EK-CI750-TD-PRE

CI750 Hardware Technical Description

Preliminary

digital™

EK-CI750-TD-PRE

CI750 Hardware Technical Description

Preliminary

Prepared by Educational Services
of
Digital Equipment Corporation

First Edition, July 1984

Printed in U.S.A.

This book was produced on a DIGITAL Word Processing System. Book production was done by Educational Services Development and Publishing in Nashua, NH.

CONTENTS

Page

CHAPTER 1 INTRODUCTION

1.1 MANUAL SCOPE....1-1
1.2 THE COMPUTER INTERCONNECT (CI)....1-1
1.3 RELATED DOCUMENTS....1-3
1.4 THE CI750 INTERFACE....1-5
1.4.1 Link Module....1-5
1.4.2 Packet Buffer Module (PB)....1-9
1.4.3 Data Path Module (DP)....1-10
1.4.4 CMI CIPA Interface Module (CCI)....1-11
1.4.5 CI750 Power....1-13

CHAPTER 2 LINK MODULE

2.1 PACKET FORMATS....2-1
2.1.1 Information Packet....2-1
2.1.1.1 Bit Synchronization....2-1
2.1.1.2 Character Synchronization....2-2
2.1.1.3 Packet Type/Length (High)....2-2
2.1.1.4 Packet Length (Low)....2-2
2.1.1.5 Destinations (True and Complement)....2-2
2.1.1.6 Source....2-3
2.1.1.7 Body....2-3
2.1.1.8 Cyclical Redundancy Check (CRC) Bytes....2-3
2.1.1.9 Trailer....2-3
2.1.2 Acknowledge/Negative Acknowledge (ACK/NACK) Packet....2-3
2.2 LINK OVERVIEW....2-5
2.2.1 Information Packet Reception....2-7
2.2.2 ACK/NACK Packet Transmission....2-8
2.2.3 Information Packet Transmission....2-9
2.2.4 ACK/NACK Packet Reception....2-10
2.3 LINK OPERATING STATES....2-10
2.4 RECEIVE CHANNEL....2-12
2.4.1 CI Carrier Detection and Path Selection....2-12
2.4.1.1 Carrier Detect Logic....2-12
2.4.1.2 Receive Path Select Mux -- ECL Logic....2-15
2.4.2 Manchester Decoder....2-15
2.4.2.1 Phase Encoding....2-15
2.4.2.2 Decoder Logic....2-18
2.4.3 Sync Character Detect Enable PAL....2-18
2.4.4 Byte Framer....2-20
2.4.5 RCVR CLK Generator....2-24
2.4.6 CRC Check....2-26
2.4.7 Destination Compare....2-28
2.4.8 ACK Source Comparison....2-29
2.4.9 Receive Data Parity and Channel Output....2-29

CONTENTS

		Page
2.5	TRANSMIT CHANNEL	2-30
2.5.1	Transmit Data Input	2-30
2.5.2	Bit Sync, Sync Character, and Trailer Bytes	2-30
2.5.3	ACK Packet Inserts	2-33
2.5.3.1	Packet Type Byte	2-33
2.5.3.2	Source Byte	2-33
2.5.3.3	Destination Bytes	2-33
2.5.4	Destination Address Register	2-33
2.5.5	Transmit Data Parity Check	2-34
2.5.6	CRC Generation	2-34
2.5.7	XMIT CLK Generator	2-34
2.5.8	Parallel to Serial Data Conversion	2-37
2.5.9	Manchester Encoder	2-37
2.5.10	XMIT ECL Drivers	2-39
2.6	CRC GENERATOR AND CHECKER	2-41
2.6.1	CRC Generator	2-41
2.6.2	CRC Checker	2-43
2.7	ARBITRATION	2-43
2.7.1	General	2-43
2.7.2	Arbitration Logic	2-46
2.8	LINK FUNCTIONS	2-51
2.9	LINK INTERFACE SIGNALS	2-54
2.10	OPERATING STATES	2-54
2.10.1	Message Transmit	2-58
2.10.1.1	Transmit Control Logic	2-62
2.10.1.2	Transmit Status	2-63
2.10.2	ACK Receive	2-65
2.10.2.1	ACK Receive PAL States	2-65
2.10.2.2	Sync Character Detect Enable PAL	2-67
2.10.3	Message Receive	2-69
2.10.4	ACK Transmit	2-72

