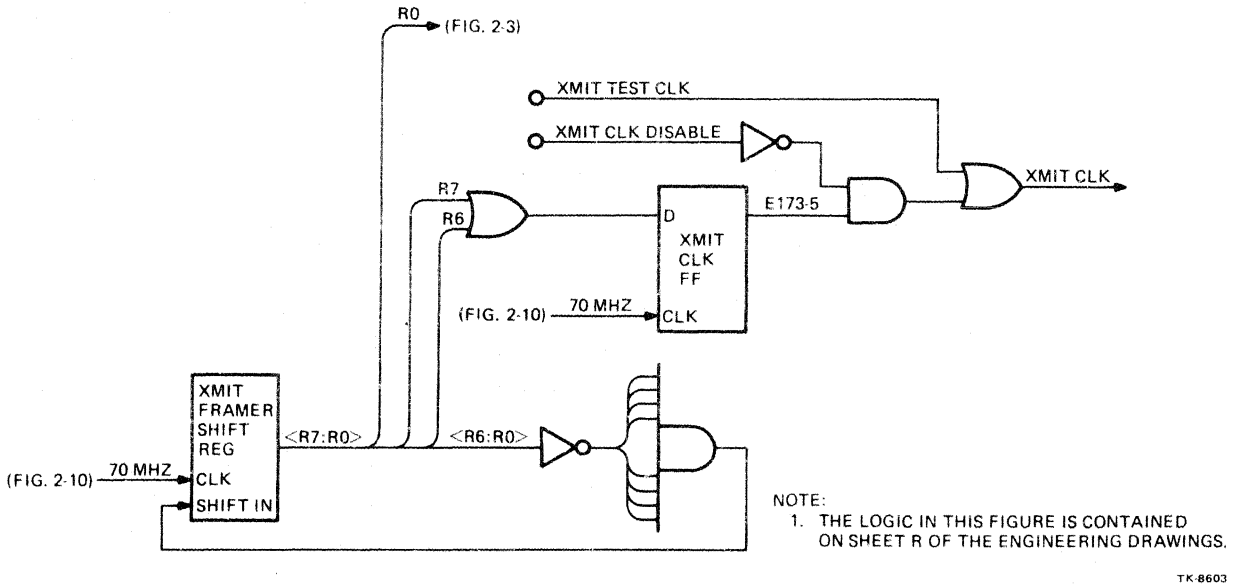


Figure 2-14 XMIT CLK Generator Block Diagram

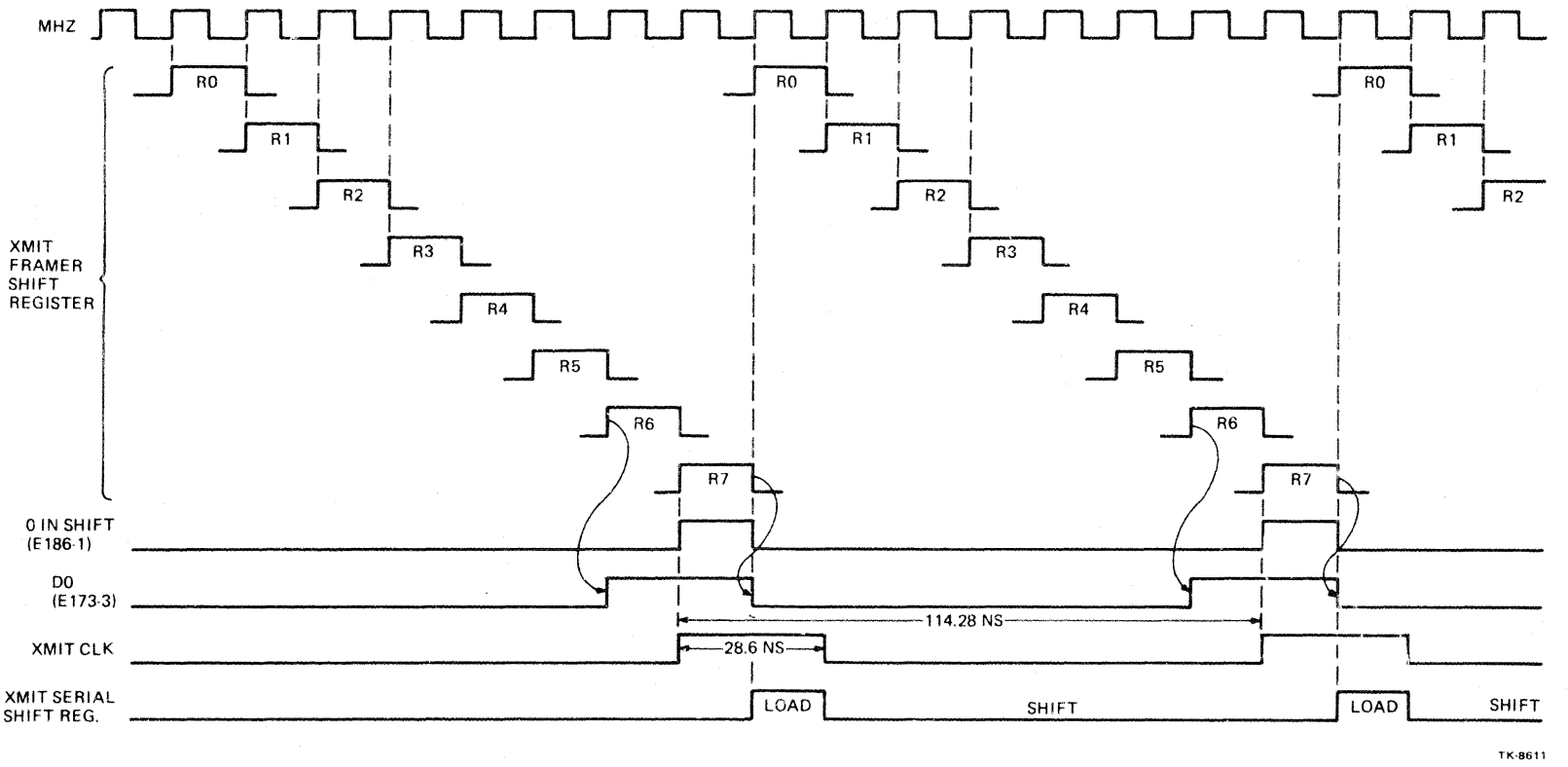


Figure 2-15 XMIT CLK Generator Timing Diagram

2.5.9 Manchester Encoder

The Manchester encoder modulates the serial data with the data bit rate clock to produce the signal format that is placed onto the CI bus.

The encoder logic consists of XORing the E183-11 output of the serial data flip-flop with the 70 MHz clock. The output of the XOR gate is inverted and applied to the Manchester encoder flip-flop. The encoder flip-flop is clocked at 140 MHz (twice the data rate) as required for phase encoded (PE) data (see Paragraph 2.4.2.1). The output of the Manchester encoder flip-flop (ME DATA) is the packet data ready to be transmitted onto the CI bus.

The action of the Manchester decoder can be seen from the timing diagram of Figure 2-16. The E183-11 output of the serial data flip-flop is shown for the given data bits. The result of XORing E183-11 with the 70 MHz is seen. Using the inverse of the XOR output for the encoder flip-flop D input, and the 140 MHz for the clock, the resultant ME DATA waveform is derived. The ME DATA signal format is identical to the format of the serial data received from the CI bus as shown in Figure 2-6.

2.5.10 XMIT ECL Drivers

The ME DATA from the Manchester encoder is transferred to the CI bus via XMIT ECL drivers (Figure 2-17). The XMIT drivers are divided into two channels feeding the A and B paths on the CI bus. Path selection is made by the port via the transmit control logic which enables the driver in the selected channel.

The ME DATA is routed to drivers in both channels and then through coupling transformers to the CI bus as CIA XMIT and CIB XMIT. The XMIT drivers are enabled by redundant XOR gates. When the transmit control logic selects channel A, A DRIVER ENA asserts (B DRIVER ENA false) and in turn asserts E151-1 from the channel A AND gate. The assertion of E151-1 causes outputs from both channel A XOR gates which in turn enables the channel A driver.

Likewise, the assertion of B DRIVER ENA from the transmit control logic causes the assertion of the E151-2 output of the channel B AND gate and thus enables the channel B driver.

Redundancy exists in the driver enabling logic to prevent the possibility of a single component failure causing the A and B channels to be enabled simultaneously. If through a logic component failure, the outputs of both the channel A and channel B AND gates were asserted (E151-1 and 2 both true), one of the channel A XOR gates and one of the channel B XOR gates would be inhibited. This would hold the enabling inputs to the channel drivers high to inhibit the drivers and isolate the node from the CI bus.*

The port can also select internal maintenance loop operation where the ME DATA from the transmit channel is looped back into the receive channel. To do this, the port control logic asserts INT MLOOP which inhibits both E151 AND gates, shutting off both output drivers. In addition, the signal lines into both the A and B channel drivers are held high by INT MLOOP to inhibit any signal data variations into the drivers.

* The operation of ECL logic is described in Paragraph 2.4.1.2.

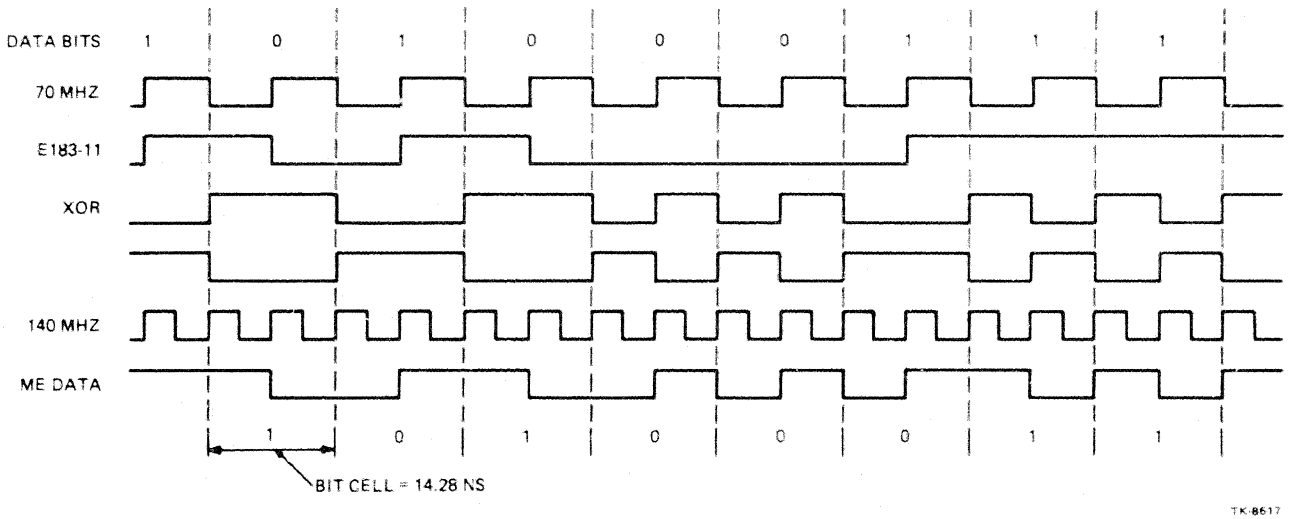


Figure 2-16 Manchester Encoder Timing Diagram

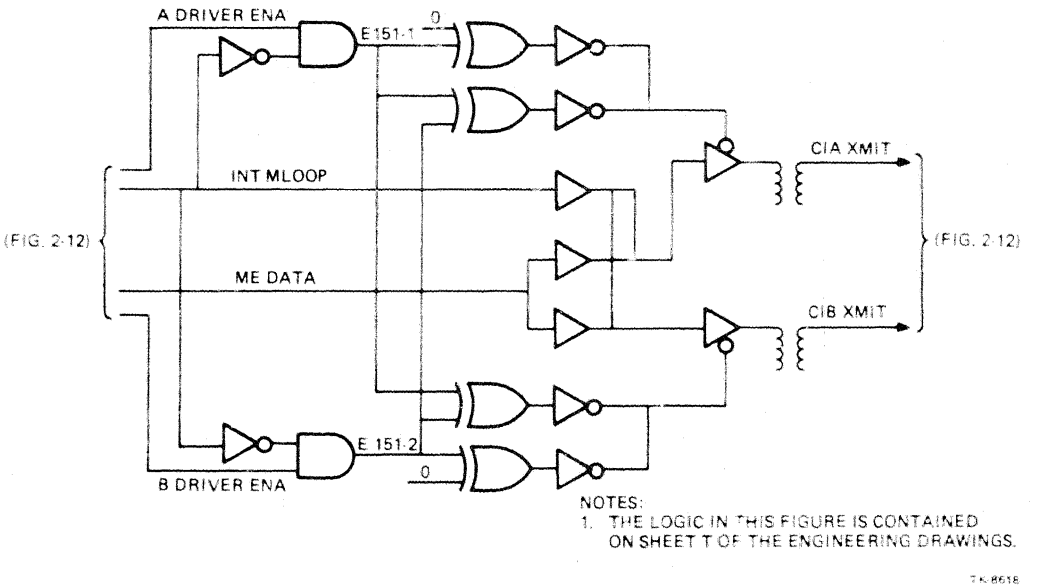


Figure 2-17 XMIT ECL Drivers

2.6 CRC GENERATOR AND CHECKER

Figure 2-18 is a block diagram of the CRC generator and checker.

2.6.1 CRC Generator

Packet bytes from the XMIT data bus in the transmit channel are input to the CRC input mux as XMIT DATA BUS ⟨7:0⟩. The transmit control logic asserts XMIT CRC ENA to the mux to select the bytes from the transmit channel. The mux output is applied to the CRC input register which outputs the bytes as NEW DATA ⟨7:0⟩.

NEW DATA ⟨7:0⟩ is applied to a CRC lookup table via an XOR gate. The lookup table logic generates the CRC longword for the packet being transmitted. CRC TABLE ⟨31:00⟩ from the lookup table logic is applied to a CRC register which outputs CRC ⟨31:00⟩.* CRC ⟨31:00⟩ is looped back into the lookup table logic in two parts. The first three bytes (CRC ⟨23:00⟩) are applied directly into the table logic while the upper byte (CRC ⟨31:24⟩) is XORed with the new input byte from the CRC input register. Thus, the new data bytes are continuously integrated into the compilation of the previous data bytes such that the CRC-generated longword is always a function of the packet bytes received from the transmit channel.

The CRC longword from the CRC register is also coupled to the BUS TDATA bus in the transmit channel via four drivers. When enabled, each driver places a byte onto the BUS TDATA bus to insert a CRC byte into the packet being transmitted.

The drivers are enabled from a CRC byte counter. The counter receives a SHIFT IN input (E29-5) when the last byte of the packet body is on the BUS TDATA bus. The input is shifted through the counter by CRC CLOCK asserting R0 through R3 in sequence. R0 through R3 are applied to four AND gates which are enabled at the appropriate time from the PAL state logic.

In addition, ENA XMIT DATA REG must be false before the R0 AND gate is enabled to place the first CRC byte (CRC ⟨7:0⟩) onto the TDATA bus. This insures that the TDATA bus is isolated from the XMIT data register before the CRC logic is connected to the bus (see Figure 2-12). Likewise, each AND gate must be disabled in sequence before the next AND gate can be enabled. This insures that only one source is driving the TDATA bus at any one time.

The CRC generator logic is clocked by CRC CLOCK which is seen to be XMIT CLK during the transmit states.

2.6.2 CRC Checker

Packet bytes from the RDAT register bus in the receive channel are input to the CRC input mux as RDAT REG ⟨7:0⟩. The transmit control logic negates the XMIT CRC ENA input to the mux selecting the bytes from the receive channel. The mux output is applied to the CRC input register which outputs the bytes as NEW DATA ⟨7:0⟩.

The CRC logic functions to generate the CRC longword (CRC ⟨31:00⟩) from the packet bytes as described in Paragraph 2.6.1. The last four bytes input to the CRC logic is the CRC longword generated for the packet. When the CRC longword is entered into the CRC lookup table, an output of DEBB 28E3 (hexadecimal) will be obtained if the packet transfer was error free.

The longword is applied to a CRC comparator which checks the value of the longword and asserts CRC STATUS if the proper value was obtained.

---

* The CRC register is initially preset to all 1's.

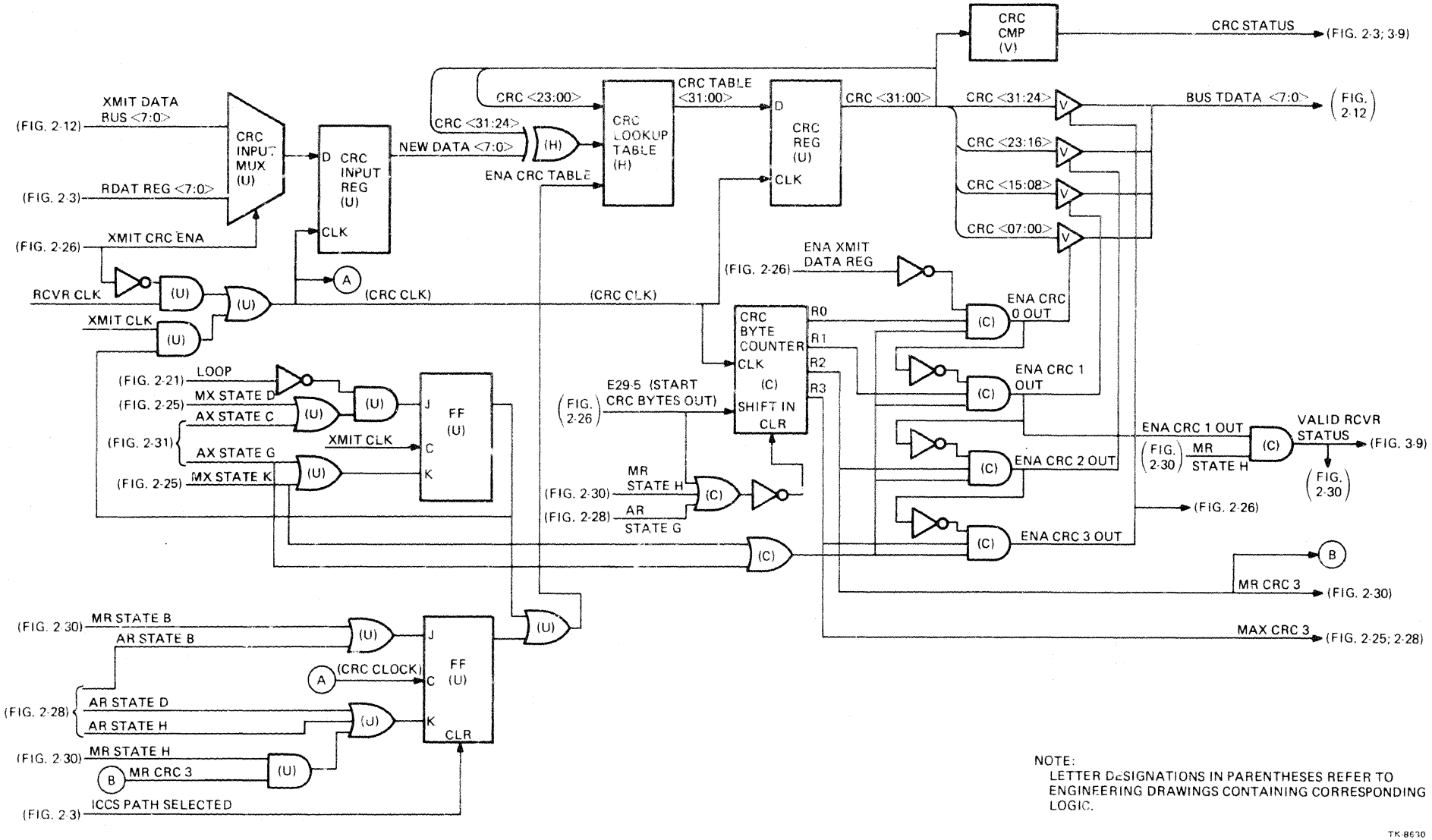


Figure 2-18 CRC Generator/Checker

2.7 ARBITRATION

2.7.1 General

To prevent collisions on the bus, only one node should be transmitting at a time. When the port commands a node to transmit an information packet, the link goes through an arbitration process in order to "gain control" of the bus.* For a node to "gain control" of the bus means that it is the node's turn to have the bus within the arbitration process that is being executed by all the nodes competing for the bus. There is no hardware or software control by which a node may seize the bus and exclude other nodes.

The arbitration process consists of counting down a specific number of "quiet slots" on the bus. A quiet slot is a time period of approximately 800 ns during which there is no activity on the bus. Eight hundred ns is sufficient time for a one-way trip on the bus and to detect a carrier presence. Thus, a quiet slot is a time period allocated for an arbitrating node to detect another node's transmission.

If a node completes its quiet slot countdown (reaches 0), the node wins the bus and may transmit. If the node detects activity on the bus (another node is transmitting) before the countdown is complete, the arbitration process is interrupted and started over once the bus is quiet again. If several nodes are competing for the bus, all but the winner will have their arbitration countdowns interrupted. When the bus goes quiet again, the losing nodes will restart their countdowns simultaneously, thus, placing them in sync with each other. This synchronism occurs only on a busy bus where the competing nodes will sense a "loss of carrier" to synchronize their countdowns.

The arbitration countdown is a round robin dual countdown algorithm such that, if more than one node is trying to transmit, the lower numbered node will be given the bus first. The other nodes, however, can each gain the bus before the lower node can gain the bus again. This is implemented by the number of quiet slots each node must count.

The number of quiet slots to be counted down is determined by the number of the node attempting to transmit and the number of the node that last had the bus. A node may count N + I + 1 slots or I + 1 slots where:

N = 16 (the maximum number of allowable nodes)
I = the node number

When a node starts to arbitrate, it counts N + I + 1 slots. If the countdown is interrupted, the node determines the number of the winning node. If the winning node was a lower number, the node restarts an I + 1 countdown. If the winning node was a higher number, the node restarts an N + I + 1 countdown. Thus, when several nodes are competing for the bus, the lowest number node wins the bus first but must count down the N + I + 1 slots to gain the bus again. The higher nodes will restart their arbitration with the I + 1 countdown and all will win the bus before the first winner can gain the bus again. As each node wins the bus, the N term is added to its countdown value and the next higher numbered node wins the bus. Thus, each competing node will have a turn at the bus, starting with the lowest numbered node and working up to the highest.

The arbitration algorithm is illustrated in Figure 2-19. Note that whenever a node completes its countdown (reaches 0), it checks that the receiver is free (ALT PATH BUSY false) before transmitting. Transmission should not occur from a node unless the node receiver is free to accept the ACK response. Although the node may have completed its countdown and gained one path of the bus, the node receiver could be busy receiving a packet on the other path. When this happens, the transmission is delayed by loading 16 into the node's counter and continuing the countdown.

The 1 term is included in the two countdown expressions because the lowest node number is 0. When node 0 is executing an I + 1 countdown, then it will be looking for 1 slot – not 0 slots.

---

* There is no arbitration process when transmitting an ACK packet as it is assumed the bus has already been acquired for the information and ACK packet transfers.

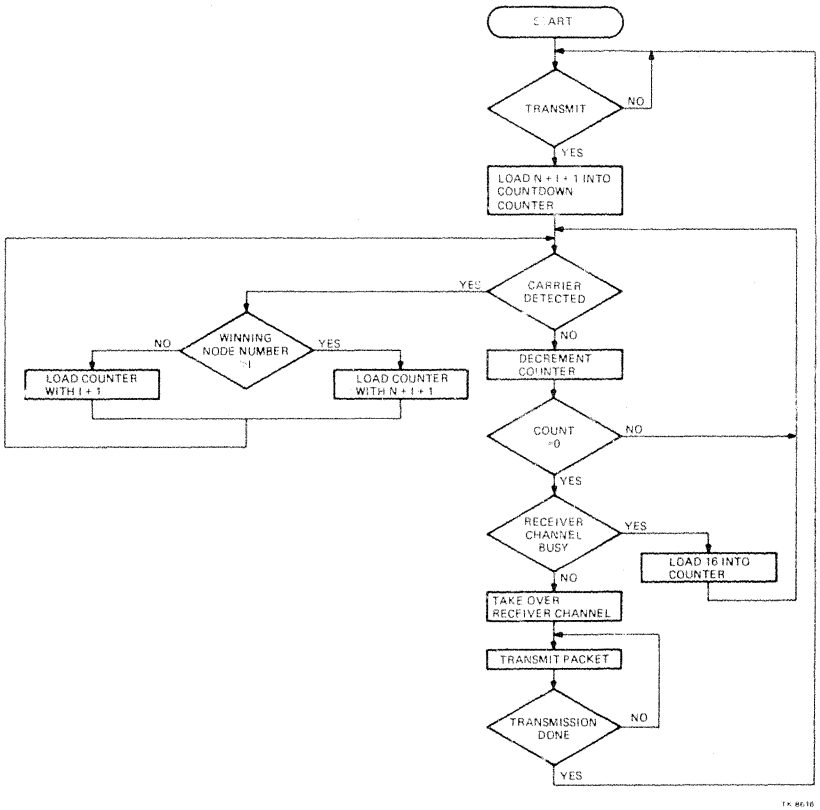


Figure 2-19 Arbitration Flow Diagram

2.7.2 Arbitration Logic

Figure 2-20 is a block diagram of the arbitration logic. Prior to receiving a transmit command from the port, the link is in the idle state (MX state A). In MX state A, the true state of LOAD ARB COUNT loads the arbitrator in preparation for the quiet slot countdown. The basic slot counter is loaded with 1001 (binary) and the down counter is loaded with N + I + 1.

SEE NEXT FRAME

FOR LARGER ART

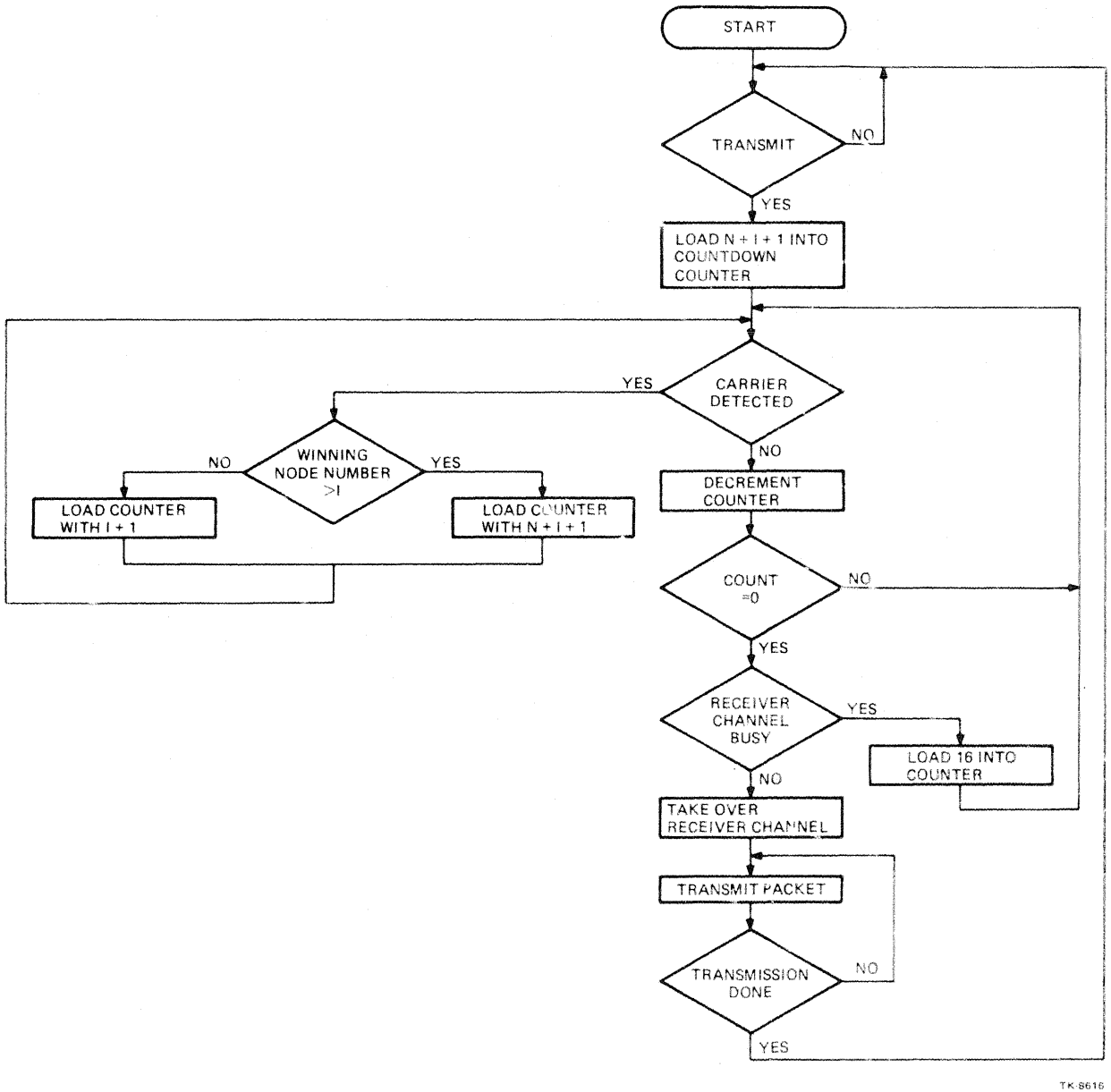


Figure 2-19 Arbitration Flow Diagram

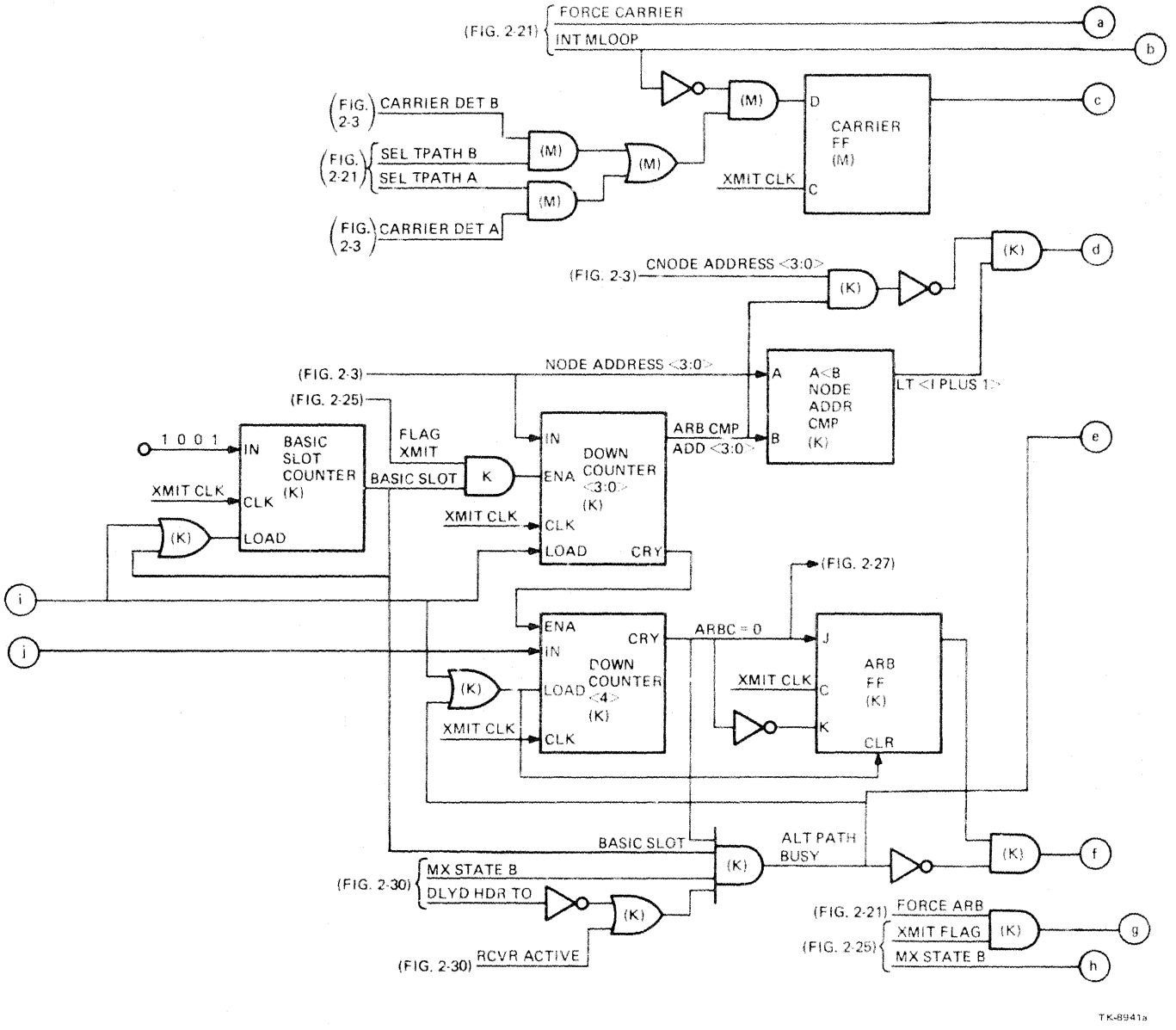
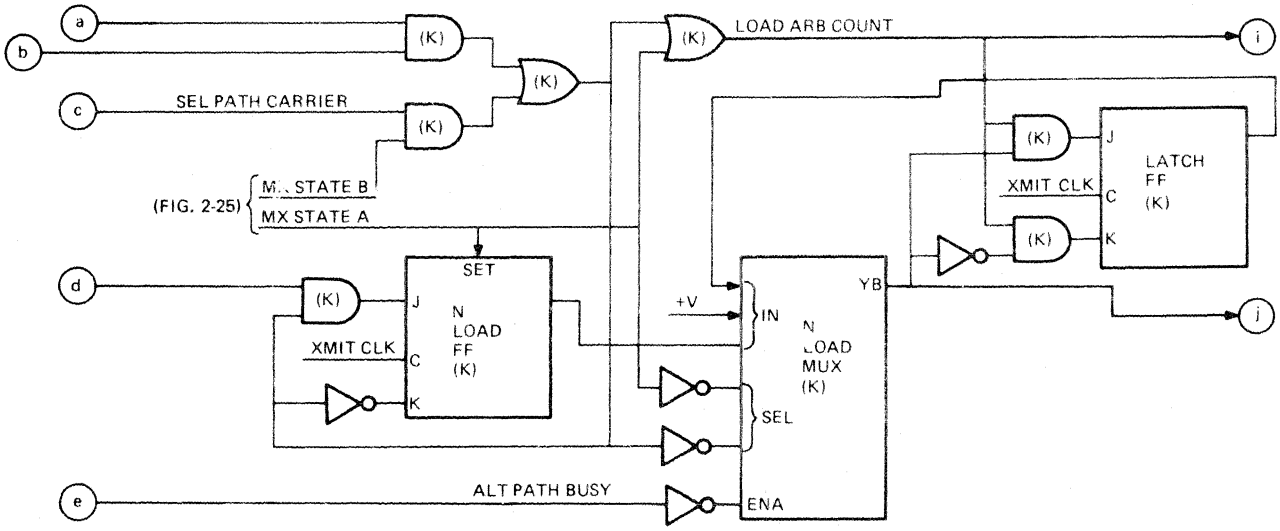
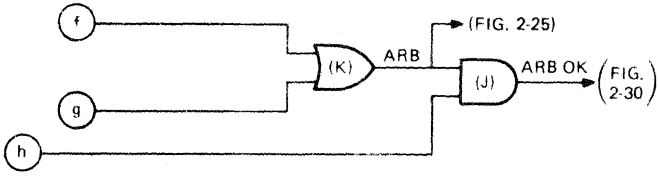


Figure 2-20 Arbitration Block Diagram (a)



NOTE:
LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.



TK-8941

Figure 2-20 Arbitration Block Diagram (b)

The down counter is in two sections: the lower four bits and the fifth bit. The four-bit section is loaded with the node address (NODE ADDRESS ⟨3:0⟩). The fifth-bit section is loaded from an N load mux that supplies the N term in the arbitration countdown expression. The mux select inputs are shown in Table 2-3.

While the link is idling in MX STATE A, the mux selects the +V input to load a 1 into the fifth bit section of the down counter. The 1 represents the N term in the N + I + 1 countdown expression.

Table 2-3 N Load Mux Selection

Select Code		Input Selected
MX STATE A	SEL PATH CARRIER	
True	X	+V
True	X	+V
False	True (load arb.)	N Load FF
False	False	Latch FF

X = don't care

When the link shifts to MX STATE B, LOAD ARB COUNT negates and the arbitrator starts its countdown. The slot counter is clocked from 1001 by XMIT CLK and outputs BASIC SLOT after seven clock pulses. BASIC SLOT is looped back to reload the counter with 1001 and the cycle is repeated. The time period of XMIT CLK is 114.28 ns; hence, BASIC SLOT asserts every 800 ns (7 × 114.28).

Each time BASIC SLOT asserts it enables the four-bit section of the down counter which is decremented by XMIT CLK. When this section of the down counter reaches 0, the next assertion of BASIC SLOT asserts the carry (CRY) output which enables the fifth bit section to decrement. If the fifth bit section contains a 1 (N + I + 1 count), the 1 becomes a 0, the four-bit section becomes all 1's, and the countdown continues. If the fifth bit section contains a 0 (I + 1 count), the CRY output goes true asserting ARBC = 0 (arbitration counter = 0) which conditions the ARB flip-flop to set on the next XMIT CLK. If the alternate bus path is not busy (ALT PATH BUSY false) ARB and ARB OK assert signifying a successful countdown and causing the link to shift to MX state C.

Note that after the counter has reached 0 count, one more assertion of BASIC SLOT is required to assert the CRY output and cause ARB to go true. The additional assertion of BASIC SLOT represents the 1 term in the two countdown expressions.

If a carrier from another node is detected during the arbitration countdown, the arbitrator is reloaded and the countdown starts over. The node address comparator determines whether the interrupting (winning) node is above or below this node in order to determine the new countdown value. (See Paragraph 2.7.1 for a general discussion of the arbitrator.) The comparator compares the node address with ARB CMP ADD (3:0) from the four-bit section of the down counter and asserts, LT (I PLUS 1)* if this node number is less than the winning node number. For example, assume this to be node 5 and the winner to be node 2. ARB CMP ADD (3:0) is down counted to 3, the comparator A input is greater than the B input; therefore, LT (I PLUS 1) is false. This node is not less than the winning node. If the winner were node 7, ARB CMP ADD (3:0) would be 14 (the fifth bit having been decremented), the comparator A input is less than the B input; therefore, LT (I PLUS 1) is true. This node is less than the winning node. The LT (I PLUS 1) signal is used to determine which count down value is to be reloaded into the down counter for the next countdown.

When a carrier is detected (interrupting the countdown), CARRIER DET A or CARRIER DET B asserts. If the carrier is detected on the SEL TPATH selected by the link control PAL, SEL PATH CARRIER is asserted. The assertion of SEL PATH CARRIER causes LOAD ARB COUNT to assert and reload the basic slot counter and both sections of the down counter. The fifth bit section of the down counter is again loaded from the N load mux; however, now the mux is selecting its input from the N load flip-flop (see Table 2-3).

During the countdown, the false state of SEL PATH CARRIER holds the N load flip-flop reset. When SEL PATH CARRIER asserts, it allows the J input to the flip-flop to look at LT (I PLUS 1) from the node comparator. If LT (I PLUS 1) is true (this node is less than the winning node), the flip-flop is set and a 1 is loaded into the fifth bit section. If LT (I PLUS 1) is false (this node is higher than the winning node), the flip-flop remains reset and a 0 is loaded into the fifth bit section.

The output from the N load mux is latched up in a latch flip-flop. When SEL PATH CARRIER negates, the N load mux selects the output of the latch flip-flop thus maintaining the fifth bit selection after SEL PATH CARRIER negates.

As described in Paragraph 2.7.1, a round robin arbitration algorithm is used in which the lowest-numbered node wins the bus first, then the next higher, and so forth in a continuous loop. For the loop to be continuous, node 0 must follow node 15 in the same way that any node follows the node preceding it. When node X is beaten by the preceding node (X-1), it restarts its countdown as I + 1. Node X is not less than the winner, therefore, LT (I PLUS 1) is false and the fifth bit section of the counter is loaded with a 0. Likewise, when node 0 is beaten by node 15 it must appear that it was beaten by a lower node and restart its countdown as I + 1; however, in this case, LT (I PLUS 1) is true. Logic has been added to the input of the N load flip-flop to force a 0 into the fifth bit section of the counter when node 0 is beaten by node 15. Thus, when this is node 0 (CNODE ADDRESS (3:0) = all 1's) and it has just been beaten by node 15 (ARB CMP ADD (3:0) = all 1's), the AND gate transferring LT (I + 1) into the N load flip-flop is inhibited and the flip-flop remains reset. Hence, a 0 is reloaded into the fifth bit section of the down counter and node 0 does an I + 1 countdown.

If the link receive channel is busy on the alternate bus path, RCVR ACTIVE will be true, causing ALT PATH BUSY to also be true. This condition inhibits the assertion of ARB and loads 16 into the down counter. ALT PATH BUSY loads only the fifth bit section of the down counter. The four-bit section remains enabled in count mode. ALT PATH BUSY generates the 16 by disabling the N load mux causing it to output a 0 into the fifth bit section. The four-bit section is at all 0's (countdown successfully completed), hence, as the fifth bit section is loaded with a 0, the four-bit section is decremented to all 1's. Thus, when the entire counter is enabled again, it contains a count of 16.

* LT = less than

The true state of RCVR ACTIVE inhibits a successful arbitration by reasserting ALT PATH BUSY. RCVR ACTIVE negates after the message on the alternate path has been received. The transmission that is arbitrating for the bus, however, still cannot be allowed because the transmit channel must be used to transmit an ACK response. This point in the message receive state sequence is state I. Hence, MR STATE I is used to keep ALT PATH BUSY true to inhibit the assertion of ARB.

The false state of DLYD HDR TO also asserts ALT PATH BUSY and inhibits a successful arbitration. DLYD HDR TO is false if a transmission is occurring from this node (A DRIVER ENA or B DRIVER ENA true) as shown in Figure 2-30. The transmission in this case would be the transmission of an ACK packet on the alternate path.

2.8 LINK FUNCTIONS

Link functions (Figure 2-21) are commanded from the port via four link control lines (LINK CONTROL ⟨3:0⟩) and eight port data lines (PORT DATA ⟨7:0⟩). The port asserts SELECT when a valid function exists on the link control lines.

A function decoder decodes the link control lines and outputs the specific function commanded by the port. The function commands are described below:

A. XMIT FCN – This function initiates arbitration and transmission on one of the CI paths. The CI path used is selected by port data bit 7 (0 = path A; 1 = path B).

B. RESET XMIT STATUS – This function resets transmission status bits at the end of a transmission operation.

C. ABORT XMIT FCN – This function aborts a currently active transmit operation.

The link mode control, PAL, receives the link control lines and the port data lines from the port. The port data lines carry control information relating to the commanded function, and specify various maintenance functions for the link.

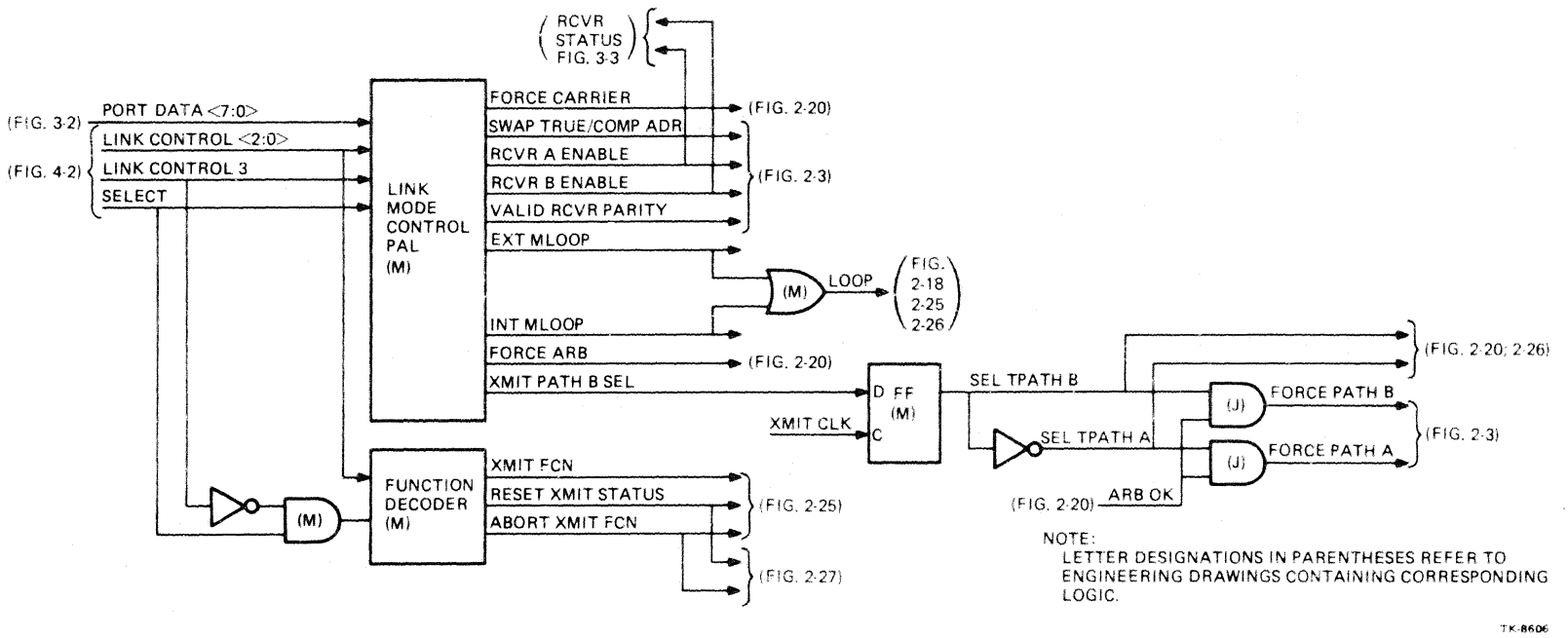


Figure 2-21 Link Functions

The control information and maintenance functions are described below:

A. XMIT PATH B SEL – This signal selects the CI path associated with the XMIT FCN command.

B. RCVR A ENABLE – This signal enables path A in the link receiver making the node accessible on CI path A.

C. RCVR B ENABLE – This signal enables path B in the link receiver making the node accessible on CI path B.

D. EXT MLOOP – This is a maintenance function that allows the link to receive its own transmission by looping on the selected CI path.

E. INT MLOOP – This is a maintenance function that allows the link to receive its own transmission by looping inside the transmit drivers and input receiver detectors. This operation will not interfere with the CI operation of other nodes.

F. FORCE CARRIER – This is a maintenance function that causes the link to see a detected carrier.

G. FORCE ARB – This is a maintenance function that causes the link to force a successful arbitration.

H. VALID RCVR PARITY – This is a maintenance function that is used to generate parity errors in the receive channel.

I. SWAP TRUE/COMP ADR – This is a maintenance function that causes the true and complementary address sources to be swapped resulting in an address mismatch.

The transmission path select signal (SEL PATH A or SEL PATH B) asserts a corresponding FORCE PATH signal after the node has successfully arbitrated for the bus (ARB OK true). The FORCE PATH signal enables the corresponding path in the receive channel in preparation to receive the ACK response.

2.9 LINK INTERFACE SIGNALS

Figure 2-22 illustrates the link interface signals. Most of the link interfacing is with the PB. Figures 2-23 and 2-24 are flow diagrams of a typical transmit and receive operation. The flow diagrams highlight the interface signals to illustrate their basic functions. Some other major signals, internal to the link, are included for completeness. The two flow diagrams utilize most of the interface signals and explain their basic functions. Interface signals not included in the flow diagrams are the three clocks (PORT CLK, XMIT CLK, RCVR CLK), the node address (NODE ADDRESS (7:0)), and INITIALIZE.

PORT CLK is received from the PB while XMIT CLK and RCVR CLK are generated within the link. All three clocks are used in both the PB and the link.

NODE ADDRESS (7:0) is sent to the port (via the PB) and inserted into the transmitted packet as the source byte.

INITIALIZE is used for system initialization.

The flow diagrams illustrate a typical error-free sequence of a transmit and a receive operation. They can be used in conjunction with the receive channel block diagram (Figure 2-3) and the transmit channel block diagram (Figure 2-12), or with the more detailed state diagrams discussed in Paragraph 2.10.

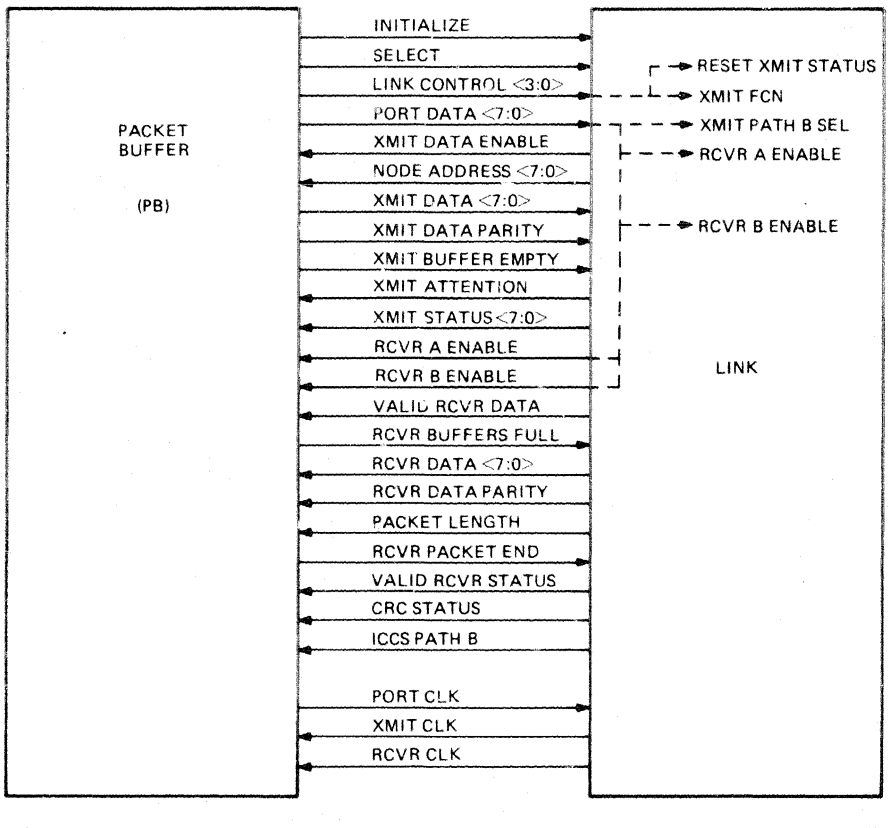


Figure 2-22 Link Interface Signals

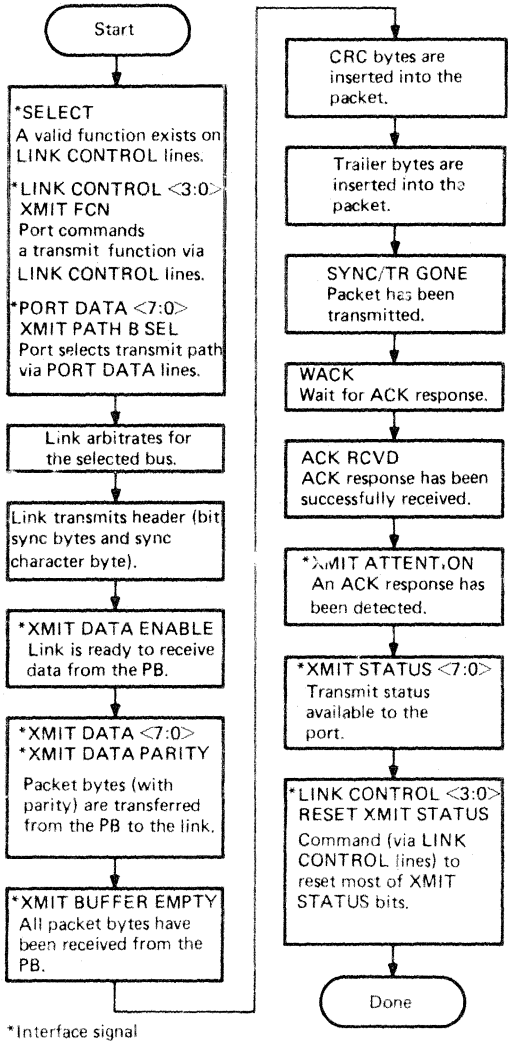


Figure 2-23 Interface Flow Diagram – Transmit Operation

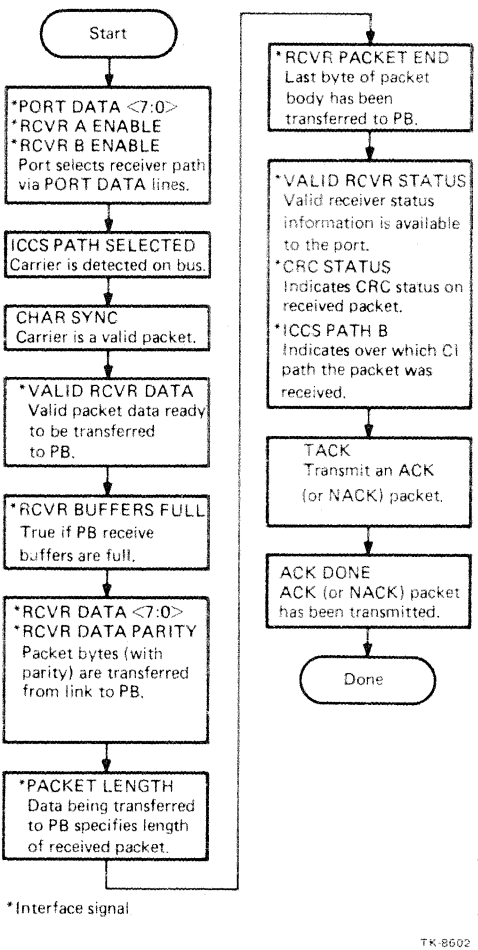


Figure 2-24 Interface Flow Diagram – Receive Operation

2.10 OPERATING STATES

The following description of the four link operations utilizes the state diagrams contained in the engineering drawing set. The various states are shown in the diagrams as circles. A path looping back into a circle holds the link in that state so long as the signal condition shown in the loopback path is true. The link goes to its next state if the signal's condition shown in the connecting path to the next state is true. Where no loopback paths are shown, the link stays in that state for one clock pulse to perform the indicated task(s) and then advances to the next state.

Also included in the drawing set is a XMIT/RCVR MSG State Flow Diagram. The diagram shows the normal state flows for a message transmission and ACK reception operation and for a message reception and ACK transmission operation. The diagram illustrates what PALs are used and how the sequence shifts from one PAL to another as the operation is executed. The diagram illustrates a basic point in link operations; that an ACK receive sequence is a part of the message transmit sequence in that the message transmit sequence is not complete until the ACK receive sequence is done. Likewise, the ACK transmit sequence is part of the message receive sequence and that the message receive sequence is not complete until the ACK transmit sequence is done.

2.10.1 Message Transmit

Figure 2-25 illustrates the message transmit state logic and is used in conjunction with the MESSAGE XMIT STATE diagram in the engineering drawing set. Two PALs are used for the message XMIT state sequence.

INITIALIZE from the port asserts TINIT which initializes the link and asserts MX STATE A from PAL no. 1. MX State A is the transmit idle state. When the port commands a transmit function, XMIT FCN asserts from the link control PAL causing TXMIT to assert and transfer the link to state B.

The link arbitrates for the bus in state B. When the arbitration is successful, ARB asserts and the link transfers to state C.

In state C the link transmits the bit synchronization bytes and the sync character byte. After the sync character byte has been transmitted, SYNC/TR GONE asserts and sends the link to state D.

In state D the CRC generator is enabled (except for maintenance loop operations), the second MSG XMIT State PAL is enabled, and the link goes to state E.

PAL no. 1 stays in state E for the rest of the transmission long as there is no parity error. If a parity error occurs, PE asserts and transfers the link to state F.

If the link is placed in state F, PAL no. 2 is reset and XMIT ATTENTION is asserted to the port which will then abort the transmission. The link then returns to state A.

PAL no. 2 moved from its idle state (state G) to state H when PAL no.1 asserted MX STATE D.

From state H the link goes to state I where the destination byte is clocked into the destination address register.

The link then transfers to state J where it waits for the body of the packet to be transmitted. When the last byte of the body is transmitted, XMIT BUFFER EMPTY is received from the PB and transfers the link to state K [if this is not a maintenance operation; if this is a maintenance operation (LOOP true), the link goes directly to state L].

In state K the CRC bytes are transmitted. MAX CRC 3 asserts when the last CRC byte is transmitted. MAX CRC 3 causes the link to transfer to state L.

In state L the packet trailer bytes are transmitted. After the trailer bytes are transmitted, SYNC/TR GONE asserts and transfers the link to state M.

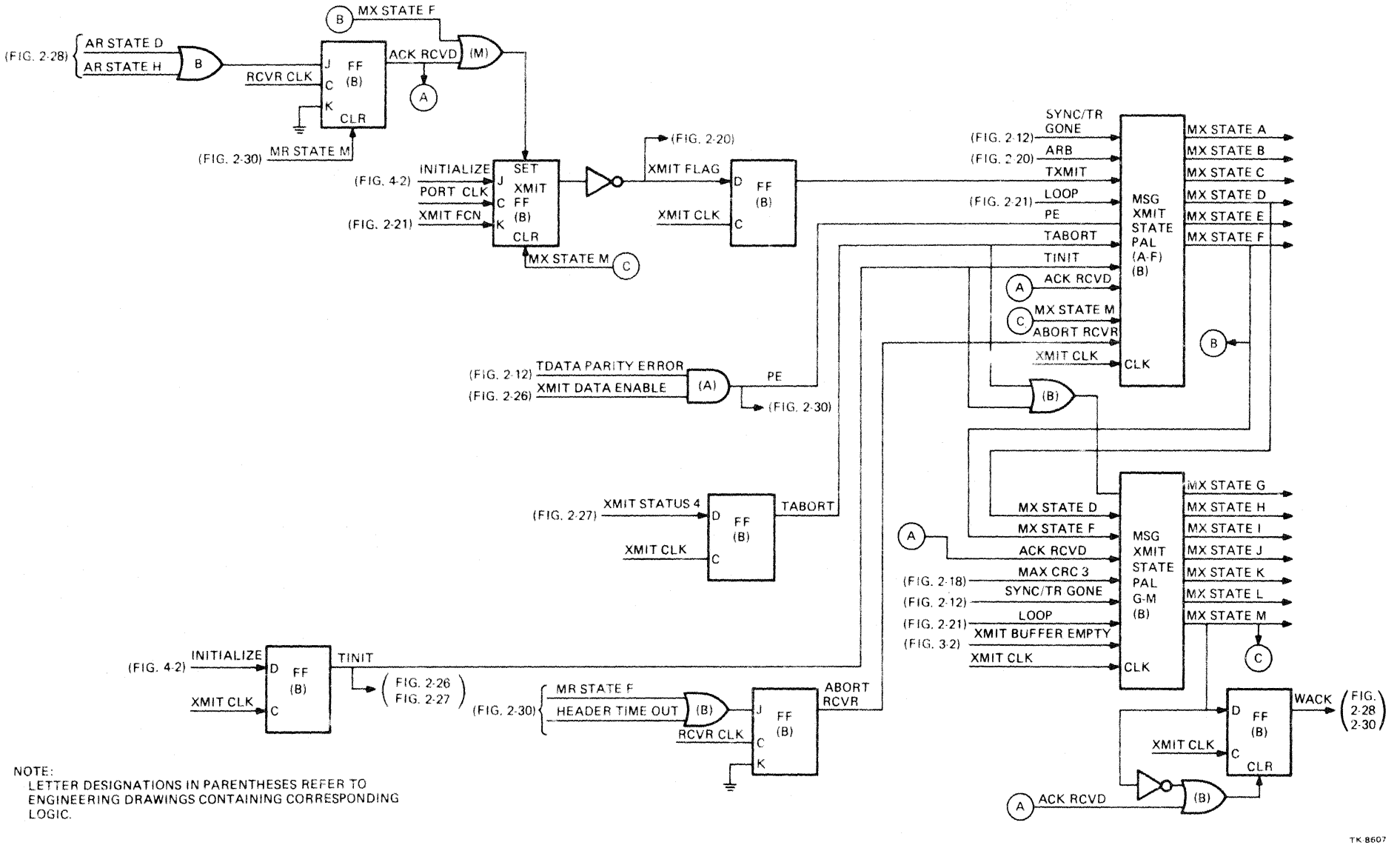


Figure 2-25 Message Transmit State Logic

In state M the link has completed its transmission and is waiting for the ACK receive sequence to complete. The end of the ACK receive sequence is indicated by the assertion of AR STATE D or AR STATE H from the ACK receive state logic. Either of these signals asserts ACK RCVD to both PALs causing them to return to their idle states. ACK RCVD also negates TXMIT to complete the message XMIT sequence.

When the link enters state M, WACK (wait for ACK) is asserted to the ACK receive state logic enabling the ACK RCVR PAL to start the ACK receive sequence. When the ACK response is received, ACK RCVD asserts and negates WACK.

The port can abort the transmission by asserting ABORT XMIT FCN via the LINK CONTROL lines. ABORT XMIT FCN asserts XMIT STATUS 4 and then TABORT via two flip-flops. TABORT is applied to both message XMIT PALs resetting them to their idle states.

The MSG XMIT sequence is also reset by HEADER TIME OUT which asserts ABORT RCVR to PAL no. 1. HEADER TIME OUT is asserted by the MSG RCVR state logic when a carrier is detected but sync character recognition does not occur.

2.10.1.1 Transmit Contr' Logic – Figure 2-26 illustrates the logic that controls the flow of data through the transmit channel shown in Figure 2-12. The control signals are regulated by the state signals generated by the XMIT state PALs. The assertion and negation of the control signals can be related to the task(s) performed in the various states as shown in the XMIT state diagrams.

ENA SYNC/TR CNT is asserted by the appropriate STATE signals and enables the sync/trailer counter to start counting.

ENA SYNC/TR gates the bit sync bytes and the trailer bytes onto the XMIT DATA bus. ENA SYNC/TR is negated by ENA XMIT DATA LATCH which gates the packet bytes from the PB onto the XMIT DATA bus. ENA XMIT DATA LATCH also asserts ENA XMIT DATA PARITY.

ENA XMIT DATA REG isolates the BUS TDATA bus from the XMIT DATA bus while the CRC bytes are being placed onto the TDATA bus. TINIT initially asserts ENA XMIT DATA REG which passes the packet bytes onto the BUS TDATA bus until XMIT BUFFER EMPTY is received from the PB. XMIT BUFFER EMPTY negates ENA XMIT DATA REG which remains negated until all the CRC bytes are placed onto the BUS TDATA b us. When this occurs, MX STATE L asserts thereby re-asserting ENA XMIT DATA REG for the trailer bytes. MX STATE L also asserts SEL TRAILER to gate the trailer bytes out of the sync/trailer PROM onto the XMIT DATA bus.

A DRIVER ENA and B DRIVER ENA enable the drivers that output the transmitted packet onto the selected CI path. During a message XMIT operation, the selected DRIVER ENA signal is asserted by the SEL TPATH signal selected by the port via the LINK CONTROL lines, and by MX STATE C. The DRIVER ENA signal is negated during MX state L when SYNC/TR GONE asserts. During an ACK XMIT operation, the selected DRIVER ENA signal is asserted by AX STATE B and LAST RCVR = B. LAST RCVR = B is true if the last message was received on CI path B (ICCS PATH B true). In this case, B DRIVER ENA asserts to transmit the ACK over the same path on which the message was received. Conversely, if the message was received on CI path A, A DRIVER ENA would assert, transmitting the ACK over CI path A. The DRIVER ENA signal is negated during AX state H when SYNC/TR GONE asserts.

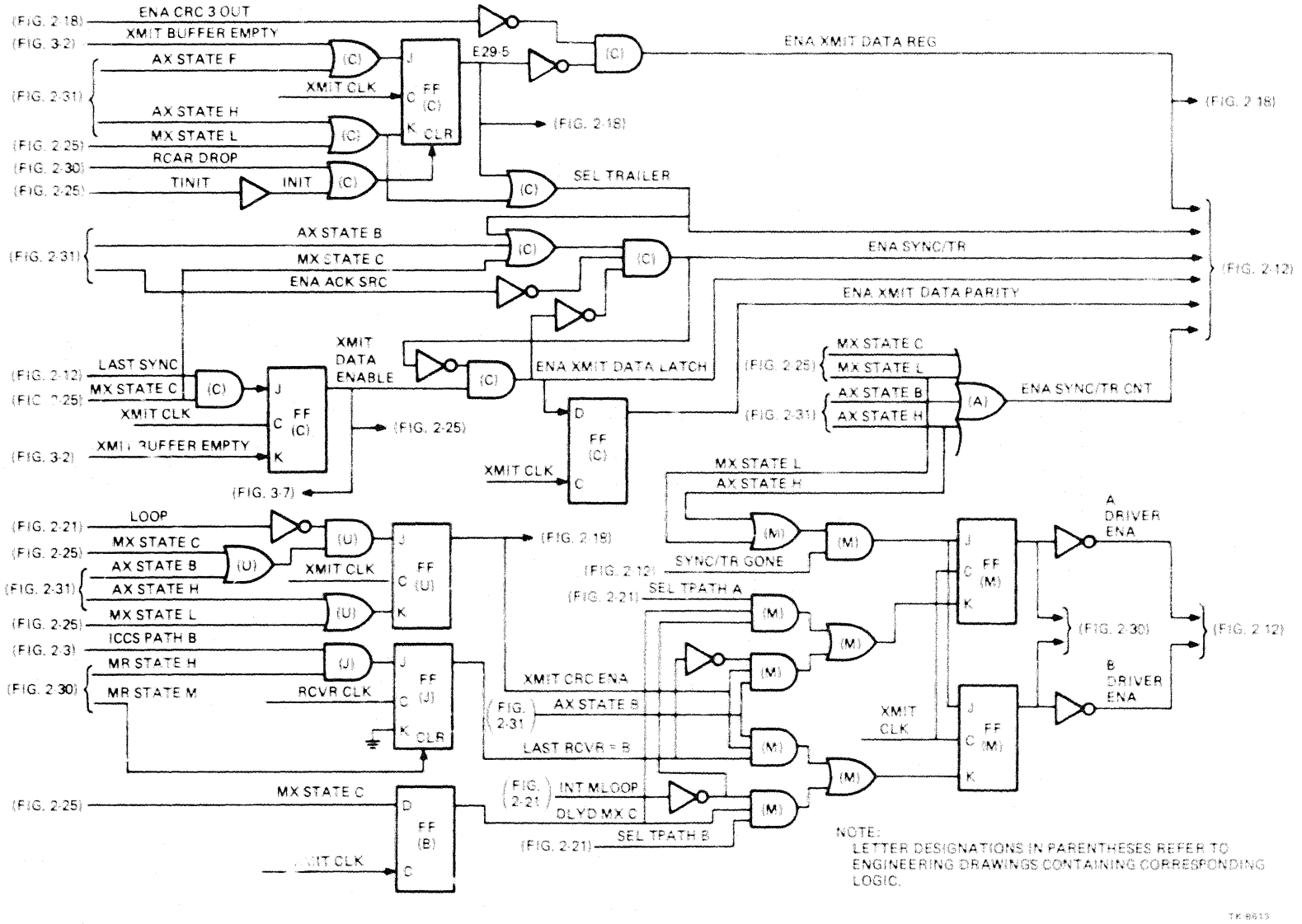


Figure 2-26 Transmit Control Logic

2.10.1.2 Transmit Status – Eight transmit status bits (XMIT STATUS (7:0)) are used to indicate the status of a transmit operation. (See Figure 2-27.) The bits are available to the port along with XMIT ATTENTION.

XMIT ATTENTION is asserted when a response is received from the destination node (ACK or NACK), when no response is received from the destination node (ACK packet timeout occurs), when a transmit parity error occurs, or when an abort transmission command is issued (ABORT XMIT FCN is asserted).

The XMIT STATUS bits are asserted as described below:

XMIT STATUS 7 – This bit is set if a parity error is detected on the data on the BUS TDATA bus in the transmit channel during a transmission. A parity error will cause XMIT ATTENTION to be asserted to the port which will then abort the transmission.

XMIT STATUS 6 – When set, this bit indicates the presence of a carrier on CI bus A.

XMIT STATUS 5 – When set, this bit indicates the presence of a carrier on CI bus B.

XMIT STATUS 4 – This bit is set when a transmission is aborted by the abort function (ABORT XMIT FCN) commanded by the port via the LINK CONTROL lines.

XMIT STATUS 3 – This bit is set when an arbitration countdown has reached 0. It does not necessarily mean that a transmission will occur (see Paragraph 2.7).

XMIT STATUS 2 – This bit is set when a NACK is received from the destination node. A NACK response causes XMIT ATTENTION to assert to the port.

XMIT STATUS 1 – This bit is set when an ACK is received from the destination node. An ACK response causes XMIT ATTENTION to assert to the port.

XMIT STATUS 0 – This bit is set when a transmit operation is in progress or whenever XMIT ATTENTION is asserted.

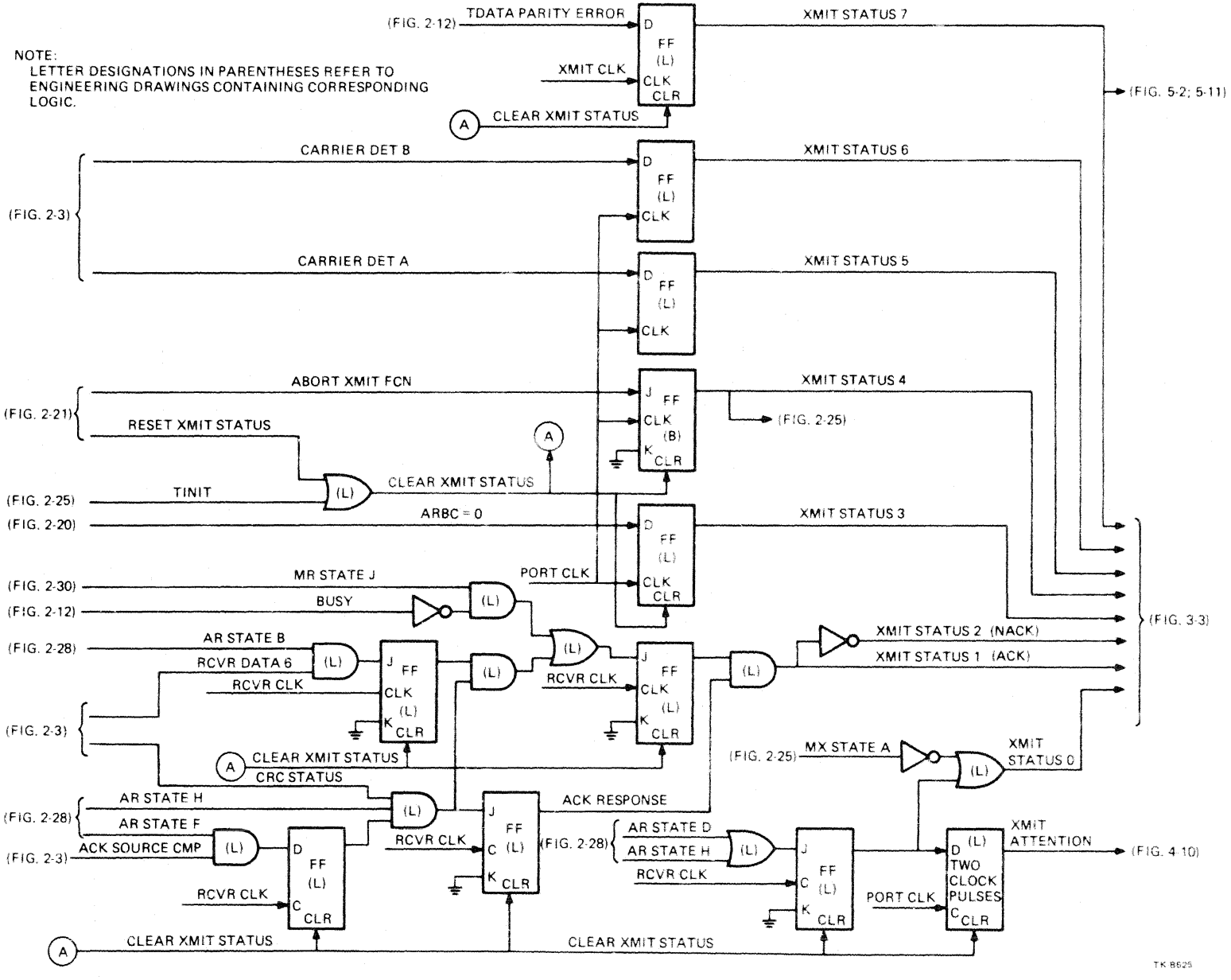


Figure 2-27 Transmit Status

2.10.2 ACK Receive

Figure 2-28 illustrates the ACK receive state logic and is used in conjunction with the the ACK RCVR STATE diagram in the engineering drawing set. Two PALs are used for the ACK RCVR state sequence.

2.10.2.1 ACK Receive PAL States – INITIALIZE from the port initializes the receive channel and asserts RINIT to both ACK RCVR PALs placing them into their idle states (state A for PAL no. 1; state E for PAL no. 2). The link is transferred to AR state B when PAL no. 1 senses that a valid packet is being received (CHAR SYNC true), that the receiver is waiting for an ACK response (WACK true), and that the packet is an ACK (RDAT REG 7 = 1) rather than a message packet.

In state B the packet true destination byte is checked. If a match is obtained (DST CMP true), the link transfers to state C.

In state C, ACK RCVR state PAL no. 2 is enabled (AR STATE E asserts) and the complement destination byte is checked. If a destination match is obtained (DST CMP true) PAL no. 2 moves to state F. PAL no. 1 remains in state C until the ACK RCVR state sequence is completed.

State D of PAL no. 1 is a "receiver clear" state which is entered if an improper response is obtained in states A, B, or C. State D is entered from state A if CHAR SYNC and WACK are true but RDAT REG 7 = 0 (this is a message packet, not an ACK response). State D is entered from state B if a true destination mismatch occurred. State D is entered from state C if a complementary destination mismatch occurred. After clearing the various receiver functions, PAL no. 1 returns to the idle state (state A).

In state F the packet source byte is checked. The link then passes to state G provided this is not a maintenance operation (INT MLOOP false). If this is a maintenance operation (INT MLOOP true), the link goes to state H.

In state G the CRC bytes are input to the CRC checker which checks for any CRC error. When MAX CRC 3 asserts (last CRC byte into the CRC checker) the link moves to state H.

State H is the last state in the ACK RCVR sequence. In this state the various receive functions are cleared and then both PALs are returned to their idle states.

The last state in a MESSAGE XMIT state sequence is state M. When MX STATE M asserts, an ACK timeout counter is enabled and starts counting. If, after 3.66 μs, the ACK RCVR sequence is not completed, the counter asserts ACK TO which terminates the sequence and returns both ACK RCVR PALs to their idle states. The port then reads status bits to determine the trouble.

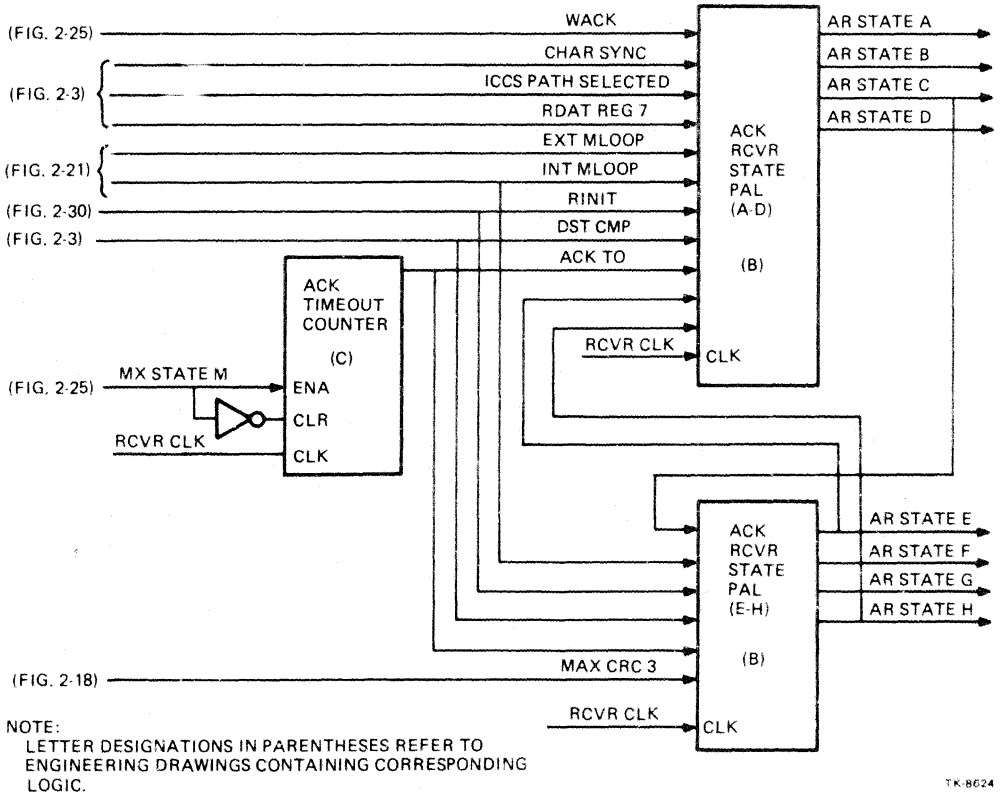


Figure 2-28 ACK Receive State Logic

2.10.2.2 Sync Character Detect Enable PAL – The purpose of the sync character detect enable PAL (Figure 2-3) is to enable the sync character detector when a packet is expected and to inhibit the detector when transmitting from this node. The sync character detector should be enabled during the following times:

A. During an internal maintenance loop operation

B. After a transmission when an ACK packet is expected

C. To receive a message packet from another node taking care not to respond to transmissions from this node (transmission of an ACK packet).

Figure 2-29 functionally illustrates the sync character detect enable PAL. ENA SYNC DET is asserted by any of five signals applied to an output OR gate.

When in maintenance loop operation, INT MLOOP is true and enables the sync detector.

The next two signals enable the sync detector when an ACK packet is received. One is generated by ANDing CARRIER DET A with the negated state of ICCS PATH B while the other is generated by ANDing CARRIER DET B and the asserted state of ICCS PATH B. Thus, the two gates look for a carrier presence in both CI paths. Enabling of the two gates is restricted to ACK packets by ACK ENA which asserts while waiting for an ACK packet (WACK true) and after a loss of carrier has been sensed. The carrier lost would be the message transmit carrier from this node. ICCS PATH B (true or false) enables one of the AND gates in the ACK ENA logic. When that gate senses a loss of carrier (CARRIER DET negates), ACK ENA asserts and is latched. The next time a carrier is sensed (the ACK response), the output AND gate is enabled and asserts ENA SYNC DET via the output OR gate.

The last two signals enable the sync detector when a message packet is received. The signals are generated by AND gates which are enabled when the node is not transmitting a message packet (both FORCE PATH signals false), and a carrier is detected on one of the CI paths. The gates are inhibited by trailer delay (TR DLY) which is true at the end of an ACK transmission when the packet trailer is being transmitted.

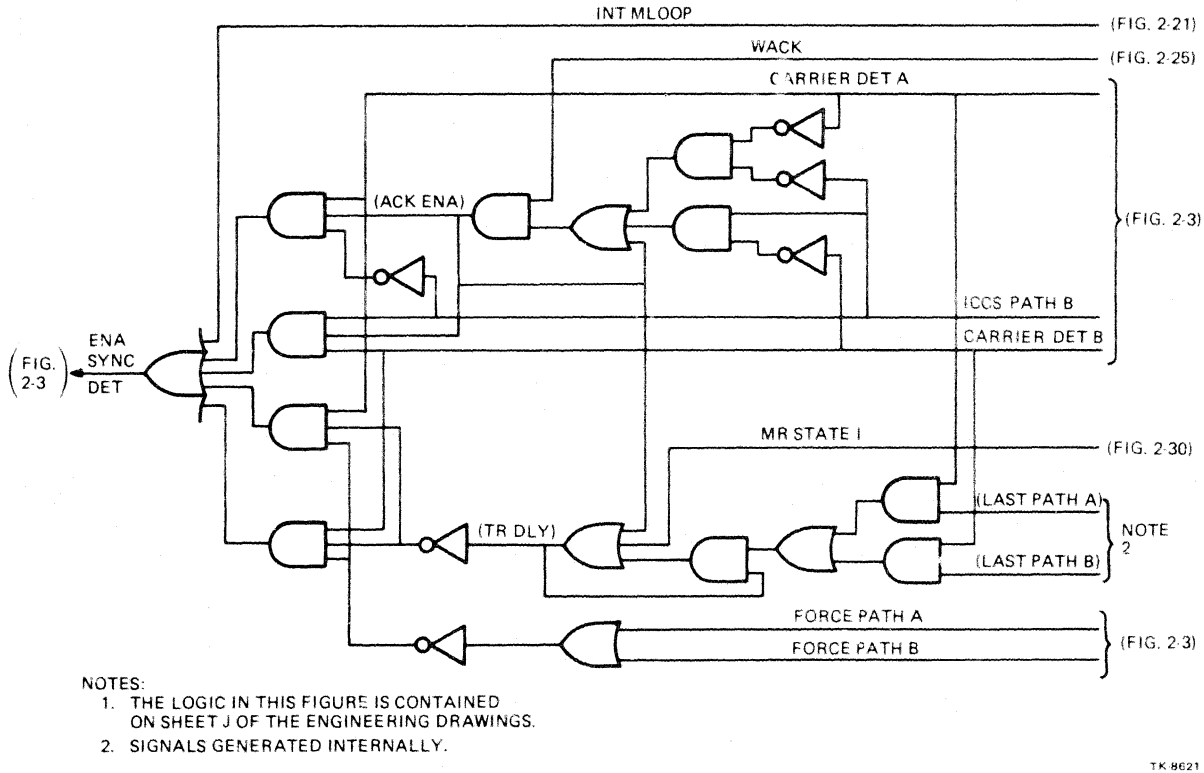


Figure 2-29 Sync Character Detect Enable PAL

2.10.3 Message Receive

Figure 2-30 illustrates the message receive state logic. It is used in conjunction with the MSG RCVR STATE diagram in the engineering drawing set. Two PALs are used for the message RVCR state sequence.

INITIALIZE from the port asserts ABORT + INIT FCN which in turn asserts RINIT. RINIT initializes the logic in the receive channel and places the two MSG RCVR state PALs into their idle states (state A for PAL no.1; state M for PAL no. 2). When the receiver is not disabled due to transmission from the transmit channel (RXMIT false), a valid packet is in the receive channel (CHAR SYNC true), and the packet is recognized as a message (RDAT REG 7 = 0) and not an ACK; PAL no. 1 transfers to MR state B.

In MR state B, VALID RCVR DATA is asserted to the PB indicating that a valid packet is being received, and PACKET LENGTH is asserted to the PB indicating that the byte being transferred contains packet length information. Also, the CRC checker is enabled and starts receiving the packet bytes. The link moves to MR state C on the next clock pulse.

In MR state C the true destination byte is checked. If a match is obtained (DST CMP true), the link moves to state D. PACKET LENGTH remains asserted in state C as the byte being transferred to the PB contains packet length information.

In MR state D the complement destination byte is checked. If a match is obtained (DST CMP true), the link moves to MR state E.

In MR state E the packet source byte is clocked into the true and complement ACK destination registers to serve as the destination for the ACK response. The next RCVR CLK pulse moves the link to MR state G.

PAL no. 1 remains in MR state G for the rest of the MSG RCVR state sequence. The assertion of MR STATE G enables PAL no. 2 in that it allows it to move from its idle state (state M) to state H when its condition signal (RCVR PACKET END) is asserted.

If any of the three condition tests made by PAL no. 1 fails, the link is transferred to MR state F. Failing any of the three tests would be:

1. While in state A with RXMIT false, CHAR SYNC asserts but RDAT REG = 1 (this is an ACK packet)

2. A true destination mismatch occurred in state C

3. A complement destination mismatch occurred in state D

In MR state F the receive logic is cleared, and PAL no. 1 returns to the idle state (state A) on the next RCVR CLK pulse.

PAL no. 2 remains in its idle state (state M) while the packet body is being transferred to the PB. After the last byte of the body has been sent to the PB, the PB asserts RCVR PACKET END and PAL no. 2 goes to MR state H.

In state H the packet CRC bytes are input to the CRC checker. When the last byte is in the checker, MR CRC 3 asserts. If there is no CRC error, CRC OK is true when MR CRC 3 asserts. In this case, the link moves to state I. If there is a CRC error, CRC OK is false and the link goes to state L.

In state L the MSG RCVR state sequence is aborted. The receive channel is cleared, PAL no. 1 is moved to its idle state (state A), and PAL no. 2 moves to its idle state (state M).

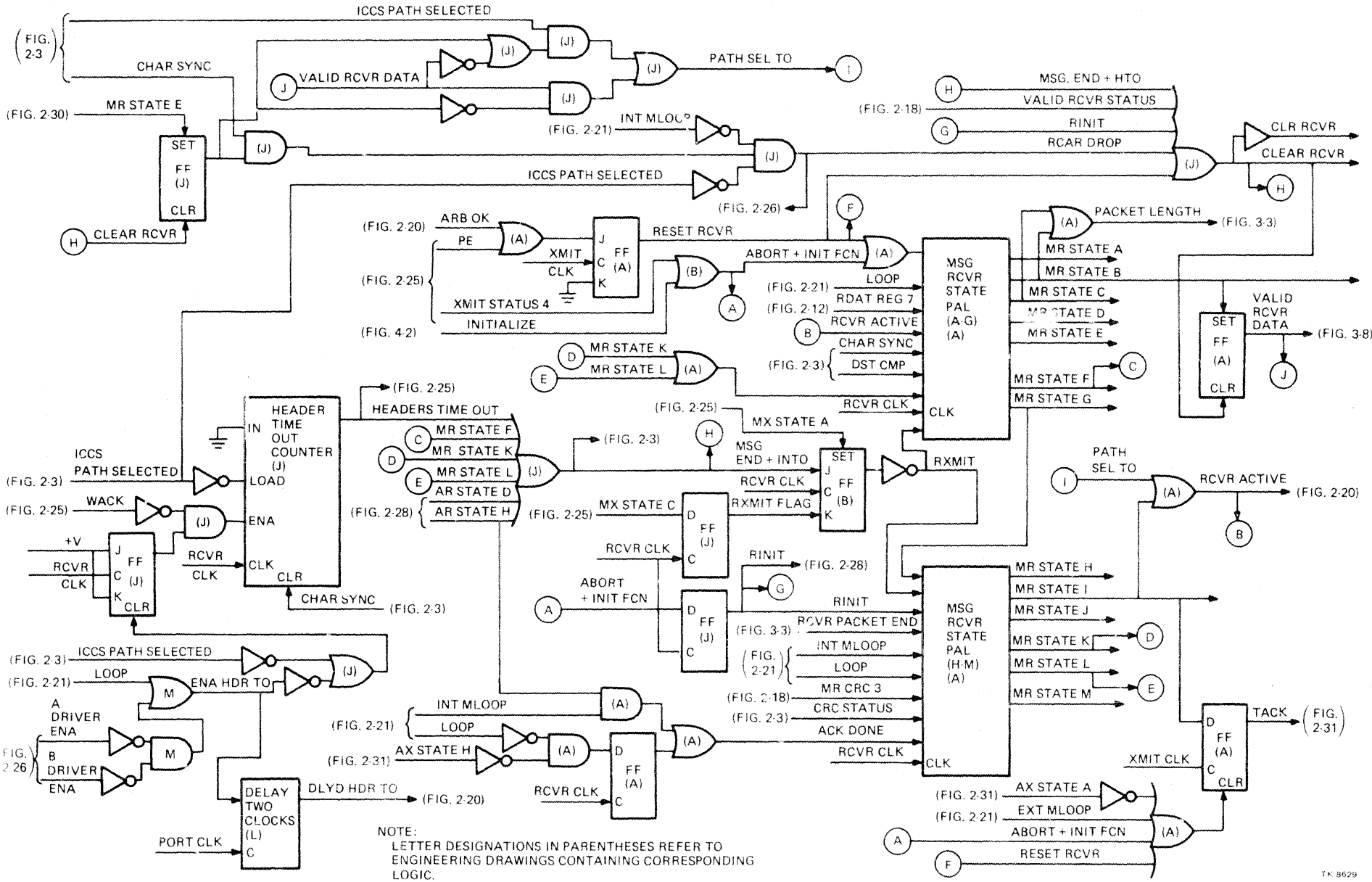


Figure 2-30 Message Receive State Logic

The message receive state sequence remains in state I while the link transmits the ACK response. The assertion of MR STATE I asserts TACK (transmit ACK) to the ACK transmit state PAL initiating the ACK transmit sequence. When the ACK transmission is done AX STATE H negates to assert ACK DONE to MSG RCVR PAL no. 2. The assertion of ACK DONE moves the link to MR state K.

In MR state K the receive channel is cleared and PAL no. 1 is returned to its idle state (MR state A). The next RCVR CLK pulse return PAL no. 2 to its idle state (state M).

The message receive state logic contains a header timeout counter to prevent receive channel hangups. The counter is turned on by ICCS PATH SELECTED (removes the counter LOAD signal) and cleared by CHAR SYNC. It thus starts counting when a carrier is detected and is cleared when the carrier is recognized as being a valid packet. If SYNC CHAR fails to assert, the counter times out (in 3.66 μs) and outputs HEADER TIME OUT. The assertion of HEADER TIME OUT causes MSG END + HTO to assert, thereby asserting CLEAR RCVR to reset the receive logic.

The header timeout counter is enabled and disabled at the RCVR CLK rate via a flip-flop. Thus, the four-bit counter is extended to five bits, producing the 3.66 μs timeout period (32×114.28 ns = 3.66 μs). Note that the counter is disabled by WACK. WACK asserts in MX state M when the transmit channel is transmitting a message packet. Thus, WACK prevents the detection of the transmitted carrier from starting the header timeout period.

Other signals besides MSG END + HTO assert CLEAR RCVR. One of these is RCAR DROP (receive carrier dropped) which asserts if a carrier is lost during a message reception. ICCS PATH SELECTED asserts before CHAR SYNC asserts and negates after CHAR SYNC negates. If a receive carrier is prematurely lost, ICCS PATH SELECTED will negate while CHAR SYNC is still true, causing RCAR DROP to assert. Note that CHAR SYNC is not applied to the ANDing operation until MR STATE E sets a flip-flop which gates CHAR SYNC to the RCAR DROP AND gate. Delaying CHAR SYNC until MR state E allows the header portion of the packet to pass before the node looks for carrier drop-out.

2.10.4 ACK Transmit

Figure 2-31 illustrates the ACK transmit state logic and is used in conjunction with the ACK XMIT STATE diagram in the engineering drawing set.

INITIALIZE from the port asserts TINIT which initializes the link and asserts AX STATE A from the ACK XMIT PAL. AX state A is the ACK transmit idle state. When TACK (transmit ACK) is received from the MSG RCVR state PAL, the link goes into AX state B.

In state B the sync/trailer PROM logic is enabled and outputs the bit synchronization bytes and the sync character byte onto the XMIT DATA BUS. The selected transmit driver is also enabled. When SYNC/TR GONE asserts, the link transfers to state C.

The link is in AX state C for one clock pulse. While in state C, the ACK type byte is placed onto the XMIT DATA BUS and the CRC generator is enabled. The next XMIT CLK pulse moves the link to AX state D.

In AX state D the ACK true destination byte is placed onto the XMIT DATA BUS. The link then advances to AX state E.

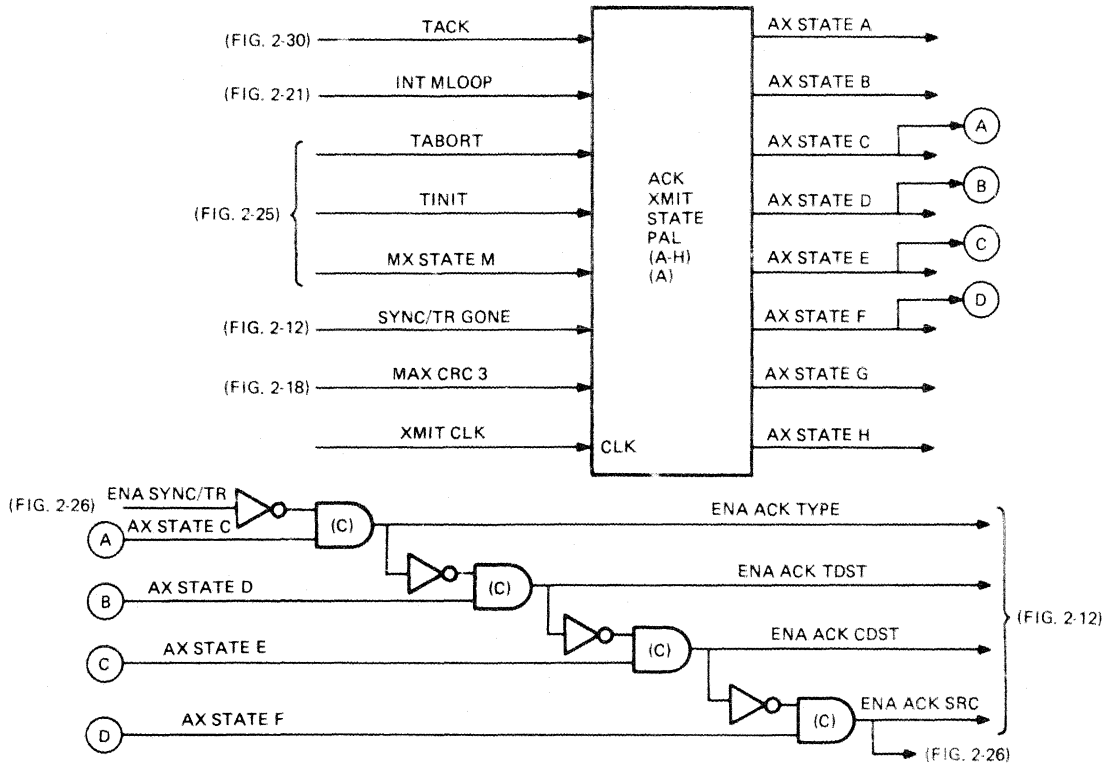
In AX state E the ACK complement destination byte is placed onto the XMIT DATA BUS. The link then advances to AX state F.

In AX state F the ACK source byte is placed onto the XMIT DATA BUS. The link then moves to state G.

In AX state G the CRC bytes generated by the CRC generator are output onto the BUS TDATA bus. When the last CRC byte has been placed onto the bus, MAX CRC 3 asserts and moves the link to AX state H.

In AX state H the sync/trailer PROM is enabled again and the packet trailer bytes are output from the PROM onto the XMIT DATA BUS. After the trailer bytes have been placed onto the bus, SYNC/TR GONE asserts and returns the ACK XMIT PAL to its idle state (state A).

Note in Figure 2-31 that the assertion of each gate coupling a byte to the XMIT DATA BUS depends on the negation of the gate that coupled the preceding byte to the bus. This insures that only one source is driving the XMIT DATA BUS at any one time.



NOTE:
LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

TK-8619

Figure 2-31 ACK Transmit State Logic

CHAPTER 3
PACKET BUFFER MODULE

NOTE

The functional block diagrams in Chapter 3 use logical AND and OR symbols. It does not necessarily follow that a corresponding gate exists on the packet buffer logic prints. The assertion of inputs A and B causing the assertion of output C may be represented on a block diagram by a single AND gate, yet the engineering drawing may show that several circuit stages are involved in the ANDing operation.

The functional block diagrams in this chapter are keyed to the packet buffer module (PB) engineering circuit schematics (CS prints) by letter designations in parentheses. The letters specify the PB CS sheet that contains the detailed logic associated with the functional blocks in the diagram.

The signal names used in the functional block diagrams are the names used on the engineering CS prints. Where other signal names or notes are used, they are enclosed in parentheses.

3.1 DATA FLOW; GENERAL DISCUSSION

Figure 3-1 is a block diagram of data flow through the packet buffer. Information in the form of messages and data, flows through the packet buffer module (PB) in packets of various size. Data going to the CI bus flows from the data path module (DP) to the link while data received from the CI bus flows from the link to the DP.

A transmit buffer (TBUF) is in the data path to the CI bus and a receive buffer (RBUF) is in the data path from the CI bus. The buffers are loaded and read under control of the port microcode. Six operations are used in transferring data in and out of the buffers. Four are used for normal transfer of data. The other two are used for maintenance and self-directed commands. The six operations are listed below:

1. TBUF LOAD
2. TRANSMIT
3. TBUF READ
4. VALID RCVR DATA
5. RBUF MLOAD
6. RBUF READ

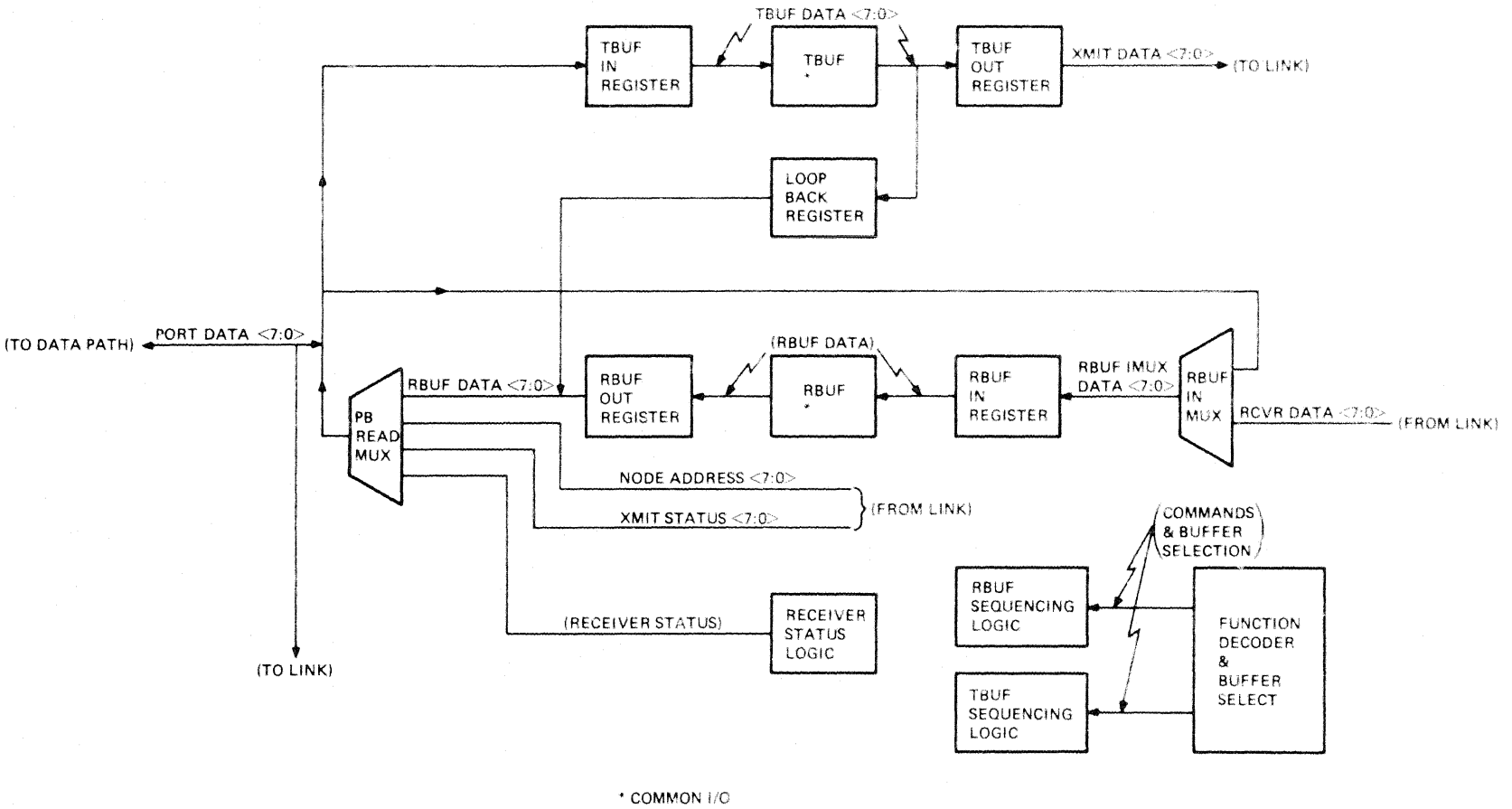


Figure 3-1 Packet Buffer Data Flow

3.1.1 TBUF LOAD
Data from the DP is loaded into the TBUF via the TBUF in register. The TBUF LOAD operation is controlled from the PB.

3.1.2 TRANSMIT
Data is read out of the TBUF into the link via the TBUF out register. The TRANSMIT operation is controlled by the link.

3.1.3 TBUF READ
Data is read out of the TBUF back into the DP via the loopback register. The loopback data is muxed with the received data on the RBUF DATA (7:0) data lines and returned to the port bus via the PB read mux. This operation is controlled by the PB and is used for maintenance and self-directed commands.

3.1.4 VALID RCVR DATA
Received data (RCVR DATA (7:0)) from the link is loaded into the RBUF via the RBUF in mux and the RBUF in register. The VALID RCVR DATA operation is controlled from the link.

3.1.5 RBUF MLOAD (Maintenance Load)
Data from the DP (PORT DATA (7:0)) is loaded into the RBUF via the RBUF in mux and the RBUF in register. The RBUF MLOAD operation is controlled by the PB and is used for maintenance purposes.

3.1.6 RBUF READ
Data is read out of the RBUF to the DP via the RBUF out register and the PB read mux. The data from the RBUF out register is muxed with the loopback data on the RBUF DATA (7:0) data lines. The RBUF READ operation is controlled by the PB.

3.1.7 PB Read Mux
Other data is provided to the DP over the PORT DATA (7:0) bus via the PB read mux. This data is NODE ADDRESS (7:0) and XMIT STATUS (7:0) from the link, and receive status from the receive status logic in the PB.

3.1.8 Control Logic
The PB operations are controlled by decoding and sequencing logic. A function decoder issues commands that specify the operation to be executed. Buffer select logic selects the buffer for the operation specified by the function decoder. If a TBUF is selected (there are two), the TBUF sequencing logic generates the control signals for the operation. Corresponding sequencing logic exists for the RBUFs which generate the control signals for an RBUF operation.

The function decoder and buffer select logic are controlled by the port microcode.

3.2 TBUF DATA FLOW OPERATIONS
The TBUF (Figure 3-2) is divided into two parts (TBUF A and TBUF B) with each TBUF having a separate, parallel data path. Thus, throughput is increased in that TBUF A can be loaded from the DP while TBUF B is being transmitted to the link. Each TBUF has 1K of storage. The following discussion will describe TBUF A and its data path. TBUF B and its data path are identical to TBUF A.

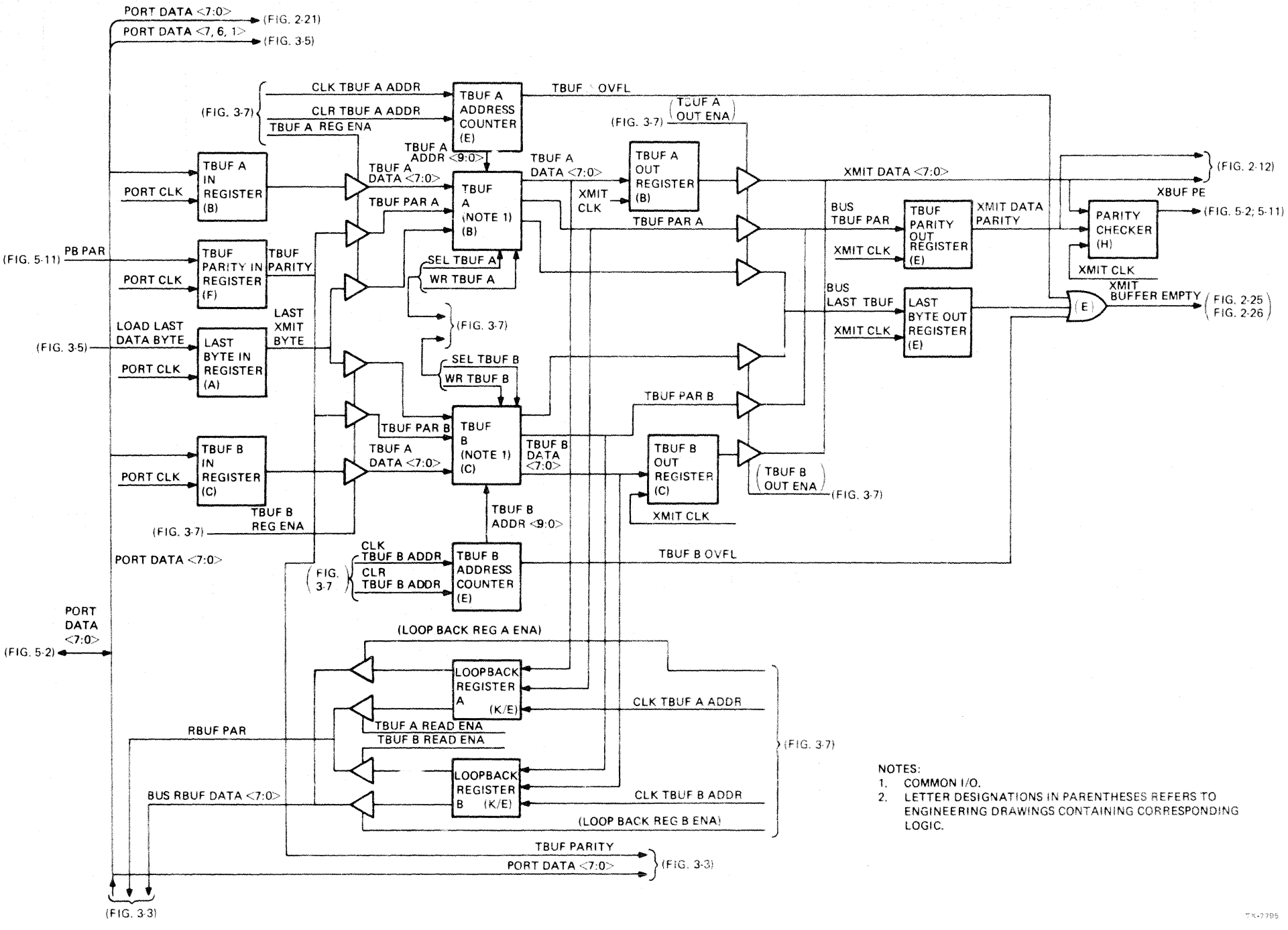


Figure 3-2 TBUF Operations

3.2.1 TBUF LOAD

SEL TBUF A enables TBUF A, selecting it for a TBUF A operation. WR TBUF A enables the TBUF A input and disables the output thereby setting up TBUF A for a write. (TBUF A has a common I/O.) A data byte (PORT DATA ⟨7:0⟩) is clocked into the TBUF A in register by PORT CLK. PORT CLK also clocks a parity bit (PB PAR) from the DP into the TBUF parity in register. TBUF A REG ENA then asserts to enable the data byte (TBUF A DATA ⟨7:0⟩) and the parity bit (TBUF PAR A) to be written into TBUF A.

The TBUF A address (TBUF A ADDR ⟨9:0⟩) is obtained from the TBUF A address counter. The counter is cleared by CLR TBUF A ADDR prior to loading a data packet into TBUF A. As each byte is written, the counter is incremented by CLK TBUF A ADDR to the next location in the buffer.

When the last byte of the data packet is on the port data bus, a LOAD LAST DATA BYTE flag is asserted and clocked into a "last byte in" register by PORT CLOCK. The flag is written into TBUF A along with the last data byte and its parity bit. The flag is used to indicate the end of the data packet to the link during a TRANSMIT operation.

3.2.2 TRANSMIT

SEL TBUF A enables TBUF A, selecting it for a TBUF A operation. WR TBUF A is false to inhibit the TBUF A input and enable the output for a read. The TBUF A address counter is cleared by CLR TBUF A ADDR to address location 0 in TBUF A.

The first data byte is read out of TBUF A from address 0. The byte (TBUF A DATA ⟨7:0⟩) is clocked into the TBUF A output register by XMIT CLK from the link. "TBUF A OUT ENA" is true and gates the data byte out of the register as XMIT DATA ⟨7:0⟩. The parity bit from TBUF A (TBUF PAR A) is gated to the TBUF parity out register where it is clocked in by XMIT CLK. The data byte is clocked into the TBUF A out register at the same time the parity bit is clocked into the TBUF parity out register.

The data byte is now availble to the link as XMIT DATA ⟨7:0⟩ and to a parity checker. The parity bit (XMIT DATA PARITY) from the TBUF parity out register is also applied to the parity checker. If a parity error is detected, XBUF PE is asserted to the DP where it sets an error bit in the port maintenance control and status register (PMCSR).

XMIT DATA PARITY is also applied to the link as the parity bit for the XMIT DATA ⟨7:0⟩ data byte.

CLK TBUF A ADDR increments the TBUF A address counter to the next location in the buffer. The address counter is a 1K counter capable of addressing the 1K locations of TBUF A. In practice, a packet will be less than 1K bytes of data; thus, the address counter should never reach a full count. If the counter is not cleared prior to a TRANSMIT operation, a full count may be reached. In this event, TBUF A OVFL comes true and asserts XMIT BUFFER EMPTY to the link.

When the last data byte is read from TBUF A, the BUS LAST TBUF bit is also read out and clocked into the "last byte out" register by XMIT CLK. This in turn asserts XMIT BUFFER EMPTY to the link as an indication that it has received the entire data packet.

3.2.3 TBUF READ (Loopback)

SEL TBUF A enables TBUF A, selecting it for a TBUF A operation. WR TBUF A is false to inhibit the TBUF A input and enable the TBUF A output for a read. The TBUF A address counter is cleared by CLR TBUF A ADDR to address location 0 in TBUF A.

The first data byte at address 0 (TBUF A DATA ⟨7:0⟩) and its parity bit (TBUF PAR A) is clocked into loopback register A by CLK TBUF A ADDR. Signals "LOOPBACK REG A ENA" and TBUF A READ ENA are true and respectively couple the data byte (RBUF DATA ⟨7:0⟩) to the PB read mux and the parity bit (RBUF PAR) to the DP.