CHAPTER 3 PACKET BUFFER MODULE

3.1	DATA FLOW; GENERAL DISCUSSION	3-1
3.1.1	TBUF Load	3-3
3.1.2	Transmit	3-3
3.1.3	TBUF Read	3-3
3.1.4	Valid RCVR Data	3-3
3.1.5	RBUF MLOAD (Maintenance Load)	3-3
3.1.6	RBUF READ	3-3
3.1.7	PB Read Mux	3-3
3.1.8	Control Logic	3-3
3.2	TBUF DATA FLOW OPERATIONS	3-4
3.2.1	TBUF Load	3-4
3.2.2	Transmit	3-6
3.2.3	TBUF Read (Loopback)	3-6

CONTENTS

		Page
3.3	RBUF DATA FLOW OPERATIONS	3-7
3.3.1	Valid RCVR Data	3-7
3.3.2	RBUF MLOAD (Maintenance Load)	3-9
3.3.3	RBUF Read	3-9
3.3.4	PB Read Mux	3-10
3.4	CLOCKS	3-10
3.5	FUNCTION DECODER AND BUFFER SELECT LOGIC	3-12
3.5.1	SEL Load Buf	3-12
3.5.2	SEL Read Buf	3-12
3.5.3	Load Buf	3-12
3.5.4	Load Last Data Byte	3-16
3.5.5	Read Buf	3-16
3.5.6	Transmit	3-16
3.5.7	RSET TBUF	3-16
3.5.8	Release RBUF	3-16
3.5.9	Read Node ADR	3-16
3.5.10	Read Xmit Status	3-16
3.5.11	Read RCVR Status	3-17
3.5.12	Link Enable and Link Disable	3-17
3.6	PB LOAD	3-17
3.7	SEQUENCING LOGIC	3-17
3.7.1	TBUF Load	3-19
3.7.2	Transmit	3-21
3.7.3	TBUF Read (Loopback)	3-21
3.7.4	Valid RCVR Data	3-22
3.7.5	RBUF MLoad	3-24
3.7.6	RBUF Read	3-25
3.8	RCVR STATUS	3-26
3.8.1	CRC ERR	3-26
3.8.2	RBUF A Full, RBUF B Full	3-28
3.8.3	RBUF B First	3-28
3.8.4	RBUF A Bus	3-29
3.8.5	RBUF B Bus	3-29
3.8.6	RCVR A Enable	3-29
3.8.7	RCVR B Enable	3-29

CHAPTER 4 CONTROL STORE

4.1	SIMPLIFIED BLOCK DIAGRAM	4-1
4.2	MICROWORD PARITY	4-5
4.3	CS MICROWORD	4-5
4.3.1	Microword Fields	4-5
4.3.2	Microword Register	4-5
4.4	MAINTENANCE MUX	4-11
4.5	CONTROL STORE SPACE AND LOGIC	4-11
4.5.1	Control Store Space	4-11
4.5.2	Control Store Logic	4-14

CONTENTS

Page

4.6 CONTROL STORE ADDRESS SOURCE....4-16
4.6.1 Maintenance Address Register....4-16
4.6.2 Microsequencer Logic....4-16
4.6.2.1 2911 Microsequencer....4-16
4.6.2.2 Microsequencer Control Logic....4-18
4.6.2.3 Branch Logic....4-22
4.7 MICROCODE START-UP....4-27