CLK TBUF A ADDR increments the TBUF A address counter to the next location in the buffer.

3.3 RBUF DATA FLOW OPERATIONS

The RBUF (Figure 3-3) is divided into two parts (RBUF A and RBUF B) with each RBUF having a separate, parallel data path. RBUF A can be loaded from the link while RBUF B is being read by the DP, thus allowing greater throughput. Each RBUF has 1K of storage. The following discussion will describe RBUF A and its data path. RBUF B and its data path is identical to RBUF A.

3.3.1 VALID RCVR DATA

A VALID RCVR DATA operation is an RBUF load of received data from the link. The operation is initiated and controlled from the link.

SEL RBUF A enables RBUF A, selecting it for an RBUF A operation. WR RBUF A enables the RBUF A input and disables the output, setting up RBUF A for a write. (RBUF A has a common I/O.)

The data byte and parity bit from the link are input to the PB through an RBUF in mux. The mux uses two select signals; one for the data byte and one for the parity bit. When mux select signal RBUF INPUT MUX SEL is false, the data byte from the link (RCVR DATA ⟨7:0⟩) is applied to the RBUF A in register as RBUF IMUX DATA ⟨7:0⟩. The byte is clocked into the register by RBUF REG CLK and then gated to RBUF A by the true state of RBUF A REG ENA.

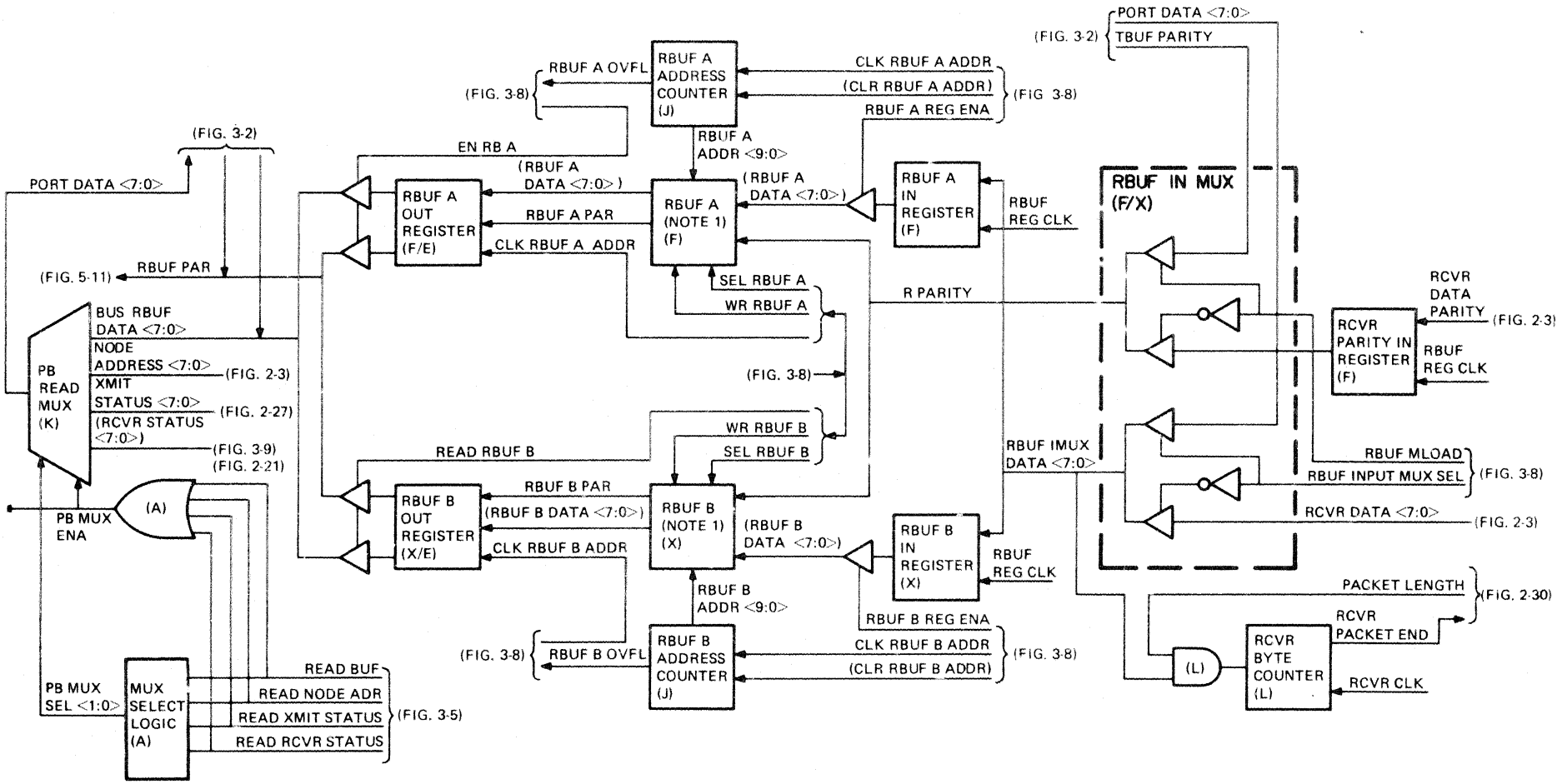
RBUF REG CLK also clocks the parity bit (RCVR DATA PARITY) into the RCVR parity in register. When mux select signal RBUF MLOAD is false, the parity bit from the register is applied to RBUF A as R PARITY.

The RBUF A address (RBUF A ADDR ⟨9:0⟩) is obtained from the RBUF A address counter. The counter is cleared by "CLR RBUF A ADDR" prior to loading in a data packet. As each byte is written, the counter is incremented by CLK RBUF A ADDR to the next location in the buffer. The address counter is a 1K counter capable of addressing the 1K locations of RBUF A. In practice, a packet will be less than 1K bytes of data; thus, the address counter should never reach a full count. If the counter is not cleared prior to a VALID RCVR DATA operation, a full count may be reached. In this event, RBUF A OVFL asserts and terminates the VALID RCVR DATA operation.

The link uses a RCVR byte counter to indicate when the data packet has been loaded into RBUF A. The first two bytes of a data packet specify how many data bytes are in the packet (packet length). PACKET LENGTH from the link asserts and loads the first two packet length bytes into the RCVR byte counter. The counter is a down counter which is decremented by RCVR CLK each time a byte is loaded into RBUF A. RCVR PACKET END asserts when the packet is completely loaded.

3.3.2 RBUF MLOAD (Maintenance Load)

Sel RBUF A enables RBUF A, selecting it for an RBUF A operation. WR RBUF A enables the RBUF A input and disables the output, setting up RBUF A for a write.



NOTES:
1. COMMON I/O.
2. LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.

Figure 3-3 RBUF Operations

The data packet is obtained from the DP via the port data bus and input to the PB through the RBUF in mux. When mux select signal RBUF INPUT MUX SEL is true, data bytes from the port bus (PORT DATA ⟨7:0⟩) are applied to the RBUF A in register as RBUF IMUX DATA ⟨7:0⟩. The bytes are clocked into the register by RBUF REG CLK and then gated to RBUF A by the true state of RBUF A REG ENA.

The parity bit from the port bus (PB PAR) is clocked into the TBUF parity in register (Figure 3-2) and then applied to the RBUF in mux as TBUF PARITY. With mux select signal RBUF MLOAD true, TBUF PARITY is coupled to RBUF A as R PARITY.

The RBUF A address (RBUF A ADDR ⟨9:0⟩) is obtained from the RBUF A address counter. The counter is cleared by "CLR RBUF A ADDR" before loading in a data packet. As each byte is written, the counter is incremented by CLK RBUF A ADDR to the next location in the buffer.

3.3.3 RBUF Read

SEL RBUF A enables RBUF A, selecting it for an RBUF A operation. WR RBUF A is false to inhibit the RBUF A input and enable the output for a read. The RBUF A address counter is cleared by "CLR RBUF A ADDR" to address location 0 in RBUF A.

A data byte ("RBUF A DATA ⟨7:0⟩") and parity bit (RBUF A PAR) read out of RBUF A are clocked into the RBUF A out register by CLK RBUF A ADDR. EN RB A is true, gating out the data byte and parity bit as RBUF DATA ⟨7:0⟩ and RBUF PAR, respectively. (READ RBUF B in the RBUF B data path corresponds to EN RB A.) RBUF PAR is applied to the DP while RBUF DATA ⟨7:0⟩ is placed on the port data bus via the PB read mux.

When reading RBUF A out to the DP, EN RB A asserts and couples the data in the RBUF A out register to the BUS RBUF DATA ⟨7:0⟩ bus before CLK RBUF A ADDR asserts. The data in the RBUF A out register is undetermined until CLK RBUF A ADDR asserts and clocks the first data byte from RBUF A into the register. Thus, when reading RBUF A, the DP discards the first byte as invalid data.

The reading of a data packet from RBUF A does not have to be done in consecutive cycles. The packet can be partially read and the remainder of the packet read at a later time. If a read operation is interrupted, the first data byte read when the read operation is continued, is valid data.

3.3.4 PB Read Mux

The PB read mux muxes four signal groups of eight bits each onto the port data bus as PORT DATA ⟨7:0⟩. When READ BUF is asserted, the RBUF DATA ⟨7:0⟩ lines are selected. READ NODE ADR, READ XMIT STATUS, and READ RCVR STATUS respectively select NODE ADDRESS ⟨7:0⟩, XMIT STATUS ⟨7:0⟩, and "RCVR status". NODE ADDRESS ⟨7:0⟩ and XMIT STATUS ⟨7:0⟩ come directly from the link and do not pertain to the PB. "RCVR status" is comprised of eight status signals relating to received data from the link (Paragraph 3.8).

The PB read mux is enabled by PB MUX ENA whenever any of the four select signals is asserted.

3.4 CLOCKS

Three clocks are used within the PB and these are obtained from the DP and the link (Figure 3-4). The three clocks used are:

1. PORT CLK*
2. XMIT CLK
3. RCVR CLK

---

* PORT CLK T3 also appears on the PB logic prints but is identical to PORT CLK. The two signals fan out from different drivers, hence the different mnemonics.

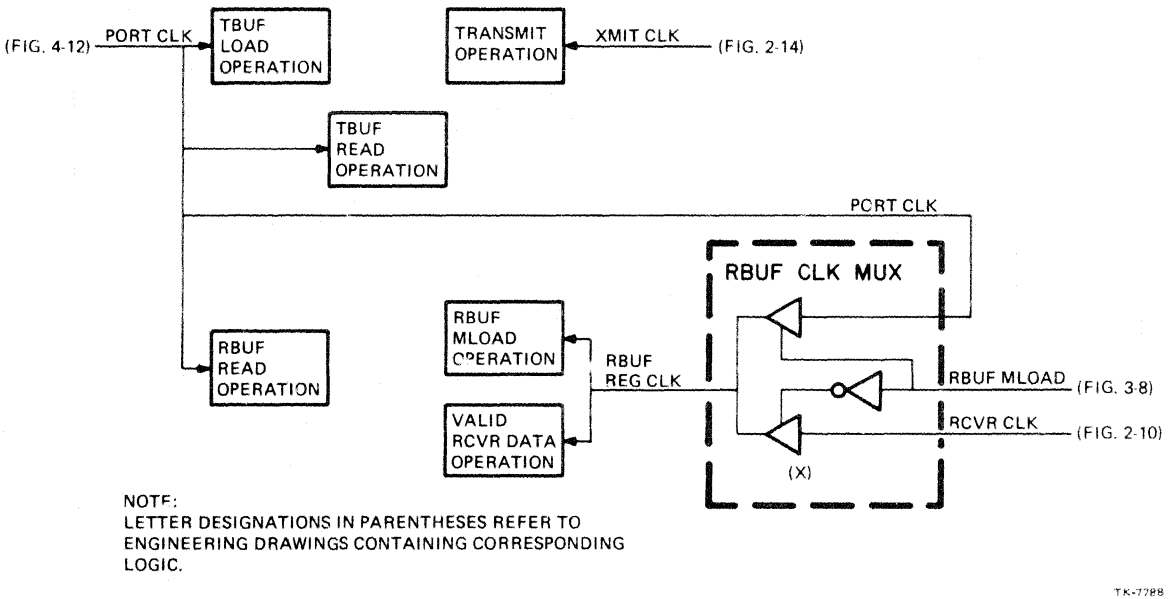


Figure 3-4 Packet Buffer Clocks

PORT CLK is obtained from the DP and synchronizes all operations that involve data flow to or from the DP. PORT CLK has a 200 ns period.

XMIT CLK is obtained from the link and synchronizes the TRANSMIT operation in which data flows from the PB to the link. XMIT CLK has a 114 ns period.

RCVR CLK is obtained from the link and synchronizes the VALID RCVR DATA operation in which data flows from the link to the PB. RCVR CLK has a 114 ns period.

Figure 3-4 illustrates the six PB operations and the clocks that synchronize them. Note that the two operations that load the RBUF, (RBUF MLOAD and VALID RCVR DATA) are synchronized by RBUF REG CLK. RBUF REG CLK is PORT CLK when the RBUF is being loaded from the DP (RBUF MLOAD operation), and is RCVR CLK when the RBUF is being loaded from the link (VALID RCVR DATA operation).

The TBUF and RBUF address counters are clocked by whichever clock is synchronizing the particular operation.

3.5 FUNCTION DECODER AND BUFFER SELECT LOGIC

The SELECT bit from the microword asserts for one microcycle and enables the function decoder and the buffer select logic (see Figure 3-5). Four link control bits from the microword (LINK CONTROL ⟨3:0⟩) carry the PB function command to the function decoder which outputs one of thirteen possible commands for one microcycle. The function commands and their associated link control codes are shown in Table 3-1.

The following paragraphs describe each of the function commands.

3.5.1 SEL LOAD BUF

Prior to issuing a load buffer command (LOAD BUF or LOAD LAST DATA BYTE), or a RESET TBUF command, the microcode selects the buffer with the SEL LOAD BUF command. The selection is made by the buffer select logic during the microcycle, in which the microword SELECT bit is true. The selected output is latched and remains true until SELECT asserts again and another buffer is selected.

SEL LOAD BUF enables the "load" section of the buffer select logic which outputs one of four "buffer load enable" signals according to port data bits PORT DATA ⟨7:6⟩ (Table 3-2).

3.5.2 SEL READ BUF

Before issuing a read buffer command (READ BUF) or a RELEASE RBUF command, the microcode selects the buffer with the SEL READ BUF command. The selection is made by the buffer select logic during the microcycle in which the microword SELECT bit is true. The selected output is latched and remains true until SELECT asserts again and another buffer is selected.

SEL READ BUF enables the "read" section of the buffer select logic which outputs one of four "buffer read enable" signals according to port data bits PORT DATA ⟨7:6⟩ (Table 3-3).

3.5.3 LOAD BUF

The LOAD BUF command loads port data into the buffer selected by the SEL LOAD BUF command. The load operations are TBUF LOAD and RBUF MLOAD. The VALID RCVR DATA operation (loading of the RBUF from the link) is not a function of the PB microword.

A data packet does not have to be loaded in consecutive cycles. A packet can be partially loaded and the remainder of the packet loaded at a later time.

When loading a TBUF, the last byte of data must be loaded with a LOAD LAST DATA BYTE command.

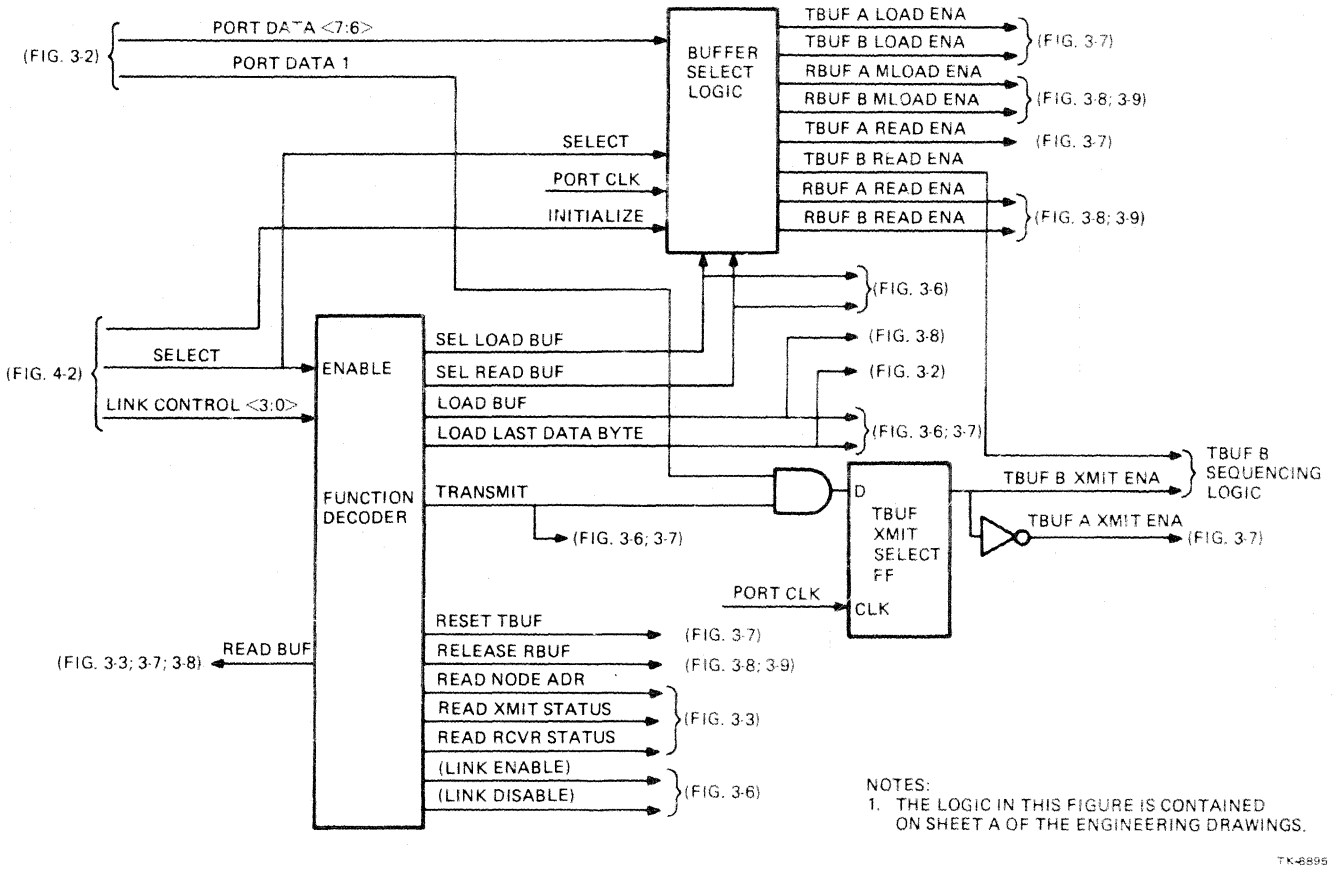


Figure 3-5 Function Decoder and Buffer Select Logic

Table 3-1 Link Control Codes Vs PB Function Commands

LINK CONTROL 3	2	1	0	Function Command
0	0	0	0	READ NODE ADR
0	0	0	1	LOAD LAST DATA BYTE
0	0	1	0	—
0	0	1	1	TRANSMIT
0	1	0	0	—
0	1	0	1	—
0	1	1	0	"Enable link"
0	1	1	1	"Disable link"
1	0	0	0	READ RCVR STATUS
1	0	0	1	READ XMIT STATUS
1	0	1	0	READ BUF
1	0	1	1	LOAD BUF
1	1	0	0	RELEASE RBUF
1	1	0	1	RESET TBUF
1	1	1	0	SEL READ BUF
1	1	1	1	SEL LOAD BUF

Table 3-2 Load Buffer Select Code

PORT DATA 7	6	Buffer Selected
0	0	TBUF A LOAD ENA
0	1	TBUF B LOAD ENA
1	0	RBUF A MLOAD ENA
1	1	RBUF B MLOAD ENA

Table 3-3 Read Buffer Select Code

PORT DATA 7	6	Buffer Selected
0	0	RBUF A READ ENA
0	1	RBUF B READ ENA
1	0	TBUF A READ ENA
1	1	TBUF B READ ENA

3.5.4 LOAD LAST DATA BYTE

The LOAD LAST DATA BYTE command is the load command for the last byte of data loaded into one of the TBUFs. It performs the same function as a LOAD BUF command and in addition, loads a "last data byte" bit into the TBUF along with the data byte.

3.5.5 READ BUF

The READ BUF command reads data from the buffer selected by the SEL READ BUF command. The data is read out to the port data bus via the PB read mux. The read operations are TBUF READ and RBUF READ. The TRANSMIT operation (reading of the TBUF to the link) is initiated by the PB microword but is a separate command.

3.5.6 TRANSMIT

The TRANSMIT command reads data from the selected TBUF to the link. After the command is issued, the link controls the read operation. The link continues reading the selected TBUF until the "last data byte" flag is read out.

During the microcycle that TRANSMIT is true, one of the port data bits (PORT DATA 1) is sampled to determine which TBUF will be transmitted. A TBUF XMIT flip-flop asserts TBUF A XMIT ENA if the port data bit is false, and TBUF B XMIT ENA if the bit is true.

Only one TRANSMIT operation can be executed at a time. (Only one TBUF can be read at a time by the link.) A TBUF must be completely read, or the operation aborted and the transmit status cleared, before another TRANSMIT command can be issued.

3.5.7 RESET TBUF

The RESET TBUF command resets the address counter associated with the selected TBUF.

3.5.8 RELEASE RBUF

The RELEASE RBUF command resets the address counter associated with the selected RBUF. It also clears the "full" flag (negates RBUF FULL; Figure 3-9) for the selected buffer making it available to the link for a VALID RCVR DATA operation.

3.5.9 READ NODE ADR

The READ NODE ADR command selects the node address (NODE ADDRESS (7:0)) from the link to be muxed onto the port data bus by the PB read mux.

3.5.10 READ XMIT STATUS

The READ XMIT STATUS command selects the transmit status (XMIT STATUS (7:0)) from the link to be muxed onto the port data bus by the PB read mux.

3.5.11 READ RCVR STATUS

The READ RCVR STATUS command selects the eight "receive status" bits to be muxed onto the port data bus by the PB read mux. The "receive status" bits are discussed in Paragraph 3.8.

3.5.12 Link Enable and Link Disable

The "link enable" and "link disable" commands are used in the link module and perform no function on the PB other than to assert PB LOAD. PB LOAD must be true to enable the path to the link for the commands. (See PB LOAD; Paragraph 3.6.)

3.6 PB LOAD

Data placed on the port data bus from the DP is obtained from a 32-bit PB OUT register. The register output is enabled by PB LOAD from the PB. PB LOAD is asserted for all commands that require data to be transferred from the PB OUT register to the port data bus. (See Figure 3-6.)

An eight-bit enable and an eight-bit disable command function for the link is transferred to the link from the DP via the port data bus (Figure 3-1). Although these commands do not pertain to the PB, it is required that PB LOAD be true in order to transfer the commands from the PB OUT register to the port data bus.

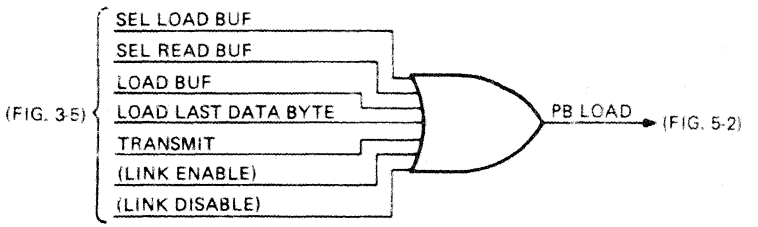
Referring to Figure 3-6:

1. SEL LOAD BUF and SEL READ BUF commands require port data bits PORT DATA (7:6) to select which buffer to load or read.

2. LOAD BUF and LOAD LAST DATA BYTE commands obtain the byte to be loaded from the port data bus.

3. The TRANSMIT command requires PORT DATA 1 to select which TBUF to transmit to the link.

4. "Link enable" and "link disable" commands require a path from the PB OUT register on the DP to the port data bus.



NOTES:
1. THE LOGIC IN THIS FIGURE IS CONTAINED ON SHEET A OF THE ENGINEERING DRAWINGS.

TK-7789

Figure 3-6 PB Load Logic

3.7 SEQUENCING LOGIC

The PB function decoder and buffer select logic generates the necessary signals to enable the TBUF and RBUF load/read operations. The signals pertinent to each of the six operations are discussed in Paragraphs 3.7.1 through 3.7.6. The A buffer is used in all the discussions. Corresponding logic exists for the B buffer. Figure 3-7 illustrates the sequencing logic associated with the three TBUF operations. Figure 3-8 illustrates the sequencing logic associated with the three RBUF operations.

3.7.1 TBUF LOAD

The TBUF LOAD sequencing logic is illustrated in Figure 3-7. Before a TBUF LOAD operation is initiated, a RESET TBUF command is issued to clear the selected TBUF address counter. The RESET TBUF command is ANDed with TBUF A LOAD ENA to assert CLR TBUF A ADDR. The next PORT CLK pulse asserts CLK TBUF A ADDR which clears the counter. (The address counter is an asynchronous counter which requires a clock pulse while the clear input is true in order to reset.)

The TBUF LOAD operation is initiated by the LOAD BUF command. The LOAD BUF command (or LOAD LAST DATA BYTE if this is the last byte) is ANDed with TBUF A LOAD ENA (or TBUF B LOAD ENA) to enable the pulse width flip-flop to be set by the next PORT CLK pulse. The flip-flop output is ANDed with TBUF A LOAD ENA to assert WR TBUF A and SEL TBUF A. SEL TBUF A enables TBUF A and WR TBUF A enables it for a load.

The output of the pulse width flip-flop is delayed 80 ns, and then used to clear the flip-flop. Thus, SEL TBUF A and WR TBUF A become 80 ns pulses.

Another output of the pulse width flip-flop is delayed 20 ns and ANDed with TBUF A LOAD ENA to assert TBUF A REG ENA and CLK TBUF A ADDR. These two signals are also 80 ns wide and are delayed 20 ns with respect to SEL TBUF A and WR TBUF A.

TBUF A REG ENA gates the output of the TBUF A in register to TBUF A. Delaying TBUF A REG ENA allows time for the tri-state output of TBUF A to be disabled by WR TBUF A before the write data is gated into TBUF A from the TBUF A in register.

The TBUF A address counter is incremented on the trailing edge of CLK TBUF A ADDR. Delaying CLK TBUF A ADDR assures that TBUF A is disabled (SEL TBUF A is negated) before the address is incremented to the next location.

3.7.2 TRANSMIT

The TRANSMIT sequencing logic is illustrated in Figure 3-7. A TRANSMIT operation requires both a TRANSMIT command from the function decoder and the XMIT DATA ENA signal from the link. XMIT DATA ENA is true when the link is ready to receive transmitted data from the PB.

Before a TRANSMIT operation can be executed, the selected TBUF address counter must be cleared. In a TRANSMIT operation the counter is cleared by the assertion of TRANSMIT instead of by a RESET TBUF command. TRANSMIT is ANDed with TBUF A XMIT ENA to assert CLR TBUF A ADDR. The next PORT CLK pulse asserts CLK TBUF A ADDR. Clocking the counter with the clear input asserted resets it to zero.

XMIT DATA ENA is ANDed with TBUF A XMIT ENA to assert "TBUF A OUT ENA" and SEL TBUF A. SEL TBUF A enables TBUF A. "TBUF A OUT ENA" gates the data byte out of the TBUF A register to the link.

CLK TBUF A ADDR increments the TBUF A address counter during the TRANSMIT operation. The clock is asserted by the ANDing of XMIT DATA ENA, TBUF A ENA, and XMIT CLK. Thus, the link synchronizes the address counter with XMIT CLK.

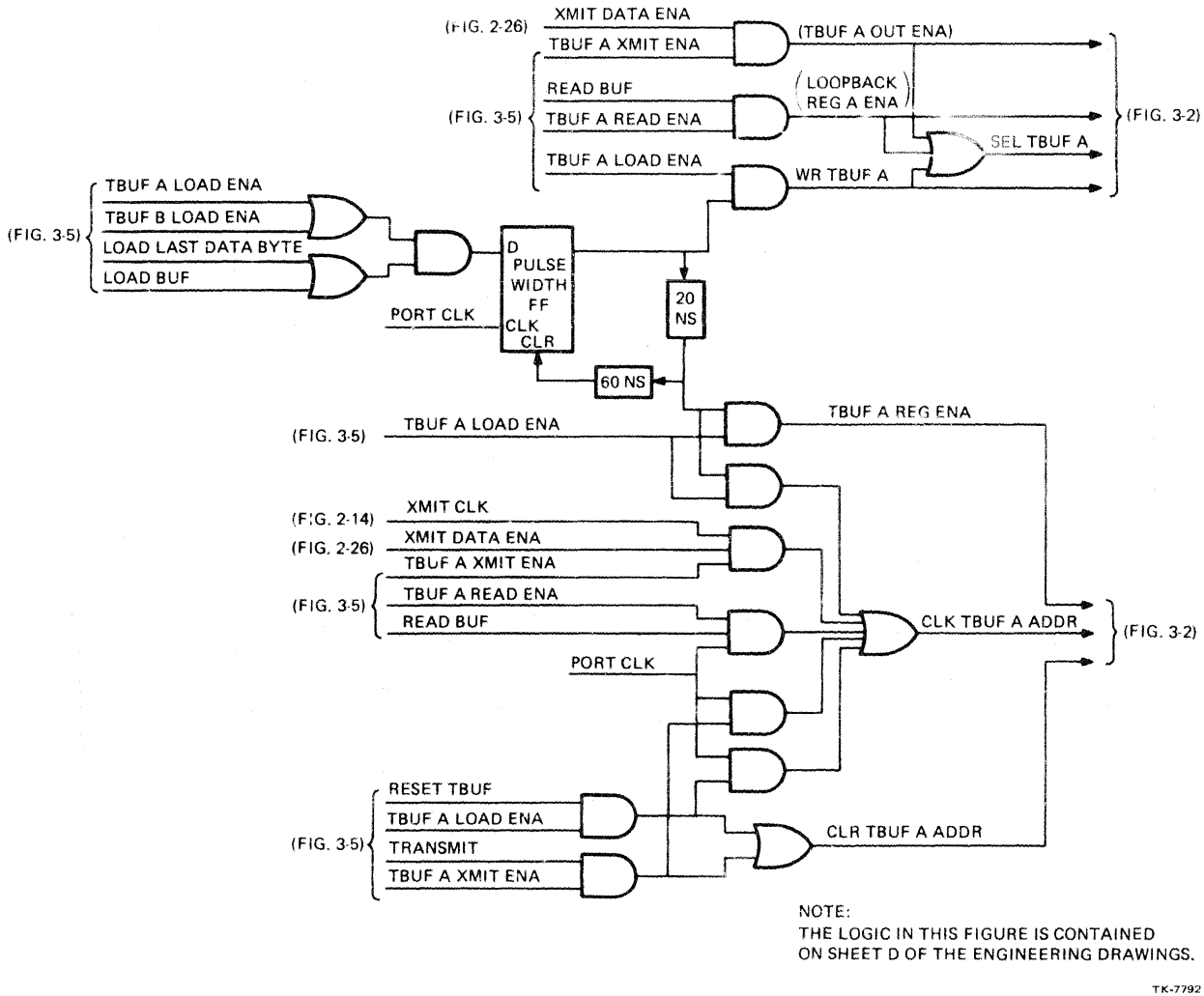


Figure 3-7 TBUF Sequencing Logic

3.7.3 TBUF READ (Loopback)

The TBUF READ (loopback) sequencing logic is shown in Figure 3-7. The TBUF A address counter must be reset to zero before the TBUF READ operation can be executed. The microcode resets the address counter by selecting TBUF A with a SEL LOAD BUF command (asserting TBUF A LOAD ENA from the buffer select logic) and then asserting the RESET TBUF command. The ANDing of RESET TBUF and TBUF A LOAD ENA asserts CLR TBUF A ADDR. The next PORT CLK pulse asserts CLK TBUF A ADDR thereby resetting the counter.

With the address counter reset to zero, READ BUF and TBUF A READ ENA are ANDed to assert "LOOPBACK REG A ENA" and SEL TBUF A. SEL TBUF A enables TBUF A. "LOOPBACK REG A ENA" gates the data from loopback register A onto the RBUF data lines.

The ANDing of READ BUF, TBUF A READ ENA, and PORT CLK asserts CLK TBUF A ADDR. Thus, the address counter is synchronized by PORT CLK from the DP.

3.7.4 VALID RCVR DATA

The VALID RCVR DATA logic is illustrated in Figure 3-8. The RBUF address counter is cleared at the end of all RBUF operations. Thus, the VALID RCVR DATA operation will start with the address counter already set to zero.

The VALID RCVR DATA operation is initiated and executed entirely under link control. Consequently, the selection of the receive buffer (RBUF A or RBUF B) is not made by the buffer select logic but by the "RBUF load selection" logic shown in Figure 3-8.

When both RBUFs are empty, RBUF A is selected to receive the data packet as described below. The RBUF A LOAD ENA and the RBUF B LOAD ENA flip-flops are initially in the reset state. Signals RBUF A FULL ENA and RBUF B FULL ENA are false (both RBUFs are empty). When VALID RCVR DATA asserts, the VRD and the RBUF A LOAD ENA flip-flops are enabled and become set by the next RCVR CLK pulse. The corresponding RBUF B LOAD ENA flip-flop does not set due to the negated state of RBUF A FULL ENA. VALID RCVR DATA stays true while the entire data packet is being loaded, holding "VRD" true and keeping the RBUF A LOAD ENA flip-flop set via a feedback gate.

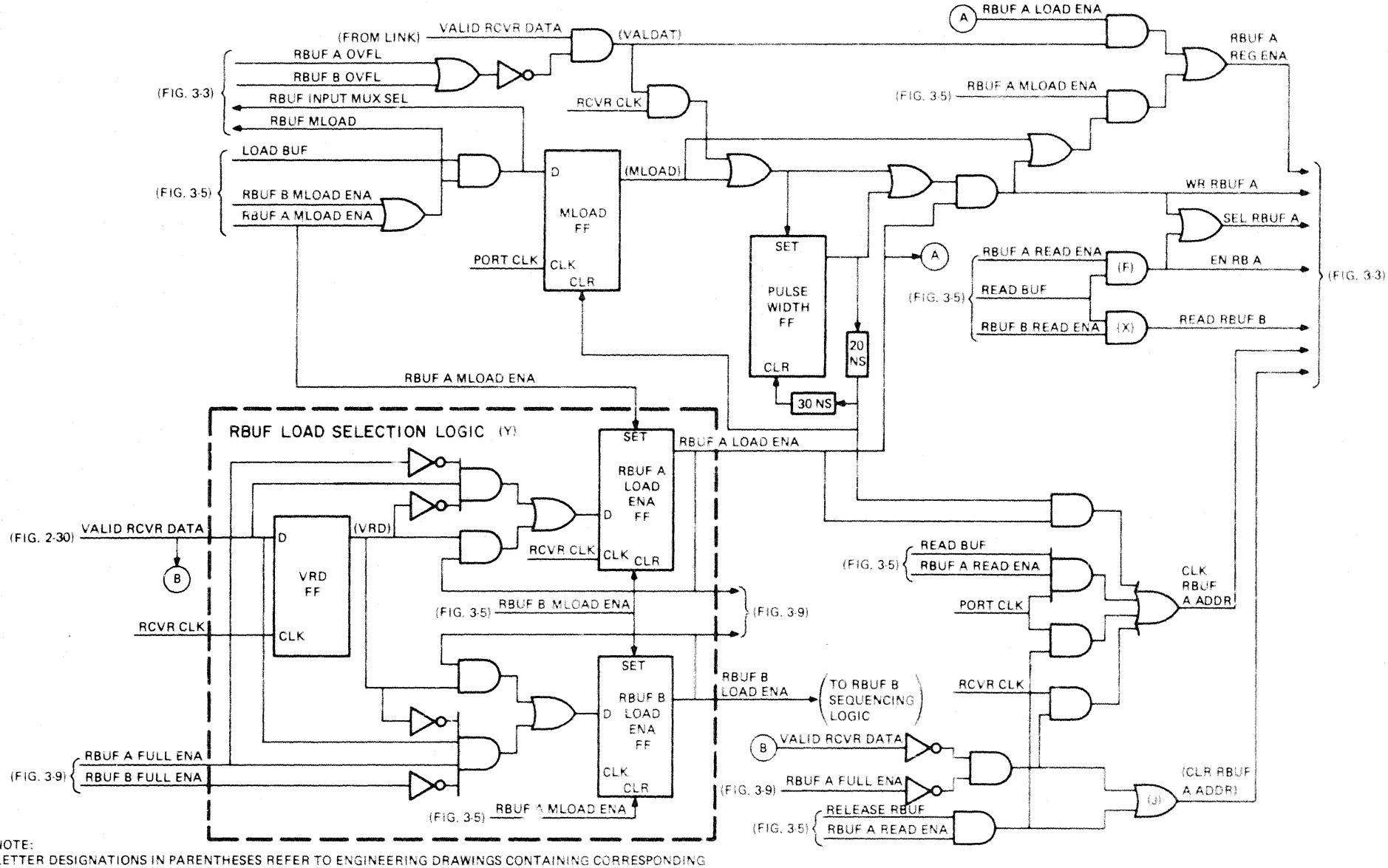
After the packet is loaded into RBUF A, RBUF A FULL ENA is asserted by the receive status logic. When VALID RCVR DATA asserts to load another packet, the true state of RBUF A FULL ENA inhibits the setting of the RBUF A LOAD ENA flip-flop but allows the RBUF B LOAD ENA flip-flop to be set. Thus, RBUF B is selected to receive the next data packet.

Selection will continue to alternate to the empty RBUF. If both RBUFs are full, neither RBUF A LOAD ENA nor RBUF B LOAD ENA will assert and the load operation will not be executed. This condition causes the receive status logic to raise a flag to both the link and the DP (see Paragraph 3.8).

The load operation is initiated by the assertion of VALID RCVR DATA. If neither address counter has overflowed (both RBUF A OVFL and RBUF B OVFL are false), "VALDAT" asserts and is ANDed with RBUF A LOAD ENA to assert RBUF A REG ENA. RBUF A REG ENA gates the output of the RBUF A in register to RBUF A.

"VALDAT" is synchronized by RCVR CLK and sets the pulse-width flip-flop. The flip-flop output is ANDed with RBUF A LOAD ENA to assert WR RBUF A and SEL RBUF A. SEL RBUF A enables RBUF A. WR RBUF A enables RBUF A for a load operation. The output of the pulse width flip-flop is delayed 50 ns and then fed back to reset the flip-flop, converting the SEL RBUF A and the WR RBUF A signals into 50 ns pulses.

Another output from the pulse-width flip-flop is delayed 20 ns and ANDed with RBUF A LOAD ENA to assert CLK RBUF A ADDR. The setting of the pulse-width flip-flop is synchronized by RCVR CLK, hence the incrementation of the RBUF A address counter is also synchronized by RCVR CLK.



NOTE:
LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC. THE RBUF LOAD SELECTION LOGIC IS ON SHEET Y. LOGIC NOT DESIGNATED IS ON SHEET H.

Figure 3-8 RBUF Sequencing Logic

The RBUF A address counter is incremented on the trailing edge of CLK RBUF A ADDR. By shifting CLK RBUF A ADDR 20 ns, it is assured that RBUF A is disabled (SEL RBUF A negated) before the address is changed to the next location.

After the data packet has been loaded into RBUF A, the RBUF A address counter must be reset to zero. At the end of the load operation, VALID RCVR DATA negates. One cycle later RBUF A FULL ENA asserts indicating that RBUF A is full and ready to be read out to the port. During this cycle, the negated state of both of these signals asserts CLR RBUF A ADDR and, on the next RCVR CLK pulse, asserts CLK RBUF A ADDR. This clears the RBUF A address counter, preparing it to clock an RBUF A READ operation.

3.7.5 RBUF MLOAD

Refer to the RBUF load selection logic in Figure 3-8. The assertion of RBUF A MLOAD ENA directly sets the RBUF A LOAD ENA flip-flop and directly resets the RBUF B LOAD ENA flip-flop. Thus, RBUF A LOAD ENA is true during the RBUF MLOAD operation.

The RBUF MLOAD operation is initiated by the assertion of LOAD BUF. The LOAD BUF command is ANDed with RBUF MLOAD (asserted by either RBUF A MLOAD ENA or RBUF B MLOAD ENA) to assert RBUF INPUT MUX SEL. RBUF MLOAD and RBUF INPUT MUX SEL switch the RBUF in mux to select the parity bit and the data byte from the DP. RBUF INPUT MUX SEL also enables the MLOAD flip-flop to be set by the next PORT CLK pulse. The flip-flop output ("MLOAD") is ANDED with RBUF A MLOAD ENA to assert RBUF A REG ENA. RBUF A REG ENA gates the output of the RBUF A in register to RBUF A.

"MLOAD" also sets the pulse width flip-flop. The flip-flop output is ANDed with RBUF A LOAD ENA to assert WR RBUF A and SEL RBUF A. SEL RBUF A enables RBUF A and WR RBUF A enables it for a load. The output of the pulse width flip-flop is delayed 50 ns and then fed back to reset the flip-flop, converting SEL RBUF A and the WR RBUF A signals into 50 ns pulses.

Another output from the pulse width flip-flop is delayed 20 ns and ANDed with RBUF A LOAD ENA to assert CLK RBUF A ADDR. The setting of the pulse-width flip-flop is synchronized by PORT CLK (via the MLOAD flip-flop), hence the incrementation of the RBUF A address counter is also synchronized by PORT CLK.

The RBUF A address counter is incremented on the trailing edge of CLK RBUF A ADDR. By shifting CLK RBUF A ADDR 20 ns, it is assured that RBUF A is disabled (SEL RBUF A false) before the address is changed to the next location.

After the MLOAD operation is completed, the RBUF A address counter must be reset to zero. The microcode accomplishes the reset by selecting RBUF A with the SEL READ BUF command (asserting RBUF A READ ENA from the buffer select logic) and then asserting the RELEASE RBUF command. The ANDing of RELEASE RBUF and RBUF A READ ENA asserts CLR RBUF A ADDR. The next RCVR CLK pulse asserts CLK RBUF A ADDR, thereby resetting the counter.

3.7.6 RBUF READ

The RBUF READ logic is illustrated in Figure 3-8. The RBUF READ operation is initiated by the assertion of READ BUF. The READ BUF command is ANDed with RBUF A READ ENA to assert EN RB A and SEL RBUF A. SEL RBUF A enables RBUF A and EN RB A gates the data from the RBUF A out register onto the RBUF data lines. (The signal in the RBUF B data path corresponding to EN RB A is READ RBUF B.)

The ANDing of READ BUF, RBUF A READ ENA, and PORT CLK asserts CLK RBUF A ADDR. Thus, the RBUF A address counter is synchronized by PORT CLK from the DP.

After the READ RBUF operation is completed, the RBUF A address counter must be reset to zero. The microcode does this by selecting RBUF A with the SEL READ BUF command (asserting RBUF A READ ENA from the buffer select logic) and then asserting the RELEASE RBUF command. The ANDing of RELEASE RBUF and RBUF A READ ENA asserts CLR RBUF A ADDR. The next PORT CLK pulse asserts CLK RBUF A ADDR thereby resetting the counter.

3.8 RCVR STATUS

"RCVR status" is placed on the port data bus from the PB read mux when the READ RCVR STATUS command is asserted. "RCVR status" consists of eight signals. The signals, described in Paragraphs 3.8.1 through 3.8.7, are listed below:

1. CRC ERR
2. RBUF A FULL
3. RBUF B FULL
4. RBUF B FIRST
5. RBUF A BUS
6. RBUF B BUS
7. RCVR A ENABLE
8. RCVR B ENABLE

Figure 3-9 illustrates the RCVR status logic.

3.8.1 CRC ERR

The link does a CRC check on received data packets. The receive status CRC ERR bit is asserted if a CRC error is detected. The CRC ERR bit is used only in maintenance loop modes. It is not used in normal operation.

The CRC ERR bit asserts after the associated data packet has been loaded into the RBUF. Thus, if a CRC error is flagged, the packet containing the error is in the RBUF.

VALID RCVR STATUS asserts after a data packet has been loaded into the RBUF with a VALID RCVR DATA operation. If no CRC error occurred, CRC STATUS is true when VALID RCVR STATUS is asserted. This causes CRC OK to assert. CRC OK enables the CRC OK flip-flop to set on the next RCVR CLK pulse. The asserted output from the flip-flop results in a negated CRC ERR bit for RCVR STATUS.

3.8.2 RBUF A FULL, RBUF B FULL

If RBUF A had just been loaded with a data packet having no CRC error, CRC OK is asserted and ANDed with RBUF A LOAD ENA to enable the RBUF A FULL ENA flip-flop to set. RCVR CLK sets the flip-flop asserting RBUF A FULL ENA. The flip-flop is held set via a feedback gate holding RBUF A FULL ENA true. The next PORT CLK pulse asserts RBUF A FULL via the RBUF A FULL flip-flop. When RBUF A FULL is true it asserts REC ATTN to the DP.

RBUF A is emptied (read out to the DP) by a READ RBUF operation. After a READ RBUF operation, a RELEASE RBUF command is issued to reset the RBUF A address counter and to release RBUF A back to the link. The RELEASE RBUF command releases RBUF A to the link by asserting CLR RBUF A via two flip-flops. RELEASE RBUF is ANDed with the negated state of RBUF B READ ENA to enable the first CLR RBUF A flip-flop to be set by PORT CLK. (RBUF A has just been read out; therefore, RBUF B READ ENA will be false.) The output from the first flip-flop enables the second CLR RBUF A flip-flop which is set by RCVR CLK. Thus, CLR RBUF A is synchronized by RCVR CLK.

CLR RBUF A breaks the feedback latch holding the RBUF A FULL ENA flip-flop set. This negates both RBUF A FULL ENA and RBUF A FULL, indicating that RBUF A is ready for another load from the link.

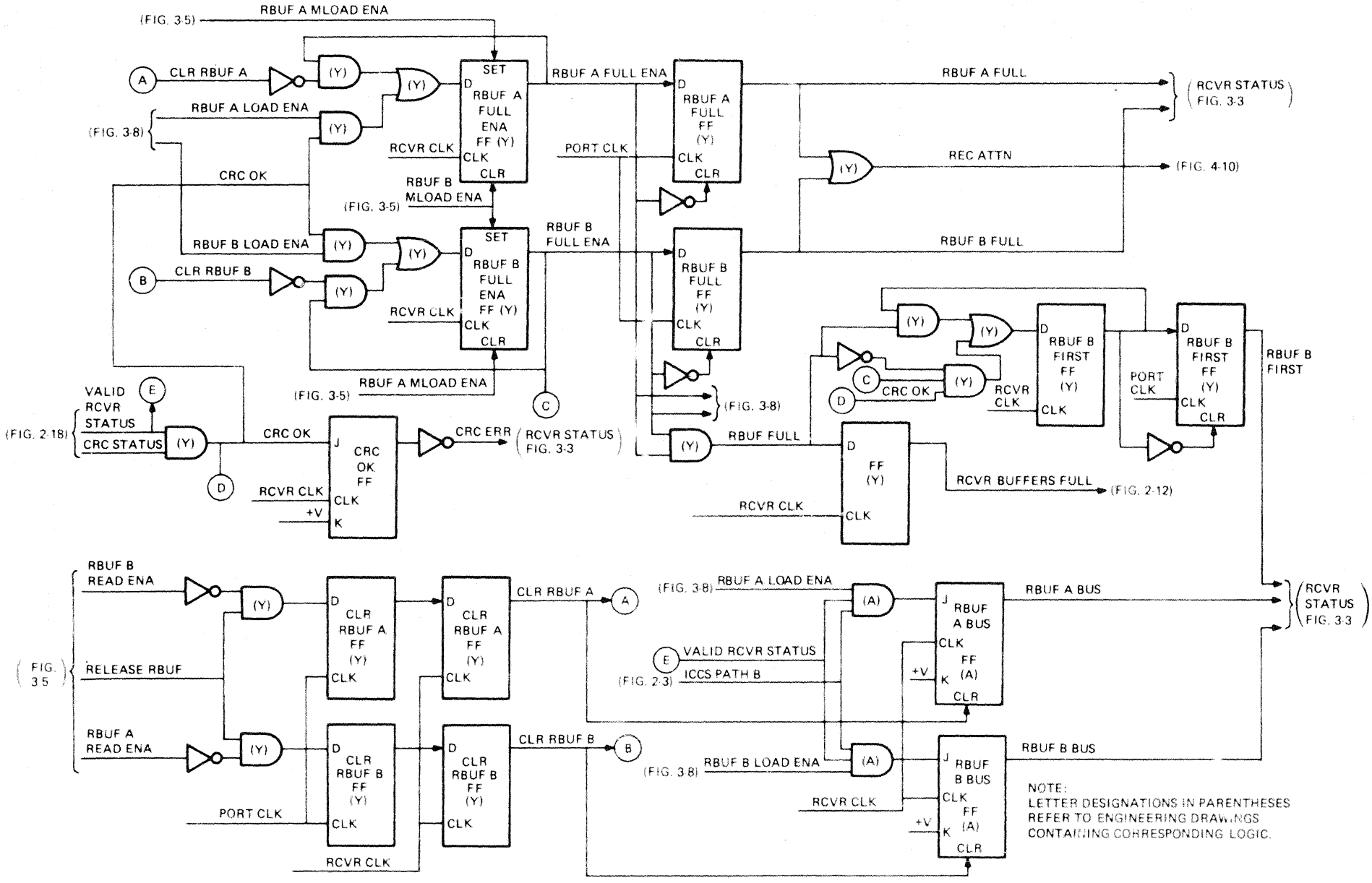


Figure 3-9 RCVR Status Logic

Identical "RBUF FULL" logic exists for RBUF B. If the data packet had been loaded into RBUF B instead of RBUF A, an identical sequence would have occurred in the corresponding RBUF B logic causing "RCVR status" bit RBUF B FULL to assert.

Should both RBUF A FULL and RBUF B FULL be true, RCVR BUFFERS FULL is asserted to the link preventing it from initiating another VALID RCVR DATA operation.

3.8.3 RBUF B FIRST

If both RBUFs are full (RBUF FULL true), the RBUF B FIRST status bit indicates which RBUF was filled first. The RBUF B FIRST status bit is invalid (not sampled) until both RBUFs are filled.

RBUF B FULL ENA is ANDed with CRC OK and the negated state of RBUF FULL to enable the first RBUF B FIRST flip-flop to be set by RCVR CLK. The flip-flop is set if RBUF B is full but not RBUF A. The second RBUF B FIRST flip-flop is set by PORT CLK asserting RBUF B FIRST.

If RBUF A is loaded while RBUF B is still full, RBUF FULL asserts holding the first RBUF B FIRST flip-flop set via a feedback gate. With both RBUFs full, the RBUF B FIRST bit is sampled and found to be true.

3.8.4 RBUF A BUS

This bit indicates which CI bus received the last data packet loaded into RBUF A. If the bit is negated, the pack was received on CI bus A. If the bit is asserted, the pack was received on CI bus B.

While RBUF A is being loaded, RBUF A LOAD ENA is true. RBUF A LOAD ENA is ANDed with VALID RCVR STATUS and ICCS PATH B. Thus, when VALID RCVR STATUS asserts, the ICCS PATH B signal is sampled. If the signal is true, the data packet just loaded into RBUF A was received on CI bus B. In this case, the RBUF A BUS flip-flop is enabled and sets on the next RCVR CLK. When the flip-flop sets, the RBUF A BUS bit is asserted as part of "RCVR status."

3.8.5 RBUF B BUS

This bit indicates which CI bus received the last data packet loaded into RBUF B. If the bit is negated, the pack was received on CI bus A. If the bit is asserted, the pack was received on CI bus B.

The RBUF B BUS logic is identical to the RBUF A BUS logic with RBUF B replacing RBUF A.

3.8.6 RCVR A ENABLE

This bit is set if the RCVR A ENB bit (bit⟨00⟩) of a "link enable" command byte is set. The RCVR A ENB bit must be set for the link to respond to traffic on CI bus A.

3.8.7 RCVR B ENABLE

This bit is set if the RCVR B ENB bit (bit ⟨07⟩) of a "link enable" command byte is set. The RCVR B ENB bit must be set for the link to respond to traffic on CI bus B.

CHAPTER 4
CONTROL STORE

NOTE

The functional block diagrams in Chapter 4 use logical AND and OR symbols. It does not necessarily follow that a corresponding gate exists on the engineering logic prints. The assertion of inputs A and B causing the assertion of output C may be represented on a block diagram by a single AND gate, yet the engineering drawing may show that several circuit stages are involved in the ANDing operation.

The block diagrams are keyed to the engineering circuit schematics (CS prints) by letter designations in parentheses. The letters specify the CS sheet that contains the logic associated with the functional blocks in the diagram. The logic for the CS function discussed in this chapter, is divided between the DP and the PB modules. A note on each block diagram specifies which module contains the logic used in the diagram.

The signal names used in the functional block diagrams are the names used on the engineering CS prints. Where other signal names or notes are used, they are enclosed in parentheses.

4.1 SIMPLIFIED BLOCK DIAGRAM

The control store (Figure 4-1) consists of 3K bytes of storage used to store the port microcode. The microcode uses 48-bit microwords. Each microword consists of 47 control bits (BUS U⟨46:00⟩) and a sync bit used for maintenance purposes. The 3K of storage consists of 2K of RAM and 1K of PROM.

The RAM area of the CS is written during the uninitialized state. IB IN ⟨31:00⟩ from the DP is placed on the CS I/O bus (BUS U⟨46:00⟩) and then written into the CS. The lower 32 bits are written first and then the upper bits.

Bit 46 is the parity bit for the microword (excluding the sync bit). A parity check is performed on each microword read out of the CS during the initialized state when the microcode is running. If a parity error is detected, CSPE is asserted to the DP as an error flag.

Most of the microword read from the CS is latched into the microword register. The register outputs control signals to all of the port modules.

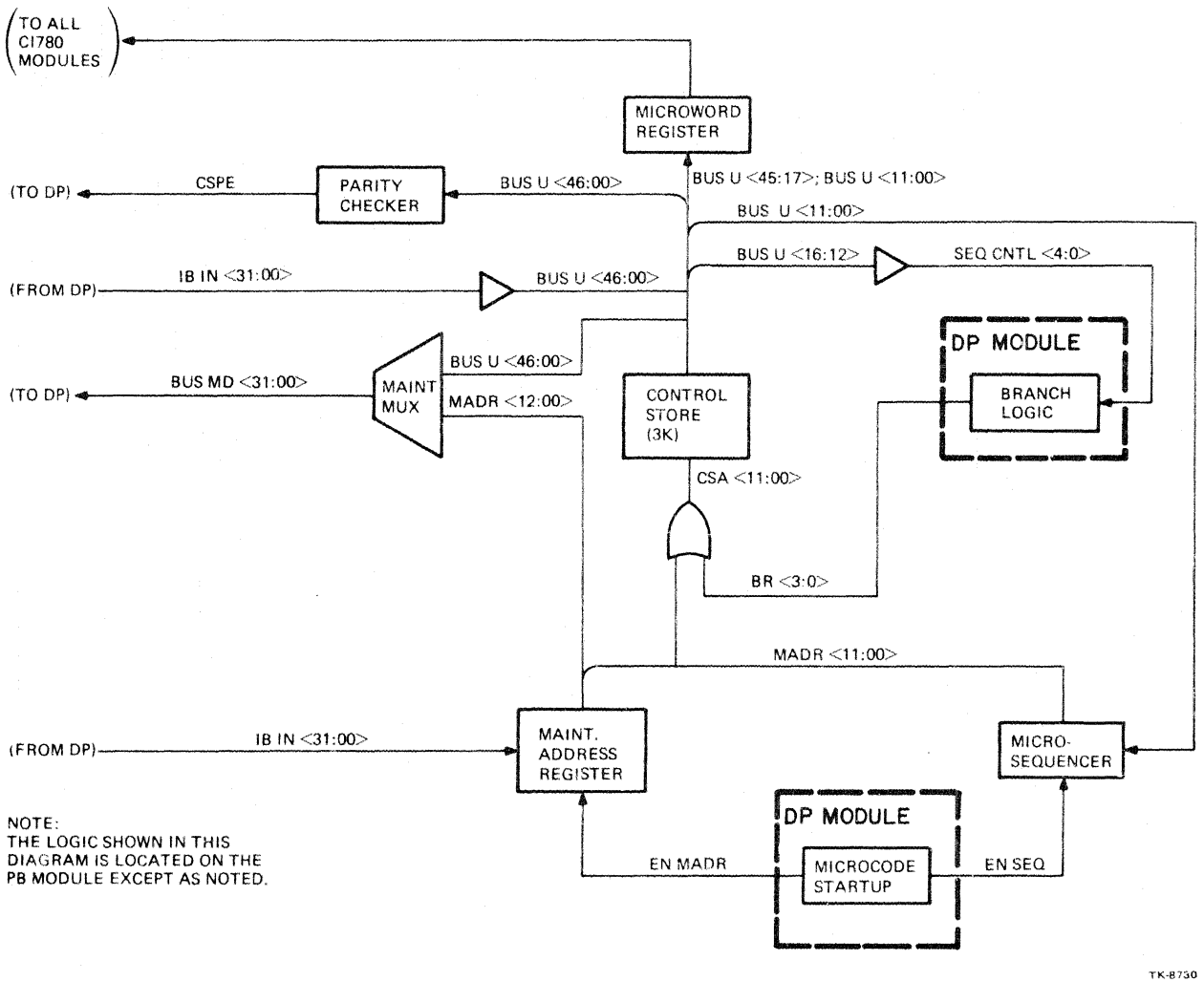


Figure 4-1 Control Store Simplified Block Diagram

The CS is addressed via 12 address bits (CSA ⟨11:00⟩) obtained from either the microsequencer or the maintenance address register. In the uninitialized state (e.g. during power up) the maintenance address register provides the address (MADR ⟨11:00⟩). The register input is IB IN ⟨12:00⟩ from the DP. The microcode start-up logic enables the maintenance address register by asserting EN MADR.

In the initialized state (while the microcode is running) the address is provided by the microsequencer. The microsequencer is enabled by EN SEQ from the microcode start-up logic. The microsequencer uses bits BUS U⟨11:00⟩ from the microword as the base address. Branching logic is used to specify the lower four address bits. The branching conditions are selected by sequential control bits SEQ CNTL ⟨4:0⟩ which are actually bits BUS U⟨16:12⟩ of the microword. The microsequencer contains a memory stack and a PC counter for address control.

The CS microword and the contents of the maintenance address register can be read by the DP via the maintenance mux. The mux selects the lower 32 bits of the microword, the upper bits of the microword, or the 13 bits from the maintenance register for the MD (miscellaneous data) bus to the DP (BUS MD⟨31:00⟩).

Figure 4-2 is a detailed block diagram of the control store area and should be referred to throughout the rest of this chapter.

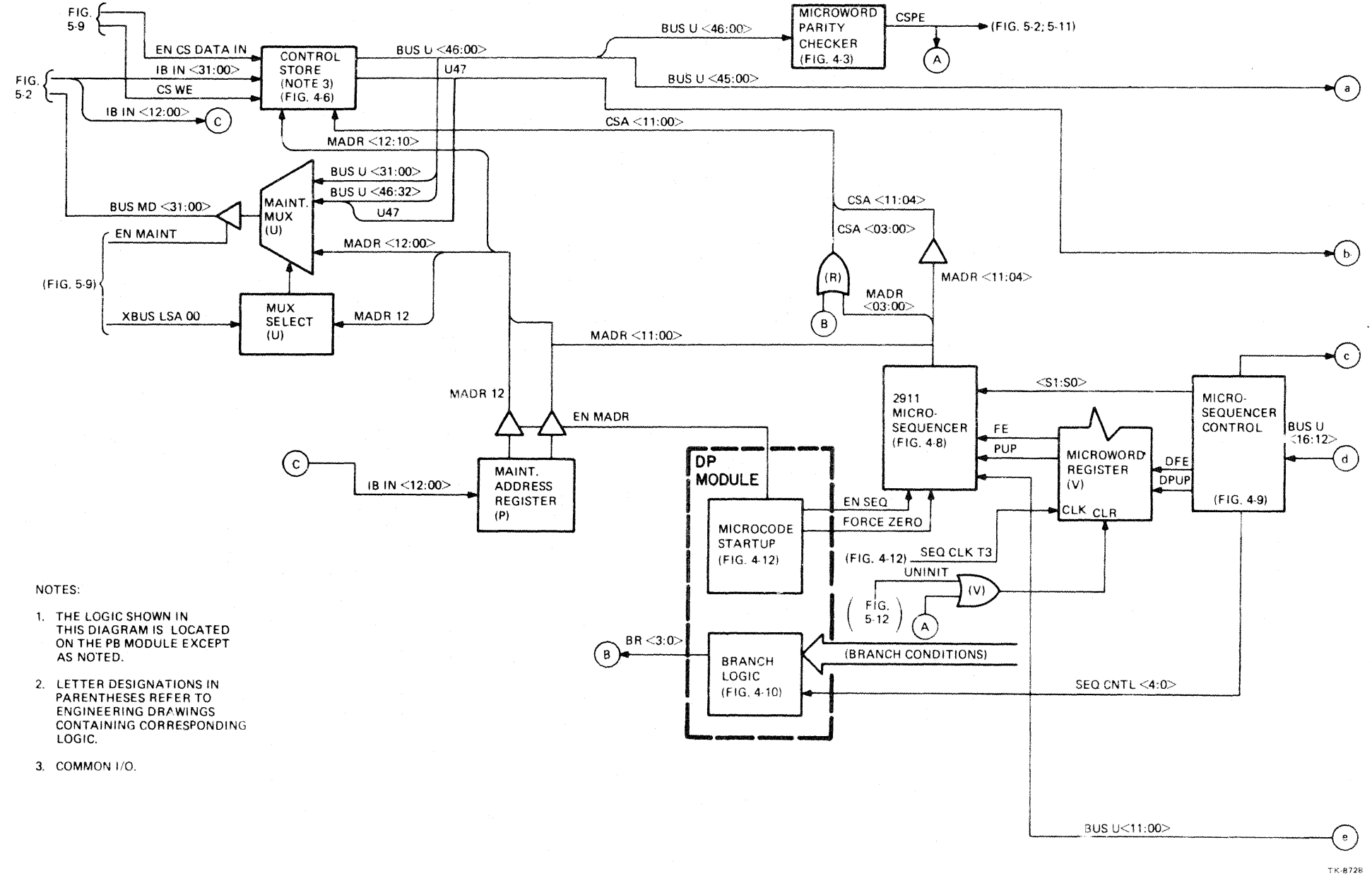
4.2 MICROWORD PARITY

A parity check is made on each microword as it is read out of CS. BUS U⟨46:00⟩ is input to a microword parity checker which outputs CSPE to the DP if a parity error is detected. Bit 46 is the parity bit generating odd parity for each microword. Also, note that a CS parity error resets the microword register containing the microword with the error.

The SYNC bit (U47) is not included in the parity check as it is a programmable bit that can be used with any of the CS microwords, even the microwords in the PROM area whose parity bits cannot be changed.

Figure 4-3 is a block diagram of the parity checker. Each byte of the microword is checked for odd parity in parity generators. Those bytes with an odd number of bits asserted will assert the output of their respective generator. The generator outputs are themselves input into a summation parity generator where again an asserted output means an odd number of asserted inputs. This is a "no error" state which would condition the parity error flip-flop to reset.

If the number of asserted inputs to the summation parity generator is even, the generator output is false and the parity error flip-flop sets on the next SEQ CLK T3 pulse. When the flip-flop sets, CSPE is asserted.



NOTES:

1. THE LOGIC SHOWN IN THIS DIAGRAM IS LOCATED ON THE PB MODULE EXCEPT AS NOTED.
2. LETTER DESIGNATIONS IN PARENTHESES REFER TO ENGINEERING DRAWINGS CONTAINING CORRESPONDING LOGIC.
3. COMMON I/O.

Figure 4-2 Control Store Block Diagram (a)

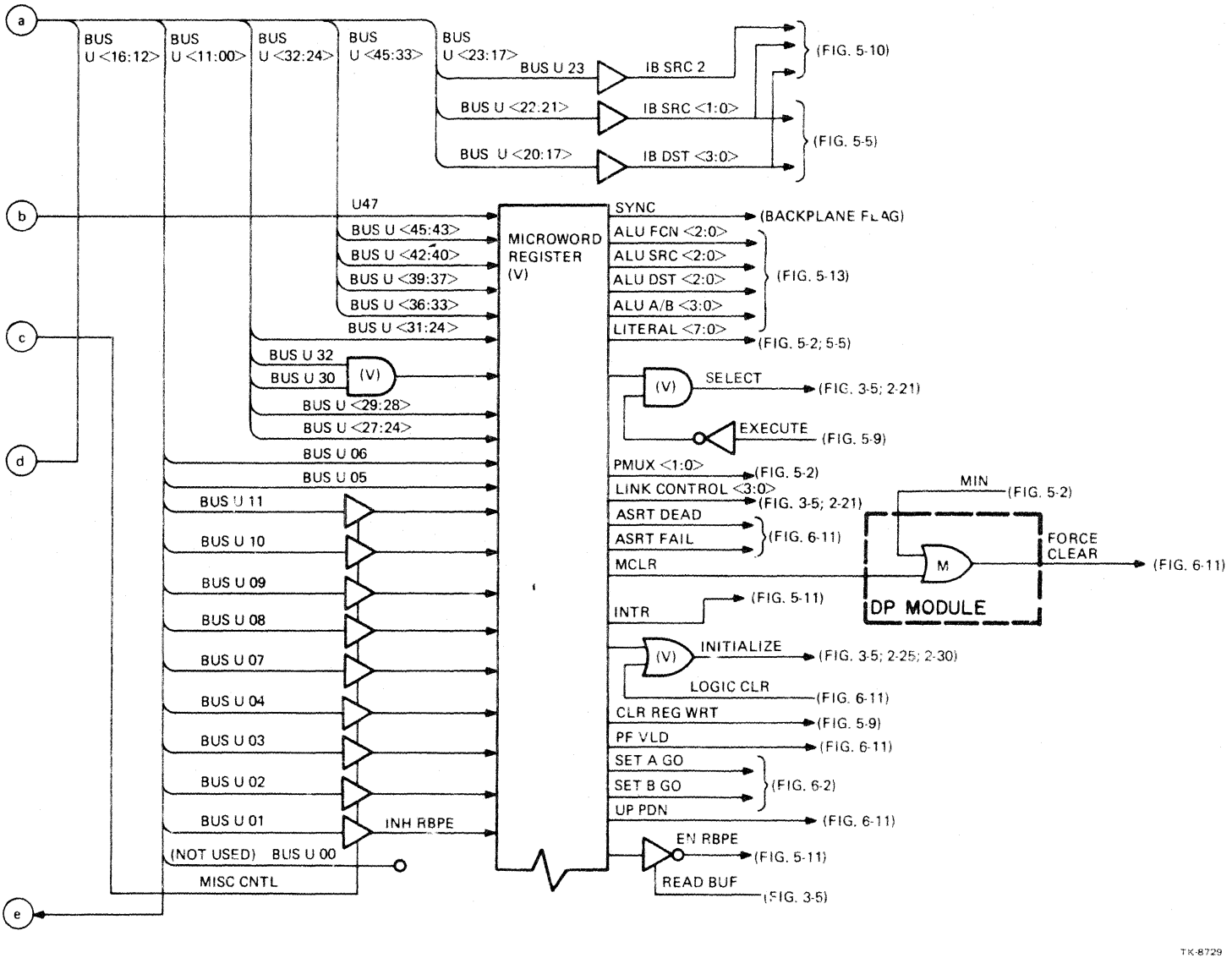


Figure 4-2 Control Store Block Diagram (b)

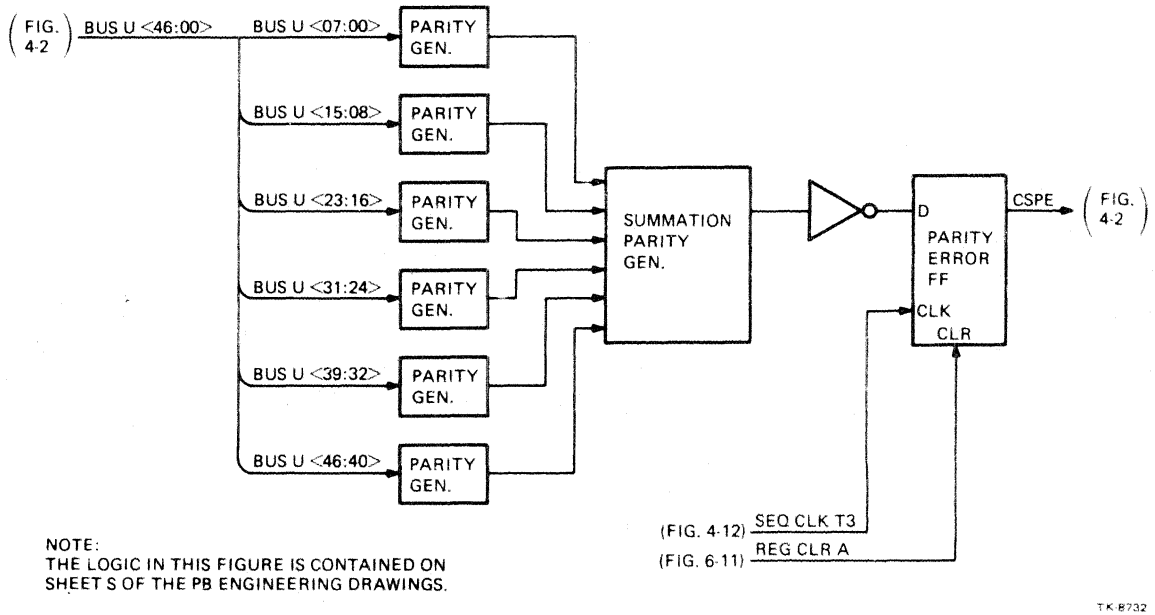


Figure 4-3 Microword Parity Checker

4.3 CS MICROWORD

4.3.1 Microword Fields

The 48 bits of the CS microword are shown in Figure 4-4, grouped by fields. Table 4-1 describes each of the fields shown in the figure.

4.3.2 Microword Register

When a microword is read out of CS, most of the bits are latched into the microword register by SEQ CLK T3. The remaining bit fields are the next address field and the SEQ CNTL field used to select the next microaddress, and the IB SRC and IB DST fields. The IB SRC and IB DST fields must be present in the DP at the start of the microcycle, hence, they cannot wait for SEQ CLK T3 to clock the microword register.

The register is reset in the uninitialized state and whenever the current microword produces a parity error.

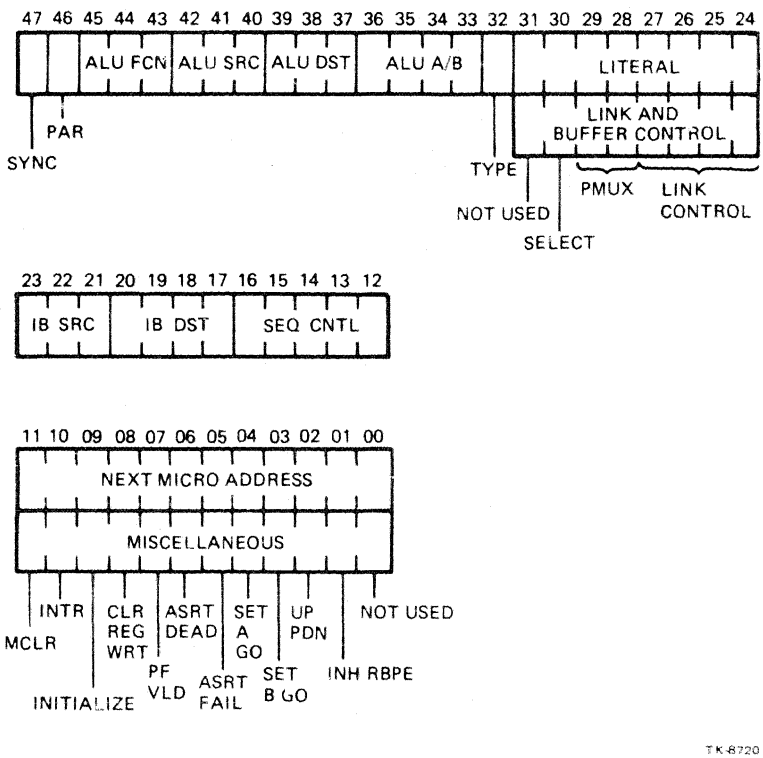


Figure 4-4 Microword Fields

Table 4-1 Microword Fields

Bit	Name	Description
47	SYNC	A programmable bit that is used during port debugging to indicate the execution of a specific microword. The SYNC bit is not included in the parity check of the microword. The SYNC bit can be written in both the RAM and PROM areas of the CS. The bit is available on the port backplane.
46	PAR	The odd parity bit on bits ⟨45:00⟩ of the CS microword.
⟨45:43⟩	ALU FCN ⟨2:0⟩	Function code for the 2901 ALU on the DP.
⟨42:40⟩	ALU SRC ⟨2:0⟩	Operand source code for the 2901 ALU on the DP.
⟨39:37⟩	ALU DST ⟨2:0⟩	Destination code for the 2901 ALU on the DP.
⟨36:33⟩	ALU A/B ⟨3:0⟩	The A and B address lines for the 2901 scratch pads on the DP.
32	TYPE	Selects the definition of bits ⟨31:24⟩ as shown below.
⟨31:24⟩	LITERAL ⟨7:0⟩	Valid when TYPE = 0. Used in the DP as a number or as an address.
⟨31:24⟩	—	Link and PB control bits. Valid when TYPE = 1. The bit fields are defined below.
31	—	Not used.
30	SELECT	Indicates that the LINK CONTROL lines (⟨27:24⟩) are valid.
⟨29:28⟩	PMUX ⟨1:0⟩	Selects a byte in the packet buffer input and output registers on the DP.
⟨27:24⟩	LINK CONTROL ⟨3:0⟩	Specifies operations on the link and PB. This field is valid when SELECT = 1.
⟨23:21⟩*		IB SRC ⟨2:0⟩ Selects the source of BUS IB data in the DP.
⟨20:17⟩*		IB DST ⟨3:0⟩ Selects the destination for BUS IB data in the DP.

* These bits bypass the microword register and go directly to the DP.

Table 4-1 Microword Fields (Cont)

Bit	Name	Description
(16:12)	SEQ CNTL (4:0)	Specifies the operation of the 2911 microsequencer, selects the branch conditions that alter the microaddress, and selects the definition of bits (11:00).
(11:00)	Next microaddress	This field is the base address that is modified by the branch bits to form the address of the next microword. It allows the microcode to jump to any address in the CS. This field is valid so long as the SEQ CNTL field is not all 1s.
(11:00)	MISC CNTL	This field (miscellaneous control) allows the microcode to control miscellaneous flags and functions in the port. The field is valid when the SEQ CNTL field is all 1s. The MISC CNTL bits are described below.
11	MCLR	This bit (maintenance clear) causes the port to enter the uninitialized state.
10	INTR	Sets the interrupt request flag that initiates an interrupt sequence to the host CPU.
09	INITIALIZE	Generates an initialize signal to the link.
08	CLR REG WRT	Clears the REG WRT flag in the DP.
07	PF VLD	When the power-fail valid bit is set, the ASRT DEAD and ASRT FAIL bits are valid.
06	ASRT DEAD	Facilitates processor initialization and booting.
05	ASRT FAIL	Facilitates processor initialization and booting.
04	SET A GO	Starts an external bus transfer with the host using the A parameters.
03	SET B GO	Starts an external bus transfer with the host using the B parameters.
02	UP PDN	Allows the microcode to set the PDN (power down) bit in the port configuration register.
01	INH RBPE	This bit is set during a DP read of the first byte from a packet buffer. The first byte read is always undefined data. INH RBPE prevents a parity error from asserting on the undefined data.
00	—	Not used.

4.4 MAINTENANCE MUX

During the uninitialized state the CS can be read by the DP for maintenance purposes. The CS microword is input to the DP via a maintenance mux and a 32-bit miscellaneous data bus (BUS MD(31:00)). The microword is applied to the maintenance mux where the mux first selects the lower 32 bits (BUS U(31:00)) for the MD bus, and then the upper 16 bits (BUS U(46:32); U47).

The DP can also read the 13 bits from the maintenance address register (MADR (12:00)) via the maintenance mux.

Mux selection is accomplished using one of the local store address bits from the DP (XBUS LSA 00) and MADR 12 from the maintenence address register. XBUS LSA 00 selects either the microword or the maintenance address. MADR 12 is used here and throughout the CS logic to select the upper or lower portion of the microword. MADR 12 false selects the lower portion (BUS U(31:00)). MADR 12 true selects the upper portion (BUS U(46:32); U47). Table 4-2 lists the mux selection code.

Table 4-2 Maintenance Mux Selection Code

XBUS LSA 00	MADR 12	BUS MD(31:00)
0	0	BUS U(31:00)
0	1	BUS U(46:32); U47
1	X	MADR (12:00)

0 = negated
1 = asserted
X = don't care

4.5 CONTROL STORE SPACE AND LOGIC

4.5.1 Control Store Space

The control store space (Figure 4-5) has a microword store area and a flag store area. The microword store area consists of 1K × 47 of PROM and 2K × 47 of RAM. The area is addressed by 12 control store address bits CSA (11:00). The two most significant address bits (CSA (11:10)) divide the store area into three banks and are used as the bank select bits. Bits CSA (09:00) address the 1024 (1K) word locations within each bank.

The flag store area is 3K × 1 of RAM used to store the programmable SYNC bit (U47). CSA (11:00) also addresses the 3K of flag storage thus giving a SYNC bit location in the flag store area for each word location in the microword store area. The SYNC bit can be written anywhere across the address spectrum; thus, even the microwords in the PROM area (bank 0) could have a SYNC bit written in as bit 47.

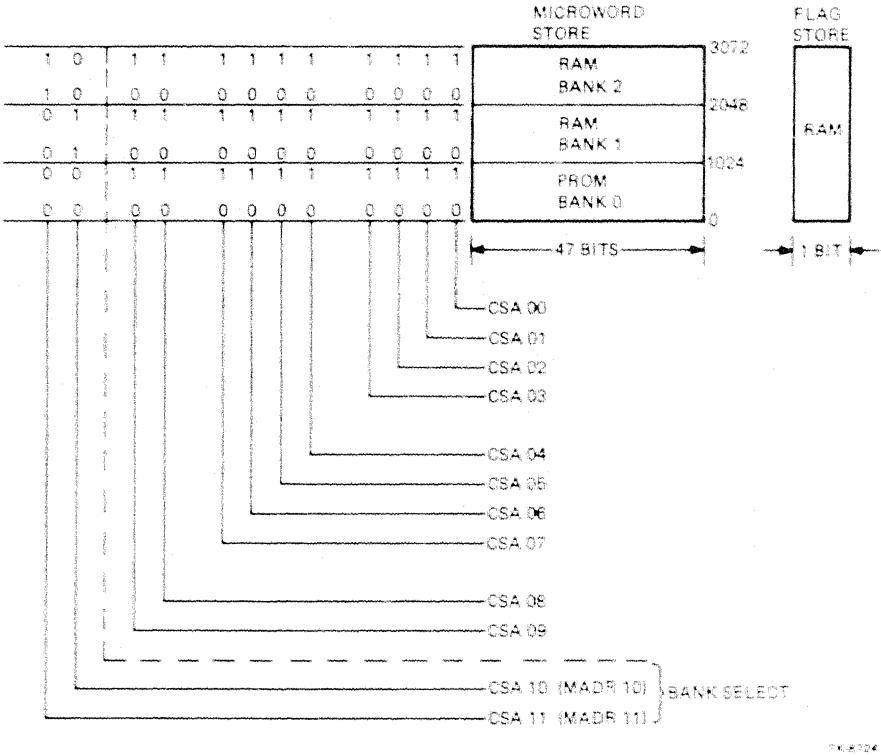


Figure 4-5 Control Store Space

4.5.2 Control Store Logic

Figure 4-6 is a block diagram of the control store logic. Bank 0 is comprised of six 1K × 8 PROMs. Each PROM outputs eight bits onto the microword I/O bus (BUS U⟨46:00⟩). The high-order PROM outputs only seven bits (BUS U⟨46:40⟩). Banks 1 and 2 are each made up of twelve 1K × 4 RAMs. Each RAM has a four-bit I/O to the microword bus. The high-order RAM in each bank uses only three of its four I/O lines (BUS U⟨46:44⟩).

Bits MADR ⟨11:10⟩ (identical to CSA ⟨11:10⟩ shown in Figure 4-5) are the bank select bits. They are applied to bank select logic where they are decoded to output one of three SEL BANK enabling signals. When true, each SEL BANK signal enables all the RAMs (or PROMs) in its respective bank. Address bits CSA ⟨09:00⟩ are applied to all the RAMs and PROMs; however, only the RAMs (or PROMs) in the enabled bank will respond to the address. The address bits select a location in each of the RAMs (or PROMs) of the selected bank.

All 47 bits from the addressed location in the selected bank are available on the microword bus for reading except during a CS write operation. All 47 bits are read simultaneously.

The two writable CS banks are divided into three parts of four RAMs each. The parts are 16 bits each and are designated as LO (BUS U⟨15:00⟩), MID (BUS U⟨31:16⟩), and HI (BUS U⟨46:32⟩). Each part receives a separate write enable signal.

To write the CS RAMs, the signal CS WE is asserted from the DP and then ANDed with MADR 12. MADR false asserts WR CS LO and WR CS MID thus enabling the LO and MID parts for a write. MADR 12 true asserts WR CS HI, enabling the HI part for a write.

Write data (IB IN⟨31:00⟩) and a data in enabling signal (EN CS DATA IN) is received from the DP. MADR 12 is ANDed with EN CS DATA IN to again select the high or low portion of the microword. When MADR 12 is false, IB IN⟨31:00⟩ is coupled to BUS U⟨31:00⟩ and written into the LO and MID parts of the selected RAM bank. When MADR 12 is true, IB IN⟨14:00⟩ is coupled to BUS U⟨46:32⟩ and written into the HI part of the selected RAM bank.

The flag store RAM is addressed by CSA⟨11:00⟩ to select bit 47 of the microword being addressed in the microword store area. The flag store output (U47) is available on the microword bus for reading except during a CS write operation. Bit U47 is read out along with its associated microword.

The flag store is written as bit 47 of the input microword. The input to the flag store RAM is IB IN 15. The flag store is enabled by WR CS HI. Thus, the flag is written when IB IN ⟨14:00⟩ is being coupled to BUS U ⟨46:00⟩ and the upper portion of the microword is being written.

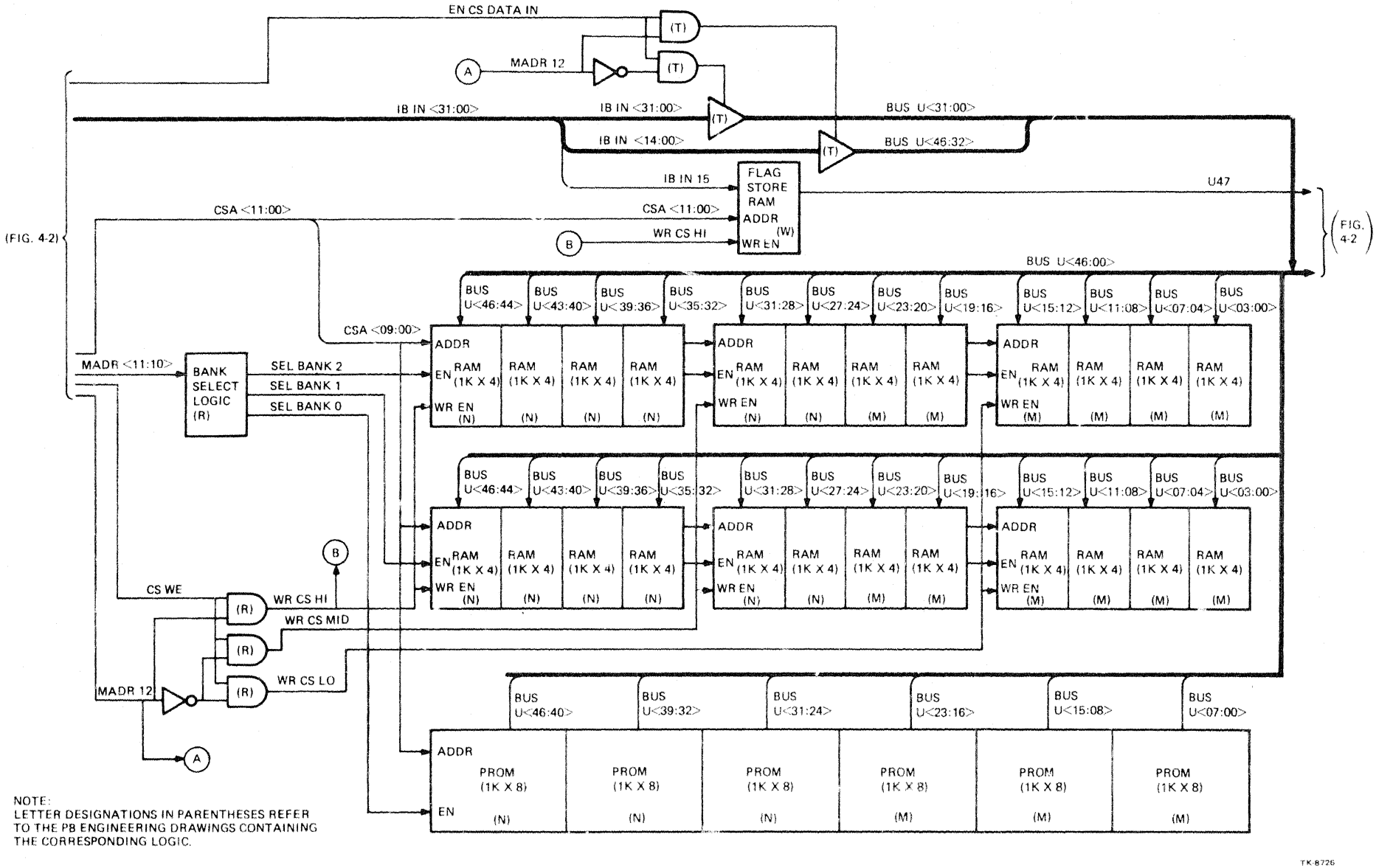


Figure 4-6 Control Store Logic

4.6 CONTROL STORE ADDRESS SOURCE

The CS addressing bits (CSA (11:00)) are obtained from either the maintenance address register or the microsequencer logic. The selection is made by the microcode start-up logic which asserts EN MADR to enable the output from the maintenance address register, or EN SEQ to enable the 2911 microsequencer.

4.6.1 Maintenance Address Register

The maintenance address register has 13 bits and receives IB IN (12:00) from the DP. When enabled the register outputs MADR (12:00). All 13 bits are applied to the maintenance mux for read back into the DP over the MD bus. MADR 12 is used in the mux select logic to select the high or low portion of the microword for the MD bus. MADR 12 is also used in the CS logic to select the high or low portion of the microword to be written from the IB bus. MADR (11:10) is used in the CS logic for CS bank selection.

MADR (11:00) is muxed onto common lines with the 12-bit output from the 2911 microsequencer.

4.6.2 Microsequencer Logic

The microsequencer logic consists of the 2911 microsequencer, the microsequencer control logic which regulates and controls the various microsequencing functions, and the branch logic.

4.6.2.1 2911 Microsequencer – The 2911 microsequencer outputs a 12-bit address onto common address lines MADR (11:00) where it is muxed with the 12-bit output from the maintenance address register. Figure 4-7 illustrates the muxing function. Also note that the microsequencer comprises three 2911 chips, each outputting four bits onto the MADR lines. The upper eight bits on the MADR lines (MADR (11:04)) become address bits CSA (11:04), respectively. The lower four bits (MADR (03:00)) are ORed with branch bits BR (03:00) from the branch logic in the DP to produce address bits CSA (03:00).

The lower 12 bits of the CS microword (BUS U(11:00)) are used by the microsequencer to formulate the next address. Each chip receives the four bits from the microword that correspond to its four outputs onto the MADR lines.

Figure 4-8 is a functional block diagram of a 2911 microsequencer chip. The source of the four-bit chip output could be an address register (which would be the four next address bits from the microword), a 4 × 4 memory stack, or a PC counter/incrementer. A mux selects the address source according to select code (S1:S0) from the microsequencer control logic.

The memory stack is enabled by file enable (FE) received from the CS microword via the microword register. The stack push/pop control (PUP) is also obtained from the microword via the microword register.

FORCE ZERO from the microcode start-up logic negates the microsequencer output causing the output to be all zeros.

4.6.2.2 Microsequencer Control Logic – Figure 4-9 is a block diagram of the microsequencer control logic. BUS U (16:12) is the microsequencer control field in the CS microword. The field specifies how the next CS address is formulated.

BUS U (16:12) becomes SEQ CNTL (4:0) respectively within the control logic.

If SEQ CNTL 4, SEQ CNTL 3, or SEQ CNTL 2 is false, control bits SEQ CNTL (1:0) are inhibited from the mux select logic and the stack enable logic. In this case, the mux select logic output defaults to S1 false (0) and S0 true (1), and decoded file enable (DFE) from the stack enable logic is negated. The push/pop stack control logic responds to SEQ CNTL (1:0); however, decoded push/pop (DPUP) has no effect, while the stack file enable signal (DFE) is false.

SEQ CNTL (4:0) also goes to the branch logic to control the branching function.

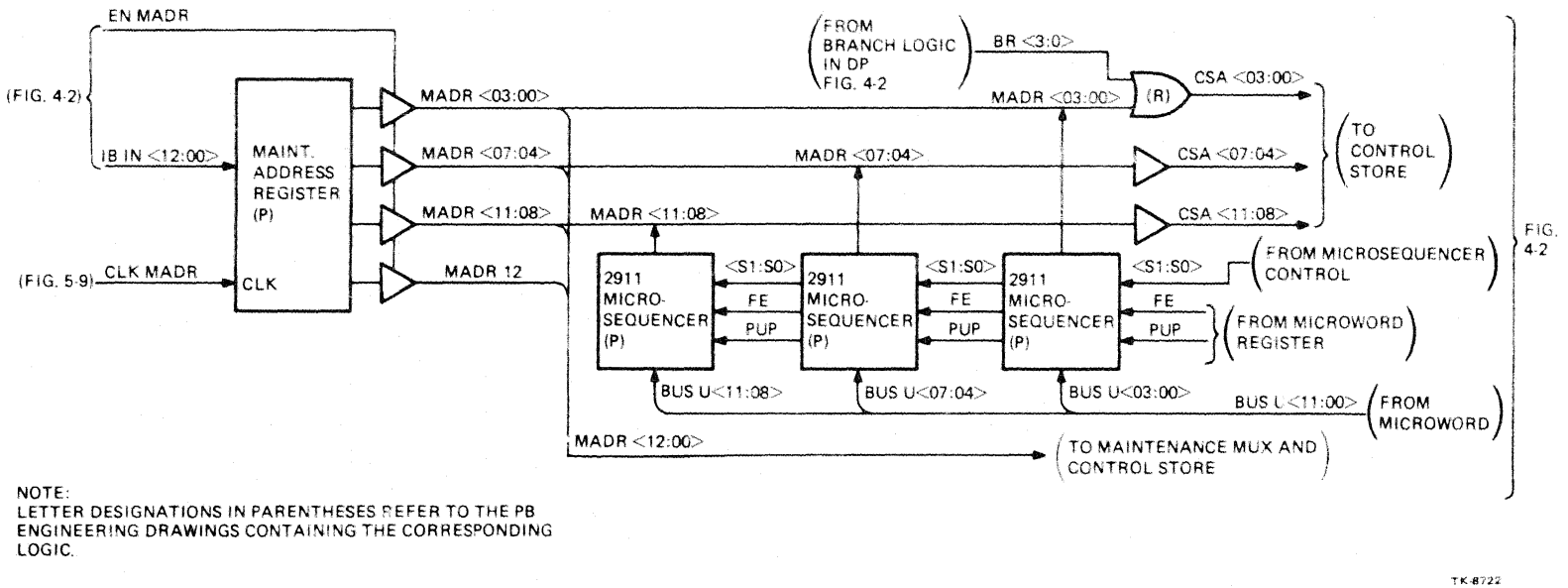
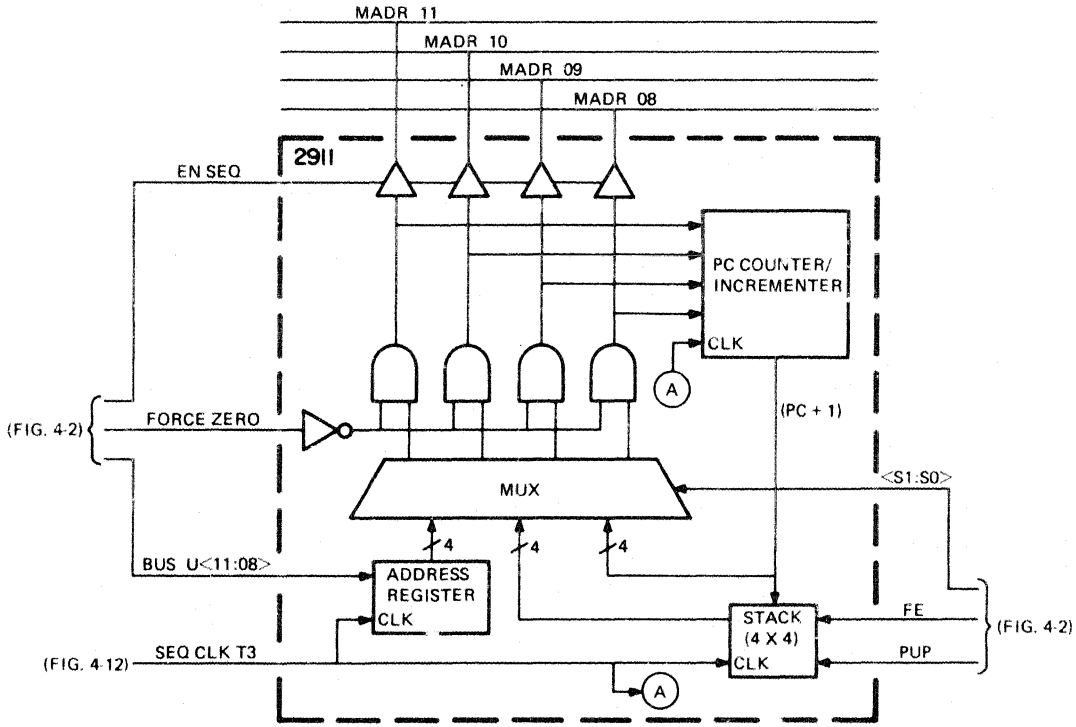


Figure 4-7 Control Store Address Multiplexing



NOTE:
THE LOGIC IN THIS FIGURE IS CONTAINED ON
SHEET P OF THE PB ENGINEERING DRAWINGS.

TK-8723

Figure 4-8 2911 Microsequencer

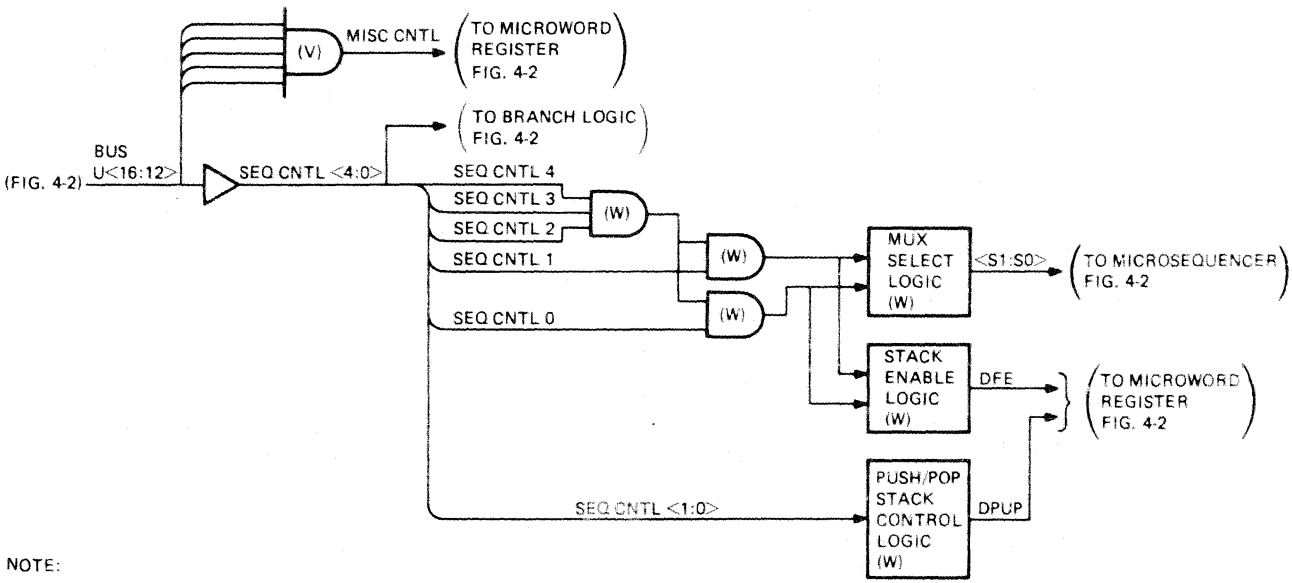


Figure 4-9 Microsequencer Control Logic

Table 4-3 lists the five SEQ CNTL bits in binary sequence and shows how the bits control the various sequencing functions. All 32 bit counts (or bit states) are listed.

The first 28 counts (bit states) operate the branch logic. During the first four bit states, branches 3, 2, 1, and the A portion of branch 0 are enabled. During the next four bit states, only branch 1 and the A portion of branch 0 are enabled. For the next eight states, the B portion of branch 0 is enabled. The next eight states find the C portion of branch 0 enabled. The last four bit states of branch logic operation has select condition code (SEL CC) asserted. SEL CC is actually the D portion of branch 0. The branch logic is described in Paragraph 4.6.2.3.

Note that during the 28 bit states of branch logic operation, either SEQ CNTL 4, SEQ CNTL 3, or SEQ CNTL 2 is false, hence the S1 and S0 control bits from the mux select logic are in the default state (S1 = 0; S0 = 1) and the DFE signal from the stack enable logic is false. With the control bits in the default state, the microsequencer mux selects the address register and the microsequencer serves only to couple the microword next address field (BUS U ⟨11:00⟩) to the MADR ⟨11:00⟩ common address lines as the base address for branching operations. The stack is disabled by the negated state of DFE during branching operations, hence, the state of DPUP is meaningless.

During the last four bit states, the SEQ CNTL ⟨4:2⟩ bits are true, disabling the branch logic and causing the microsequencer to be used as the addressing control. As shown in Figure 4-9, SEQ CNTL ⟨1:0⟩ are now input to the mux select logic and the stack enable logic. Table 4-3 shows the state of the S1,S0 control bits and the stack enabling signal (DFE) for the last four bit states.

The first of the four bit states is a (jump to subroutine) JSR function. In this state SEQ CNTL ⟨1:0⟩ are both 0 hence S1 and S0 remain in their default state and the address register is still selected; however, now the stack is enabled and DPUP is asserted. DPUP true causes the output of the PC counter/incrementer (PC + 1) to be pushed onto the stack. The microcode jumps to the address of a subroutine but saves the next address (PC + 1) to return to the main flow after the subroutine is finished.

The second state is a return from subroutine (RTS) function. In this state the mux selects the stack for the next address. The stack is enabled and DPUP is false which pops the stored address from the stack to the mux. The microcode, returning from a subroutine, uses the address stored on the stack to return to the main flow.

The third state is a "pop the stack" housecleaning function. In this state the mux selects the PC counter/incrementer for the next address, hence the microcode simply advances to the next address in the main flow. The stack is enabled and DPUP is false which pops the stack of an unwanted address. Clearing the stack in this manner is necessary when the microcode jumps to a subroutine and continues on from the subroutine without returning to the main flow via an RTS.

The fourth state is the MISC CNTL function. In this state the mux again selects the PC counter/incrementer for the next address and the microcode advances to the next address in the main flow. The stack is disabled by the negated state of DFE. The MISC CNTL function is the utilization of the next address field of the microword (BUS U ⟨11:00⟩) for one microcycle for miscellaneous flags and control functions. In this state, the sequential control bits (SEQ CNTL ⟨4:0⟩) are all 1s, hence BUS U ⟨16:12⟩ are all 1s and MISC CNTL is asserted (Figure 4-9). MISC CNTL gates the microword next address field (now carrying the miscellaneous flags and controls) into the microword register (Paragraph 4.3).*

* Bits 5 and 6 of the next address field (ASRT FAIL and ASRT DEAD) are not gated directly by MISC CNTL. However, they are indirectly gated by MISC CNTL because they are subsequently gated by PF VLD.

Table 4-3 Microsequencer Control Functions

Bit State	SEQ CNTL 4 3 2 1 0	EN BR 2/3	EN BR 0A/1	EN BR 0B	EN BR 0C	SEL CC	Microsequencer Mux Select Code ⟨S1:S0⟩	Microsequencer Address Source	Stack Enable (DFE)	Push/ Pop (DPUP)
1	0 0 0 0 0						0 1	Address Register	0	X
2	0 0 0 0 1									
3	0 0 0 1 0									
4	0 0 0 1 1									
5	0 0 1 0 0									
6	0 0 1 0 1									
7	0 0 1 1 0									
8	0 0 1 1 1									
9	0 1 0 0 0									
10	0 1 0 0 1									
11	0 1 0 1 0									
12	0 1 0 1 1									
13	0 1 1 0 0									
14	0 1 1 0 1									
15	0 1 1 1 0									
16	0 1 1 1 1									
17	1 0 0 0 0									
18	1 0 0 0 1									
19	1 0 0 1 0									
20	1 0 0 1 1									
21	1 0 1 0 0									
22	1 0 1 0 1									
23	1 0 1 1 0									
24	1 0 1 1 1									
25	1 1 0 0 0									
26	1 1 0 0 1									
27	1 1 0 1 0									
28	1 1 0 1 1						0 1	Address Register	0	X
29	1 1 1 0 0						0 1	Address Register	1	1
30	1 1 1 0 1						1 0	Stack	1	0
31	1 1 1 1 0						0 0	PC Counter/ Incrementer	1	0
32	1 1 1 1 1						0 0	PC Counter/ Incrementer	0	1

1 = Asserted
0 = Negated
X = Don't care

In describing the 32 states of sequential control bits SEQ CNTL ⟨4:0⟩, four special microsequencer states and 28 branch states were discussed. It may have been noticed that there appeared to be no state that used the next address field of the microword unchanged. As will be seen in the section on branching, (Paragraph 4.6.2.3), one of the branching states is a null wherein no conditions are checked. This allows the next address field to pass to the CS unchanged.

4.6.2.3 Branch Logic – Figure 4-10 is a block diagram of the branch logic. Four branch bits (BR ⟨3:0⟩) are generated by the branch logic to modify the base address from the 2911 microsequencer. Branch bits BR ⟨3:1⟩ each have a mux for selecting the various conditions affecting that branch. Branch bit BR 0 has four muxes to select its branch conditions.

The branch muxes are controlled by sequential control bits SEQ CNTL ⟨4:0⟩. The muxes function during 28 of the 32 bit states of SEQ CNTL ⟨4:0⟩ as shown in Table 4-3; however, not all the muxes are enabled during all of these states. When a branch mux is not enabled, the associated addressing bit is determined by the corresponding bit from the microsequencer.

Control bits SEQ CNTL ⟨4:3⟩ are applied to the branch 0 mux select logic. The control bits are decoded to assert one of four outputs to enable one of the branch 0 muxes. The control bits divide the 32 bit states into groups of eight. Table 4-3 illustrates this and also shows the state of the four outputs from the branch 0 mux select logic for the eight-bit groups.

EN BR 0A/1 enables the branch 1 mux and the A mux of branch 0. It is asserted for the eight bit states that SEQ CNTL ⟨4:3⟩ are false.

EN BR 0A/1 is ANDed with the negated state of SEQ CNTL 2 to assert EN BR 2/3. EN BR 2/3 enables the branch 2 mux and the branch 3 mux. Making EN BR 2/3 a function of SEQ CNTL 2 limits the enabled state of the branch 2 mux and the branch 3 mux to only four bit states.

EN BR 0B enables the B mux of branch 0 for the eight bit states that SEQ CNTL ⟨4:3⟩ are 0 and 1, respectively.

EN BR 0C enables the C mux of branch 0 for the eight bit states that SEQ CNTL ⟨4:3⟩ are 1 and 0, respectively.

The fourth output from the branch 0 mux select logic is asserted by the 1:1 state of SEQ CNTL ⟨4:3⟩. It is ANDed with the negated state of SEQ CNTL 2 to produce SEL CC. SEL CC selects the branch conditions of the branch 0 "D" mux. Making SEL CC a function of SEQ CNTL 2 limits the asserted state of SEL CC to only four bit states.

Table 4-4 lists the branching conditions for the 32 bit states of SEQ CNTL ⟨4:0⟩. Refer to it during the following discussion of the branch muxes. When a condition is sampled by the branch logic, the corresponding bit from the microsequencer is always 0.

The branch 3 mux is enabled for the first four bit states. The mux selects IB IN 19 when SEQ CNTL ⟨1:0⟩ are in the 1:1 state. The mux selects 0 (ground) for the other three states of SEQ CNTL ⟨1:0⟩. The mux output routes to the branch output register and then to the BR 3 output line.

The branch 2 mux is also enabled for the first four bit states. The mux selects one of four condition inputs as determined by SEQ CNTL ⟨1:0⟩. The BUS ERR condition (negated) is selected for both the 0:0 and the 0:1 states of SEQ CNTL ⟨1:0⟩. The mux output is placed on the BR 2 output line via the branch output register.

The branch 1 mux is enabled for the first eight bit states. The mux selects one of eight condition inputs as determined by SEQ CNTL ⟨2:0⟩. The mux output is placed on the BR 1 output line via the branch output register.

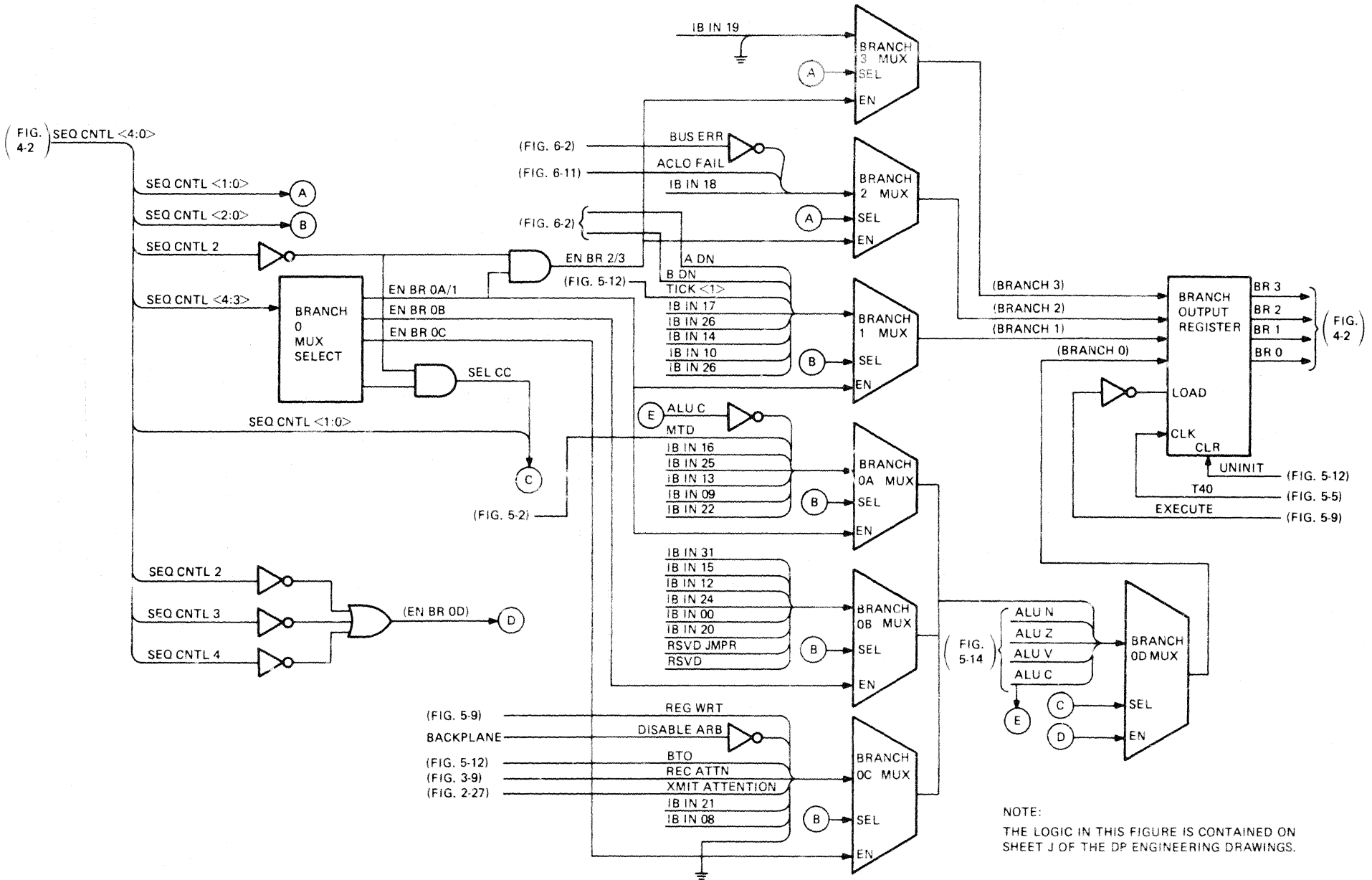


Figure 4-10 Branch Logic

Table 4-4 Branch Conditions

Bit State	SEQ CNTL ⟨4:0⟩	Function	Branch 3	Branch 2	Branch 1	Branch 0
1	00000	Branch	0	$\overline{\text{BUS ERR}}$	A DN	$\overline{\text{ALU C}}$
2	00001	"	0	$\overline{\text{BUS ERR}}$	B DN	$\overline{\text{ALU C}}$
3	00010	"	0	ACLO FAIL	TICK ⟨1⟩	MTD
4	00011	"	IB IN 19	IB IN 18	IB IN 26	IB IN 16
5	00100	"	0	0	IB IN 17	IB IN 25
6	00101	"	0	0	IB IN 14	IB IN 13
7	00110	"	0	0	IB IN 10	IB IN 09
8	00111	"	0	0	IB IN 26	IB IN 22
9	01000	"	0	0	0	IB IN 31
10	01001	"	0	0	0	IB IN 15
11	01010	"	0	0	0	IB IN 12
12	01011	"	0	0	0	IB IN 24
13	01100	"	0	0	0	IB IN 00
14	01101	"	0	0	0	IB IN 20
15	01110	"	0	0	0	RSVD JMPR
16	01111	"	0	0	0	RSVD
17	10000	"	0	0	0	$\overline{\text{REG WRT}}$
18	10001	"	0	0	0	$\overline{\text{DISABLE ARB}}$
19	10010	"	0	0	0	BTO
20	10011	"	0	0	0	REC ATTN
21	10100	"	0	0	0	XMIT ATTENTION
22	10101	"	0	0	0	IB IN 21
23	10110	"	0	0	0	IB IN 08
24	10111	"	0	0	0	0
25	11000	"	0	0	0	ALU N
26	11001	"	0	0	0	ALU C
27	11010	"	0	0	0	ALU V
28	11011	"	0	0	0	ALU Z
29	11100	JSR	0	0	0	0
30	11101	RTS	0	0	0	0
31	11110	POP STACK	0	0	0	0
32	11111	MISC CNTL	0	0	0	0

The A, B, and C mux of branch 0 have their outputs connected to a common output line. Mux A is enabled for the first group of eight bit states, mux B for the second group, and mux C for the third group. The enabled mux selects one of eight condition inputs as determined by SEQ CNTL ⟨2:0⟩. Thus, the common mux output line receives a branch condition for the first 24 bit states.