CHAPTER 5 DATA PATH MODULE

5.1 GENERAL....5-1
5.2 CIPA BUS....5-5
5.3 DP BUSES AND INTERFACE....5-5
5.3.1 PB OUT Register....5-5
5.3.2 PB IN Register....5-10
5.3.3 XBOR Register....5-10
5.3.4 XBIR Register....5-11
5.4 LS AND VCDT....5-11
5.4.1 LS/VCDT Address Selection....5-13
5.4.2 LS/VCDT Write Strobe Logic....5-16
5.5 MD BUS....5-17
5.5.1 PB IN Register....5-19
5.5.2 MADR and MDATR....5-19
5.5.3 PMCSR And Microword LITERAL Field....5-20
5.6 DP ALU....5-23
5.6.1 2901A Microprocessor....5-23
5.6.2 Data Manipulation....5-26
5.6.3 Carry Look-Ahead Logic....5-27
5.7 DP PARITY GENERATION AND CHECKING....5-27
5.7.1 Parity Generation and Checking Logic....5-27
5.7.2 Receive Buffer Parity Error (RBPE)....5-29
5.7.3 PB IN Register Parity Error (PBIR PE) [Output Parity Error (OPE)]....5-29
5.7.4 CIPA Error....5-32
5.7.4.1 DP To CCI Parity Check....5-32
5.7.4.2 CCI To DP Parity Check (IPE)....5-33
5.7.5 Packet Buffer Parity (PB PAR)....5-34
5.7.6 Local Store Parity Error (LSPE)....5-35
5.7.7 Parity Error (PE)....5-36
5.8 UNSOLICITED CMI REQUESTS....5-36
5.8.1 Starting An Unsolicited Sequence....5-38
5.8.2 Unsolicited Write Sequence....5-41
5.8.2.1 Obtaining The Write Data....5-41
5.8.2.2 Register Selection....5-47
5.8.3 Unsolicited Read Sequence....5-48
5.8.3.1 Register Selection....5-49
5.8.3.2 Transferring The Read Data To The CCI....5-50

CONTENTS

		Page
5.9	DP CONTROL LOGIC	5-51
5.9.1	IB Bus Destination	5-51
5.9.2	IB Bus Source	5-53
5.9.3	Control Signals	5-54
5.10	CCI/DP INTERFACE CONTROL LOGIC	5-56
5.10.1	Port Initiated Write Of CCI	5-57
5.10.2	Port Initiatted Read Of CCI	5-57
5.10.3	Unsolicited Request Operations	5-60
5.11	PORT CLOCKS AND OPERATING MODES	5-62
5.11.1	Port Clocks	5-62
5.11.2	Operating Modes	5-64
5.11.2.1	Run Mode	5-64
5.11.2.2	Uninitialized Mode	5-64
5.11.2.3	Stall Mode	5-64
5.11.2.4	Suspend Mode	5-65
5.11.2.5	Differences Between Stall and Suspend Mode	5-69
5.12	INTERRUPT, INITIALIZE, AND POWER CONTROL FUNCTIONS	5-69
5.12.1	Interrupt Function	5-72
5.12.2	Initialize Function	5-75
5.12.2.1	CCI Initialize Logic	5-75
5.12.2.2	DP Initialize Logic	5-75
5.12.2.3	Boot Timer And Maintenance Timer	5-77
5.12.3	Power Control Function	5-79
5.12.3.1	Power-up Sequence	5-79
5.12.3.2	Power Fail Sequence	5-85
5.12.3.3	Remote Reset Function	5-87