Note that one of the branch condition inputs of mux C is 0 (ground). When this condition is selected (bit state 24), there are no branch conditions and the next address from the 2911 microsequencer is applied to the CS unchanged.

The branch condition on the common output line is applied to the four low order inputs of the branch 0 "D" mux. The three select bits for the D mux are SEL CC and SEQ ⟨1:0⟩ with SEL CC being the most significant bit. SEL CC is false for the first 24 bit states (Table 4-3) hence the mux selects only from the four low order inputs. Thus, for the first 24 bit states, the D mux simply couples the selected branch condition from the common line to the BR 0 output line via the branch output register. SEL CC is true for the next four bit states (states 25 through 28), causing SEQ CNTL ⟨1:0⟩ to select from the four high order inputs (ALU functions).

Branch 0 is active for all 28 bit states of branch operations. Also it can be seen that the branch D mux is enabled for all 28 states. It is disabled during states 29 through 32 (SEQ CNTL ⟨4:2⟩ all 1s) when the microsequencer special functions are enabled.

4.7 MICROCODE START-UP

The two CS address sources (the maintenance address register and the microsequencer) are enabled from the microcode start-up logic. EN MADR enables the maintenance address register during the uninitialized state. When the initialization process is complete, EN MADR negates and EN SEQ asserts. EN SEQ enables the microsequencer which supplies the CS address during the initialized state.

Figure 4-11 is a flow diagram of the microcode start-up process. The following discussion follows the sequence illustrated in the diagram. Figure 4-12 is a block diagram of the logic involved in the start-up process.

Upon system start-up, UNINIT asserts on the DP and places the port into the uninitialized state. When UNINIT asserts, EN MADR asserts to the maintenance address register and EN SEQ is negated. Also FORCE ZERO is asserted to the microsequencer in preparation for when the microsequencer will take over the addressing function. In addition, UNINIT asserts SUSPEND CLK, inhibiting the microsequencer clock (SEQ CLK T3).

When initialization is completed, the DP negates UNINIT and the port goes from the uninitialized to the initialized state. The negation of UNINIT negates SUSPEND CLK thereby enabling SEQ CLK T3 to the FORCE ZERO flip-flop and to the microsequencer.

The CS address source for the first microcycle of the initialized state may not be the microsequencer depending on the state of the programmable starting address (PSA) bit in the port maintenance control/status register (PMCSR). During a normal start-up, PSA = 0. In this case, the negation of UNINIT directly negates EN MADR which in turn directly asserts EN SEQ. The enabled microsequencer then responds to the true state of FORCE ZERO and outputs a starting address of 0 to the CS. The next SEQ CLK T3 pulse resets the FORCE ZERO flip-flop allowing the microsequencer to respond to the microcode in the CS.

If, while in the uninitialized state, it is determined that a diagnostic routine should be run, the PSA bit is set to 1. With PSA = 1, the negation of UNINIT does not cause EN MADR to negate (and thus EN SEQ to assert) until the next DP CLK T3 pulse resets the enable flip-flop. Thus, for the first microcycle of the initialized state, the maintenance address register still provides the CS address. The address provided would be the starting address of the desired diagnostic routine.

When the DP CLK T3 pulse occurs, EN MADR negates, EN SEQ asserts and the CS address source shifts from the maintenance address register to the microsequencer. The same DP CLK T3 pulse resets the FORCE ZERO flip-flop to allow the microsequencer to respond to the next address field of the first microword of the diagnostic routine.

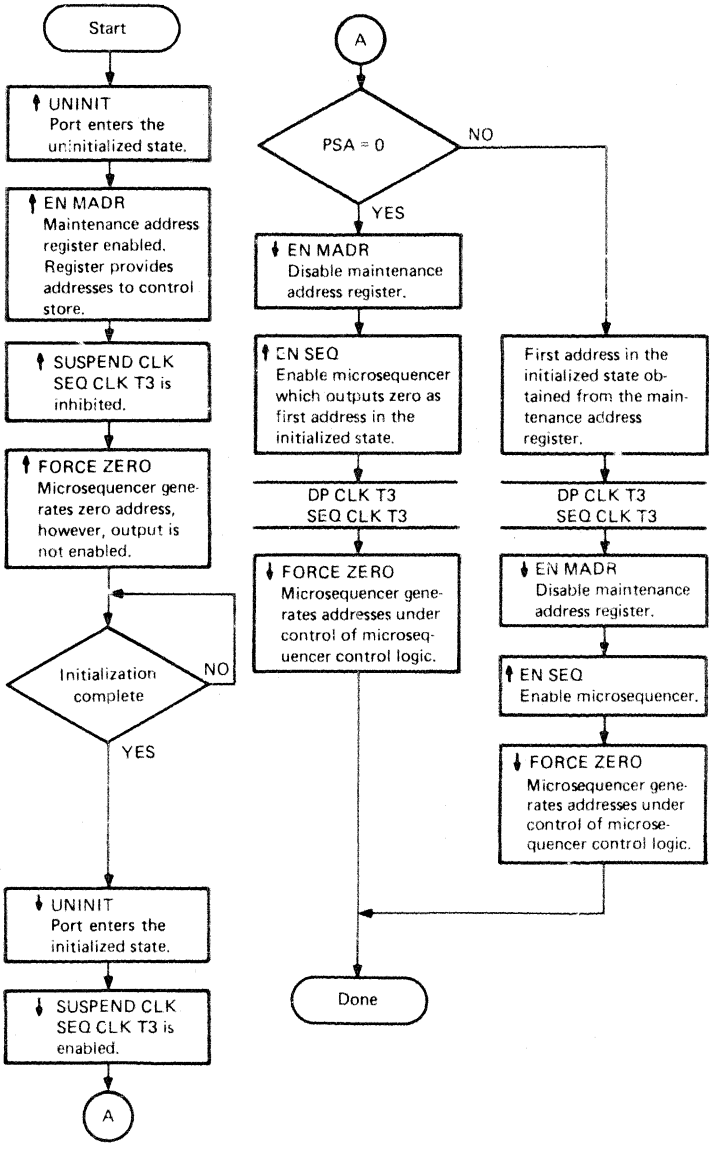
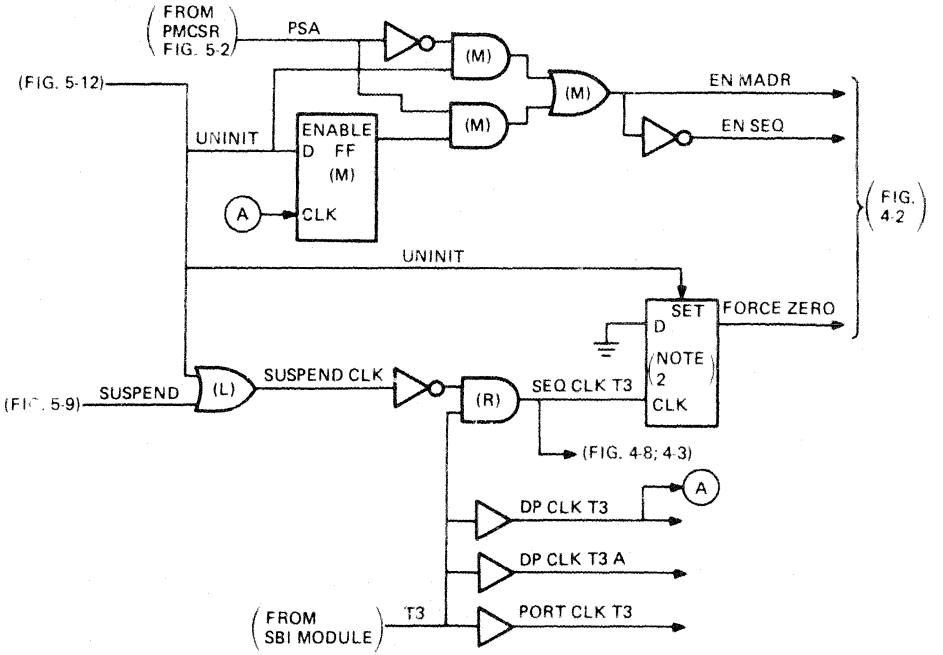


Figure 4-11 Microcode Start-Up Flow Diagram



NOTES:

1. LETTER DESIGNATIONS IN PARENTHESES REFER TO THE DP ENGINEERING DRAWINGS CONTAINING THE CORRESPONDING LOGIC.
2. LOCATED ON SHEET W OF PB ENGINEERING DRAWINGS.

TK-8727

Figure 4-12 Microcode Start-Up Logic

Figure 4-13 is a timing diagram of the microcode start-up. Figure 4-13A illustrates the start-up timing when the PSA bit = 0. Figure 4-13B illustrates the start-up timing when the PSA bit = 1. Note that the difference between the two timing sequences is the point at which EN MADR negates (and EN SEQ asserts) and what causes it to negate.

4.8 CLOCKS

Figure 4-12 illustrates the generation of the DP clocks. The DP clocks (DP CLK T3, DP CLK T3A, and PORT CLK T3) are all derived from T3 as shown in the figure. The time period for the clocks is 200 ns (see Figure 4-13).

SEQ CLK T3 is also derived from T3; however, it is subject to the negated state of SUSPEND CLK.

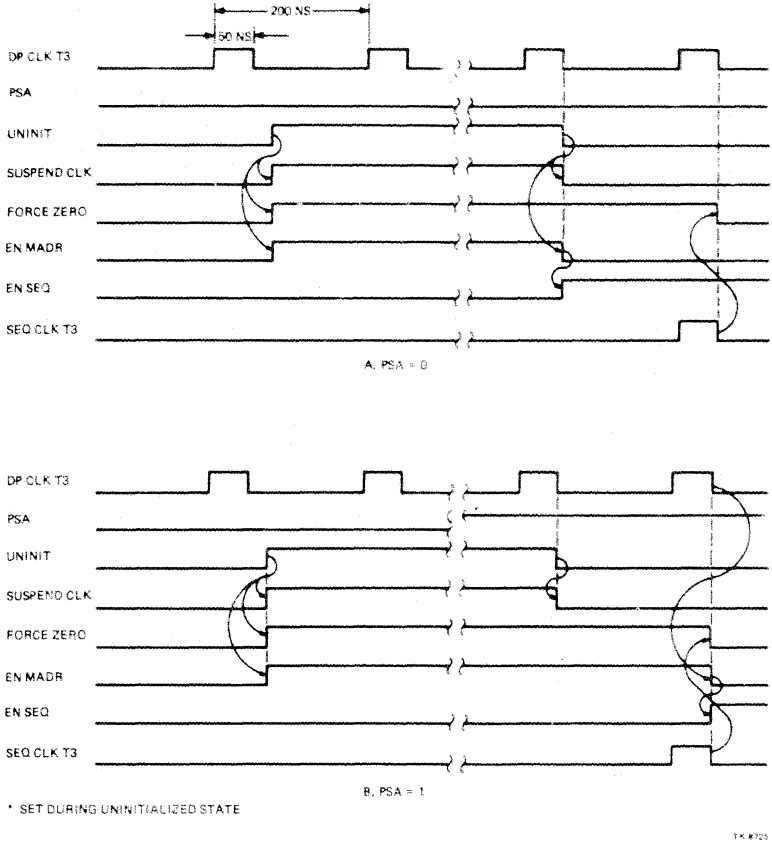
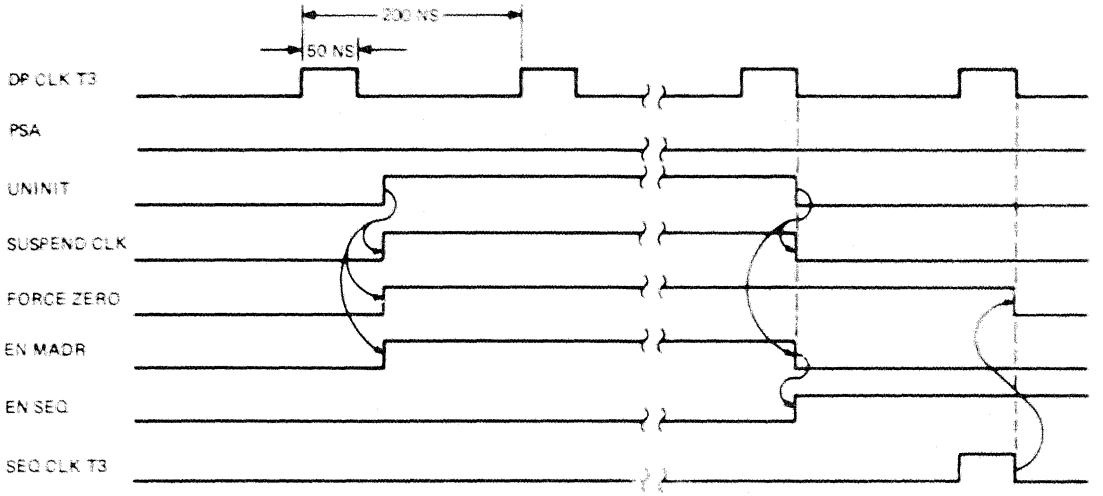
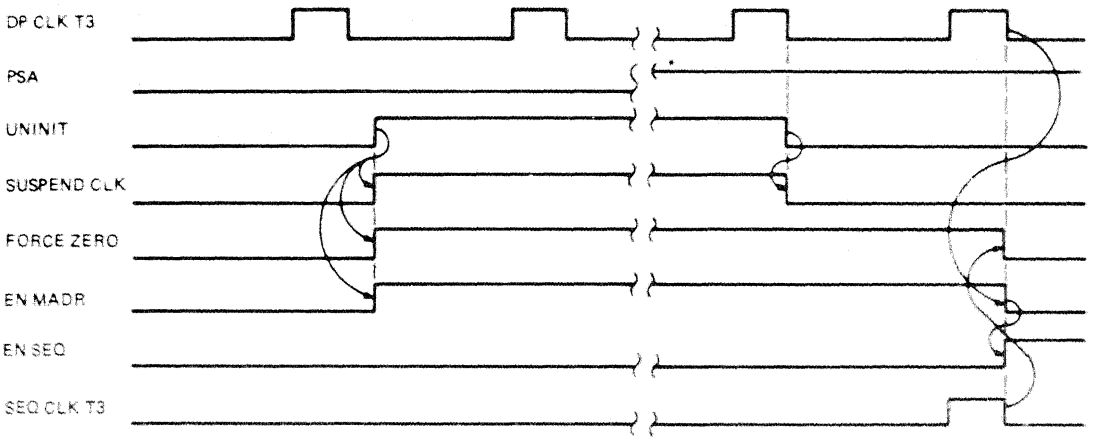


Figure 4-13 Microcode Start-Up Timing Diagram

SEE NEXT FRAME
FOR LARGER ART



A. PSA = 0



B. PSA = 1

* SET DURING UNINITIALIZED STATE

TK 8725

Figure 4-13 Microcode Start-Up Timing Diagram

CHAPTER 5
DATA PATH MODULE

NOTE

The functional block diagrams in Chapter 5 use logical AND and OR symbols. It does not necessarily follow that a corresponding gate exists on the DP logic prints. The assertion of inputs A and B causing the assertion of output C may be represented on a block diagram by a single AND gate, yet the engineering drawing may show that several circuit stages are involved in the ANDing operation.

The functional block diagrams in this chapter are keyed to the DP engineering circuit schematics (CS prints) by letter designations in parentheses. The letters specify the DP CS sheet that contains the detailed logic associated with the functional blocks in the diagram.

The signal names used in the functional block diagrams are the names used on the engineering CS prints. Where other signal names or notes are used, they are enclosed in parentheses.

5.1 OVERVIEW

Figure 5-1 is a block diagram of the data path module. Operation of the DP is under microword control except during an unsolicited SBI request operation. The microword fields control the flow of data through the DP, selecting the source of data for the BUS IB when outputting from the DP into the SBI module, and selecting the destination for the BUS IB data when receiving data from the SBI module. Possible sources for the BUS IB data are the local store (LS) RAMs, the virtual circuit descriptor table (VCDT) RAMs, the 2901A microprocessor, or the miscellaneous data (BUS MD). Possible sources for the BUS MD data are the PB IN register, the microword from the control store (CS), the port maintenance control/status register (PMCSR), or the microword literal field.

An unsolicited SBI request is a read or a write of a DP location, that is not controlled by the microword. The request sequence is initiated from the host via the SBI module and controlled via the R B(09:00) input lines from the SBI module.

The microword IB DST field and IB SRC field select the BUS IB destination and source, respectively. When the IB DST field and the IB SRC field select the PB OUT register and the PB IN register as the BUS IB destination and BUS IB source, respectively, the DP is functioning to transfer data packets between the PB and the SBI module. The data undergoes format conversion within the DP. Longwords received from the SBI module are applied to the BUS IB bus and then to the IB IN bus via a latch. From the IB IN bus the longword is loaded into a PB OUT register which then unloads a byte at a time onto the PORT DATA bus.

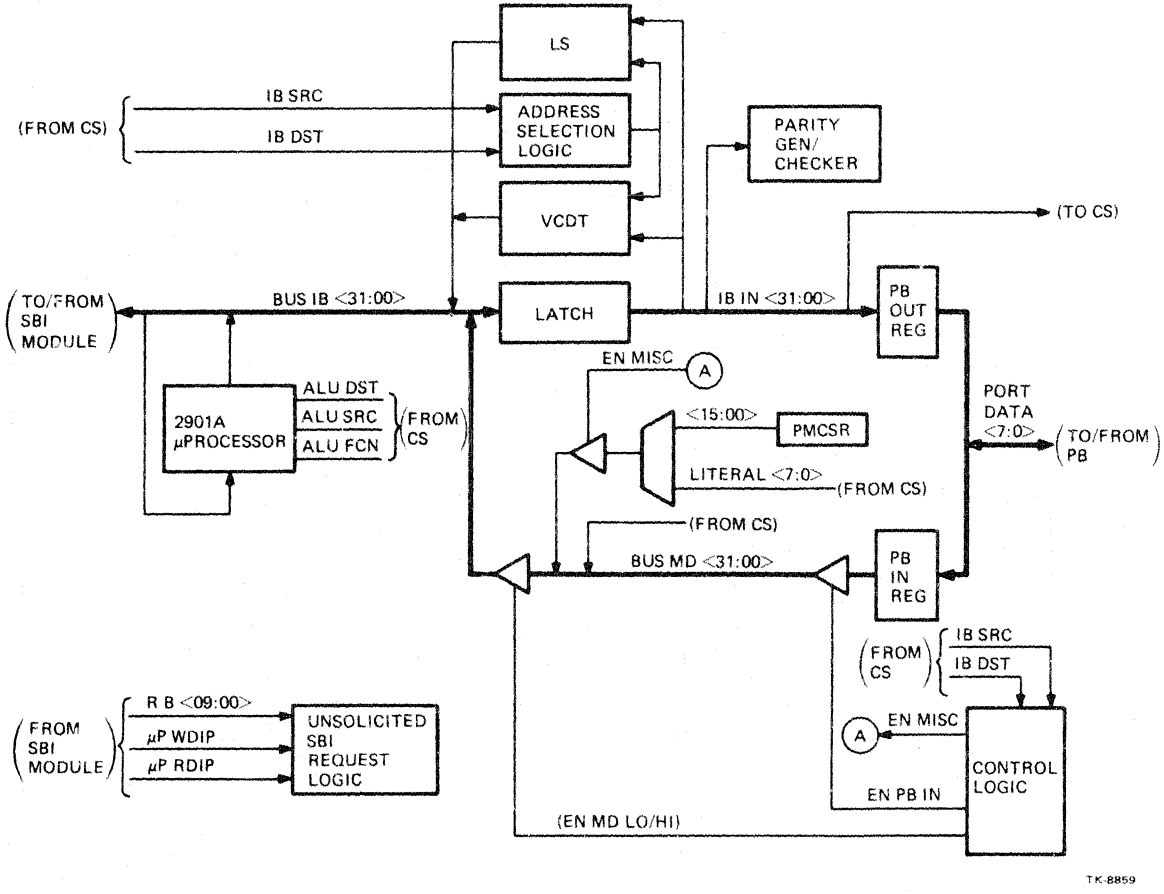


Figure 5-1 Data Path Block Diagram

Bytes received from the PB via the PORT DATA bus are applied to a 32-bit PB IN register. When four bytes are loaded into the register, the register output is enabled onto the BUS MD bus by EN PB IN from the DP control logic. The data longword is transferred from the BUS MD bus to the BUS IB bus by EN MD LO/HI (also from the control logic) and then sent to the SBI module.

The DP contains a 256 × 32 LS RAM area and a 256 × 16 VCDT RAM area. The LS contains software status blocks and many software registers associated with the port architecture. The VCDT is used to store CI node parameters. The LS or VCDT can be selected as a BUS IB source (read the RAMs) or a BUS IB destination (write the RAMs via the IB IN bus). The LS and VCDT are addressed in parallel from the address selection logic. The address source may be the IB IN bus data, the IB IN bus data translated, or the microword literal field. If the LS/VCDT is the BUS IB source, the microword IB SRC field selects the LS/VCDT address. If the LS/VCDT is the BUS IB destination, the microword IB DST field selects the LS/VCDT address.

In an unsolicited SBI request sequence, the microcode is suspended and the host writes or reads a selected DP location as specified by the R B(09:00) inputs. If the LS or VCDT RAMs are selected, the R B(09:00) control lines supply the LS/VCDT address to the LS/VCDT address selection logic.

The DP control logic receives both the source and the destination control fields from the microword. It decodes the control fields to determine if a BUS IB source or BUS IB destination is being specified, and what source (or destination) has been selected. The logic then generates the required enabling signals for the selected area, and the required gating signals to connect that area to the BUS IB.

A parity generator/checker is connected to the IB IN bus and does all the parity generating and checking for the DP. The BUS IB bus is connected to the IB IN bus through a latch which makes the BUS IB data available to the parity logic. Byte parity is generated and checked on data for the PB interface, word parity is generated and checked on data in the LS and the VCDT, and longword parity is generated and checked on data at the SBI module interface.

The DP contains a 2901A microprocessor which performs general purpose arithmetic and logical operations under control of the microword ALU control fields. The 2901A can be a BUS IB source or a BUS IB destination. The function performed by the 2901A is specified by the ALU FCN field from the microword.

5.2 DP BUSES AND PB INTERFACE

Figure 5-2 is a block diagram of the DP buses and the PB interface. Data transferring between the SBI module and the PB is in two different formats. Data at the DP/SBI module interface is in 32-bit longword format. Data at the DP/PB interface is in eight-bit byte format. Longword-to-byte and byte-to-longword conversion occurs at the DP/PB interface.

5.2.1 DP-to-PB Interface

Data from the SBI module inputs as BUS IB (31:00) into a transparent latch. The latch output follows the latch input so long as the latch HOLD input (LATCH IB) is true. The latch output (IB IN (31:00)) is then applied in eight-bit bytes to four sections of the PB OUT register. The 32-bit longword is clocked into the register by CLK PB OUT which is asserted by the LD PB OUT command from the DP control logic. LD PB OUT is asserted when the PB OUT register is selected as the BUS IB destination.

The PB OUT register is unloaded by the PB. A PB LOAD command and a PMUX (1:0) code from the PB control the data flow from the PB OUT register to the PORT DATA bus. The PB LOAD command enables the PB out byte select logic while the PMUX (1:0) code asserts one of the four EN PB BYTE output signals. The PMUX (1:0) code asserts the four EN PB BYTE (3:0) signals in sequence to unload the PB OUT register onto the PORT DATA bus a byte at a time. After the last byte has been unloaded, LD PB OUT is again asserted by the DP control logic to load the next longword into the PB OUT register.

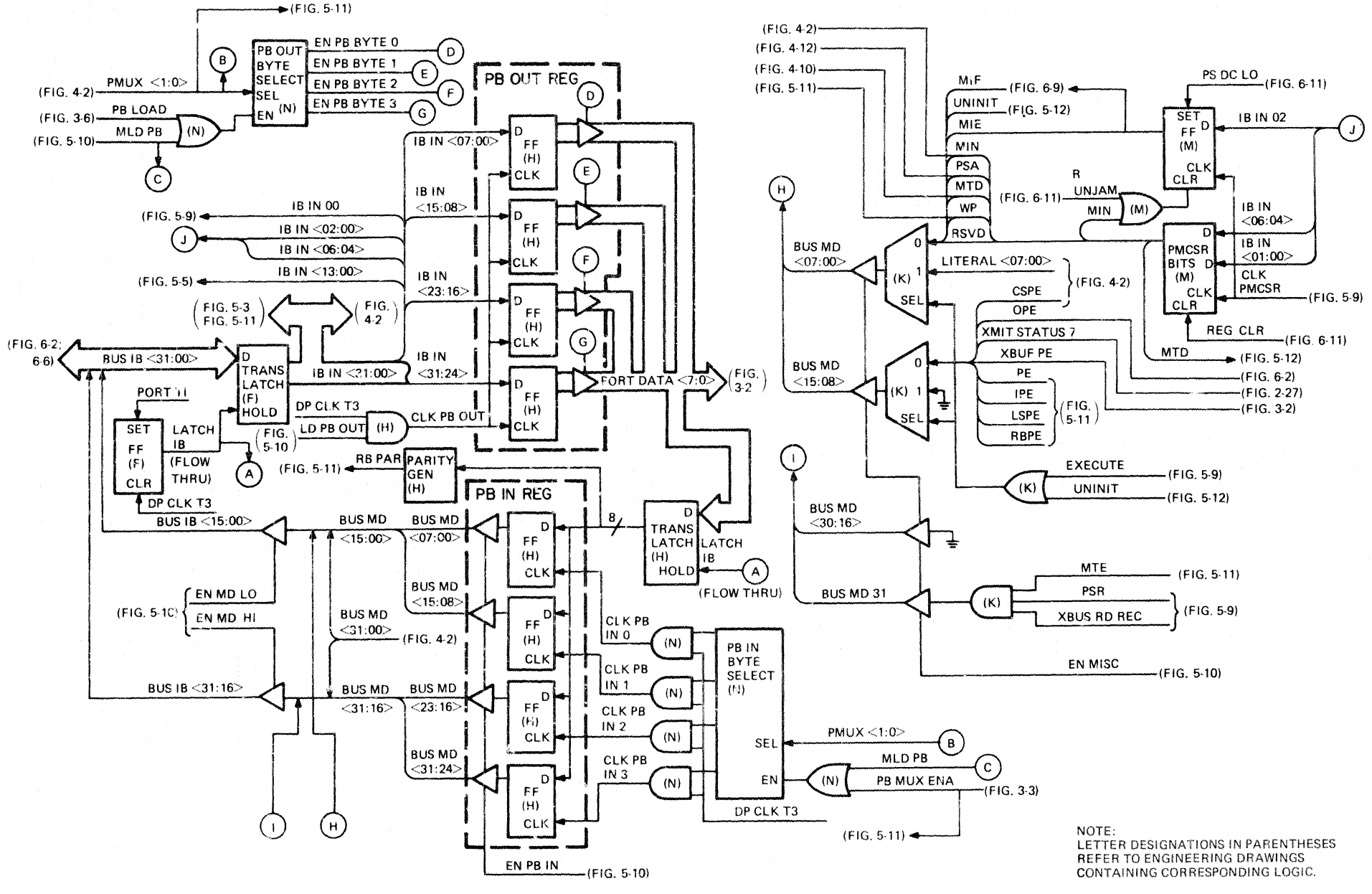


Figure 5-2 DP Buses and PB Interface

5.2.2 PB-to-DP Interface

Input data bytes from the PB (PORT DATA ⟨7:0⟩) are applied to a transparent latch. The latch output follows the latch input so long as the latch HOLD input (LATCH IB) is true. The latch output byte is then applied to four sections of the 32-bit PB IN register.

The PB IN register is loaded by the PB. A PB MUX ENA command and a PMUX ⟨1:0⟩ code from the PB control the data flow from the PORT DATA bus to the PB IN register (via the transparent latch). The PB MUX ENA command enables the PB in byte select logic while the PMUX ⟨1:0⟩ code asserts one of the four CLK PB IN output signals to clock a data byte into the PB IN register. The PMUX ⟨1:0⟩ code asserts the four CLK PB IN signals in sequence to load the PB IN register from the PORT DATA bus a byte at a time. After the last byte has been loaded, EN PB IN is asserted by the DP control logic to gate the 32-bit register output onto the BUS MD as BUS MD ⟨31:00⟩. EN PB IN is asserted when the PB IN register is selected as the BUS IB source. EN PB IN then negates while PB MUX ENA asserts to start loading new data bytes into the PB IN register.

The BUS MD bus is gated to the BUS IB in two sections. The lower 16 bits are gated by EN MD LO while the upper 16 bits are gated by EN MD HI. The BUS MD bus also receives a 32-bit input from the CS in the PB.

5.2.3 LITERAL/PMCSR Mux

A third source for the BUS MD is the LITERAL/PMCSR mux. When the port is in the uninitialized state (UNINIT true) or executing an unsolicited SBI request (EXECUTE true), the mux selects the 16-bit PMCSR register. An unsolicited SBI write request of the PMCSR will assert CLK PMCSR which writes six PMCSR bits (MIE, MIN,PSA, MTD, WP, RSVD). The PMCSR register bits are described in Table 5-1.

When not in the uninitialized state and not executing an unsolicited SBI request (EXECUTE and UNINIT false), the LITERAL/PMCSR mux selects LITERAL ⟨7:0⟩ for the lower eight bits of the BUS MD bus. The next eight bits (BUS MD ⟨15:08⟩) are grounded.

BUS MD 31 receives the maintenance error (MTE) bit during an unsolicited SBI read request of the port status register (PSR). XBUS RD REC asserts during an unsolicited SBI read request and is ANDed with PSR and MTE to generate bit 31 for the BUS MD.

The LITERAL/PMCSR mux output is gated to the BUS MD bus by EN MISC from the DP control logic. EN MISC asserts when the DP control logic selects the LITERAL as the BUS IB data source, and during an unsolicited SBI request when the PMCSR is to be read.

Table 5-1 PMCSR Bits

Bit	Mnemonic	Description
15	PE	Parity Error: PE is the OR of PMCSR bits (14:08). It is cleared when PMCSR (14:08) are all cleared.
14	CSPE	Control Store Parity Error: CSPE sets when a parity error is detected in the CS in the PB. CSPE can only be set when the microcode is running.
13	LSPE	Local Store Parity Error: LSPE sets when a parity error is detected while reading the LS or the VCDT. LSPE can only be set by a microcode read of LS or the VCDT. It will not set during an unsolicited SBI request.
12	RBPE	Receive Buffer Parity Error: Set when a parity error is detected while reading a packet buffer.
11	XMIT STATUS 7	Transmit Data Parity Error: Set when a parity error is detected in the link transmit channel.
10	IPE	Input Parity Error: Set when a parity error is detected on a data transfer from the SBI module to the DP.
09	OPE	Output Parity Error: Set when a parity error is detected on a data transfer from the output buffer to the transceivers within the SBI module.
08	XBUF PE	Transmit Buffer Parity Error: Set when a parity error is detected while the PB is unloading a transmit buffer.
07	UNINIT	Uninitialized State: When set the port is in the uninitialized state. The microcode is not running and the port will not respond to data packet traffic. UNINIT is set by DEAD, MIN, or MTE. The microcode is started when UNINIT is cleared by writing a 1 into the PICR, or by a boot timeout.
06	PSA	Programmable Starting Address: When set the microcode will start at the address in the MADR register in the PB when a "1" is written into the PICR or a boot time-out occurs. When reset the microcode starts at location 000.
05	RSVD	Not used.
04	WP	Wrong Parity: When set the DP parity generator/checker will generate and check even parity instead of odd. Used to generate parity errors for maintenance purposes.

Table 5-1 PMCSR Bits (Cont)

Bit	Mnemonic	Description
03	MIF	Maintenance Interrupt Flag: When set, this bit indicates that an interrupt-causing condition has occurred.
02	MIE	Maintenance Interrupt Enable: When set interrupts are enabled. This bit is set by PS DC LO from the SBI module, or by writing MIE with a "1".
01	MTD	Maintenance Timer Disable: When set, the boot and maintenance timers are disabled and cannot cause an interrupt. When reset the timers are enabled.
00	MIN	Maintenance Initialize: When set, an initialize signal is generated that clears all port errors and leaves the port in the uninitialized state.

5.3 LS/VCDT

LS (local store) consists of eight 256 × 4 RAMs addressed in parallel to form a 32-bit output. The total LS space (256 × 32) is enabled in two 16-bit segments forming a 256 × 16 LS HI section and a 256 × 16 LS LO section.

The virtual circuit descriptor table (VCDT) consists of four 256 × 4 RAMs addressed in parallel to form a 16-bit output. One signal enables the total VCDT space.

Figure 5-3 illustrates the LS and VCDT space and the addressing and enabling signals associated with each. All three areas (LS HI, LS LO, VCDT) are addressed in parallel by LSA (07:00) from the LSA mux. Thus, access is to the same location in each area.

Data placed into the LS and VCDT is from the IB IN bus. IB IN (31:16) is input into the LS HI area. IB IN (15:00) is input into the LS LO area and the VCDT.

Data out of the LS and VCDT is placed onto the BUS IB. The LS HI section outputs onto BUS IB (31:16). The LS LO section and the VCDT output onto BUS IB (15:00).

Sections LS LO, LS HI, and the VCDT are enabled by EN LS LO, EN LS HI, and EN VCDT, respectively. The enabling signal for any area must be true before data can be written into or read out of that area. In addition, to write data into an enabled area, WR RAM must be true. To read data out of an enabled area, EN LS/VCDT must be true and WR RAM must be false (assertion of the WR RAM write strobe inhibits the RAM output).

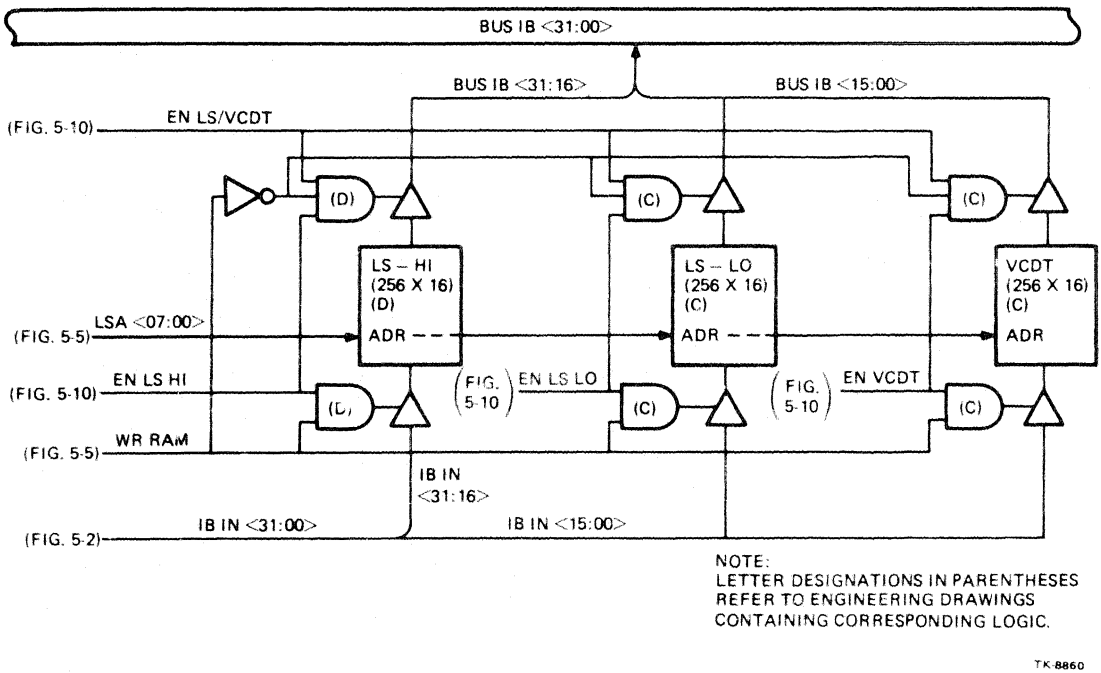


Figure 5-3 LS/VCDT Block Diagram

5.3.1 LS/VCDT Address Selection

Figure 5-4 is a simplified block diagram of the LS/VCDT address selection function. The LS and VCDT address (LSA (07:00)) is obtained from an LSA mux which functions to select the address from four possible sources. Address source decode logic monitors the IB DST and IB SRC fields from the microword to determine if the LS/VCDT is to be a BUS IB destination or a possible BUS IB source. Accordingly, the address source decode logic decodes the IB DST field or the IB SRC field to effect mux selection of the LS/VCDT address source. When the logic senses that the LS/VCDT has been selected as the bus IB destination, it asserts EN RAM WR to the write strobe logic. The write strobe logic generates the write strobe (WR RAM) for the LS/VCDT RAMs.

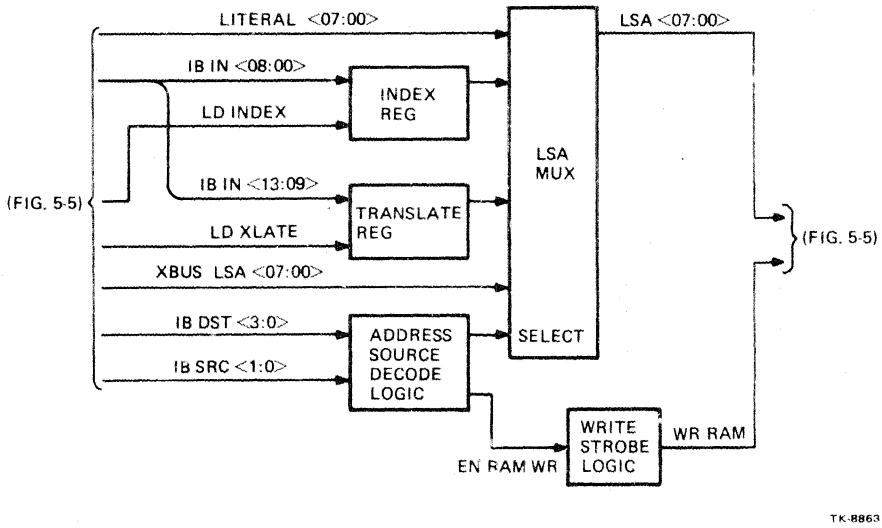


Figure 5-4 LS/VCDT Address Selection Simplified Block Diagram

Figure 5-5 is a detailed block diagram of the LS/VCDT address selection function. A two-bit input to the two mux SEL pins (SEL 2, SEL 1) select the address source. Table 5-2 lists the address source selected by the mux for the four states of SEL 2 and SEL 1.

The SEL 2 and SEL 1 inputs are obtained from two flip-flops. Both flip-flops are directly set by UNINIT when the port goes into the uninitialized state, thereby forcing SEL 2 and SEL 1 true. With SEL 2 and SEL 1 both true, the mux selects XBUS LSA ⟨07:00⟩ as the LS/VCDT address. The assertion of SUSPEND during an unsolicited SBI request also forces the LSA mux to select XBUS LSA ⟨07:00⟩.

Thus, during the uninitialized state and during an unsolicited SBI request when microcode control of the LSA address mux is suspended, the LS/VCDT address is supplied from the host via the SBI module and the XBUS LSA address lines.

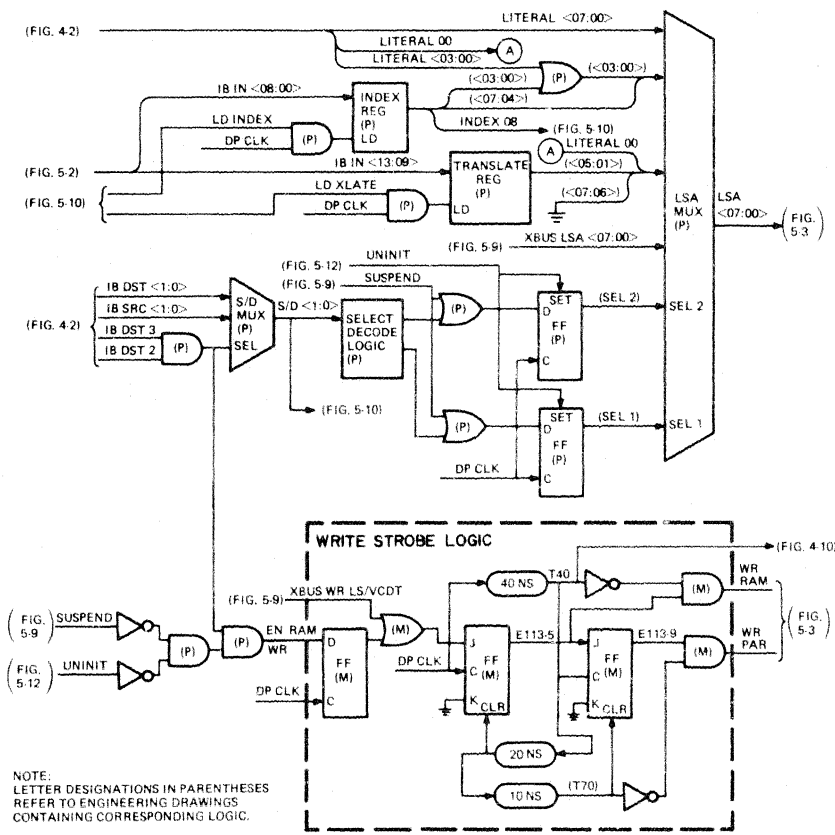


Figure 5-5 LS/VCDT Address Selection Block Diagram

SEE NEXT FRAME
FOR LARGER ART

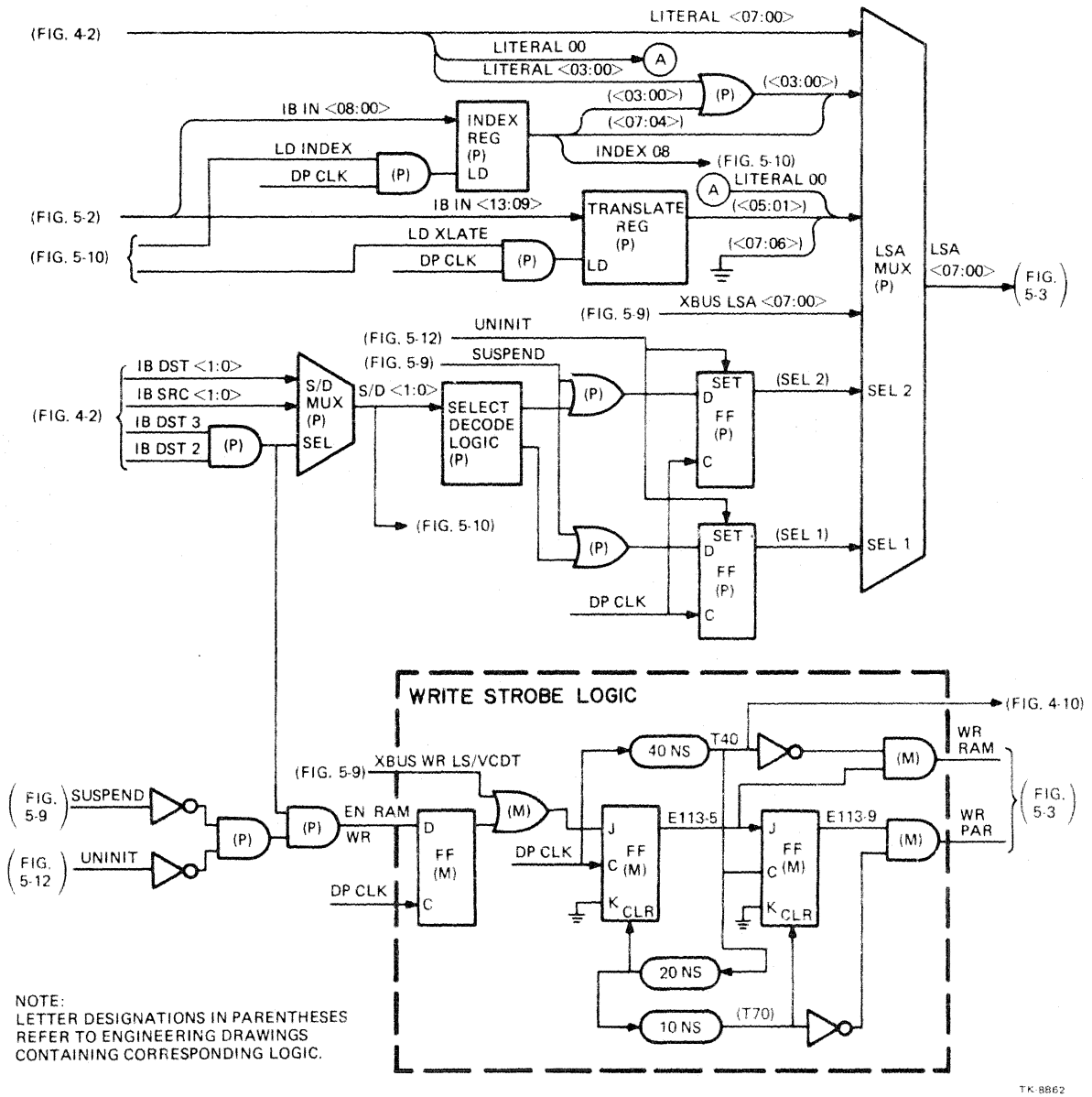


Figure 5-5 LS/VCDT Address Selection Block Diagram

Table 5-2 LSA Mux Selection Code

SEL 2	SEL 1	Address Source
0	0	Literal
0	1	Index Register
1	0	Translate Register
1	1	XBUS LSA Register

When not in the uninitialized state (UNINIT false) and when not executing an unsolicited SBI request (SUSPEND false), the select decode logic controls the two SEL bits by conditioning the two SEL flip-flops to set or reset. The decode logic causes both the flip-flops to reset, or one or the other to be set, thereby causing the LSA mux to select the LITERAL, the index register, or the translate register as the LSA address source. The decode logic will not cause both flip-flops to set and hence will never select the XBUS LSA input as the LSA address source.

The decode logic operates from a two-bit source/destination input (S/D ⟨1:0⟩) obtained from the S/D mux. The mux selects destination bits IB DST ⟨1:0⟩ or source bits IB SRC ⟨1:0⟩ for the S/D ⟨1:0⟩ output. The mux selection is made by ANDing IB DST bits 3 and 2. If both bits are true, the LS/VCDT is being selected as the BUS IB destination and the mux selects IB DST ⟨1:0⟩ for the S/D ⟨1:0⟩ bits. If either (or both) bits are false, another destination is being selected for the BUS IB (Table 5-3). In this case the mux selects IB SRC ⟨1:0⟩ for the S/D ⟨1:0⟩ bits in the event the LS/VCDT is selected as the BUS IB source.

When the LSA mux SEL bits ⟨2:1⟩ are 0:0, the literal input (LITERAL ⟨07:00⟩) from the microword in the PB is selected for the LSA ⟨07:00⟩ address lines.

When the LSA mux SEL bits ⟨2:1⟩ are 0:1, the output of the index register is selected for the LSA ⟨07:00⟩ address lines. The index register is loaded with IB IN ⟨08:00⟩ when LD INDEX asserts from the DP control logic. IB IN ⟨07:00⟩ provides the eight-bit address input to the mux. IB IN 08 provides INDEX 08 which is used in the DP control logic to select the LS or the VCDT (INDEX 08 negated = LS; INDEX 08 asserted = VCDT). Also note that the four least significant bits from the index register are ORed with the four least significant bits of the LITERAL input. This allows the literal bits to perform four-bit wide indexing into the LS or VCDT tables.

When the LSA mux SEL bits ⟨2:1⟩ are 1:0, the output of the translate register is selected for the LSA ⟨07:00⟩ address lines. The translate register is loaded with five bits from the IB IN bus (IB IN ⟨13:09⟩) when LD XLATE asserts from the DP control logic. These five bits output from the register as address lines ⟨05:01⟩. Address lines ⟨07:06⟩ are grounded. The least significant address line (00) is LITERAL 00 which allow one-bit indexing of the translate LS or VCDT entries.

5.3.2 Write Strobe Logic

As previously mentioned, when the LS/VCDT is selected as the BUS IB destination, IB DST ⟨3:2⟩ are both true. If this is not an unsolicited SBI request (SUSPEND false) and the port is not in the uninitialized state (UNINIT false), then EN RAM WR is asserted to generate an LS/VCDT write strobe. EN RAM WR is applied to a flip-flop whose output is ORed with XBUS WR LS/VCDT from the unsolicited SBI request logic. The OR gate output is applied to some delay logic where the LS/VCDT write strobe (WR RAM) and the parity write strobe (WR PAR) are generated. Delays are incorporated into the write strobe logic making the WR RAM strobe 40 ns wide and the WR PAR strobe 30 ns wide. The WR PAR strobe begins on the trailing edge of the WR RAM strobe as shown in Figure 5-6.

The delay logic consists of two flip-flops. The first flip-flop is enabled by the OR gate output and is set by DP CLK. The flip-flop output is applied to an AND gate which then asserts WR RAM. The clock pulse that set the flip-flop is applied to a delay line where it is delayed 40 ns to become T40. T40 is inverted and applied to the WR RAM AND gate causing WR RAM to negate.

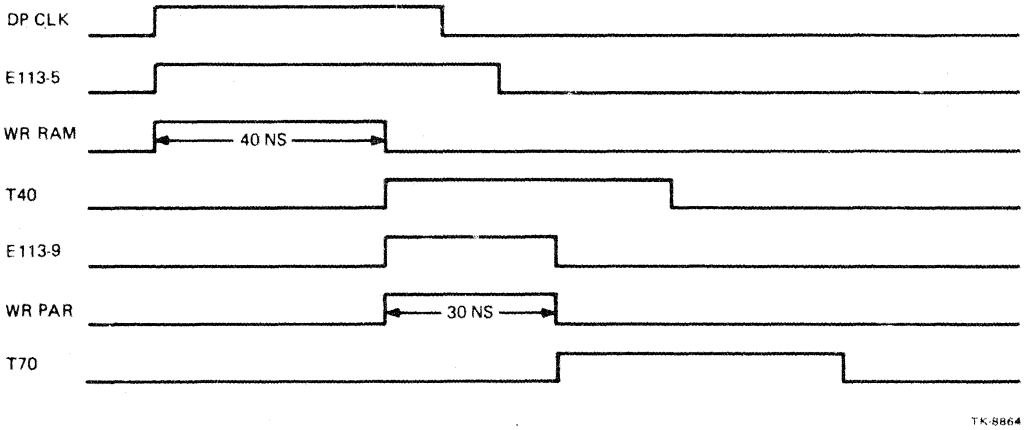


Figure 5-6 Write RAM Timing Diagram

The assertion of T40 clocks the second flip-flop. The second flip-flop output is applied to the WR PAR AND gate which then asserts WR PAR. T40 is delayed 30 ns, inverted, and then applied to the WR PAR AND gate causing WR PAR to negate.

5.4 UNSOLICITED SBI REQUESTS

An unsolicited SBI request is a read or a write of a DP location, that was initiated by the host and not by the port microcode. The request is a two-state sequence involving a suspend cycle and an execute cycle. During the suspend state the microsequencer clock is stopped for one cycle, the microcode branch flags are saved, and the BUS IB sources and destinations are disabled thus freeing up the BUS IB for the external access. During the execute state, the read or write of the DP location occurs and the data is transferred between the BUS IB and the SBI module. After the execute cycle, normal microcode execution is resumed at the address that was frozen during the suspend cycle.

An unsolicited SBI request sequence is initiated by either a write-in progress (UP WDIP) or a read-in progress (UP RDIP) flag from the SBI module. In addition, the SBI module loads control information (R B(09:00)) associated with the requested operation, into the XBUS LS address register with LD LS ADRS REG. XBUS LSA (07:00) from the register is applied to the LSA mux where it may be selected to address LS or the VCDT. XBUS LSA (09:00), also from the register, addresses a location in a 1K × 8 PROM which outputs the control information for the requested operation.

Figures 5-7 and 5-8 are flow diagrams of unsolicited SBI write and read sequences, respectively. Figure 5-9 illustrates the logic associated with the sequences.

5.4.1 Unsolicited Write Sequence

An unsolicited write sequence (Figure 5-7) is initiated by UP WDIP from the SBI module which causes XBUS RD/WRT to assert. The next DP CLK pulse sets the suspend flip-flop placing the port into the suspend state and causing SUSPEND to assert. SUSPEND in turn asserts SUSPEND CLK to the control store logic in the PB where it inhibits the CS microsequencer clock thereby suspending microsequencer operation. The next DP CLK negates SUSPEND and asserts EXECUTE. The assertion of EXECUTE causes EN RCV WD to assert. The next DP CLK pulse negates EXECUTE to complete the unsolicited SBI write request sequence.

XBUS LSA (09:00) addresses a 1K × 8 PROM which outputs control information regarding the write operation.

If PROM output 01 (MDATR) is true during the suspend cycle (SUSPEND true), and the port is in the uninitialized state (UNINIT true), maintenance data is written into the CS in the PB. EN CS DATA IN gates the maintenance data into the CS while CS WE asserts the CS write strobe.

If PROM output 03 (LS/VCDT) is true during the execute cycle (EXECUTE true), XBUS WR LS/VCDT is asserted to the LS and VCDT write logic (Figure 5-5) to write the data from the BUS IB into the location addressed by XBUS LSA (07:00). Also XBUS EN LS/VCDT HI and XBUS EN LS/VCDT LO assert to the DP control logic to enable the RAMs to be written. XBUS LSA 08 specifies either the LS RAMs or the VCDT RAMs. XBUS LSA 08 is applied to the DP control logic which then enables the selected RAMs (Figure 5-10).