CHAPTER 6 CCI MODULE

6.1	OVERVIEW	6-1
6.1.1	CMI Protocol	6-2
6.1.1.1	Bus Signals	6-2
6.1.1.2	Write Timing	6-8
6.1.1.3	Read Timing	6-8
6.1.1.4	Write Vector Timing	6-11
6.1.2	Major Components	6-11
6.1.2.1	Command/Address Hi Register	6-11
6.1.2.2	Address Lo Register	6-11
6.1.2.3	Byte Mask Register	6-14
6.1.2.4	XMIT File	6-14
6.1.2.5	Return Read Data Register	6-14
6.1.2.6	Interrupt Vector	6-14
6.1.2.7	CNFGR Register	6-14
6.1.2.8	CMI Mux	6-14
6.1.2.9	Address Decode Logic	6-15
6.1.2.10	Command/Address Hold Register	6-15
6.1.2.11	Address Offset Register	6-15
6.1.2.12	Function Register	6-15

CONTENTS

		Page
6.1.2.13	Receive Write Data Register	6-15
6.1.2.14	RCV File	6-15
6.1.3	Simplified Flow Diagrams	6-16
6.1.3.1	Port Initiated Transfers	6-16
6.1.3.2	Write Vector Function	6-19
6.1.3.3	Unsolicited CMI Transfers	6-21
6.2	PORT-INITIATED TRANSFERS	6-24
6.2.1	Load Command/Address And Byte Mask Registers	6-24
6.2.2	Write Function	6-29
6.2.2.1	Load XMIT File	6-29
6.2.2.2	Issue GO	6-31
6.2.2.3	Arbitration	6-34
6.2.2.4	Command/Address Cycle	6-37
6.2.2.5	Unload XMIT File And Status Cycle	6-41
6.2.3	Read Function	6-44
6.2.3.1	Issue GO	6-44
6.2.3.2	Arbitration	6-44
6.2.3.3	Command/Address Cycle	6-44
6.2.3.4	Status Cycle And Load RCV File	6-45
6.2.3.5	Unload RCV File	6-48
6.2.4	Write Vector Function	6-50
6.2.4.1	Issue Interrupt	6-50
6.2.4.2	Arbitration	6-52
6.2.4.3	Write Interrupt Vector	6-52
6.3	UNSOLICITED CMI OPERATIONS	6-58
6.3.1	Command/Address Cycle	6-58
6.3.1.1	Address Space Decoder	6-60
6.3.1.2	Register Decoder	6-60
6.3.1.3	Command/Address Sequence	6-63
6.3.2	Read/Write CCI	6-65
6.3.2.1	Maintenance Function	6-65
6.3.2.2	Writing The CNFGR Register	6-65
6.3.2.3	Reading The CNFGR Register	6-68
6.3.3	Read/Write DP	6-68
6.3.3.1	CIPA Transfer Request	6-69
6.3.3.2	Write DP	6-72
6.3.3.3	Maintenance Initialize (MIN)	6-74
6.3.3.4	Read DP	6-74
6.4	CNFGR REGISTER	6-77
6.4.1	Adapter Code	6-77
6.4.2	PDN, PUP, NO CIPA	6-77
6.4.3	T ACLO, T DCLO, PFD	6-79
6.4.4	NXM, UCE, CRD	6-79
6.4.5	DIAGNOSE	6-80
6.4.6	CTO	6-80
6.4.7	RLTO	6-80
6.4.8	CBPE	6-81

CONTENTS

Page

6.5 PARITY GENERATION AND CHECKING.....6-81
6.6 INITIALIZE AND POWER CONTROL FUNCTIONS.....6-81

APPENDIX A CI750 MNEMONIC GLOSSARY

APPENDIX B FLOW DIAGRAM SYMBOLS

APPENDIX C HARDWARE REGISTERS

C.1 MADR -- Maintenance Address Register..... C-1
C.2 MDATR -- Maintenance Data Register..... C-2
C.3 PMCSR -- Port Maintenance Control/Status Register. C-3
C.4 CNFGR -- Configuration Register..... C-4

FIGURES

1-1 Four-Node CI Cluster.....1-2
1-2 CI750 Connection.....1-6
1-3 CI750 Configuration.....1-7
1-4 CI750 Block Diagram.....1-8
1-5 CI750 Power Distribution.....1-14

2-1 Packet Formats.....2-4
2-2 Link Simplified Block Diagram.....2-6
2-3 Receive Channel Block Diagram.....2-13
2-4 Receive Path Select MUX ECL Logic.....2-16
2-5 PE (Phase Encoded) Data.....2-17
2-6 Manchester Decoder Timing Diagram.....2-19
2-7 Byte Framer Block Diagram.....2-21
2-8 Enabling the RCVR Serial Shift Register.....2-22
2-9 Byte Framer Timing Diagram.....2-23
2-10 RCVR CLK Generator.....2-25
2-11 RCVR CLK Synchronization.....2-27
2-12 Transmit Channel Block Diagram.....2-31
2-13 Sync/Trailer PROM Space.....2-32
2-14 XMIT CLK Generator Block Diagram.....2-35
2-15 XMIT CLK Generator Timing Diagram.....2-36
2-16 Manchester Encoder Timing Diagram.....2-38
2-17 XMIT ECL Drivers.....2-40
2-18 CRC Generator/Checker.....2-42
2-19 Arbitration Flow Diagram.....2-45
2-20 Arbitration Block Diagram.....2-47
2-21 Link Functions.....2-52
2-22 Link Interface Signals.....2-55
2-23 Interface Flow Diagram -- Transmit Operation.....2-56

CONTENTS

		Page
2-24	Interface Flow Diagram -- Receive Operation	2-57
2-25	Message Transmit State Logic	2-59
2-26	Transmit Control Logic	2-61
2-27	Transmit Status	2-64
2-28	ACK Receive State Logic	2-66
2-29	Sync Character Detect Enable PAL	2-68
2-30	Message Receive State Logic	2-70
2-31	ACK Transmit State Logic	2-73
3-1	Packet Buffer Data Flow	3-2
3-2	TBUF Operations	3-5
3-3	RBUF Operations	3-8
3-4	Packet Buffer Clocks	3-11
3-5	Function Decoder and Buffer Select Logic	3-13
3-6	PB Load Logic	3-18
3-7	TBUF Sequencing Logic	3-20
3-8	RBUF Sequencing Logic	3-23
3-9	RCVR Status Logic	3-27
4-1	Control Store Simplified Block Diagram	4-2
4-2	Control Store Block Diagram	4-4
4-3	Microword Parity Checker	4-6
4-4	Microword Fields	4-7
4-5	Control Store Space	4-13
4-6	Control Store Logic	4-15
4-7	Control Store Address Multiplexing	4-17
4-8	2911 Microsequencer	4-19
4-9	Microsequencer Control Logic	4-23
4-10	Branch Logic	4-24
4-11	Microcode Start-Up Flow Diagram	4-28
4-12	Microcode Start-Up Logic	4-30
4-13	Microcode Start-Up Timing Diagram	4-32
5-1	Data Path Module Block Diagram	5-2
5-2	CIPA Bus With DP And CCI Interfaces	5-6
5-3	DP Buses and Interfaces	5-7
5-4	LS/VCDT Block Diagram	5-12
5-5	LS/VCDT Address Selection Simplified Block Diagram	5-14
5-6	LS/VCDT Address Selection Block Diagram	5-15
5-7	Write RAM Timing Diagram	5-18
5-8	ALU Block Diagram	5-24
5-9	Carry Look-Ahead Logic	5-28
5-10	Parity Generation And Checking	5-30
5-11	Unsolicited CMI Operations - Simplified Flow Diagram	5-37
5-12	Starting An Unsolicited Operation	5-39
5-13	Unsolicited CMI Request Logic	5-40
5-14	Unsolicited CMI Write Operation Flow Diagram	5-42