If IB IN 00 is true during the execute cycle, PROM outputs 05, 06, and 07 are sampled. If output 05 is true, PICR WRT asserts which in turn negates UNINIT and takes the port out of the uninitialized state (Figure 5-12). If output 06 is true, REG WRT asserts as a branch condition to the CS branching logic. REG WRT is a flag to the microcode that a register has been written. The microcode can then check the registers for new data. If output 07 is true, PMTCR CLR is asserted to the boot timer logic to reset the BTO timer and extend the boot time period (Figure 5-12).

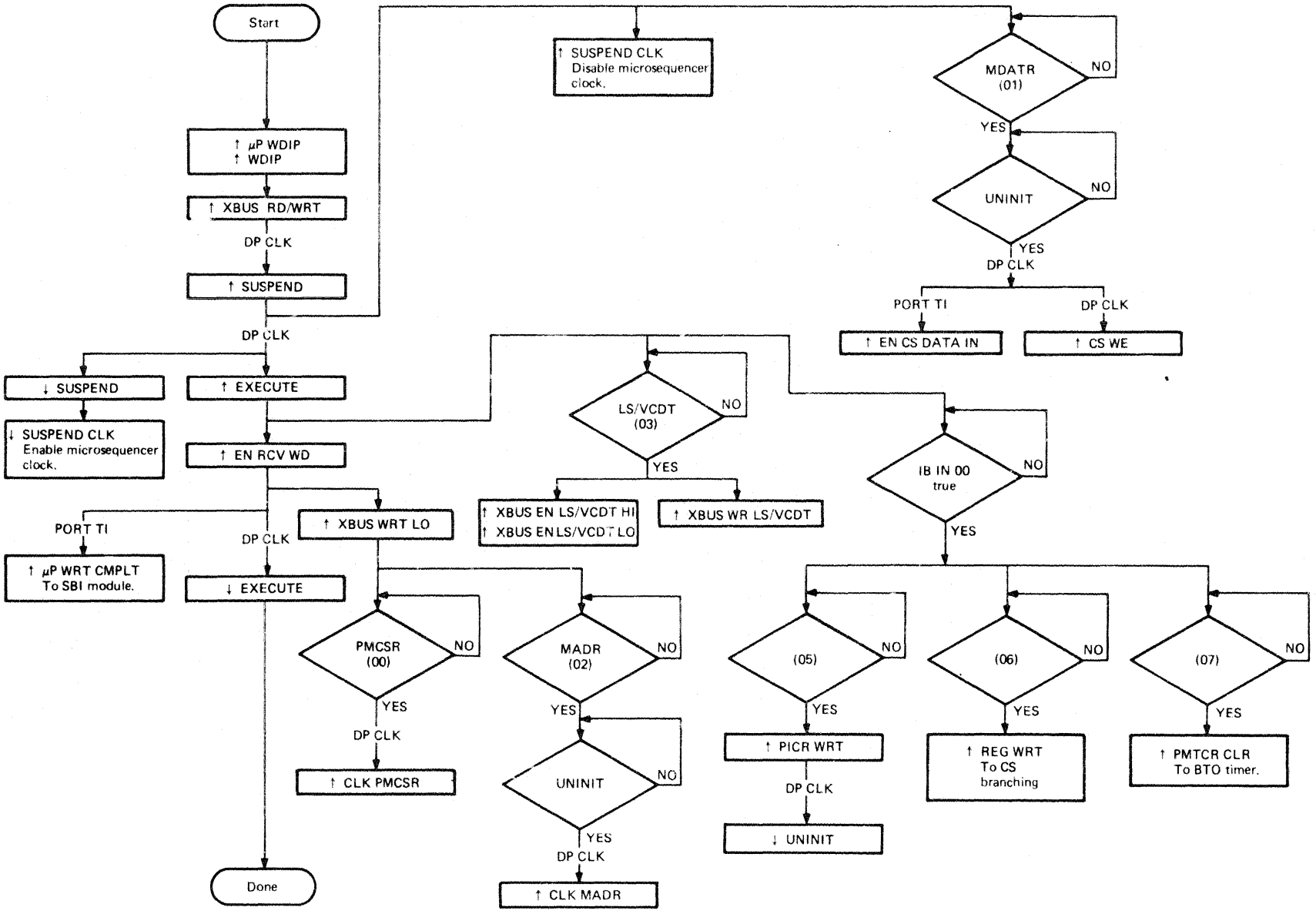


Figure 5-7 Unsolicited SBI Write Request Flow Diagram

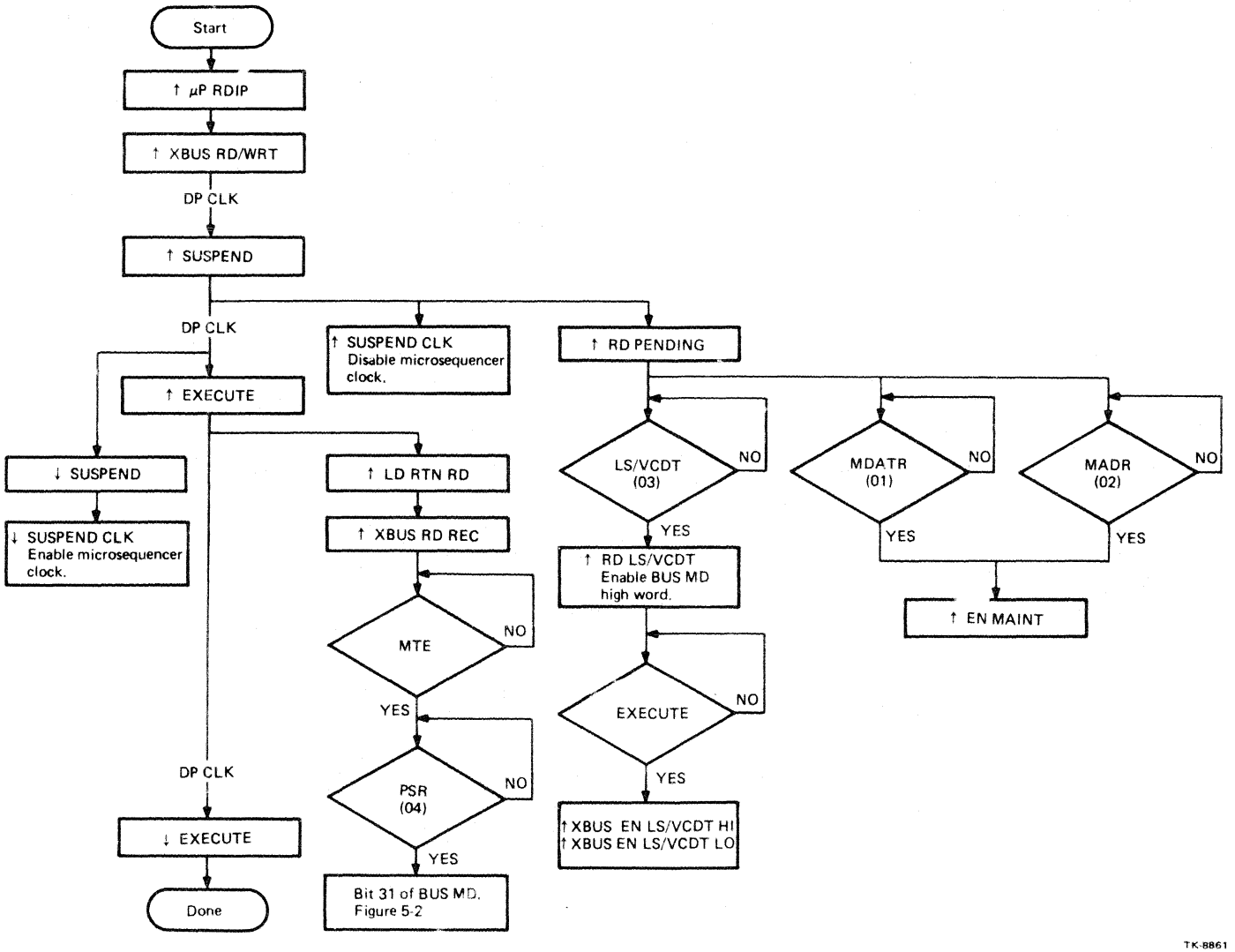
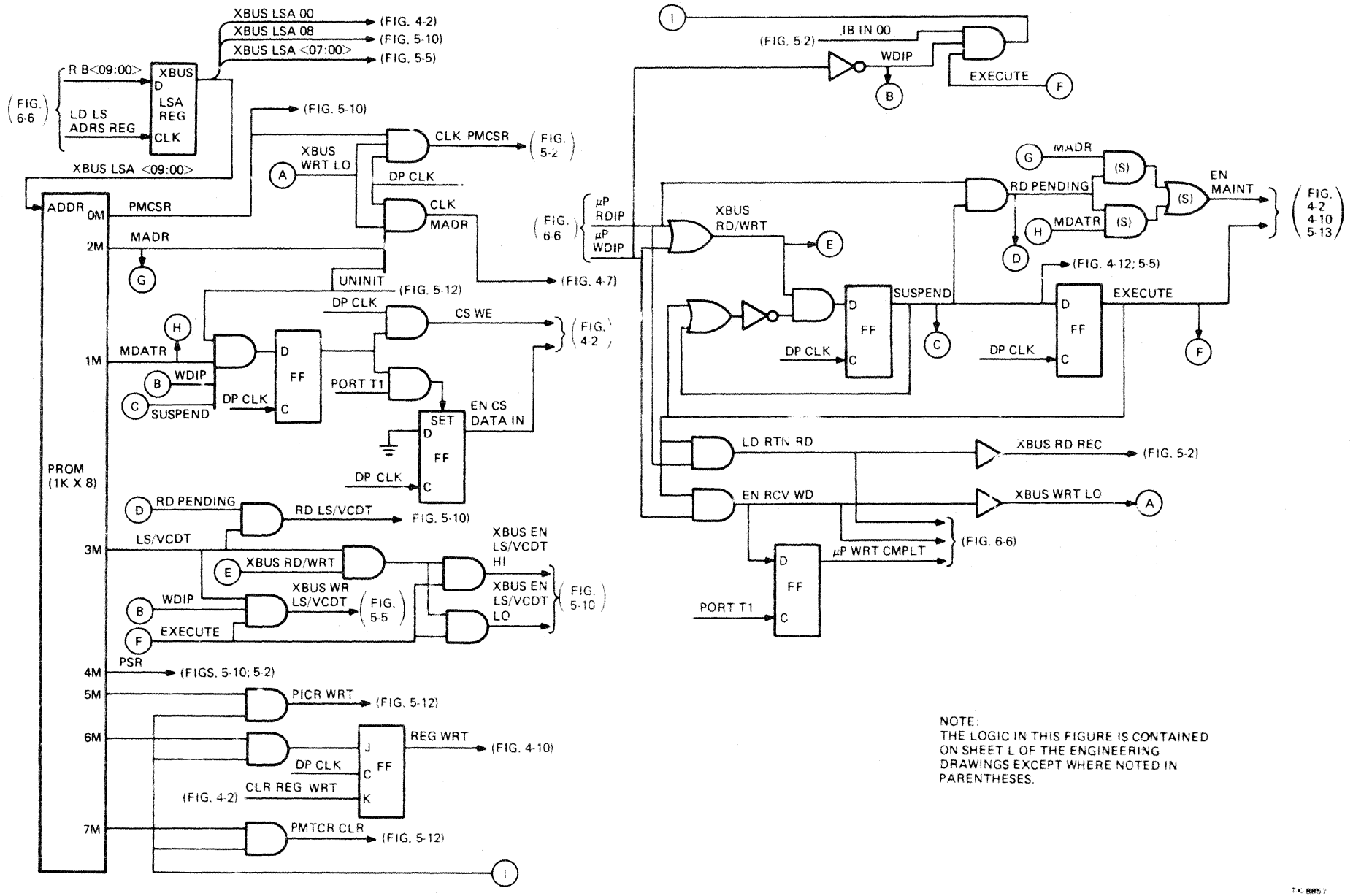


Figure 5-8 Unsolicited SBI Read Request Flow Diagram



NOTE:
THE LOGIC IN THIS FIGURE IS CONTAINED ON SHEET L OF THE ENGINEERING DRAWINGS EXCEPT WHERE NOTED IN PARENTHESES.

Figure 5-9 Unsolicited SBI Request Logic

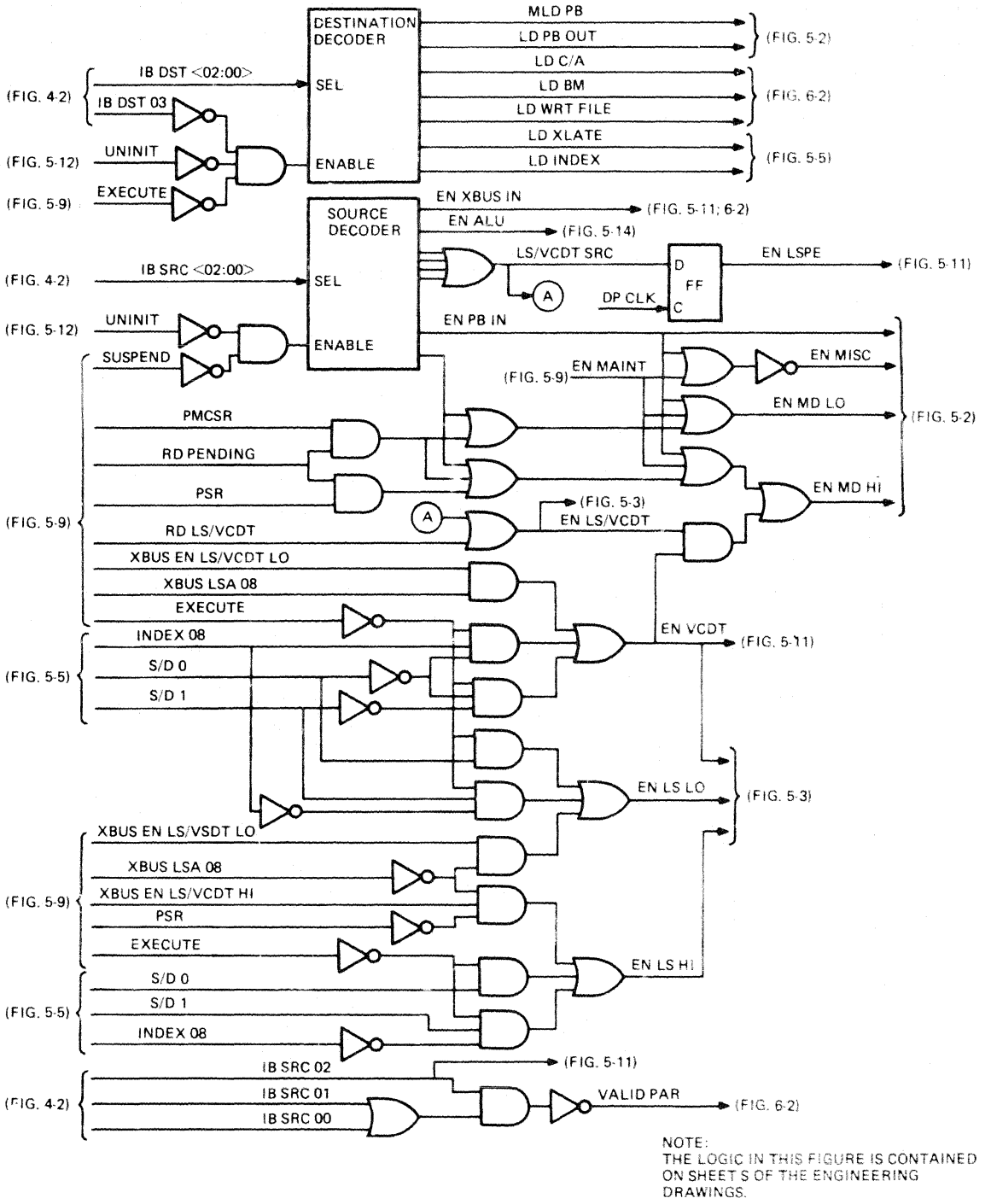


Figure 5-10 Control Logic

5.4.2 Unsolicited Read Sequence

An unsolicited read sequence (Figure 5-8) is initiated by UP RDIP from the SBI module which causes XBUS RD/WRT to assert. The next DP CLK pulse sets the suspend flip-flop placing the port into the suspend state and causing SUSPEND to assert. SUSPEND in turn asserts SUSPEND CLK to the control store logic in the PB where it inhibits the CS microsequencer clock, suspending microsequencer operation. The next DP CLK negates SUSPEND and asserts EXECUTE. The next DP CLK negates EXECUTE to complete the unsolicited SBI read request sequence.

XBUS LSA ⟨09:00⟩ addresses a 1K × 8 PROM which outputs control information regarding the read operation.

The assertion of SUSPEND causes RD PENDING to assert. If PROM output 01 (MDATR) or PROM output 02 (MADR) is true while RD PENDING is true, EN MAINT is asserted to the PB to enable maintenance data into the DP over the BUS MD bus. EN MAINT is also applied to the DP control logic where it asserts EN MD LO and EN MD HI to gate the BUS MD data to the BUS IB bus.

If PROM output 03 (LS/VCDT) is true while RD PENDING is true, RD LS/VCDT is asserted to the DP control logic to assert EN MD HI and enable the high word from the BUS MD to the BUS IB. When EXECUTE asserts, XBUS EN LS/VCDT HI and XBUS EN LS/VCDT LO assert to the DP control logic to enable the RAMs to be read. XBUS LSA 08 specifies either the LS RAMs or the VCDT RAMs. XBUS LSA 08 is applied to the DP control logic which then enables the selected RAMs.

The assertion of EXECUTE causes LD RTN RD and XBUS RD REC to assert. If a maintenance error exists (MTE true), PROM output 04 (PSR) is sampled. If PSR is true, this operation is a read of the port status register and bit 31 is asserted onto the BUS MD bus when miscellaneous data is read (Figure 5-2).

5.5 CONTROL LOGIC

The DP control logic (Figure 5-10) generates the commands and enabling signals controlling the data flow within the DP. Operation of the DP is controlled by the microword from the PB except during unsolicited SBI requests when the microcode is suspended and control shifts to the host via the SBI module.

5.5.1 BUS IB Destination

The IB DST ⟨03:00⟩ field from the microword selects the destination for the data on the BUS IB. Table 5-3 lists the IB DST codes and the destinations they select.

A destination decoder decodes the IB DST field and outputs the selected destination. The decoder is enabled when IB DST 03 = 0 (first eight codes in Table 5-3). The remaining three bits (IB DST ⟨02:00⟩) are decoded to select the destination shown in the table. Note that if the port is in the uninitialized state (UNINIT true), or an unsolicited SBI request is in progress (EXECUTE true), the decoder is disabled and the BUS IB destination is selected by the host via the SBI module and the XBUS LSA address lines (Paragraph 5.4).

The LD INDEX or LD XLATE outputs are asserted during the first cycle of a two cycle access when the LS or the VCDT is the destination. LD INDEX (or LD XLATE) functions to load the index register (or the translate register) with the LS or VCDT address (Figure 5-5). The selection between the LS or the VCDT is made during the second cycle when IB DST 03 = 1.

The next three outputs from the destination decoder (LD WRT FILE, LD BM, LD C/A) are asserted when the SBI module is selected as the BUS IB destination. All three signals are sent to the SBI module.

Decoder output LD PB OUT is asserted when the PB OUT register is selected as the BUS IB destination. LD PB OUT loads the PB OUT register from the IB IN bus for output onto the PORT DATA bus (Figure 5-2).

Table 5-3 IB DST Codes

IB DST 03	02	01	00	Destination	S/D 01	00	Address Source
0	0	0	0	No operation	–	–	—
0	0	0	1	LD INDEX	–	–	—
0	0	1	0	LD XLATE	–	–	—
0	0	1	1	LD WRT FILE	–	–	—
0	1	0	0	LD BM	–	–	—
0	1	0	1	LD C/A	–	–	—
0	1	1	0	LD PB OUT	–	–	—
0	1	1	1	MLD PB	–	–	—
1	0	0	0		–	–	—
1	0	0	1		–	–	—
1	0	1	0	Not used	–	–	—
1	0	1	1		–	–	—
1	1	0	0	LS	1	1	LITERAL
1	1	0	1	LS or VCDT*	1	0	Index Register
1	1	1	0	LS	0	1	Translate Register
1	1	1	1	VCDT	0	0	LITERAL

* Depending on INDEX 08; INDEX 08 = 0, LS selected
INDEX 08 = 1, VCDT selected

Decoder output MLD PB is a maintenance function. When asserted it enables the output of the PB OUT register onto the port data bus and allows the data to loop back into the DP by enabling the PB in register.

When IB DST ⟨03:02⟩ = 1:1, source/destination (S/D) codes are generated in the LS/VCDT address select logic which control LS/VCDT address selection by the LSA mux (Paragraph 5.3.1). The S/D codes are listed in Table 5-3 along with the address sources that they select.

5.5.2 BUS IB Source

The IB SRC (02:00) field from the microword selects the source for the BUS IB data. Table 5-4 lists the IB SRC codes and the sources they select.

A source decoder decodes the IB SRC field and outputs the selected source. Note that if the port is in the uninitialized state (UNINIT true), or an unsolicited SBI request is in progress (SUSPEND true), the decoder is disabled and the BUS IB source is selected by the host via the SBI module and the XBUS LSA address lines (Paragraph 5.4).

The first four codes listed in Table 5-4 assert LS/VCDT SRC thereby selecting the LS or the VCDT as the BUS IB source. Accordingly, the S/D codes generated in the LS/VCDT address select logic control the LS/VCDT address selection just as they did in the BUS IB destination decode process. The S/D codes are listed in Table 5-4 along with the address sources that they select.

LS/VCDT SRC asserts EN LSPE to the DP parity logic to enable the logic to check parity on the data read out of LS.

LS/VCDT SRC also asserts EN LS/VCDT to enable the output of the LS/VCDT RAMs.

The true state of EN LS/VCDT causes EN MD HI to assert if the VCDT has been selected (EN VCDT true). Thus, the high word from the BUS MD bus is placed onto the BUS IB bus along with the low word from the VCDT.

Decoder output EN PB IN is asserted when the PB IN register is selected as the BUS IB source. As shown in Figure 5-2, EN PB IN gates the output from the register onto the BUS MD bus. The data is then transferred to the BUS IB bus. Note in Figure 5-10 that EN PB IN also asserts EN MD LO and EN MD HI in order to transfer the BUS MD data to the BUS IB bus. In addition, EN MISC is negated to isolate the LITERAL/PMCSR mux output from the BUS MD bus while the PB IN register data is being transferred.

Table 5-4 IB SRC Codes

IB SRC			IB Source	S/D		Address Source
02	01	00		01	00	
0	0	0	LS	1	1	LITERAL
0	0	1	LS or VCDT*	1	0	Index Register
0	1	0	LS	0	1	Translate Register
0	1	1	VCDT	0	0	LITERAL
1	0	0	EN PB IN	–	–	—
1	0	1	LITERAL	–	–	—
1	1	0	EN ALU	–	–	—
1	1	1	EN XBUS IN	–	–	—

* Depending on INDEX 08; 0 = LS, 1 = VCDT

The next decoder output is asserted when the LITERAL field of the microword is selected as the BUS IB source. The decoder output asserts EN MD LO and EN MD HI to transfer BUS MD data to the BUS IB bus. EN MISC is true to gate the LITERAL data from the LITERAL/PMCSR mux onto the BUS MD bus.

Decoder output EN ALU is asserted when the ALU is selected as the BUS IB source. EN ALU enables the ALU logic.

Decoder output EN XBUS IN is asserted when the SBI module is selected as the BUS IB data source. EN XBUS IN is sent to the SBI module to enable the data transfer into the DP. EN XBUS IN is also applied to the DP parity logic to enable the logic to check parity on the data from the SBI module.

5.5.3 Control Signals

EN MISC is used in the interface logic (Figure 5-2) to gate the output of the LITERAL/PMCSR mux onto the BUS MD bus. EN MISC is always asserted except when the BUS MD bus is being used to transfer data out of the PB IN register (EN PB IN true), or BUS MD data is being input from the PB (EN MAINT true).

EN MAINT is applied to the PB to gate data from the PB into the DP on the BUS MD bus. EN MAINT is asserted from the unsolicited SBI request logic in order to read the maintenance data register or the maintenance address register in the PB.

EN MD LO and EN MD HI gate the BUS MD low word and the BUS MD high word respectively, onto the BUS IB bus. Both signals are asserted by EN PB IN, thus gating the assembled longword from the PB IN register to the BUS IB. Both signals are also asserted when EN MAINT is true, thus gating the maintenance data from the PB onto the BUS IB. Both signals are again asserted by the LITERAL output of the source decoder.

When executing an unsolicited SBI read request (RD PENDING true), and if the PMCSR is being read (PMCSR true), both EN MD LO and EN MD HI are again asserted. If the unsolicited SBI request is to read the PSR (PSR true), only EN MD HI asserts.

EN MD HI is also asserted by the ANDing of EN LS/VCDT and EN VCDT. EN LS/VCDT is asserted by an unsolicited SBI read request of the LS or the VCDT (RD LS/VCDT true), or by the LS/VCDT SRC output from the source decoder as described in Paragraph 5.5.2 above.

Control signals EN VCDT, EN LS LO, and EN LS HI enables the VCDT, the low word section of LS, and the high word section of LS, respectively. Each of the three enabling signals are asserted from an OR gate. Each OR gate is fed from three AND gates, any of which can assert the particular enabling signal via the OR gate.

During an unsolicited SBI request, EXECUTE asserts and disables two of the three AND gates in each signal area. The third AND gate is used during the unsolicited SBI request. XBUS EN LS/VCDT LO from the unsolicited SBI request logic is applied to the unsolicited SBI request AND gates in the EN VCDT and the EN LS LO signal areas along with XBUS LSA 08. XBUX LSA 08 selects between the LS and the VCDT. If XBUS EN LS/VCDT LO is true and XBUS LSA 08 is true, the VCDT is enabled (EN VCDT asserts). If XBUS LSA 08 is false, the low section of LS is enabled (EN LS LO asserts).

XBUS EN LS/VCDT HI from the unsolicited SBI request logic is applied to the unsolicited SBI request AND gate in the EN LS HI signal area along with XBUS LSA 08. If XBUS EN LS/VCDT HI is true and XBUS LSA 08 is false, the high section of LS is enabled (EN LS HI asserts).

When the port is under microcode control (EXECUTE false), enabling of the VCDT, the low section of LS, and the high section of LS is done via the other two AND gates in the EN VCDT, EN LS LO, and EN LS HI signal areas. The two-bit S/D code generated in the LS/VCDT address selection logic is applied to the six AND gates to select the LS and the VCDT. INDEX 08 is used to select between the LS and the VCDT when necessary.

The logic in Figure 5-10 follows the S/D code in Tables 5-3 and 5-4 to enable the appropriate RAM area.

VALID PAR is asserted to the SBI module as a flag that data being transferred from the DP to the SBI module has been checked for parity. VALID PAR is negated by the IB SRC code when the BUS IB source is a LITERAL or the ALU (Table 5-4).* Neither the LITERAL field or the ALU output is checked for parity and the negation of VALID PAR flags the SBI module of this.

5.6 DP PARITY GENERATOR/CHECKER

A multipurpose parity generator/checker (Figure 5-11) performs most of the DP parity tasks. The parity generator/checker is connected to the IB IN bus.

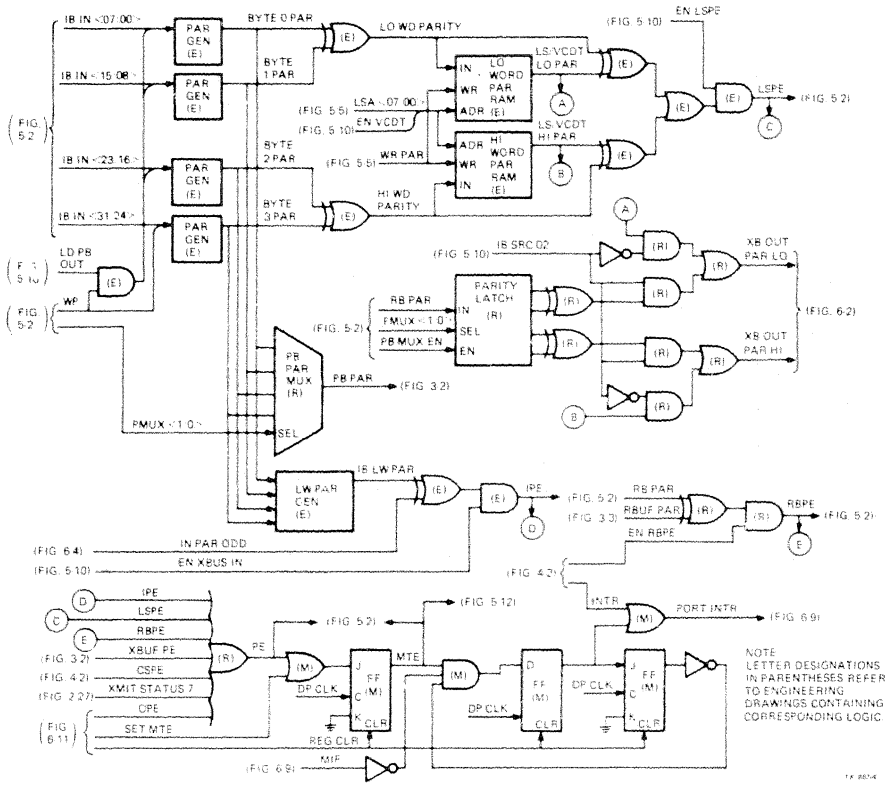


Figure 5-11 Parity Generator/Checker Logic

* EN XBUS IN also negates VALID PAR; however, this is a "don't care" condition as the SBI module is the BUS IB source and the data transfer is from the SBI module to the DP.

SEE NEXT FRAME

FOR LARGER ART

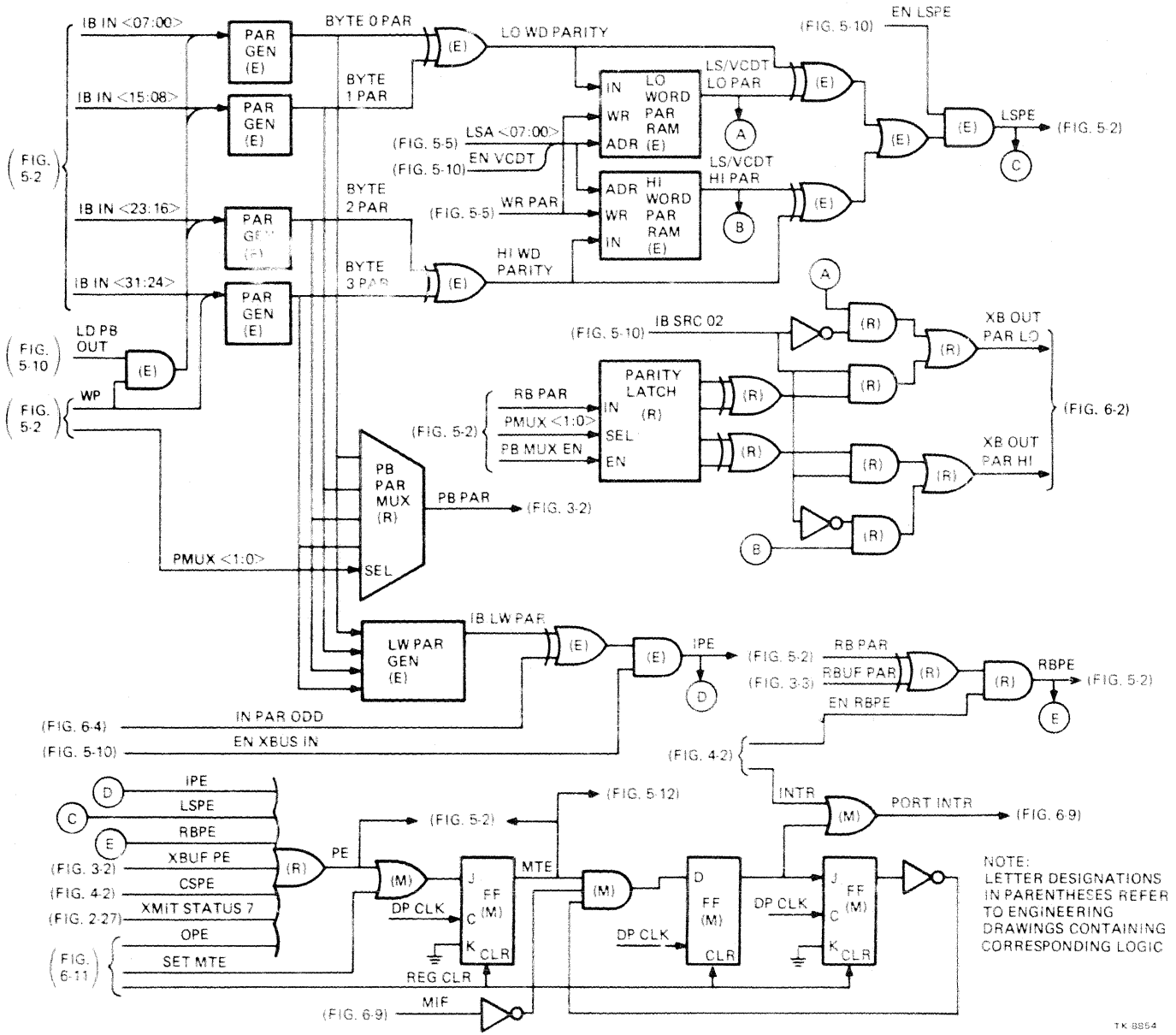


Figure 5-11 Parity Generator/Checker Logic

5.6.1 PB PAR

PB PAR are parity bits generated on data bytes output from the PB OUT register to the PORT DATA bus. PB PAR is sent to the PB along with the its associated data byte.

Data packets on the IB IN bus are in 32-bit longword format. The longword is input to the parity generator/checker logic where it is divided into bytes and input into four parity generators. Each generator outputs an odd parity bit (BYTE ⟨3:0⟩ PAR) generated on the associated input byte. The four parity bits are applied to a PB parity mux. The mux select input (PMUX ⟨1:0⟩) selects the parity bit for the mux output which is placed on the PB PAR line to the PB. PMUX ⟨1:0⟩ is the code that selects which byte of the longword is to be output from the PB OUT register onto the PORT DATA bus (Figure 5-2). Hence, the parity bit on the PB PAR line is for the data byte on the PORT DATA ⟨7:0⟩ bus.

A WP (wrong parity) bit can be input to each of the byte parity generators as a maintenance check of the parity logic. LD PB OUT is true during a data transfer from the PB OUT register to the PORT DATA bus; hence, the WP maintenance bit is applied to each of the byte parity generators.

5.6.2 Input Parity Error (IPE)

Parity is generated on the longword input from the SBI module and compared with the associated parity bit supplied by the SBI module. If the two do not compare, there is an input parity error and IPE asserts.

The four parity bits generated on the longword bytes (BYTE ⟨3:0⟩ PAR) are applied to a longword parity generator which generates a single parity bit for the entire longword. The longword parity bit (IB LW PAR) is XORed with the longword parity bit (IN PAR ODD) input from the SBI module. If the two XOR inputs do not compare and EN XBUS IN from the DP control logic is true, IPE asserts as a PMCSR bit and as an input to the PE OR gate.

The WP (wrong parity) bit is again used to generate a parity error as a maintenance check. With LD PB OUT false, only one WP bit is used for the input longword.

5.6.3 Local Store Parity Error (LSPE)

A parity bit is generated on each word written into LS and into the VCDT. The parity bits are stored in two RAMs. When the LS or the VCDT is read, parity is regenerated on the data read out and compared to the parity bits stored in the RAMs. If a match is not obtained, there is a local store parity error and LSPE asserts.

The BYTE 0 PAR and BYTE 1 PAR parity bits from the first two-byte parity generators are XORed to produce a single parity bit (LO WD PARITY) for the IB IN low word. Likewise, a single parity bit (HI WD PARITY) is produced for the IB IN high word. LO WD PARITY and HI WD PARITY are written into a low word parity RAM and a high word parity RAM, respectively. The two RAMs are addressed by LSA ⟨7:0⟩ from the LS/VCDT address selection logic to write the parity bits at the same address as the data being written into LS. EN VCDT is applied to the RAMs as the most significant address bit. While the VCDT is being written, EN VCDT is true thereby writing the VCDT parity bits in a separate location within the RAM. The RAM write strobe (WR PAR) is obtained from the write strobe logic in the LS/VCDT address select logic (Figure 5-5).

When the LS or the VCDT is read, the data read out appears on the IB IN lines for a parity check. The IB IN data generates a LO WD PARITY bit and a HI WD PARITY bit as described in the preceding paragraph. LSA ⟨7:0⟩ associated with the read, addresses the low-word parity RAM and the high-word parity RAM to access the parity bits stored when the LS/VCDT data (now being read) was written. Write strobe WR PAR is false thereby enabling the parity RAM outputs LS/VCDT LO PAR and LS/VCDT HI PAR.

LO WD PARITY and HI WD PARITY generated from the read data is XORed respectively with LS/VCDT LO PAR and LS/VCDT HI PAR from the parity RAMs. If the compared bits do not match, and EN LSPE is asserted from the DP control logic, LSPE is asserted as a bit in the PMCSR and as an input to the PE OR gate.

5.6.4 XB OUT PAR HI and XB OUT PAR LO

Parity is generated for the data being output from the DP to the SBI module. The data source may be the LS, the VCDT, or the PB IN register. The IB SRC code specifies the data source and, therefore, the source of the parity bits to be supplied with the data.

IB SRC 02 specifies the data source as shown in Table 5-4. When false, the data source is the LS or the VCDT. In this case, the LS/VCDT LO PAR and LS/VCDT HI PAR parity bits are read out of the low word parity RAM and the high word parity RAM, respectively. The bits are then gated out to the SBI module as XB OUT PAR LO and XB OUT PAR HI.

When IB SRC 2 is true, the data source is the PB IN register. In this case, PB MUX ENA is true and PMUX (1:0) selects the input byte for the PB IN register (Figure 5-2). PB MUX ENA also enables a parity latch which receives RB PAR from the PB IN parity generator. The PMUX (1:0) code places four RB PAR parity bits into the parity latch as the four data bytes are loaded into the PB IN register. The parity bits associated with the first two bytes are output from the parity latch and XORed to form the XB OUT PAR LO parity bit. The parity bits associated with the last two bytes are XORed to form XB OUT PAR HI.

5.6.5 Receiver Buffer Parity Error (RBPE)

Data from the PB into the PB IN register is checked for parity errors. A parity bit (RBUF PAR) is received from the PB along with the packet bytes. An RB PAR parity bit is generated for each byte loaded into the PB IN register. RB PAR is compared with RBUF PAR and if a match is not obtained, RBPE is asserted. RBPE is a bit in the PMCSR register and is also input to the PE OR gate logic. During valid data transfers EN RBPE from the PB is true to gate RBPE to its destinations.

5.6.6 Parity Error (PE)

PE is an OR function of seven port parity error bits. Included in the OR function are the three parity error bits generated in the parity generator/checker logic (IPE, LSPE, RBPE). Additional inputs are parity error bits from the PB transmit channel (XBUF PE) and control store RAMs (CSPE). Output parity error (OPE) from the SBI module and XMIT STATUS 7 (TDATA PARITY ERROR) from the link are also inputs to the PE OR gate.

PE is a bit in the PMCSR register. PE causes the assertion of MTE (maintenance error). MTE is applied to the boot timer logic (Figure 5-12) where it resets the boot timer and places the port into the uninitialized state. MTE can also be asserted by SET MTE from the SBI module.

If the maintenance interrupt flag (MIF) from the SBI module is false, the true state of MTE will cause PORT INTR to assert to the SBI module causing an interrupt on the SBI. A feedback loop negates PORT INTR after two DP CLK pulses. MTE remains true until REG CLR is asserted from the SBI module.

5.7 BOOT TIMER AND MAINTENANCE TIMER

The boot timer (Figure 5-12) is used to delay starting the CS microcode by holding the port in the uninitialized state. Boot jumpers select the delay which can be up to 1500 seconds in 100-second increments. The delay is used to allows time for the cluster to boot and to load the microcode.

Upon system power up, LOGIC CLR asserts from the SBI module causing CLR to assert. CLR directly sets the uninitialize flip-flop causing UNINIT to assert, placing the port into the uninitialized state.

Feedback from the output of the uninitialize flip-flop to the flip-flop input, holds the flip-flop in the reset state until CLR asserts. In the reset state, UNINIT is false, resulting in feedback that conditions the flip-flop to reset.

A boot timer one-second oscillator outputs into a decade counter at a one-cycle-per-second rate. The decade counter divides by 100 and outputs into a binary counter once every 100 seconds. A comparator compares the binary counter output with the count set into four boot jumpers. When the output from the binary counter matches the count set into the boot jumpers, the BTO (boot timeout) flip-flop sets and asserts BTO to the uninitialize flip-flop. BTO causes the uninitialize flip-flop to reset and negate UNINIT. The negation of UNINIT takes the port out of the uninitialized state and starts the microcode running.

The assertion of PICR WRT by an unsolicited SBI request from the host, also take the port out of the uninitialized state. Thus, the host can start up the microcode before the boot timeout period has expired.

The assertion of the maintenance error flag (MTE) from the parity generator/checker logic, also sets the uninitialize flip-flop and places the port into the uninitialized state. In addition, MTE clears the BTO decade counter and holds the BTO flip-flop reset so that the uninitialized state is maintained until MTE is cleared from the SBI module (Paragraph 5.6.6).

Other signals that clear the BTO flip-flop and decade counter are maintenance timer disable (MTD) from the PMCSR register, and PMTCR CLR from the SBI module. The BTO timeout period can be extended by clearing the boot timer with PMTCR CLR via an unsolicited SBI request.

A maintenance timer 400-microsecond oscillator outputs a TICK signal to the CS branching logic every 400 microseconds. TICK forms a time base used by the port microcode.

5.8 2901A MICROPROCESSOR

The DP contains eight 2901A microprocessor chips which constitutes the DP 2901A microprocessor. See Figures 5-13 and 5-14. The microprocessor functions are controlled by the microword from the CS. The following paragraphs discuss the 2901A microprocessor and its operations.

5.8.1 Data Path

Eight 2901As are used in parallel to formulate a 32-bit longword input/output for the microprocessor. The 2901A contains two 16 × 32 RAMs, an arithmetic logic unit (ALU), a Q register, and control circuitry. The RAMs are used as a scratch pad where the results of arithmetic and logical operations are stored temporarily for future use. The contents of the RAM are gated into the ALU by the source control signals supplied from the CS microword.

The two 16 × 32 RAMs are designated as RAM A and RAM B. The RAM A and RAM B address lines are tied together and are addressed by ALU A/B (3:0) from the CS microword. Hence, both RAMs address the same location simultaneously.

The 2901A operates such that when RAM B is written, RAM A is also written. When writing RAM B, the RAM B select lines select both the RAM B and RAM A internal address. Data presented at the data input is written at a location dependent on the ALU destination code from the microword.

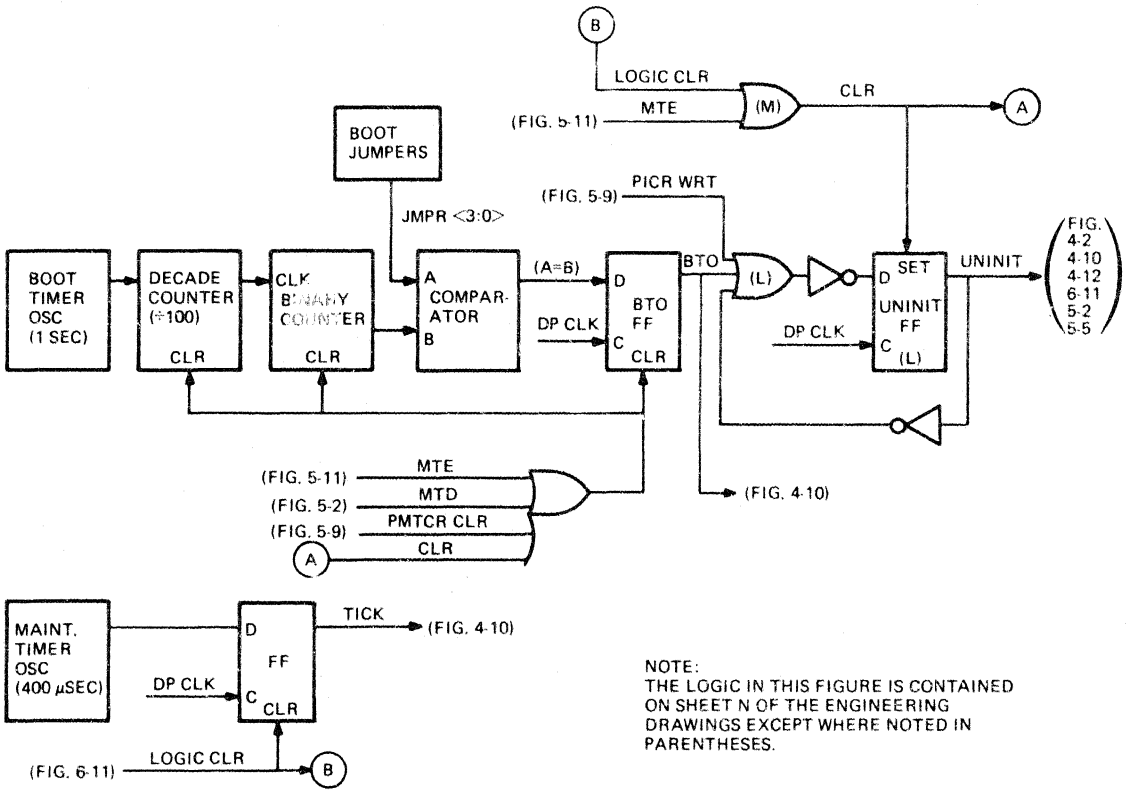


Figure 5-12 Boot Timer and Maintenance Timer

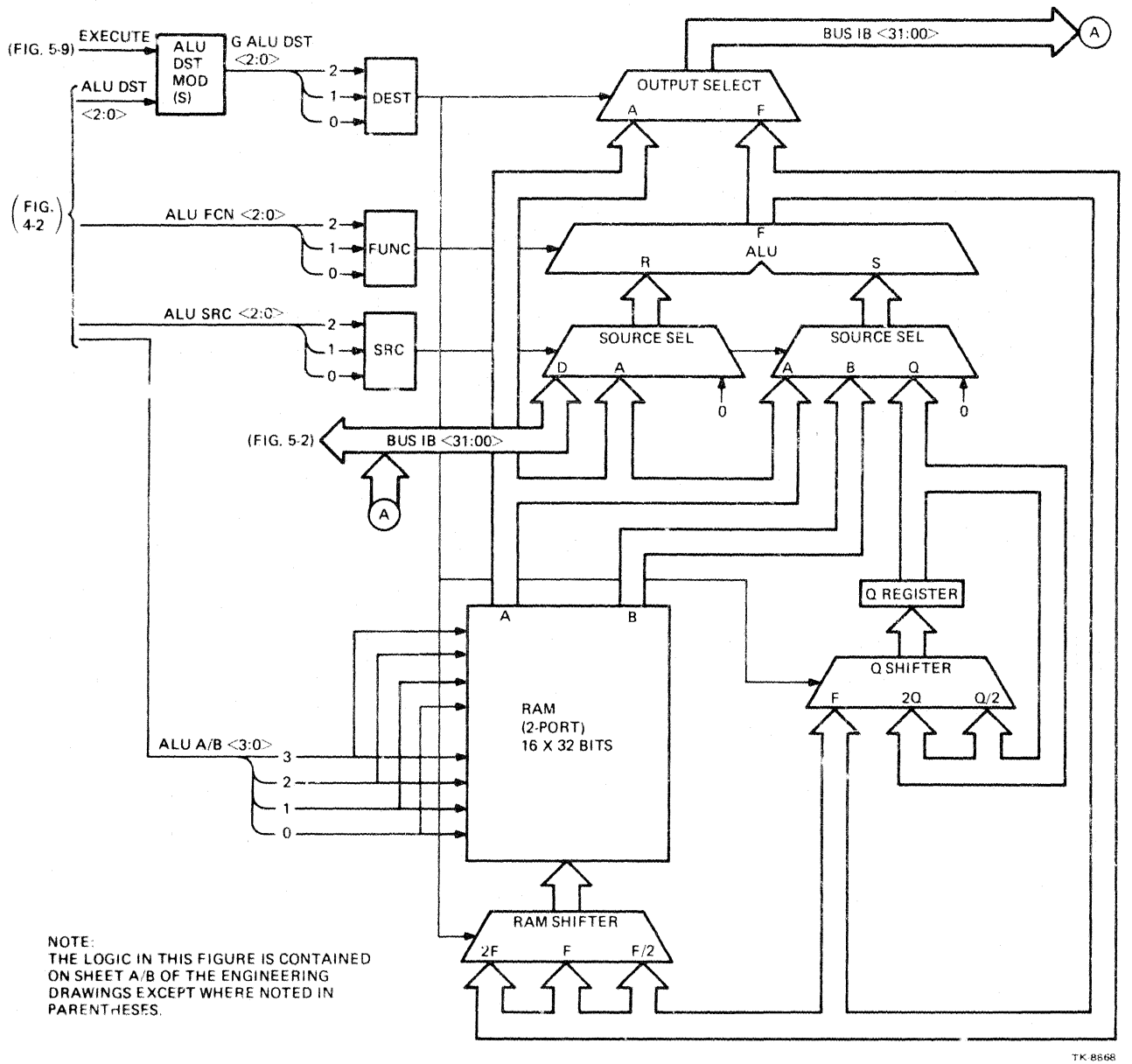


Figure 5-13 2901A Microprocessor Simplified Block Diagram

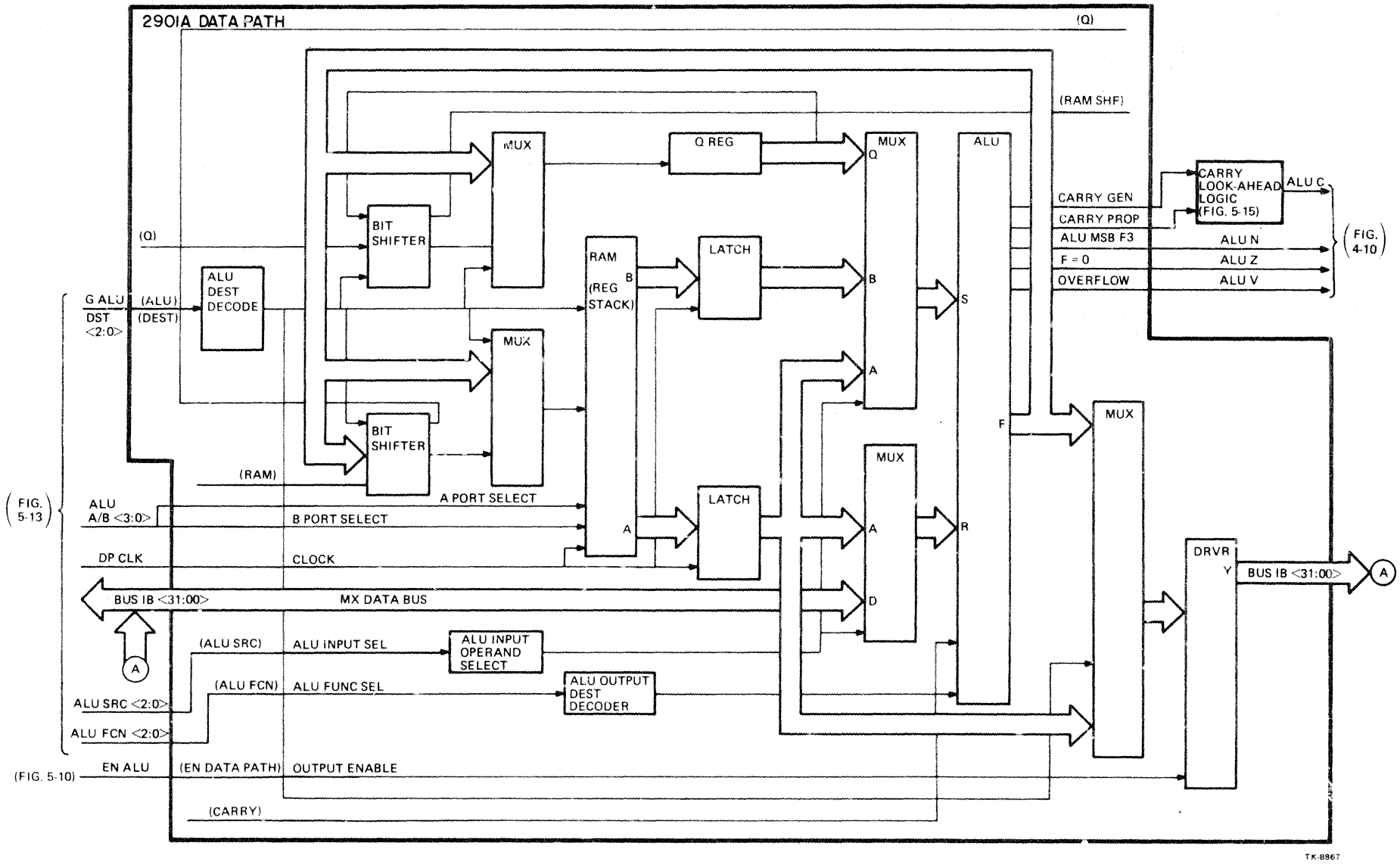


Figure 5-14 2901A Microprocessor Block Diagram

The high-speed ALU can perform three binary arithmetic and five logic operations on the two input words, R and S. The R input field is driven from a two-input mux, while the S input field is driven from a three-input mux. Both muxes have an inhibit capability; that is, no data is passed. This is equivalent to a zero source operand.

The ALU R-input mux has the RAM A port and the BUS IB bus connected as inputs. The ALU S-input mux has the RAM A port, the RAM B port, and the Q register as inputs.

The mux can select various combinations of input pairs among the A, B, D, Q, and zero inputs as source operands to the ALU. The microinstruction inputs used to select the ALU source operands are ALU SRC ⟨2:0⟩. The ALU source bits are defined in Table 5-5.

Table 5-5 ALU Source Code

Mnemonic	ALU SRC 2	1	0	Octal Code	ALU Source R	S
AQ	0	0	0	0	A	Q
AB	0	0	1	1	A	B
ZQ	0	1	0	2	0	Q
ZB	0	1	1	3	0	B
ZA	1	0	0	4	0	A
DA	1	0	1	5	D	A
DQ	1	1	0	6	D	Q
DZ	1	1	1	7	D	0

The D and Q source operands are described as follows. The D input is the direct data input from the BUS IB bus. This port is used to insert all data into the working registers inside the 2901A data path. The Q input from the Q register is a separate file used as an accumulator or holding register.

The ALU destination is selected by the ALU DST code from the microword. During an unsolicited SBI request function (EXECUTE true), the ALU DST field is altered to force a null destination code. This is done to prevent any erroneous data from being written internally within the microprocessor.

The ALU functions are selected by ALU FCN (2:0) from the microword. The ALU function bits are defined in Table 5-6.

The ALU has four status outputs: carry out (ALU C), sign bit F3 (ALU N), zero bit F=0 (ALU Z), and overflow (ALU V). ALU C is used as the carry flag. ALU N is the most significant digit of the ALU and is used to determine positive or negative results without enabling the tri-state outputs. ALU Z is used for zero detection. ALU Z is asserted when all the F outputs are low. ALU V is used to flag arithmetic operations that exceed the available 2's complement number range. The four status outputs are applied to the branching logic in the CS.

The Q register is a file loaded from the ALU and used as a temporary storage register. The Q register output can be loaded back into itself and shifted right or left as during fraction, multiplication, and division operations.

Table 5-6 ALU Function Code

Mnemonic	ALU FCN 2	1	0	Octal Code	ALU Function	Symbol
ADD	0	0	0	0	R plus S	R + S
SUBR	0	0	1	1	S minus R	S − R
SUBS	0	1	0	2	R minus S	R − S
OR	0	1	1	3	R OR S	R ∨ S
AND	1	0	0	4	R AND S	R ∧ S
NOTRS	1	0	1	5	Not R AND S	R ∧ S
EXOR	1	1	0	6	R EXOR S	R ∨ S
EXNOR	1	1	1	7	R EXNOR S	R ∨ S

5.8.2 Data Manipulation

After data is loaded into the microprocessor, both the Q register and any RAM address can be rotated or shifted left or right. During a rotate, the bit transferred out one end is transferred in on the other end. During a shift operation, the bit shifted out is lost and a new bit is generated and shifted in at the far end. To accomplish these shifts and rotations, the most significant bit (MSB) of each four-bit 2901A is connected to the least significant bit (LSB) of the adjacent 2901A via a bidirectional transfer line. To complete the wraparound required to rotate data, the MSB of the entire 32-bit longword is connected to the LSB via a bidirectional transfer line.

5.8.3 Carry Look-Ahead Logic

Circuitry associated with the 2901As contains full carry look-ahead logic that speeds the execution of arithmetic instructions and allows the data path to function with full 32-bit carry look-ahead generation. Figure 5-15 illustrates this logic. Each of the 2901A chips generates both a carry generate output (GEN) and a carry propagate output (PROP). The four pairs of GEN and PROP signals for bits ⟨15:00⟩ are combined in a carry skipper along with a C IN signal derived from ALU function codes ALU FCN ⟨1:0⟩. The sum of the outputs of the carry skipper (ALU C16) go to another carry skipper and are combined with the GEN and PROP signals from the bit ⟨31:16⟩ 2901As. The output of the second carry skipper is combined to output carry status bit (ALU C) to the CS branching logic.

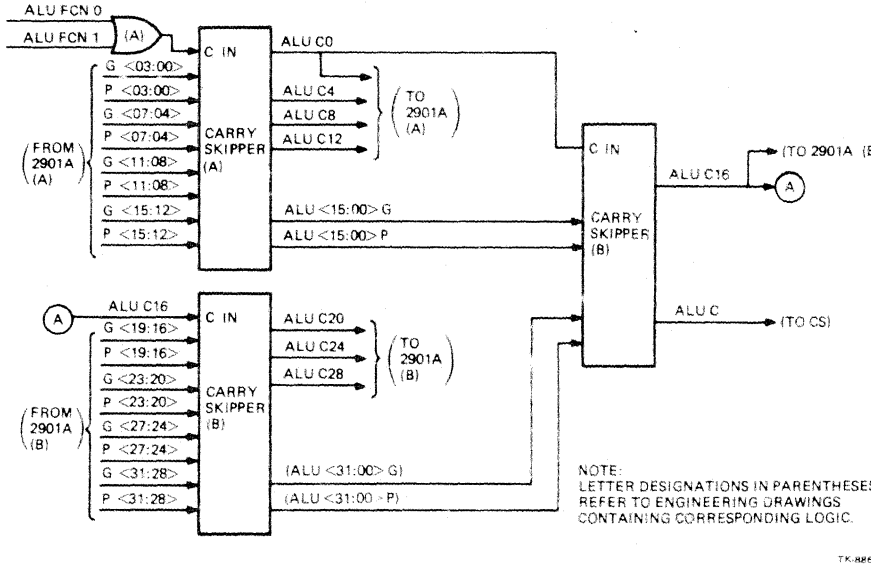


Figure 5-15 Carry Look-Ahead Logic

SEE NEXT FRAME
FOR LARGER ART

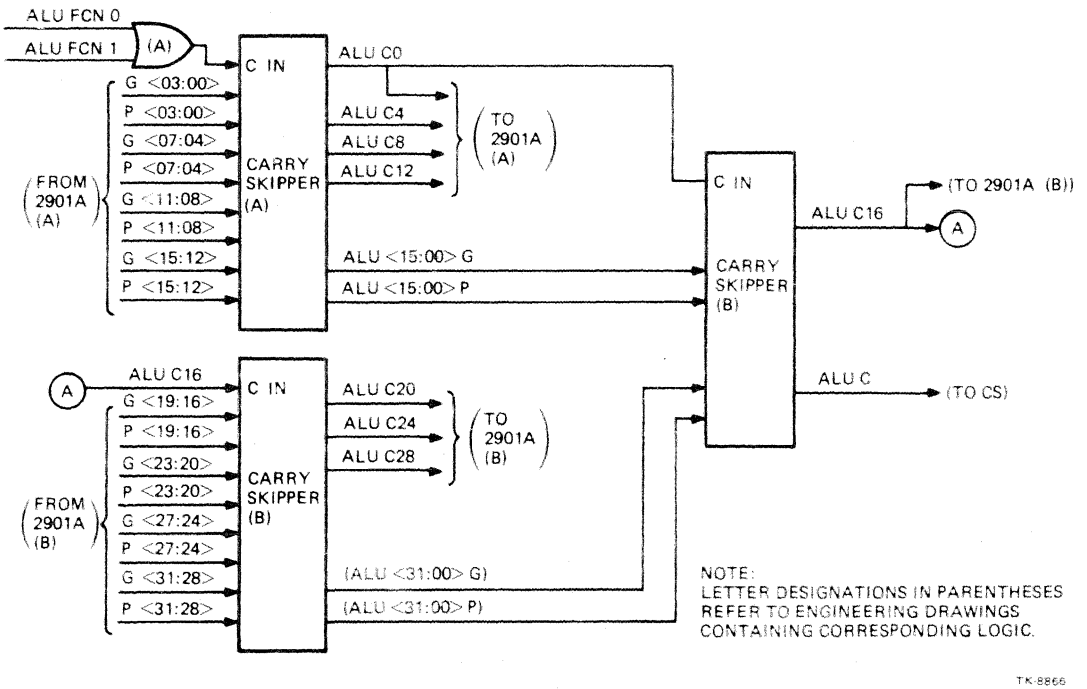


Figure 5-15 Carry Look-Ahead Logic

CHAPTER 6
SBI MODULE

NOTE

The functional block diagrams in Chapter 6 use logical AND and OR symbols. It does not necessarily follow that a corresponding gate exists on the SBI module logic prints. The assertion of inputs A and B causing the assertion of output C may be represented on a block diagram by a single AND gate, yet the engineering drawing may show that several circuit stages are involved in the ANDing operation.

The functional block diagrams in this chapter are keyed to the SBI module engineering circuit schematics (CS prints) by letter designation in parentheses. The letters specify the sheet of the CS prints that contains the detailed logic associated with the functional blocks in the diagram.

The signal names used in the functional block diagrams are the names used on the engineering CS prints. Where other signal names or notes are used, they are enclosed in parentheses.

6.1 SBI OVERVIEW

Figure 6-1 is a simplified block diagram of the SBI module. Note that the SBI module interfaces with the SBI bus via SBI transceivers. Signals shown in the block diagrams in this chapter (and on the engineering logic prints) will be prefixed with a T if they are being transmitted to the SBI, or with an R if they are being received from the SBI.

Data transfers within the SBI module are port initiated or unsolicited SBI requests.

Port-initiated transfers are controlled by the microcode. The microcode commands a transmission of data from the DP to the SBI, or a read of data from the SBI to the DP. The data is transmitted to or read from a device on the SBI.

Unsolicited SBI requests are initiated by the host CPU to read or write a DP register. The host controls the transfer sequence.

An xmit file and a receive file, each capable of holding four longwords of data, are used for port-initiated data transfers. The files are divided into A and B sections with each section capable of holding a quadword of data (two longwords). The files have separate read and write address pointers. The xmit file is used when data is being written to the SBI from the DP. The receive file is used when data is being read from the SBI to the DP.

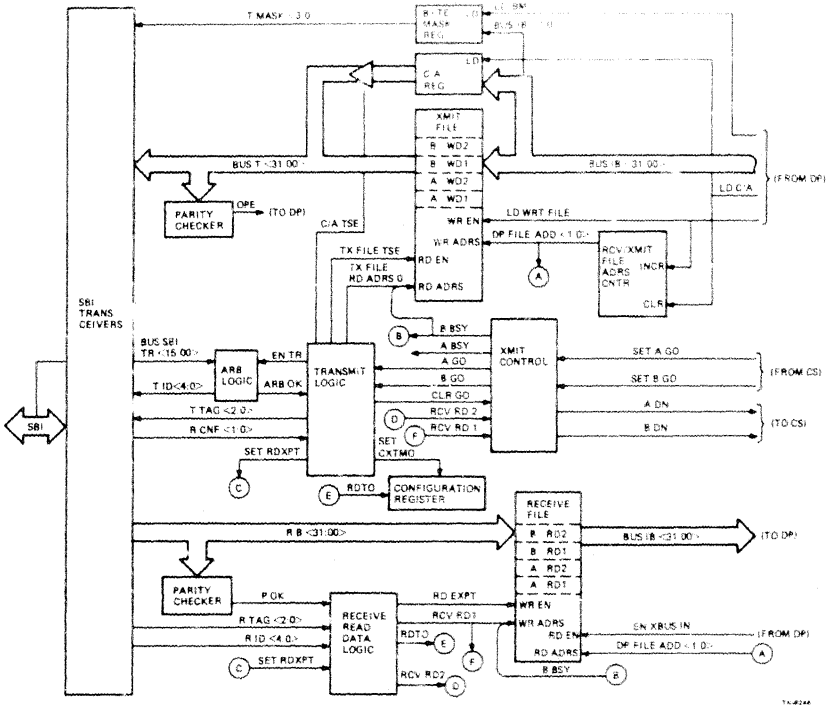


Figure 6-1 SBI Module Block Diagram

6.1.1 Write Transfers

For a port-initiated write transfer, LD C/A (command/address) is asserted by the microcode and loads a C/A register with the write command and the SBI address that is to be written. The command/address is obtained from the DP over the BUS IB.

The microcode then asserts LD WRT FILE to load the xmit file with a longword of data from the BUS IB.

The xmit file is write addressed by DP FILE ADD (1:0) from a two-bit file address counter. LD C/A from the microword resets the file address counter to zero when the C/A register is loaded. Thus, when the first data longword is loaded into the xmit file, it is placed in the first location (A WD1).

LD WRT FILE increments the file address counter to point to location A WD2. Thus, as each longword is loaded, the file pointer is incremented to the next location. After all four locations are filled, the microcode will not attempt to load any more data until the A section of the file has been read out to the SBI and there is room in the file for new data.

The byte mask associated with the data longword is loaded into the byte mask register by LD BM asserted by the microword. The byte mask (BUS IB (7:0)) is brought into the byte mask register from the DP over the BUS IB.

SEE NEXT FRAME

FOR LARGER ART

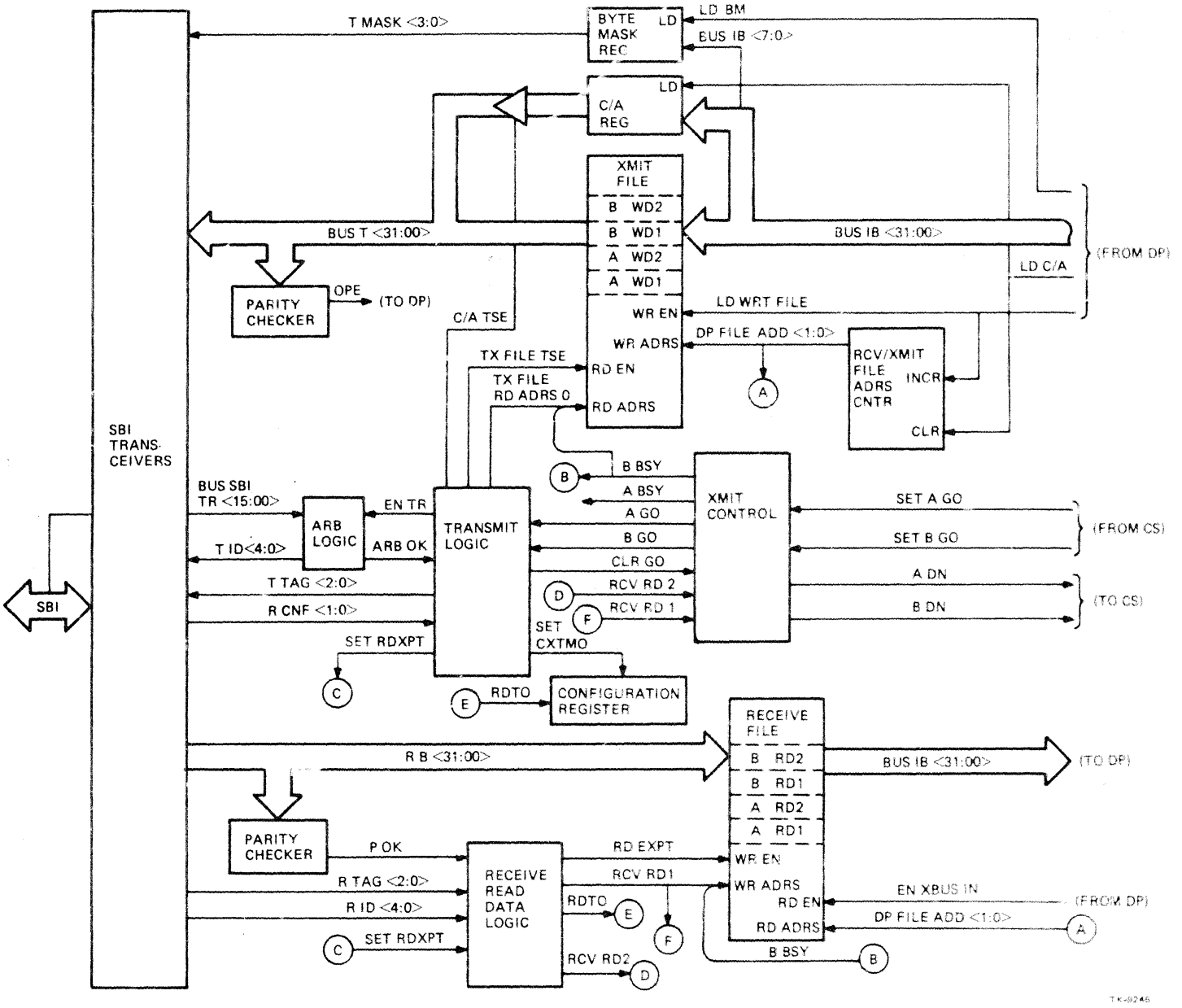


Figure 6-1 SBI Module Block Diagram

After loading the C/A register, the A section of the xmit file, and the byte mask register, the microcode asserts SET A GO to the xmit control to start the xmit file unload sequence. A GO is asserted to the transmit logic where the write operation is controlled. The xmit control asserts A BSY indicating that the A section of the file is busy being unloaded. While the A section is being unloaded (read), the B section is free to be loaded (written) with more data longwords from the DP.

The transmit logic initiates arbitration for the SBI bus by asserting EN TR to the arbitration logic. When the CI780 has won the SBI bus, the arbitration logic returns ARB OK to the transmit logic.

The transmit logic asserts C/A TSE (transfer enable) to transfer the command/address from the C/A register to the SBI bus. The transmit logic then proceeds to read the xmit file by asserting TX FILE TSE which enables the file output to the SBI bus.

The xmit file read address is a two-bit pointer in which TX FILE RD ADRS 0 from the transmit logic is the least significant bit, and B BSY from the xmit control is the most significant bit. The address pointer initially points to location A WD1.

After the first longword is read out, TX FILE RD ADRS 0 asserts to address the next location in the file (A WD2).

The tag and ID associated with the longword is also sent out to the SBI.

After the data has been placed on the SBI, the transmit logic looks for an acknowledge (ACK) confirmation from the receiving device. When the ACK confirmation (R CNF (1:0)) is received, the transmit logic resets the GO bit by asserting CLR GO to the xmit control.

When the GO bits resets, A DN is sent to the CS branching logic to indicate that the transfer is complete.

If the transmit logic does not receive an ACK confirmation, it re-tries the operation. If, after 102 microseconds of re-trying, the ACK confirmation has not been received, the transmit logic aborts the transfer and sets a timeout error bit (CXTMO) in the configuration register (see Appendix C; Paragraph C.4).

6.1.2 Read Transfers

For a port-initiated read transfer, LD C/A is asserted by the microcode and loads the C/A register with the read command and the SBI address that is to be read. The command/address is obtained from the DP over the BUS IB.

After loading the C/A register, the microcode asserts SET A GO to the xmit control to start the sequence that will load the A section of the receive file with the read data from the addressed device. A GO asserts to the transmit logic where the read operation is controlled. The xmit control asserts A BSY to indicate that the A section of the file is busy being loaded. While the A section is being loaded (written), the B section is free to be unloaded (read) by the microcode.

The transmit logic initiates arbitration for the SBI bus by asserting EN TR to the arbitration logic. When the CI780 has won the SBI bus, the arbitration logic asserts ARB OK to the transmit logic which then asserts C/A TSE to transfer the command/address from the C/A register to the SBI bus. The transmit logic then looks for an ACK confirmation (R CNF (1:0)) from the SBI. When the ACK confirmation is received, the transmit logic shifts control of the read sequence to the receive read data logic which looks for the read data to appear on the SBI. When the read data appears, the receive read data control logic checks the ID field (that the data is for the CI780), checks the tag field (that it is read data), and the parity. If everything checks good, the logic asserts RD EXPT to write the data longword into the receive file.

If the read data does not appear within 102 microseconds, the receive read data logic asserts a read data timeout (RDTO) signal which sets a bit in the configuration register.

The receive file is write addressed by a two-bit pointer with RCV RD1 being the least significant bit and B BSY being the most significant bit. The pointer initially points to location A RD1.

After the first longword is written, the receive read data logic asserts RCV RD1 which increments the address pointer to the next file location (A RD2).

If this is a simple read (of one longword), RCV RD1 negates A GO in the xmit control. If this is an extended read (of a quadword), RCV RD2 asserts when the second longword is received. In this case, RCV RD2 is used to negate A GO.

When A GO negates, A DN is sent to the CS branching logic to indicate that the transfer is complete.

6.2 PORT-INITIATED TRANSFERS

6.2.1 SBI Extended Write Transfer

An extended write transfer is the transfer of a quadword of data from the DP to the SBI bus. Figure 6-2 is a block diagram of the logic involved in a write transfer. Figure 6-3 is a flow diagram illustrating the sequence of events in an extended write transfer.

The microcode initiates the transfer by asserting LD C/A. LD C/A loads a C/A register and counter with the command and address (R IB ⟨31:00⟩) from the BUS IB in the DP. The register and counter are loaded via an IB receive register which latches the BUS IB data for 200 ns. Bits R IB ⟨31:28⟩ specify the command to be executed (in this case an extended write); bits R IB ⟨27:00⟩ specify the address on the SBI bus where the quadword is to be written. Bits R IB ⟨8:1⟩ of the address field are loaded into the C/A counter which is incrementd by INC ADRS from the transmit logic after each transfer. Thus, the C/A register does not have to be reloaded for each transfer when doing multiple transfers. The microcode reloads the C/A register with a new page address whenever a page boundry is crossed.

LD C/A clears the file address counter so the xmit file write address bits (DP FILE ADD ⟨1:0⟩) point to location A WD1 in the xmit file.

A data quadword is then loaded into the xmit file by LD WRT FILE from the DP. LD WRT FILE asserts TX FILE WT EN which loads the first longword into location WD1. LD WRT FILE also increments the file address counter to address A WD2. LD WRT FILE then reasserts to place the second longword into A WD2 and to increment the file address counter to address the first longword location in the B section of the file.

After the data quadword is in the xmit file, LD BM (load byte mask) from the DP asserts and loads the byte mask (R IB ⟨7:0⟩) into the byte mask register. R IB ⟨7:0⟩ contains two four-bit byte masks, one for each longword in the xmit file.

NOTE

If all the quadword bytes are to be used, the byte mask register need not be loaded as the logic will automatically assume a mask of all 1's.

The microcode asserts SET A GO to start the transfer sequence. SET A GO sets a flip-flop causing A GO and A BSY to assert. A GO is applied to the transmit logic which controls the data transfer. A BSY indicates that the A section of the xmit file is busy.

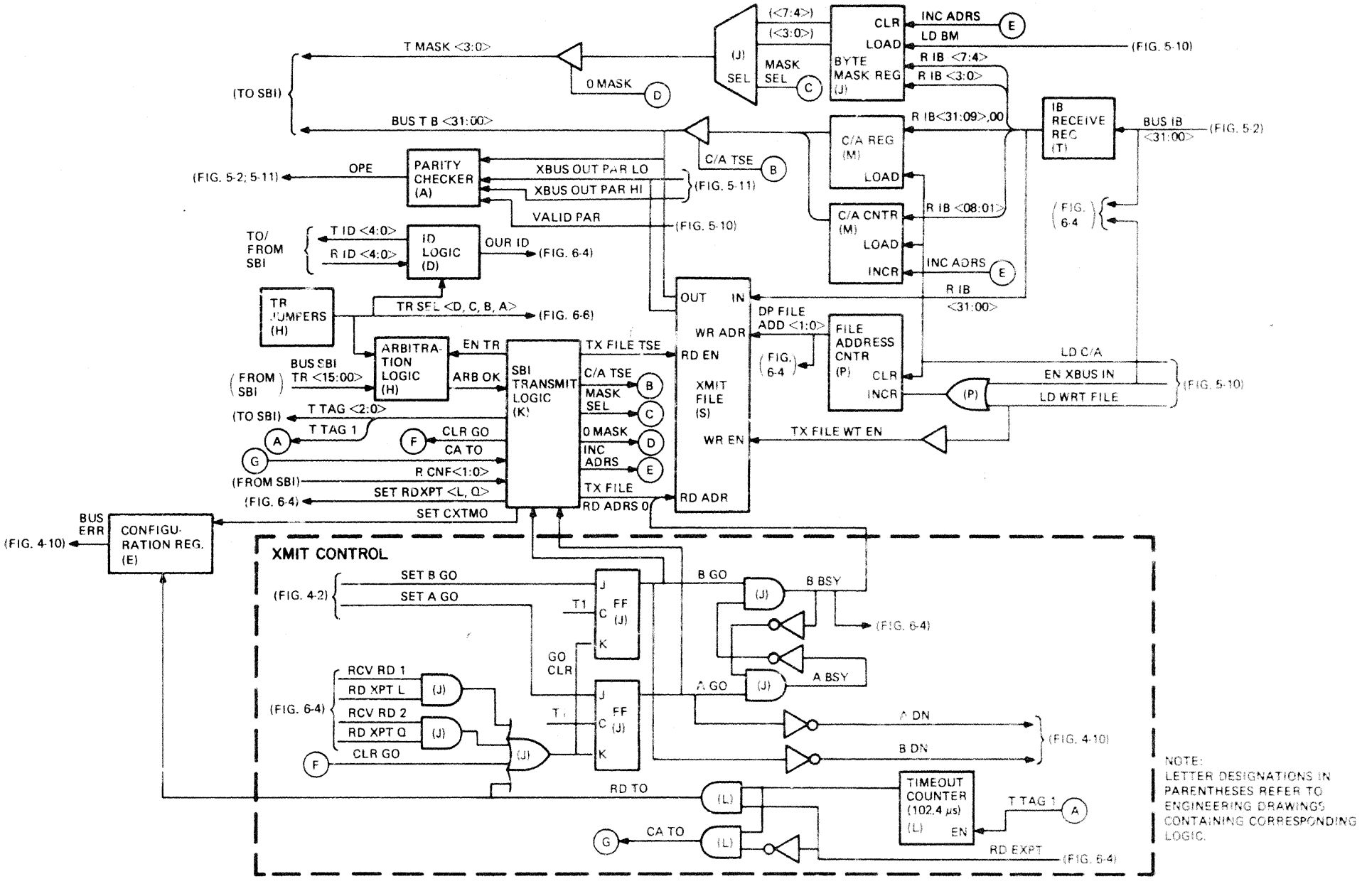


Figure 6-2 Write Transfer Block Diagram

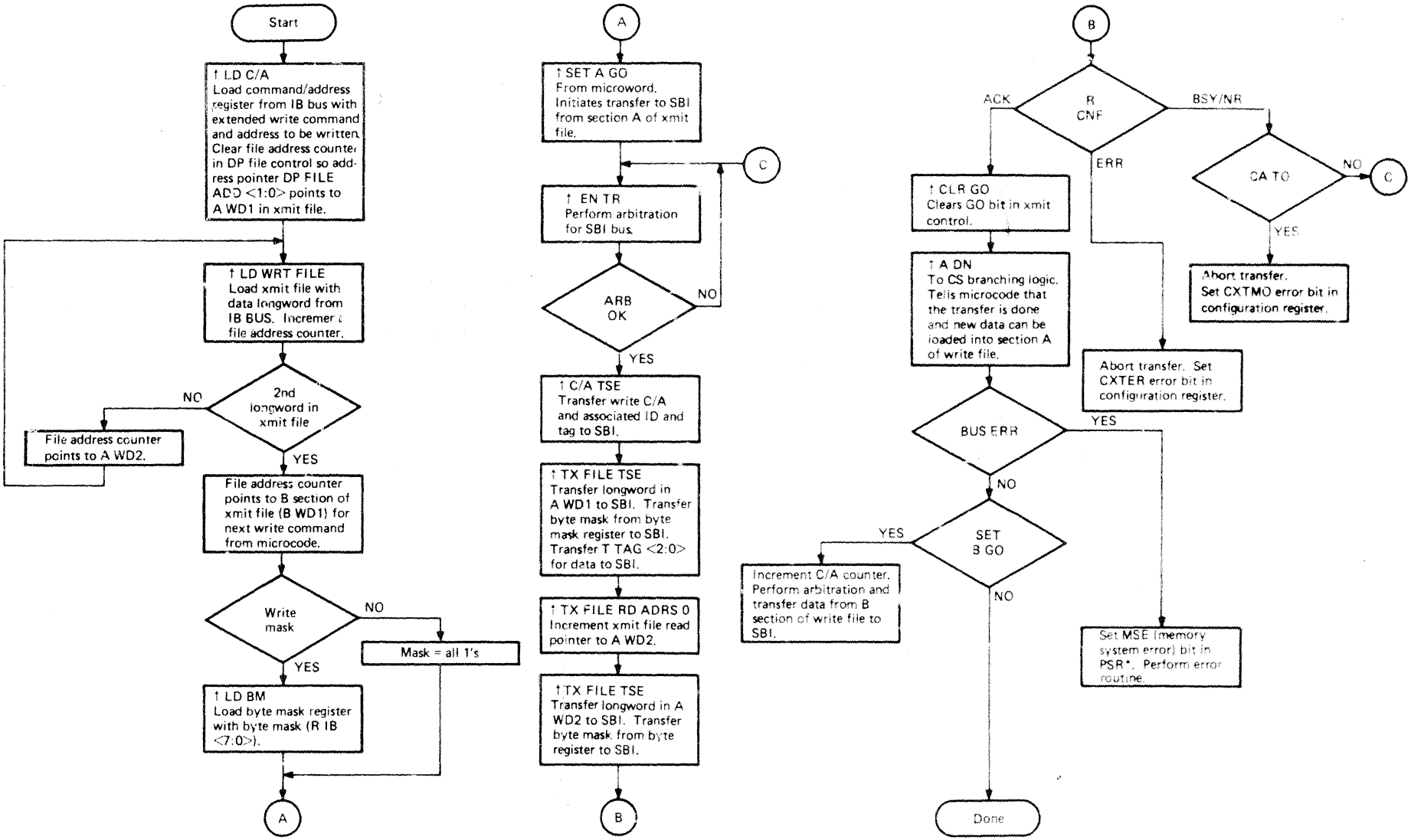


Figure 6-3 Extended Write Flow Diagram

When the transmit logic receives A GO from the xmit control, it asserts EN TR to enable the arbitration logic. Four TR jumpers generate TR SEL (D,C,B,A) to establish the priority level of the CI780 port. The arbitration logic looks at the SBI arbitration field (BUS SBI TR (15:00)) from the SBI and compares the level of any pending TR requests with the level established by the TR jumpers. If there are no TR requests pending at a higher level then that established by the TR jumpers, the port wins the SBI bus and the arbitration logic asserts ARB OK to the transmit logic.

The transmit logic responds by asserting C/A TSE (transfer enable) to transfer the command/address from the C/A register out to the SBI bus. The transmit logic outputs the tag identifying the SBI data as a command/address. Also the TR SEL priority code is output to the SBI (via the ID logic) as the ID field (T ID (4:0)).

The transmit logic then asserts TX FILE TSE which reads the longword in A WD1 out of the xmit file as BUS T B (31:00) and then to the SBI bus. The transmit logic outputs the tag identifying the longword as write data. In addition, the transmit logic outputs MASK SEL to select the four-bit byte mask associated with the longword. The transmit logic then asserts 0 MASK to gate the mask (T MASK (3:0)) to the SBI bus.

The transmit logic next asserts TX FILE RD ADRS 0 to increment the xmit file pointer to address location A WD2. TX FILE TSE then asserts again to transfer the second longword to the SBI bus. The transmit logic then selects the other four-bit byte mask from the byte mask register and places it on the SBI bus.

After transmitting the C/A, the data, and the associated information fields onto the SBI bus, the transmit logic checks the confirmation response (R CNF (1:0)) from the receiving device. If a no response or a busy confirmation is received, the transmit logic retires the extended write transfer on the SBI. If, after 102 microseconds of re-trying, an error or ACK confirmation response still has not been received, the transmit logic aborts the write operation and asserts SET CXTMO to the configuration register. SET CXTMO sets a command/address timeout error bit (CXTMO) in the configuration register.

If an error confirmation is received, the operation is immediately aborted and the CXTER error bit set in the configuration register.

If a positive confirmation (ACK) is received, the transmit logic asserts CLR GO to the xmit control logic to clear the GO bit.

A timeout counter establishes the "no response" and busy timeout periods. The counter is enabled by T TAG 1 when the C/A tag is asserted to the SBI. When the timeout period has expired (102.4 μs) the counter output is asserted causing CA TO to go true.

CLR GO from the transmit logic is applied to the xmit control where it asserts GO CLR to the A GO flip-flop. GO CLR resets the flip-flop negating A GO to the transmit logic and asserting A DN to the microcode branching logic in the CS. When the microcode senses the presence of A DN, it knows that section A of the xmit file is ready to receive more data.

Also, A BSY negates, thereby allowing B BSY to come true when the B GO bit sets.

After the extended write transfer is complete, the BUS ERR bit in the configuration register is checked by the microcode branching logic. If BUS ERR is true, an error occurred during the extended write transfer just completed.

6.2.2 SBI Extended Read Transfer

An extended read transfer is the transfer of a quadword of data from some device on the SBI to the DP. Figures 6-2 and 6-4 are block diagrams of the logic involved in a read transfer. Figure 6-5 is a flow diagram illustrating the sequence of events in an extended read transfer.

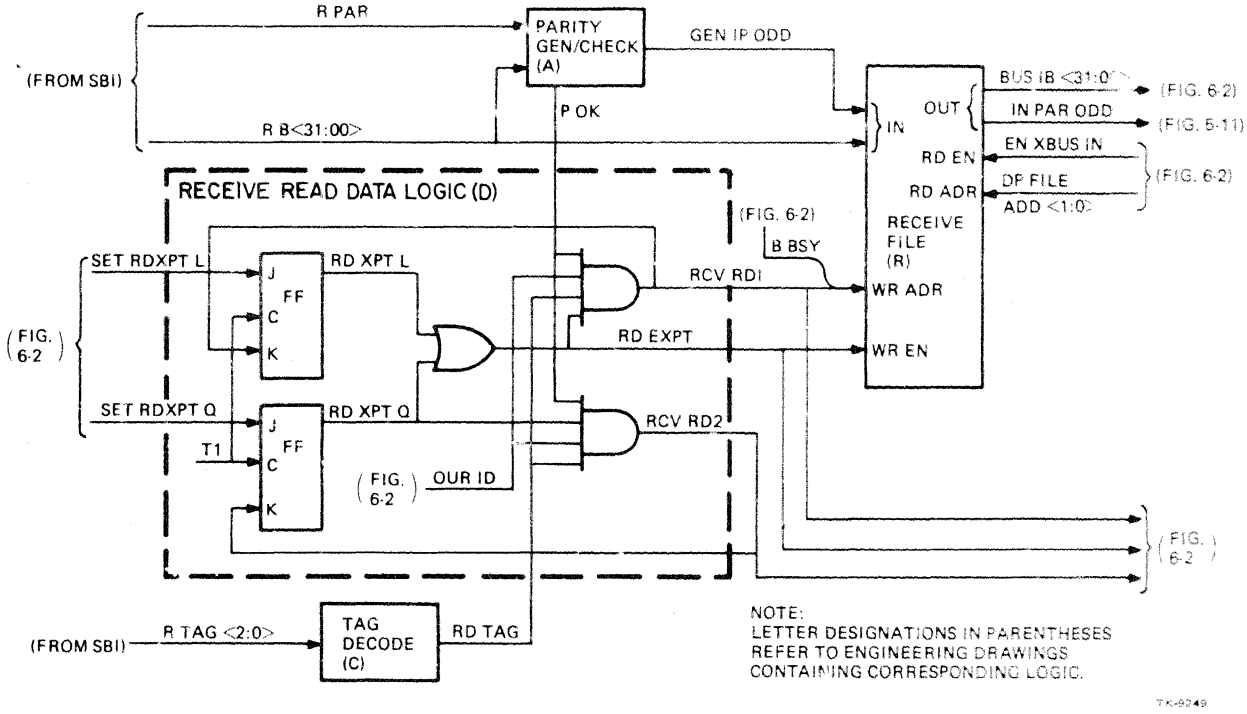


Figure 6-4 Read Transfer Block Diagram

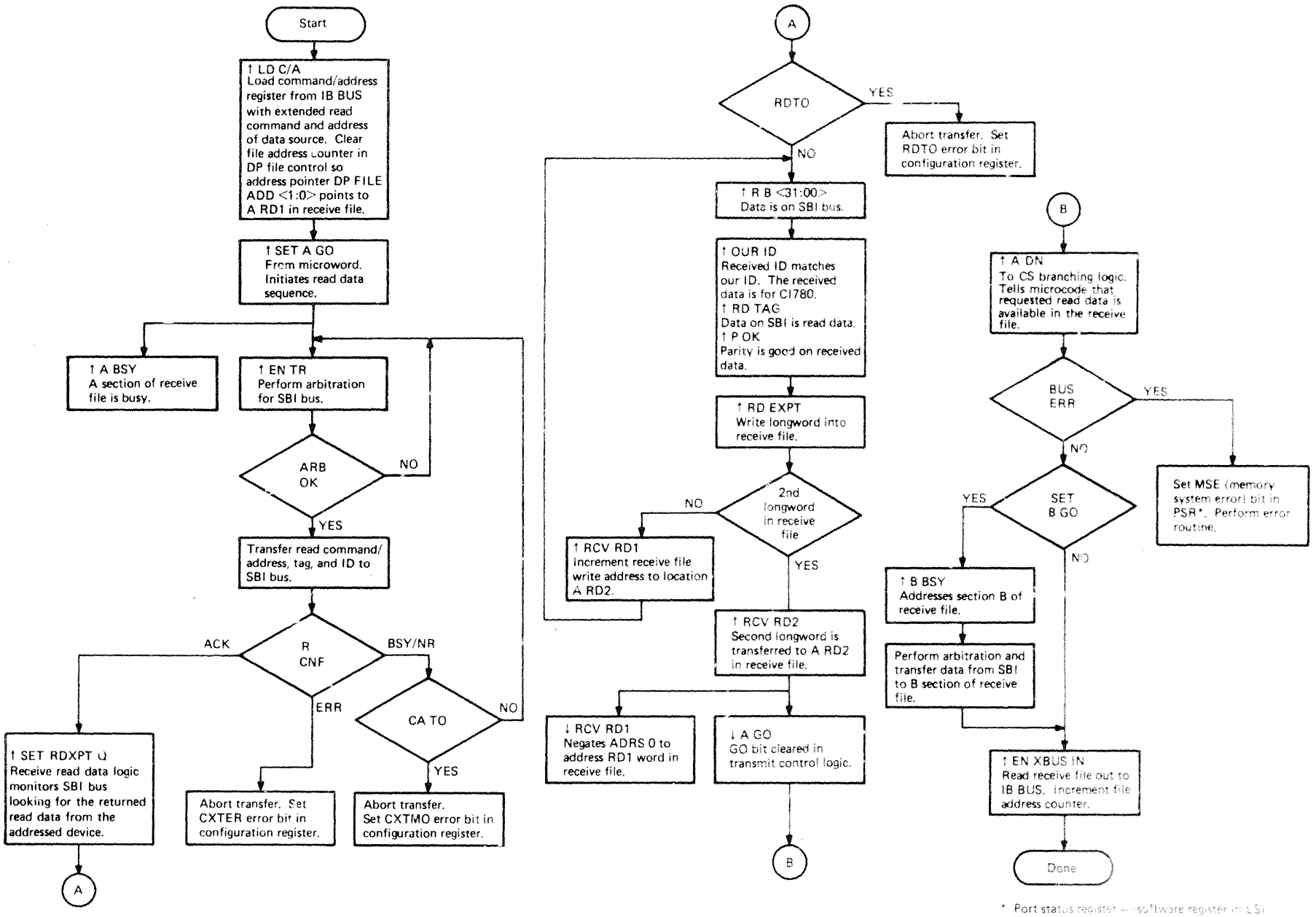


Figure 6-5 Extended Read Flow Diagram

The microword initiates the transfer by asserting LD C/A. LD C/A loads the C/A register and counter with the command and address (R IB ⟨31:00⟩) from the BUS IB in the DP. The register and counter are loaded via an IB receive register which latches the BUS IB data for 200 ns. Bits R IB ⟨31:28⟩ specify the command to be executed (in this case an extended read); bits R IB ⟨27:00⟩ specify the SBI bus address of the device to be read. Bits R IB ⟨8:1⟩ of the address field are loaded into the C/A counter which is incremented by INC ADRS after each transfer. Thus, the C/A register does not have to be reloaded for each transfer when doing multiple transfers. The microcode reloads the C/A register with a new page address whenever a page boundry is crossed.

LD C/A also clears the file address counter so the receive file read address bits (DP FILE ADD ⟨1:0⟩) point to location A RD1 in the receive file.

The microcode asserts SET A GO to start the transfer sequence. SET A GO sets a flip-flop causing A GO and A BSY to assert. A GO is applied to the transmit logic to initiate the arbitration process for the SBI bus. A BSY indicates that the A section of the receive file is busy.

The transmit logic asserts EN TR to enable the arbitration logic. Arbitration for the SBI bus proceeds as in the case of an extended write transfer. When the port has won the SBI bus, the arbitration logic returns ARB OK to the transmit logic.

The transmit logic responds by asserting C/A TSE to transfer the command/address from the C/A register out to the SBI bus. The transmit logic outputs the tag identifying the SBI data as a command/address. Also the TR SEL priority code is output to the SBI (via the ID logic) as the ID field (T ID ⟨4:00⟩).

After transferring the C/A and the associated tag and ID fields onto the SBI bus, the transmit logic checks the confirmation response (R CNF ⟨1:0⟩) from the addressed SBI device. If a no response or a busy confirmation is received, the transmit logic re-tries the extended read transfer on the SBI. If, after 102 microseconds of re-trying, an error or ACK confirmation response still has not been received, the transmit logic aborts the read operation and asserts SET CXTMO to the configuration register. SET CXTMO sets a command/address timeout error bit (CXTMO) in the configuration register.

If an error confirmation is received, the operation is immediately aborted and the CXTER error bit set in the configuration register.

If a positive confirmation (ACK) is received, the transmit logic asserts SET RDXPT L (read data expect longword) to the receive read data logic. SET RDXPT transfers control of the read sequence to the receive read data logic.

As shown in Figure 6-4, SET RDXPT L from the transmit logic sets a flip-flop asserting RD XPT L which in turn asserts RD EXPT. RD EXPT enables the receive file to write the received data longword into location A RD1.

When the data appears, the associated ID field is compared with the TR SEL code from the arbitration TR jumpers. If a match is obtained, OUR ID asserts to the receive read data logic indicating that the read data is for the CI780.

The tag field is decoded by a tag decode circuit. If the tag indicates that the data is read data, RD TAG asserts to the receive read data logic.

The incoming data is checked for parity in a parity checker. If there are no parity errors, P OK is asserted to the receive read data logic.

If the ID, the tag, and the parity are all good, the receive read data logic asserts RCV RD1. RCV RD1 is the least significant bit of the receive file write address pointer. Asserting RCV RD1 increments the write pointer to location A RD2. RCV RD1 is then negated via feedback to the RD XPT L flip-flop causing it to reset.

For an extended read sequence (Figure 6-5), the transmit control logic asserts SET RDXPT Q (read data expected quadword) which sets a flip-flop asserting RD XPT Q. RD XPT Q keeps RD EXPT asserted to write the second longword of the read transfer into location A RD2 of the receive file.

The ID, tag, and parity associated with the second longword are checked. If they are all good, the receive read data logic asserts RCV RD2. RCV RD2 is returned to the xmit control and resets the A GO bit. When A GO negates, A DN asserts to the microcode signifying that the transfer is complete.

If the requested data did not appear on the SBI after 102 microseconds, the operation is aborted and the timeout counter asserts RDTO to the configuration register setting a timeout error bit.

6.3 UNSOLICITED SBI REQUESTS

Unsolicited SBI Requests are reads and writes of a CI780 location that have been requested by a device on the SBI. The transfer sequence is not under control of the port microcode.

6.3.1 Unsolicited SBI Writes

Figure 6-6 is a block diagram of the logic involved in an unsolicited SBI write sequence. Figure 6-7 is a flow diagram illustrating the sequence of events in the write sequence.

The unsolicited SBI write command/address is placed on the SBI bus (along with the associated tag and mask) by the soliciting device. R TAG (2:0) is decoded in the tag decode logic and outputs C/A TAG identifying the information on the SBI as a command/address.

Parity is checked on the command/address field. P OK asserts if there is no parity error.

The address decode logic receives R B (27:10) from the SBI and asserts OUR ADD if the unsolicited request is addressed to the CI780 port.

The address/function decode logic decodes R B (31:28) to determine what the function is and if it is a valid function. FUNC VLD asserts if the function is valid.

The SBI receive state logic monitors and controls the unsolicited write sequence. It senses if the correct tag was received, if parity was good, and if the function is valid. If these were all good, the receive state logic asserts WD EXPT to the T confirm logic. The T confirm logic transmits an ACK confirmation back to the soliciting device.

UP ACCESS asserts if the write is to a DP register. If UP ACCESS does not assert, the write is to the configuration register. R B (09:00) supplies the address of the port register to receive the write data.

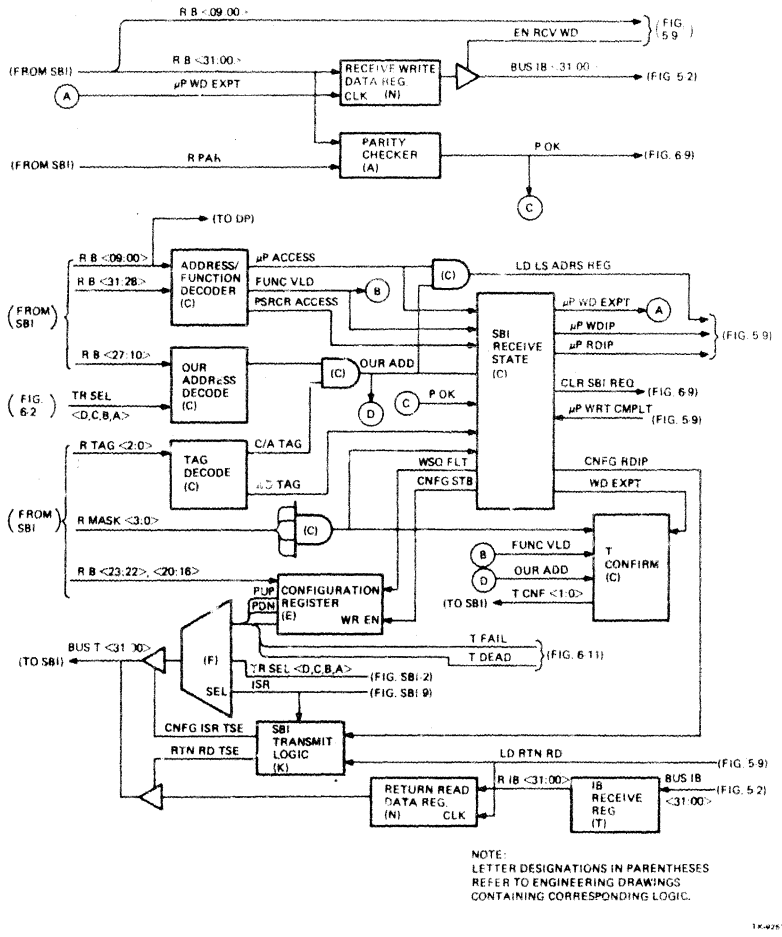


Figure 6-6 Unsolicited SBI Request Block Diagram

6.3.1.1 Writing a DP Register – If the function is a write of a DP register (UP ACCESS true), LD LS ADRS REG is asserted to the DP to load the address of the register to be written into an XBUS address register in the DP. The address of the register to be written (R B (09:00) is supplied from the SBI module to the DP.

When the write data appears on the SBI, the associated tag is decoded by the tag decode logic. The tag decode logic asserts WD TAG identifying the SBI data as write data.

Parity is checked on the write data. P OK asserts if there is no parity error.

SEE NEXT FRAME

FOR LARGER ART

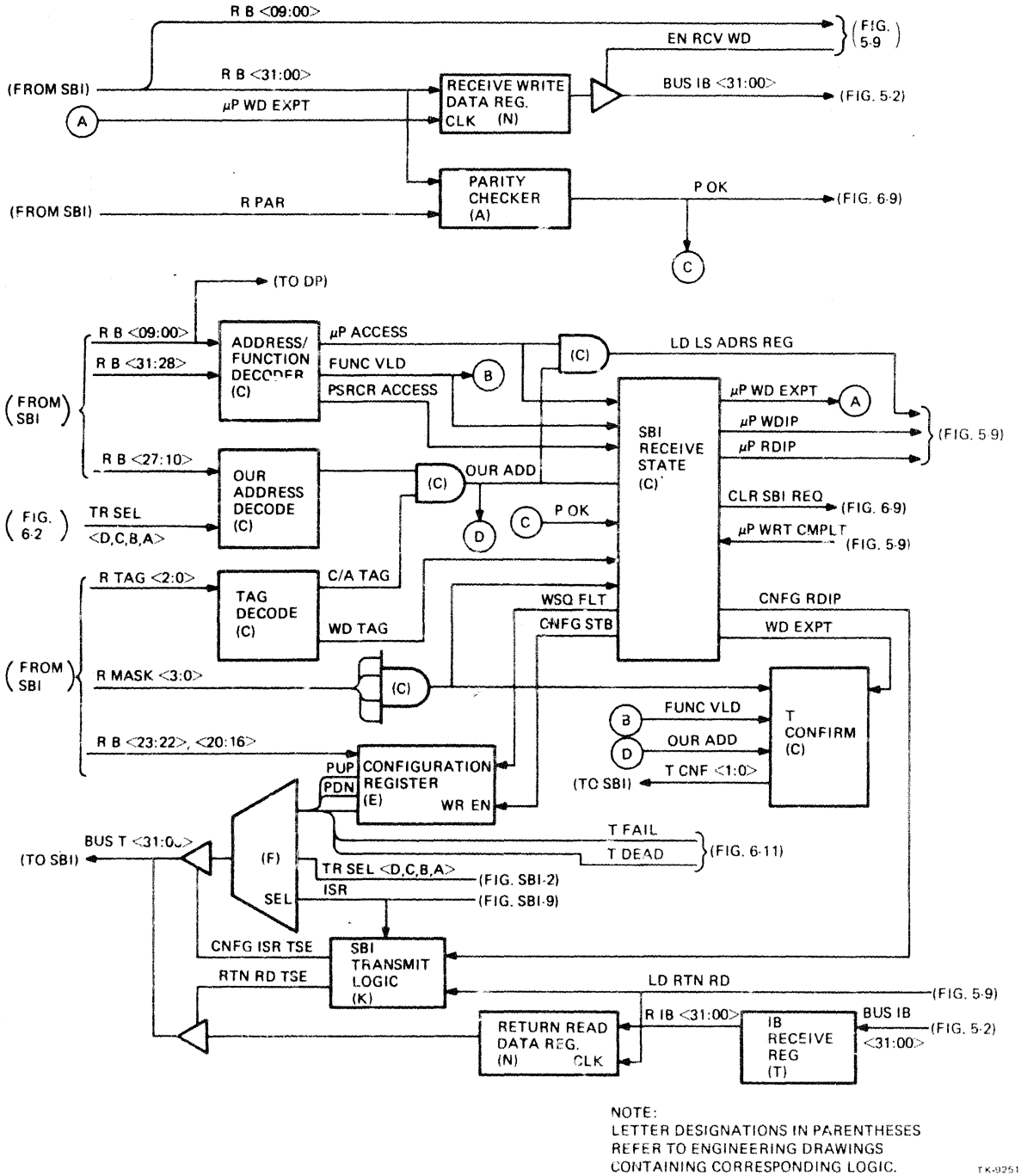


Figure 6-6 Unsolicited SBI Request Block Diagram

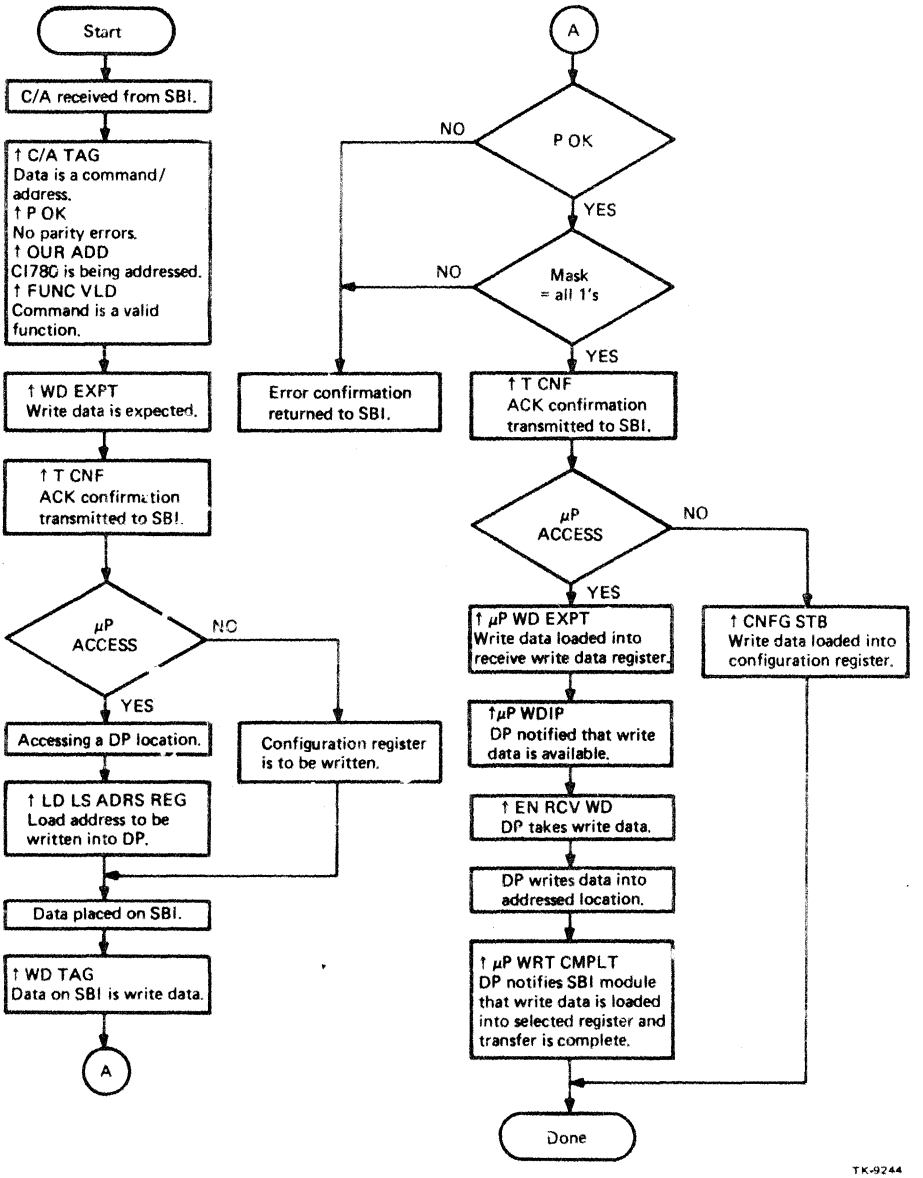


Figure 6-7 Unsolicited SBI Write Flow Diagram

The mask associated with the write data is checked for all 1's. Any other mask code for an unsolicited SBI write request operation results in an error confirmation being returned to the requesting device.

The T confirm logic chec for the presence of WD TAG, and that the mask is all 1's. The T confirm logic then transmits an ACK confirmation to the requesting device on the SBI.

The SBI receive state logic also generates UP WD EXPT which loads the write data from the SBI into the receive write data register.

The SBI receive state logic then asserts UP WDIP (write data in progress) to inform the DP that the write data is available. The DP responds with EN RCV WD which gates the write data out of the receive write data register into the DP.

A sequence is now executed in the DP to write the data into the selected location.

After the DP has deposited the write data in the selected in the location, it asserts UP WRT CMPLT to the SBI receive state logic indicating that the transfer is complete. The receive state logic then returns to the "not busy" state.

6.3.1.2 Writing the Configuration Register – If the function command decoded by the address/function decode logic is not UP ACCESS, then the write data is for the configuration register.

When the configuration register is to be written, LD LS ADRS REG does not assert to the DP as there is no DP register address to be loaded into the DP XBUS register.

The SBI receive state logic monitors and controls the writing of the configuration register. As in the case for writing a DP register, if FUNC VLD, OUR ADD and P OK all check good, then WD EXPT is true. If a write data tag is received (WD TAG true) and the write data mask is all 1's, an ACK confirmation response is sent to the requesting device. The receive state logic then asserts CNFG STB which loads the write data into the configuration register.

The receive state logic then returns to the "not busy" state.

6.3.2 Unsolicited SBI Reads

Figure 6-6 is a block diagram of the logic involved in an unsolicited SBI read sequence. Figure 6-8 is a flow diagram illustrating the sequence of events in the read sequence.

The unsolicited SBI read command/address is placed on the SBI bus (along with the associated tag and mask) by the soliciting device. R TAG ⟨2:0⟩ is decoded in the tag decode logic and outputs C/A TAG identifying the information on the SBI as a command/address.

Parity is checked on the command/address field. P OK asserts if there is no parity error.

The address decode logic receives R B ⟨27:10⟩ from the SBI and asserts OUR ADD if the unsolicited request is addressed to the CI780 port.

The address/function decode logic decodes R B ⟨31:28⟩ to determine what the function is and if it is a valid function. FUNC VLD asserts if the function is valid. UP ACCESS asserts if the read is of a DP register. If UP ACCESS does not assert, the read is of the configuration register. R B ⟨09:00⟩ supplies the address of the port register that is to be read.

When the T confirmation logic senses that P OK, OUR ADD, and FUNC VLD are all true, it transmits an ACK confirmation response (T CNF) to the requesting device on the SBI.

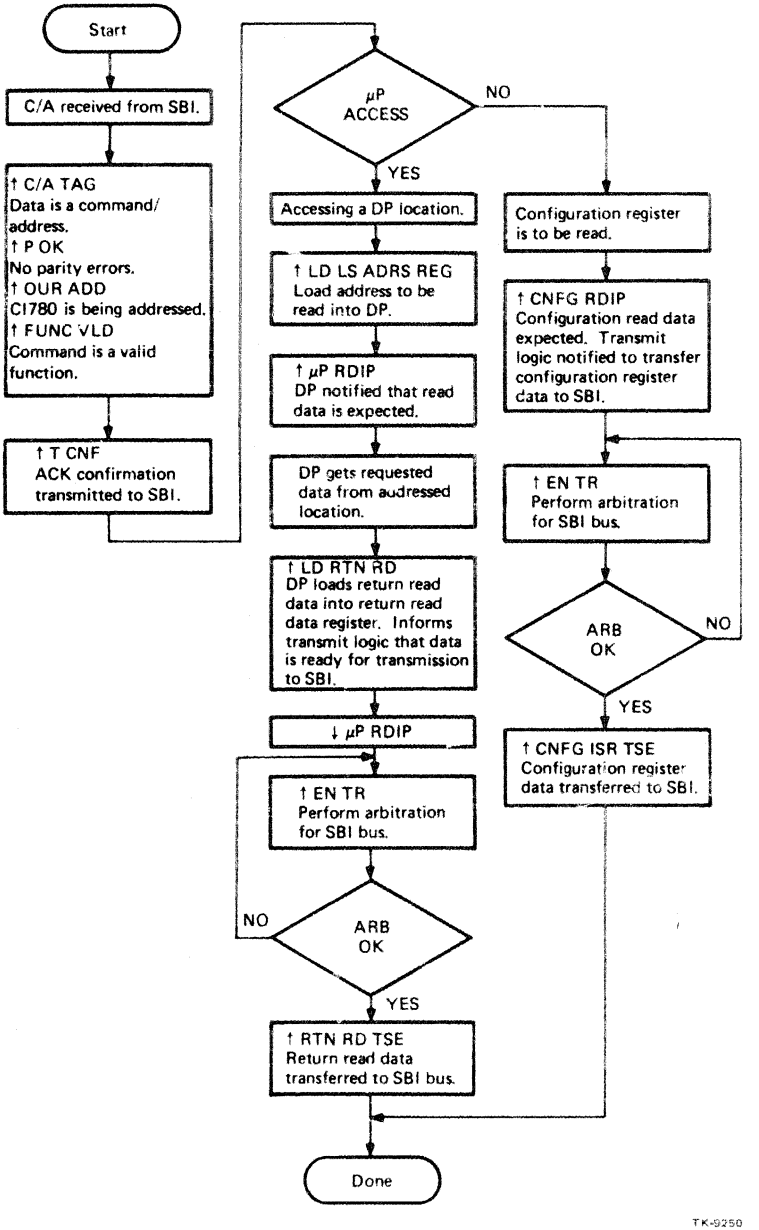


Figure 6-8 Unsolicited SBI Read Flow Diagram

6.3.2.1 Reading a DP Register – If the function is to read a DP register (UP ACCESS true), LD LS ADRS REG is asserted to the DP to load the address of the register to be read into an XBUS address register in the DP. The address of the register to be read (R D (09:00)) is supplied from the SBI module to the DP.

The SBI receive state logic monitors and controls the unsolicited read sequence. It senses that the operation is an access of a DP register (UP ACCESS true), that the operation is a valid function (FUNC VLD true), and that the request was for the CI780 (OUR ADD true). The receive state logic also checks the parity (P OK) of the SBI information field.

When the logic senses that all of the above signals are true, it asserts UP RDIP (read data in progress) to the DP notifying it that read data is expected.

UP RDIP initiates a sequence in the DP that gets the requested data from the selected location and places it on the BUS IB.

The DP then asserts LD RTN RD to load the return read data into the return read data register. LD RTN RD also informs the SBI transmit logic that the requested data is ready to be transmitted to the SBI. At this point the receive state logic negates UP RDIP.

The transmit logic asserts EN TR to the arbitration logic enabling it to arbitrate for the SBI bus. When the arbitration is successful and the SBI bus is obtained, the arbitration logic returns ARB OK to the transmit logic.

The transmit logic then asserts RTN RD TSE to transfer the return read data out of the return read data register and onto the SBI bus.

6.3.2.2 Reading the Configuration Register – If the function command decoded by the address/function decode logic is not UP ACCESS, then the register to be read is the configuration register.

The SBI receive state logic senses the false state of UP ACCESS and asserts CNFG RDIP (configuration read data in progress) indicating that configuration read data is expected.

CNFG RDIP is sent to the transmit logic which then must arbitrate for the SBI bus. When the arbitration is successful the transmit logic asserts CNFG ISR TSE.

CNFG ISR TSE gates the output from the configuration/ISR mux onto the SBI. The configuration/ISR mux selects the output from the configuration register due to the false state of ISR.

With the data in the configuration register transferred to the SBI bus, the SBI receiver state logic returns to the "not busy" state.

6.4 INTERRUPT SUMMARY REQUEST (ISR)

The CI780 port can request a CPU interrupt for the purpose of having the CPU run CI780 service routines. Figure 6-9 is a block diagram of the logic involved in requesting the interrupt. Figure 6-10 is a flow diagram of the ISR interrupt sequence.

A CPU interrupt sequence is requested by INTR from the microcode, or by the assertion of the maintenance error flag (MTE). In either case, the DP asserts PORT INTR to the SBI module to initiate the ISR sequence.

PORT INTR sets a flip-flop asserting MIF (maintenance interrupt flag). If the maintenance interrupt function is enabled (MIE bit in the DP PMCSR register set), the ISR decode logic becomes enabled.

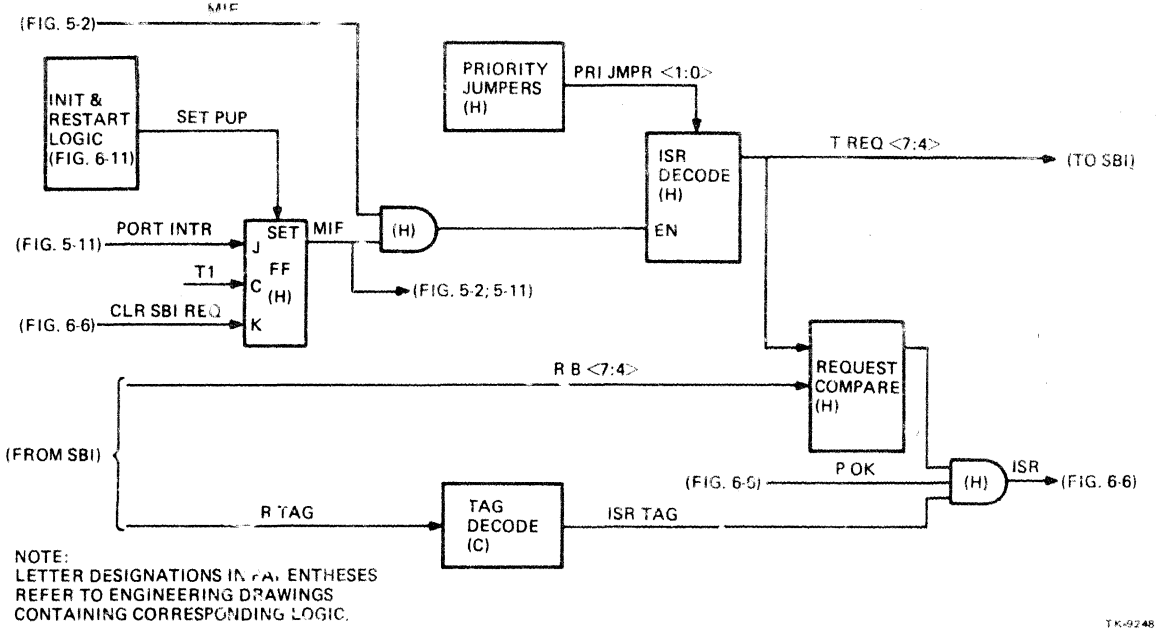


Figure 6-9 ISR Block Diagram

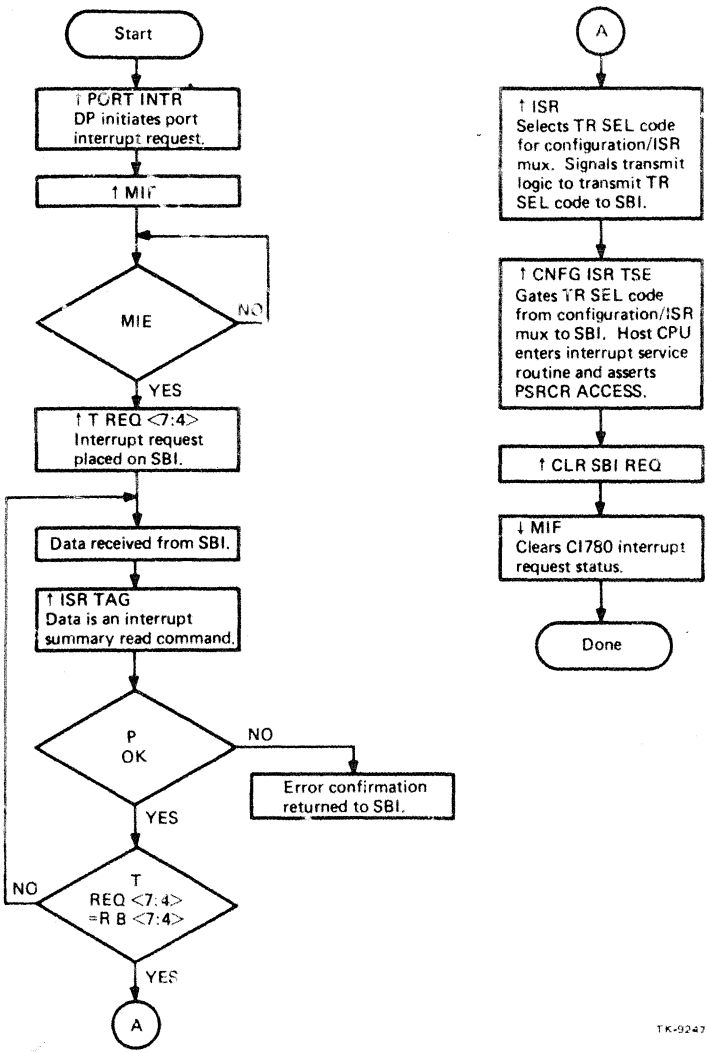


Figure 6-10 ISR Flow Diagram

The ISR decode logic asserts one of four T REQ output lines to the SBI bus. The T REQ level asserted depends on two priority jumpers that input PRI JMPR ⟨1:0⟩ into the decode logic.

T REQ ⟨7:4⟩ from the decode logic is placed on the SBI bus to request service from the CPU. The request is at the priority level established by the priority jumpers. The CPU responds by placing an interrupt summary read command on the SBI. The associated tag is decoded to assert ISR TAG identifying the bus data as an interrupt summary read command.

Parity is checked on the command and P OK asserted if there are no parity errors.

Bits R B ⟨7:4⟩ of the interrupt summary read command is the request level now being serviced by the CPU. The interrupt service request level is compared with the port request level (T REQ ⟨7:4⟩). If they match, ISR is asserted to the configuration/ISR mux and to the transmit logic (Figure 6-6).

ISR causes the configuration/ISR mux to select the TR SEL code generated by the TR jumpers.

ISR also notifies the transmit logic to transmit the TR SEL code to the SBI. The transmit logic responds by asserting CNFG ISR TSE which gates the TR SEL ⟨D,C,B,A⟩ code from the configuration/ISR mux to the SBI bus.

The CPU now generates a vector used to invoke the CI780 service routine.

To clear the port of the interrupt state, the CPU asserts a PSRCR ACCESS (port status release control register) function. The address/function decoder outputs PSRCR ACCESS to the receive state logic (Figure 6-6). This causes the receive state logic to assert CLR SBI REQ which resets the MIF flip-flop. Resetting the flip-flop negates MIF and takes the port out of the interrupt request state.

6.5 INTITIALIZATION AND RESTART LOGIC

The initialization and restart logic (Figure 6-11) initializes the port on system power-up and shuts down the port when a power failure occurs within the CI780 or the host system.

VAX-11/780 protocol requires that a power failure cause the assertion of SUPPLY ACLO followed by the assertion of SUPPLY DCLO (Figure 1-5). During power-up the reverse is true, that is SUPPLY DCLO negates and then SUPPLY ACLO negates.

6.5.1 Start-Up Sequence

Figure 6-12 is a flow diagram of the start-up sequence. When power is applied to the port, both SUPPLY ACLO and SUPPLY DCLO come true. SUPPLY DCLO asserts PS DC LO which clears the PUP and PDN bits in the configuration register and sets the MIE bit in the PMCSR. PS DC LO also causes REG CLR and LOGIC CLR to assert and reset the port hardware registers and logic circuits respectively. LOGIC CLR also asserts UNINIT and places the port into the uninitialized state (Figure 5-12).

As power comes up SUPPLY DCLO negates followed by the negation of SUPPLY ACLO. The negation of SUPPLY ACLO causes SET PUP to come true and set the PUP bit in the configuration register.

SET PUP also asserts MIF (Figure 6-9) and, with maintenance interrupts enabled (MIE bit in PMSCR true), an interrupt sequence is initiated to the host CPU.

When the boot timer start-up delay has timed out (BTO) or the host asserts PICR WRT via an unsolicited SBI write operation (Figure 5-12), the port enters the initialized state and UNINIT negates.

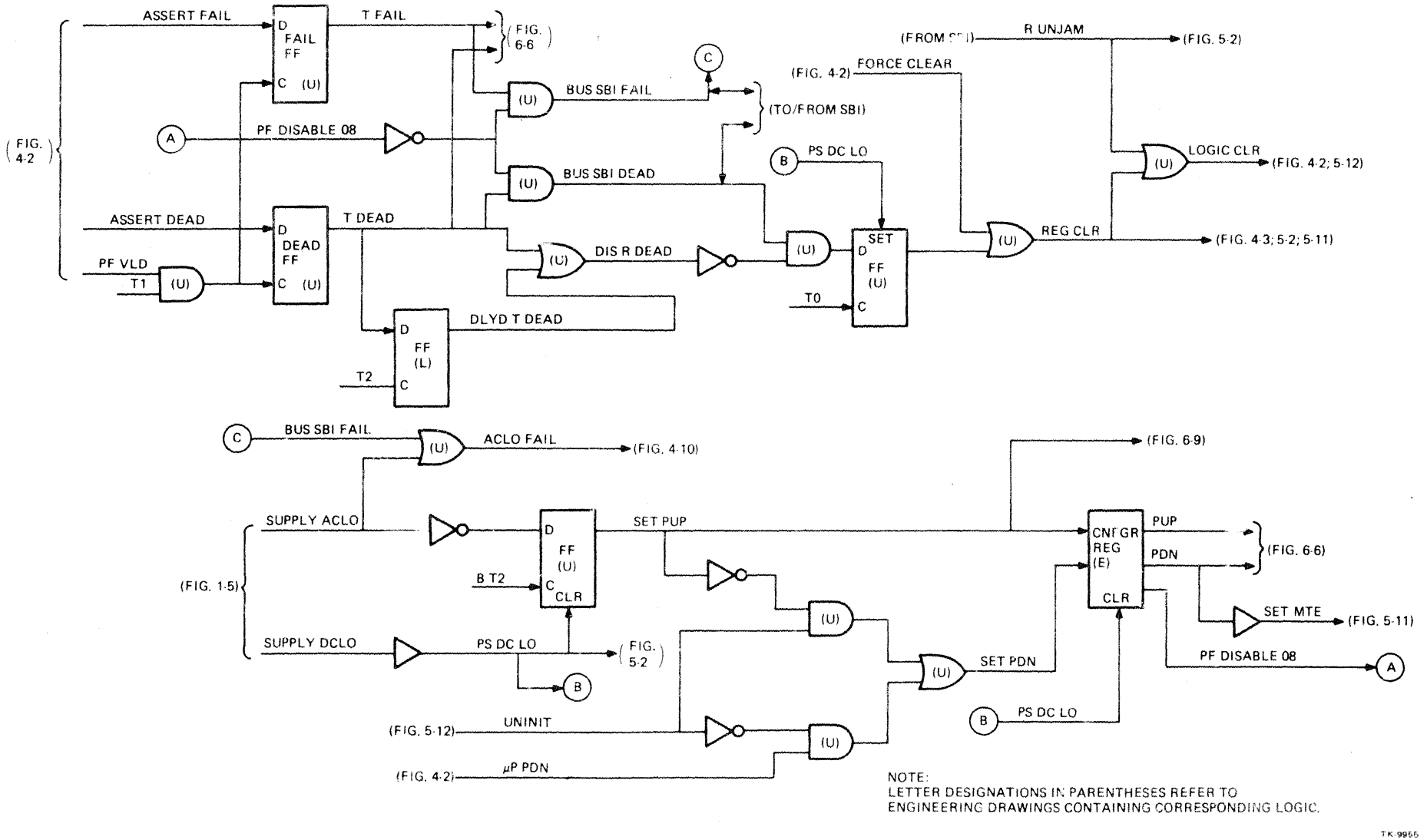


Figure 6-11 Initialization and Restart Logic

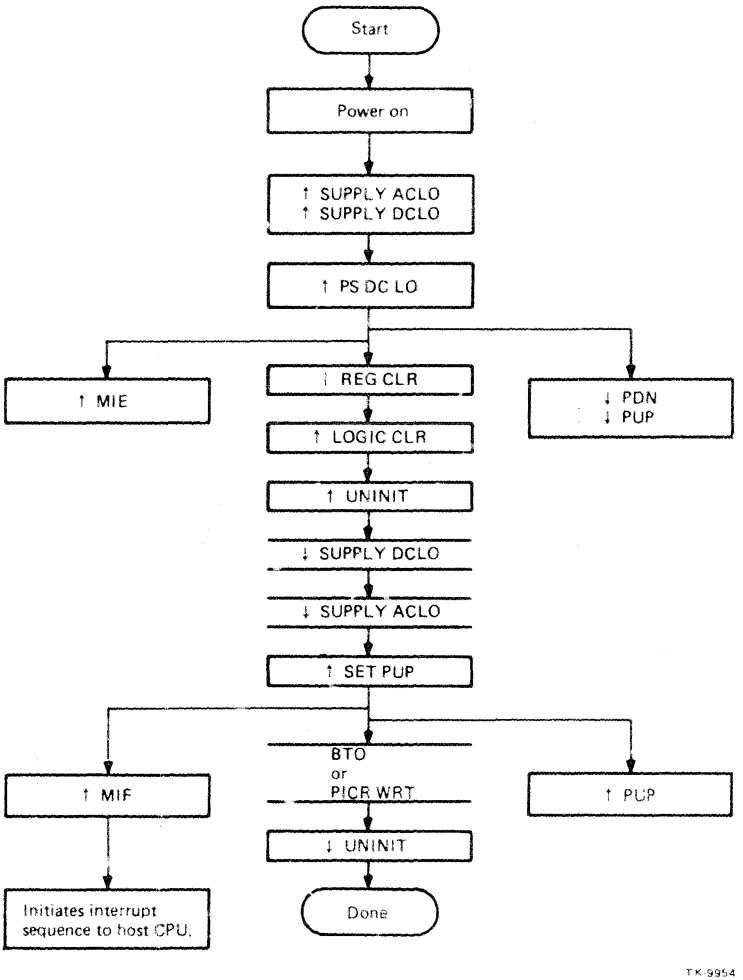


Figure 6-12 Start-Up Sequence

6.5.2 Power-Fail Sequence

Figure 6-13 is a flow diagram of the power-fail sequence. When a power failure occurs within the CI780, SUPPLY ACLO comes true and asserts ACLO FAIL to the CS branching logic. ACLO FAIL is also asserted by a power failure within the host CPU which causes BUS SBI FAIL to assert on the SBI.

If the port is initialized (UNINIT false), ACLO FAIL causes the microcode to branch to a power-fail routine. The power-fail routine causes the port to save current information so that operation may be resumed when power returns.

The microcode power-fail routine asserts UP PDN. With the port in the initialized state (UNINIT false), UP PDN asserts SET PDN to the configuration register which resets the PUP bit and sets the PDN bit in the register. PDN asserts SET MTE which then asserts MTE in the DP module. MTE places the port in the uninitialized state (Figure 5-12), thereby stopping the microcode. MTE also asserts PORT INTR which initiates an interrupt sequence to the host CPU (Figure 6-9).

If a genuine power interruption is occurring, SUPPLY ACLO will still be true and if this is a port power failure, SUPPLY DCLO will assert. When SUPPLY DCLO comes true it asserts PS DC LO. PS DC LO asserts REG CLR and LOGIC CLR thereby clearing all the port hardware registers and logic circuits.

If the power failure occurred in the host CPU, BUS SBI DEAD will be asserted on the SBI. BUS SBI DEAD asserts REG CLR and LOGIC CLR to reset the port registers and logic circuits.

If only a transient AC power dip occurred, SUPPLY ACLO may negate before SUPPLY DCLO asserts, hence the port registers and logic circuits are not cleared. In this case, the negation of SUPPLY ACLO causes SET PUP to come true which sets the PUP bit and resets the PDN bit in the configuration register. Resetting the PDN bit negates SET MTE and MTE which allows the port to enter the initialized state when the boot timer has timed out (BTO) or the host asserts PICR WRT via an unsolicited SBI write operation (Figure 5-12).

If the port is not initialized (UNINIT true) when SUPPLY ACLO first asserts, the same basic sequence is executed except that the hardware – not the microcode – powers down the port. As shown in Figures 6-11 and 6-13, the assertion of ACLO FAIL negates SET PUP. With UNINIT true, the negation of SET PUP causes SET PDN to assert to the configuration register. SET PDN sets the PDN bit and resets the PUP bit in the register.

PDN asserts SET MTE which then asserts MTE in the DP module. MTE asserts PORT INTR which initiates an interrupt sequence to the host CPU.

The power-fail sequence then completes as shown in Figure 6-13.

As illustrated in Figure 6-11, FORCE CLEAR (asserted by MCLR from the microword or by MIN from the PMCSR) also resets the CI780 by asserting REG CLR and LOGIC CLR.

R UNJAM from the SBI clears the port logic while leaving the register contents intact for diagnostic examination.

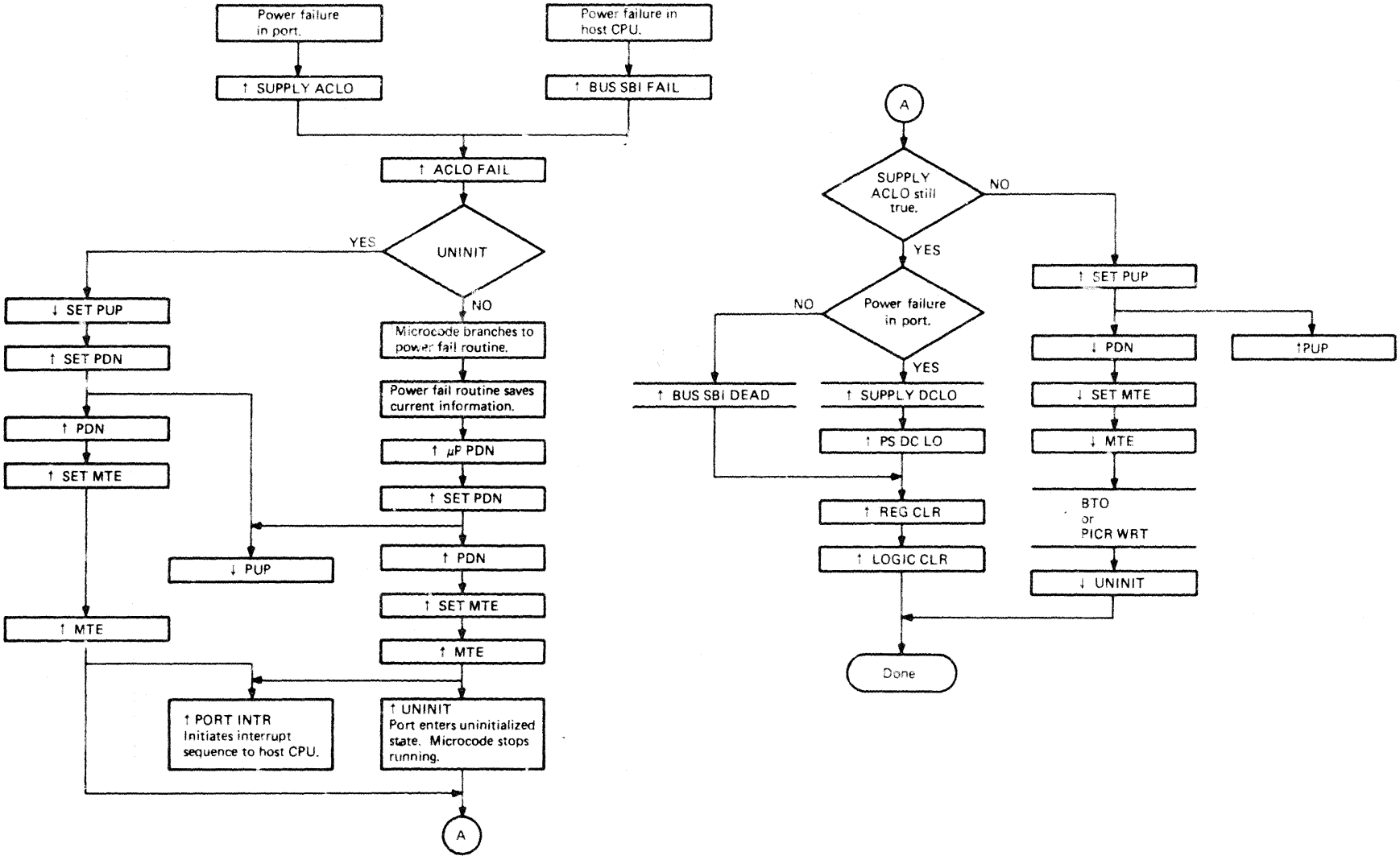


Figure 6-13 Power Fail Sequence

6.5.3 Maintenance Mode

In the maintenance mode, the CI780 port can be reset from another node by means of a reset packet. The remote node sends the reset packet to the CI780 causing the port microcode to assert ASSERT FAIL and PF VLD (power-fail valid) (Figure 6-11). ASSERT FAIL conditions a fail flip-flop to set while PF VLD enables the T1 clock to set the flip-flop. Setting the fail flip-flop asserts T FAIL which in turn asserts BUS SBI FAIL onto the SBI (PF DISABLE 08 false). BUS SBI FAIL functions to initiate a power-down sequence within the host system.

The microcode then asserts ASSERT DEAD and PF VLD which similarly results in the assertion of T DEAD and then BUS SBI DEAD on the SBI. BUS SBI DEAD completes the power-down sequence within the host system.

Note that T DEAD also asserts DIS R DEAD (disable receive dead). DIS R DEAD inhibits BUS SBI DEAD from clearing the port registers and logic. Thus the port is still able to function while the host system is powered down.

When T DEAD negates, DIS R DEAD is held true by DLYD T DEAD for one clock pulse after T DEAD negates. This insures that BUS SBI DEAD has negated before DIS R DEAD goes false, and prevents BUS SBI DEAD from possibly clearing the port registers and logic circuits.

The microcode then asserts PF VLD with ASSERT DEAD negated resulting in the resetting of the dead flip-flop and the negation of T DEAD and BUS SBI DEAD. With BUS SBI DEAD false, the host system attains a partial powered-up state.

The host system remains in the partial powered-up state until the remote port sends a start packet. The start packet causes the port microcode to assert PF VLD with ASSERT FAIL negated, thereby resetting the fail flip-flop and negating T FAIL and BUS SBI FAIL. The host system will now complete its power-up.

When the PF DISABLE 08 bit in the configuration register is true, maintenance diagnostics can test the microword ASSERT FAIL and ASSERT DEAD bits without affecting the SBI.

APPENDIX A
CI780 MNEMONIC GLOSSARY

ACK	Acknowledge
ALU	Arithmetic logic unit
AR	ACK receive (state)
ARB	Arbitrate
ARBC	Arbitration counter
AX	ACK transmit (state)
BM	Byte mask
BR	Branch
BSY	Busy
BTO	Boot timeout
C/A	Command/address
C/ATO	Command address timeout
CI	Computer interconnect (formerly ICCS and IPA)
CNF	Confirm
CNFG STB	Configuration strobe
CRC	Cyclic redundancy check
CRD	Corrected read data
CS	Control store
CSA	Control store address
CSPE	Control store parity error
CXTER	Command transmit error
CXTMO	Command transmit timeout
DIS R DEAD	Disable receive dead
DFE	Decoded file enable
DN	Done
DP	Data path (module)
DPUP	Decoded push/pop
DST CMP	Destination compare
ECL	Emitter coupled logic
FCN	Function
FE	File enable
FLT	Fault
IB	Internal bus
IB DST	IB destination
IB SRC	IB source
ICCS	Intercomputer communications switch (see CI)

INTR	Interrupt
IPA	Interprocessor adapter (see CI)
IPE	Input parity error
ISR	Interrupt summary request
JSR	Jump to subroutine
LS	Local store
LSB	Least significant bit
LSPE	Local store parity error
LT	Less than
MADR	Maintenance address register
MCLR	Maintenance clear
MD	Miscellaneous data
MDATR	Maintenance data register
ME	Manchester encoded
MIE	Maintenance interrupt enable
MIF	Maintenance interrupt flag
MIN	Maintenance initialize
MISC CNTL	Miscellaneous control
MLOAD	Maintenance load
MLOOP	Maintenance loop
MR	Message receive (state)
MSB	Most significant bit
MSE	Memory system error
MTD	Maintenance timer disable
MTE	Maintenance error
MX	Message transmit (state)
MXT	Multiple transmit
NACK	Negative acknowledge
OPE	Output parity error
PAL	Programmable array logic
PB	Packet buffer (module)
PC	Program counter
PDN	Power-down
PE	Parity error
PF VLD	Power fail valid
PICR	Port initialize control register
PMCSR	Port maintenance control/status register
PMTCR	Port maintenance timer control register
PROM	Programmable read-only memory
PROP	Propagate
PSA	Programmable starting address
PSR	Port status register
PSRCR	Port status release control register
PUP	Push/pop
PUP	Power-up

RAM	Random access memory
RBPE	Receive buffer parity error
RBUF	Receive buffer
RCAR	Receive carrier
RD	Read data
RDS	Read data substitute
RDTO	Read data timeout
RDIP	Read data in progress
RDXPT L	Read data expected longword
RDXPT Q	Read data expected quadword
RSVD	Reserved
RTS	Return from subroutine
SBI	Synchronous backplane interconnect
S/D	Source/destination
SEL CC	Select condition code
SEQ	Sequencer
TACK	Transmit ACK
TBUF	Transmit buffer
TR	Transfer request
TSE	Transfer enable
TTL	Transistor-transistor logic
UNINIT	Uninitialized
URD	Unexpected read data
VCDT	Virtual circuit descriptor table
VRD	Valid receive data
WACK	Wait for ACK
WD	Write data
WDIP	Write data in progress
WP	Wrong parity
WRT	Write
WSQ	Write sequence
XBUS	External bus

APPENDIX B
FLOW DIAGRAM SYMBOLS

The flow diagram symbols used in this manual are defined in Figure B-1. Signal mnemonics are shown in upper case. All other flow diagram text is in lower case.

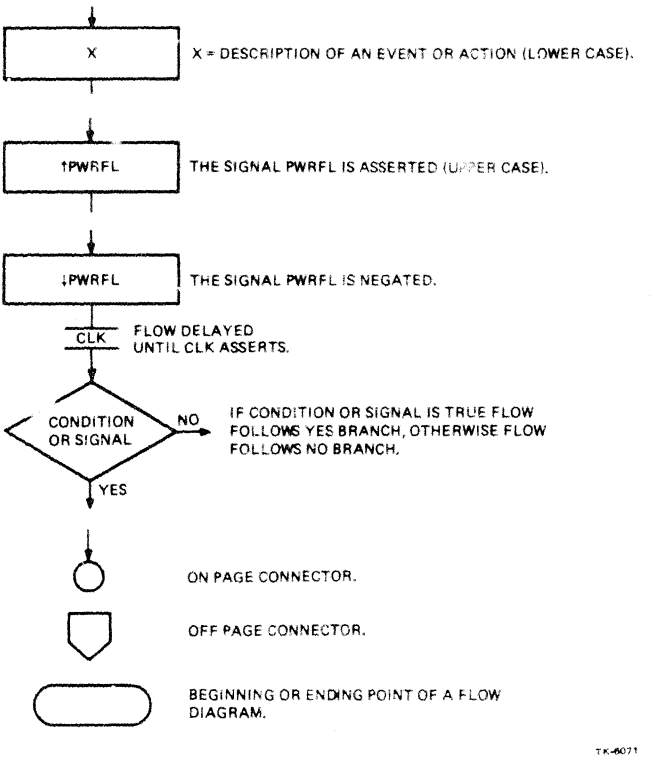


Figure B-1 Flow Diagram Symbols

SEE NEXT FRAME

FOR LARGER ART

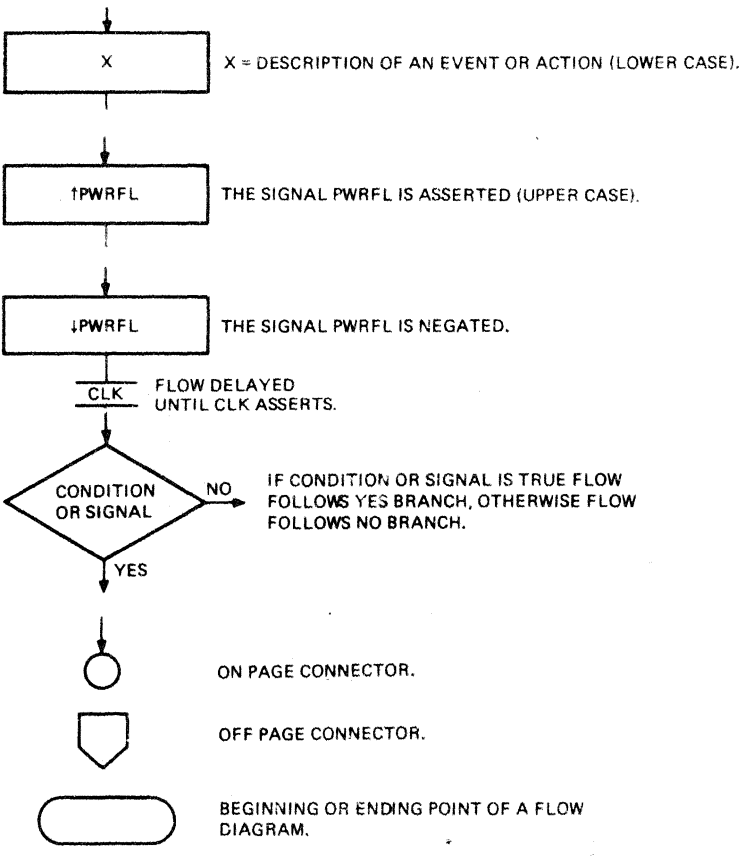


Figure B-1 Flow Diagram Symbols

APPENDIX C
HARDWARE REGISTERS

Appendix C is a description of four hardware registers that can be accessed by the port software for maintenance purposes. The registers described are:

1. MADR – Maintenance Address Register
2. MDATR – Maintenance L ta Register
3. PMCSR – Port Maintenance Control/Status Register
4. CNFGR – Configuration Register

C.1 MADR – Maintenance Address Register

Figure C-1 illustrates the function of the MADR bits. The register address = XXXXX014 (hex). MADR contains the address of the control store location to be accessed. It is read or written only in the uninitialized state.

Refer to Figure 4-1 and Paragraphs 4.1, 4.6.1, and 4.7 for a discussion of MADR operation.

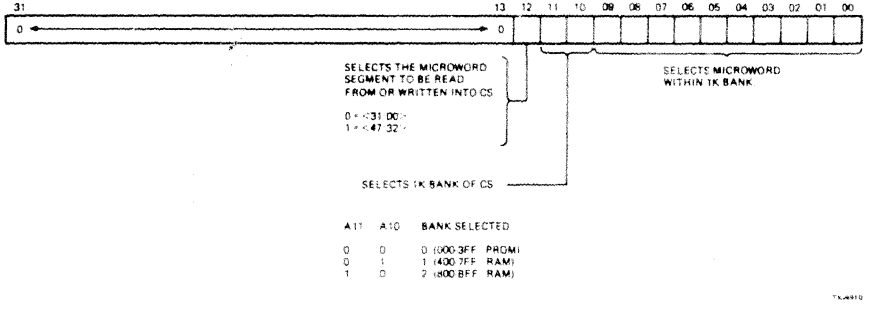


Figure C-1 Maintenance Address Register (MADR) Bit Fields

SEE NEXT FRAME

FOR LARGER ART

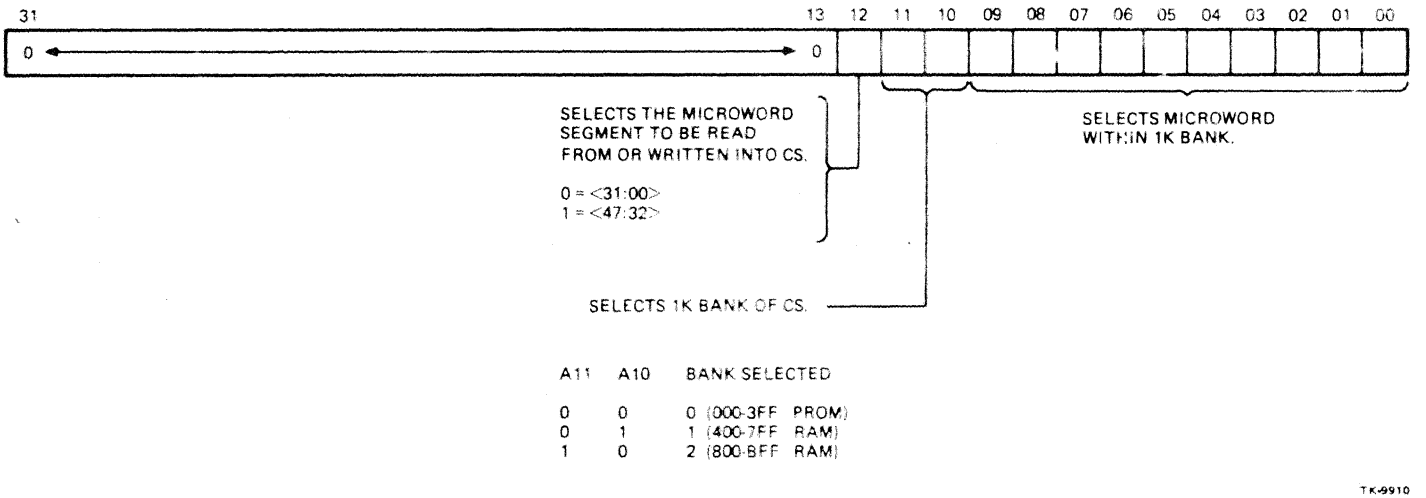


Figure C-1 Maintenance Address Register (MADR) Bit Fields

C.2 MDATR - Maintenance Data Register

Figure C-2 illustrates the MDATR bits. The register address = XXXXX018 (hex). MDATR does not exist as a physical register. A read or write of MDATR will read or write the microword in the control store location specified by the address in the MADR. When MADR 12 = 0, MDATR (31:00) contains microword bits (31:00). When MADR 12 = 1, MDATR (15:00) contains microword bits (47:32) (MDATR (31:16) are all 0s). MDATR is read or written only in the uninitialized state.

Figure 4-4 and Table 4-1 define the microword bits. Refer to Figure 4-2 and Paragraph 4.4 for a discussion of reading and writing the control store.

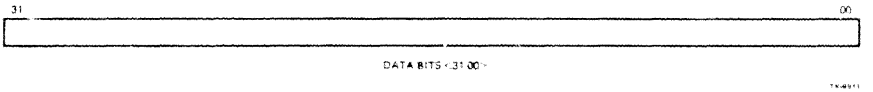


Figure C-2 Maintenance Data Register (MDATR) Bit Field

SEE NEXT FRAME

FOR LARGER ART

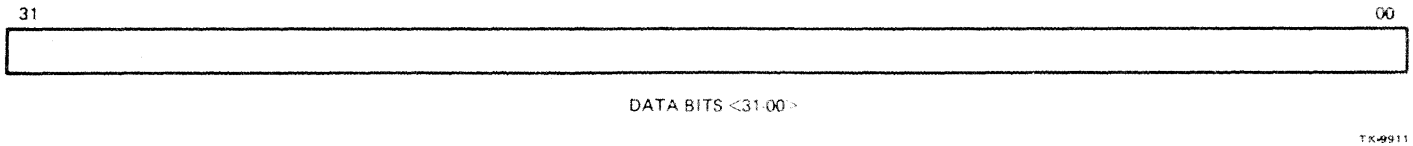


Figure C-2 Maintenance Data Register (MDATR) Bit Field

C.3 PMCSR – Port Maintenance Control/Status Register

Figure C-3 illustrates the function of the PMCSR bits.

The register address = XXXXX004 or XXXXX010. PMCSR contains port hardware error flags, interrupt bits, and initialization control bits.

A description of the PMCSR bits is given in Paragraph 5.2.3 and Table 5-1.

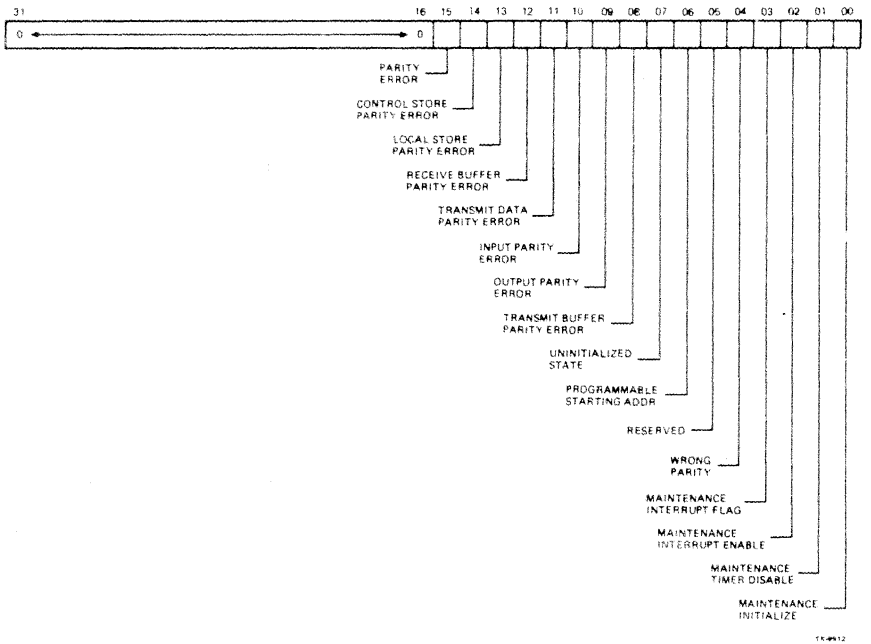


Figure C-3 Port Maintenance Control/Status Register (PMCSR) Bit Fields

SEE NEXT FRAME

FOR LARGER ART

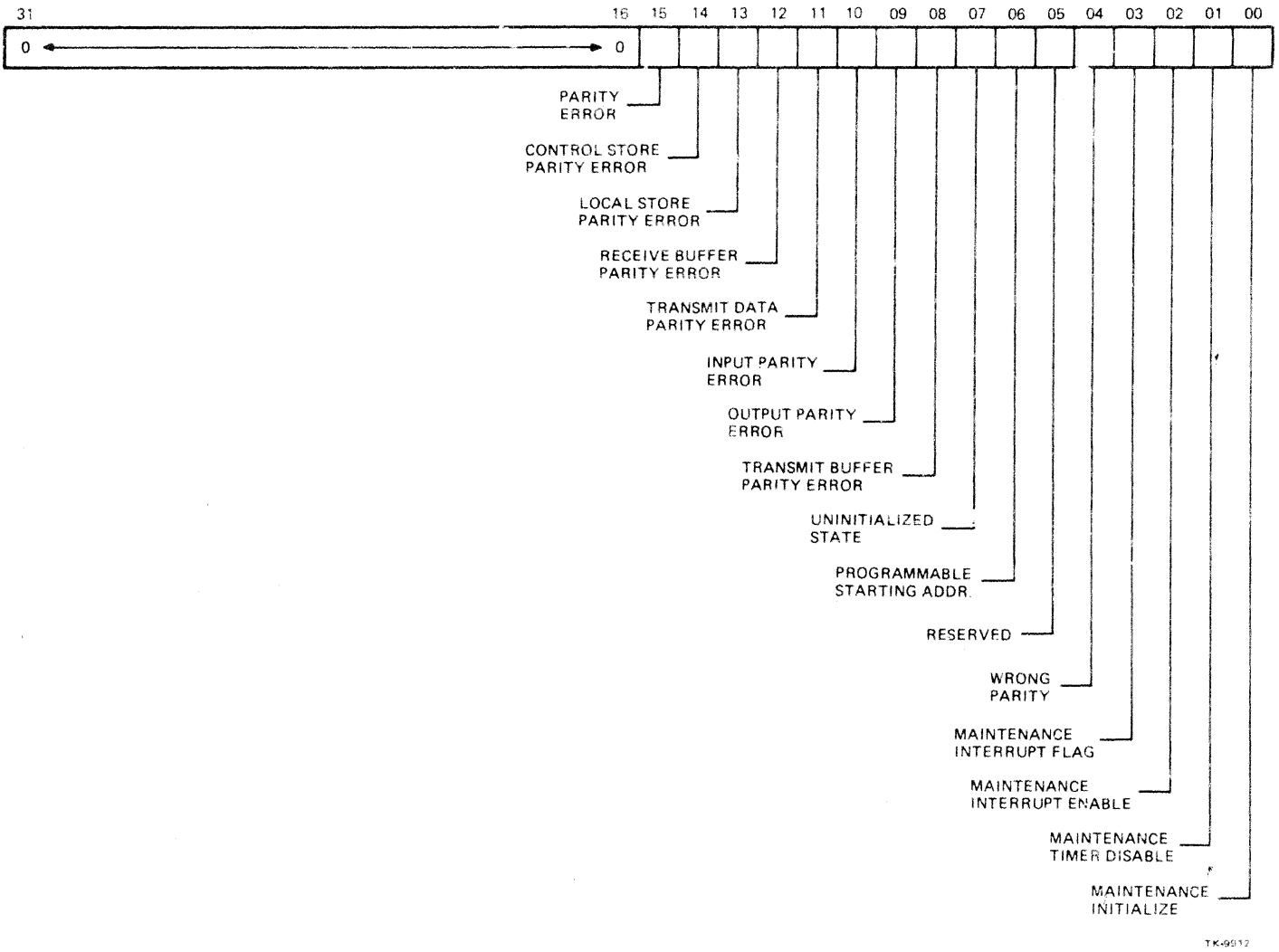


Figure C-3 Port Maintenance Control/Status Register (PMCSR) Bit Fields

C.4 CNFGR – Configuration Register

Figure C-4 illustrates the function of the CNFGR bits.

The register address = XXXXX000. CNFGR contains SBI fault bits, and status and error bits for an SBI transfer. It also contains the CI780 adaptor code. Refer to Paragraph 6.3 for a discussion of reading and writing the CNFGR.

Table C-1 describes the CNFGR bit functions.

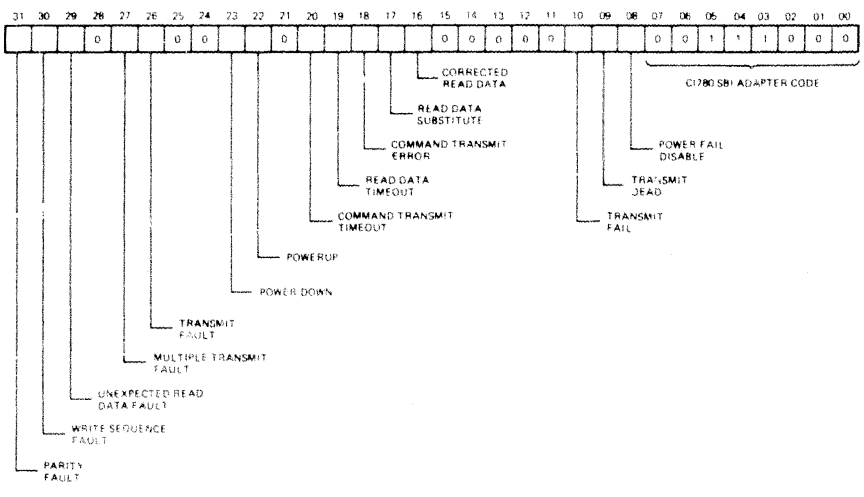


Figure C-4 Configuration Register (CNFGR) Bit Fields

SEE NEXT FRAME

FOR LARGER ART

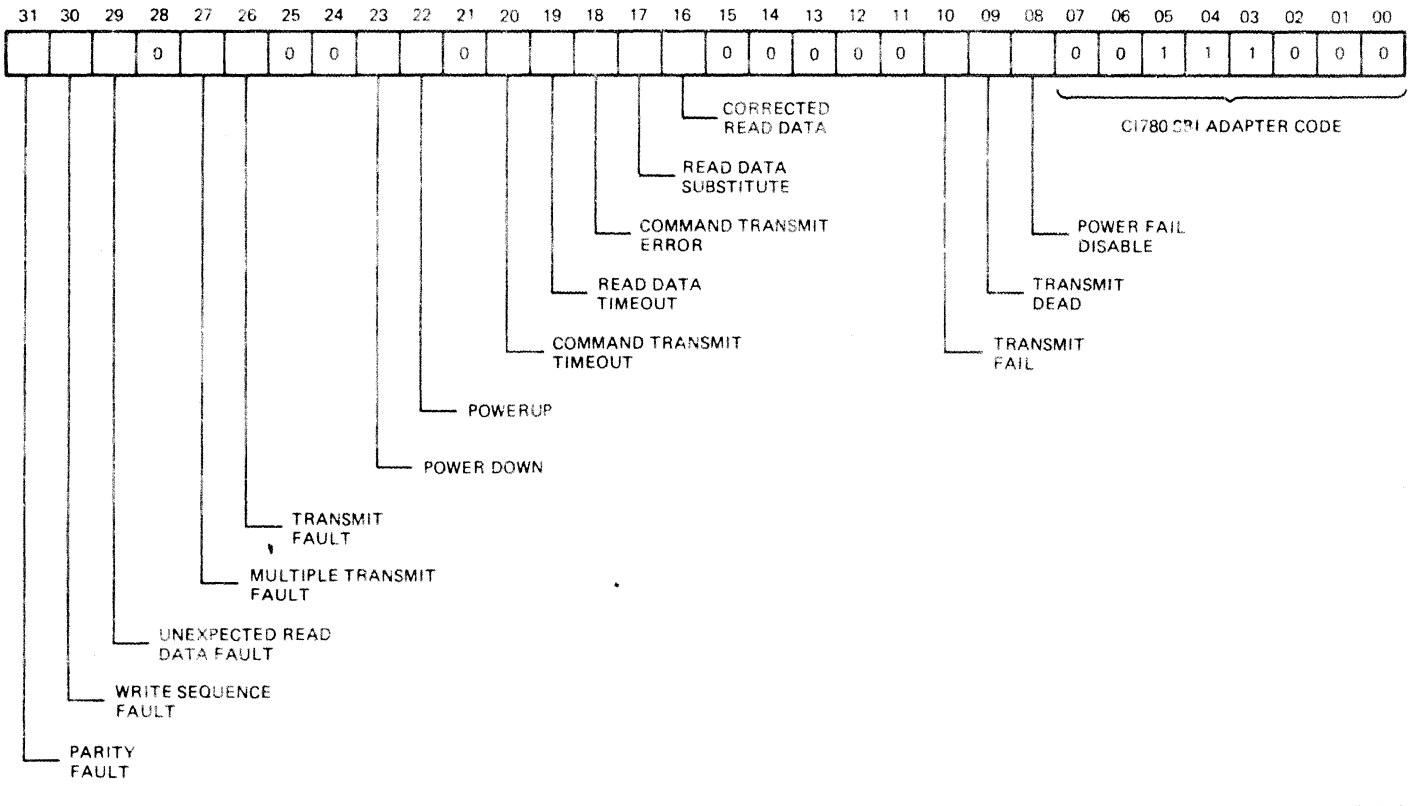


Figure C-4 Configuration Register (CNFGR) Bit Fields

Table C-1 CNFGR Bits

Bit	Mnemonic	Description
31:26	———	Bits (31:26) are the SBI fault bits. They are set when the port detects the respective fault condition as described below. The fault bits are read only.
31	PAR FLT	Parity fault: Set when the port detects an SBI parity error.
30	WSQ FLT	Write sequence fault: Set when the port receives a write mask command that is not immediately followed by the expected write data.
29	URD FLT	Unexpected read data fault: Set when the port receives read data but did not issue a read command.
28	0	This bit is reserved and read as 0.
27	MXT FLT	Multiple transmit fault: Set when the ID bits transmitted by the port do not match the ID bits received back from the SBI.
26	XMT FLT	Transmit fault: Set if the CI780 was the SBI nexus that caused the SBI FAULT line to assert.
25:24	0	Reserved. Read as 0s.
23	PDN	Power down: Set if the port is powering down. PDN is set by the assertion of SUPPLY ACLO if the port is in the uninitialized state, otherwise it is set by the microcode via the PDN bit. The PDN bit is cleared by writing a 1 to it or by setting the PUP bit.
22	PUP	Power up: Set by the negation of SUPPLY ACLO. The PUP bit is cleared by writing a 1 to it or by setting the PDN bit.
21	0	Reserved. Read as 0.
20	CXTMO	Command transmit timeout: Set when the port initiates an SBI transfer and does not receive an ACK or error confirmation within 102 microseconds. The timeout period is selectable by backplane jumpers.
19	RDTO	Read data timeout: Set when the port initiates an SBI read transfer and read data is not returned within 102 microseconds.

Table C-1 CNFGR Bits (Cont)

Bit	Mnemonic	Description
18	CXTER	Command transmit error: Set when the port receives an error confirmation in response to a port initiated SBI command transmission.
17	RDS	Read data substitute: Set when the port receives an RDS (uncorrectable read data) confirmation in response to a port initiated SBI read command.
16	CRD	Corrected read data: Set when the port receives a CRD confirmation in response to a port initiated SBI read command.
15:11	0	Reserved. Read as 0s.
10	T FAIL	Transmit fail: Set by the microcode through the miscellaneous control field when PFV and ASSERT FAIL are true.
09	T DEAD	Transmit dead: Set by the microcode through the miscellaneous control field when PFV and ASSERT DEAD are true.
08	PF DISABLE	Power fail disable: When set, the SBI FAIL and SBI DEAD drivers to the SBI are disabled.
07:00	———	Adaptor code: These bits contain the CI780 SBI adaptor code.