CONTENTS

		Page
5-15	Unsolicited CMI Read Operation Flow Diagram	5-44
5-16	DP Control Logic	5-45
5-17	CCI/DP Interface Control Logic	5-47
5-18	CCI/DP Interface Control Logic - Port Initiated Write Of CCI	5-58
5-19	CCI/DP Interface Control Logic - Port Initiated Read Of CCI	5-59
5-20	CCI/DP Interface Control Logic - Unsolicited Request Operations	5-61
5-21	Port Clocks	5-66
5-21A	Stall Mode Timing	5-67
5-21B	Suspend Mode Timing	5-68
5-22	Interrupt, Initialize, And Power Control Block Diagram	5-70
5-23	Interrupt Logic	5-73
5-24	Interrupt Sequence	5-74
5-25	Initialize Logic	5-76
5-26	Initialize Sequence	5-78
5-27	Power Control Logic	5-80
5-28	Powerup Sequence	5-81
5-29	Power Fail Sequence	5-82
5-30	Power-Fail And Power-Up Interrupt Signals	5-88
6-1	CMI Bus Signals	6-5
6-2	CMI Data And Command/Address Formats	6-6
6-3	CMI Write Timing	6-9
6-4	CMI Read Timing	6-10
6-5	CMI Write Vector Timing	6-12
6-6	CCI Block Diagram	6-13
6-7	Flow Diagram of Port Initiated Transfers	6-17
6-8	Write Vector Function Flow Diagram	6-20
6-9	Flow Diagram Of Unsolicited CMI Transfers	6-22
6-10	Load Flow Diagram for CMD/ADDR HI, ADDR LO, and Byte Mask Registers	6-25
6-11	CCI Register Control Logic	6-27
6-12	Load XMIT File Flow Diagram	6-30
6-13	Issue GO Flow Diagram	6-32
6-14	GO/DONE Logic	6-33
6-15	Arbitration Flow Diagram	6-35
6-16	Arbitration Logic	6-36
6-17	Flow Diagram Of Port Initiated Command/Address Cycle	6-38
6-18	CCI Control Logic	6-40
6-19	Unload XMIT File And Status Cycle Flow Diagram	6-42
6-20	Status Cycle And Load RCV File Flow Diagram	6-46
6-21	Unload RCV File Flow Diagram	6-49
6-22	Issue Interrupt Flow Diagram	6-51
6-23	Issue Interrupt Logic	6-53
6-24	Write Interrupt Vector Flow Diagram	6-54

CONTENTS

		Page
6-25	Interrupt Vector	6-56
6-26	Unsolicited Decode And Register Logic	6-59
6-27	CI750 Address Responses	6-61
6-28	CI750 CMI Address Space vs Register Decoder Outputs	6-62
6-29	Flow Diagram Of Unsolicited Command/Address Sequence	6-64
6-30	Read/Write CCI Flow Diagram	6-67
6-31	Flow Diagram Of CIPA Transfer Request	6-70
6-32	Write DP Flow Diagram	6-73
6-33	Read DP Flow Diagram	6-75
6-34	CNFGR Register Logic	6-78
B-1	Flow Diagram Symbols	B-1
C-1	Maintenance Address Register (MADR) Bit Fields	C-2
C-2	Maintenance Data Register (MDATR) Bit Field	C-4
C-3	Port Maintenance Control/Status Register (PMCSR) Bit Fields	C-6
C-4	Configuration Register (CNFGR) Bit Fields	C-8

CONTENTS

Page

TABLES

1-1 CI750 Related Documents....1-3

2-1 Link State Diagrams....2-11
2-2 Link Clocks....2-25
2-3 N Load Mux Selection....2-48

3-1 Link Control Codes Vs PB Function Commands....3-14
3-2 Load Buffer Select Code....3-15
3-3 Read Buffer Select Code....3-16

4-1 Microword Fields....4-8
4-2 Maintenance Mux Selection Code....4-12
4-3 Microsequencer Control Functions....4-20
4-4 Branch Conditions....4-26

5-1 CIPA Bus Signals....5-8
5-2 LSA Mux Selection Code....5-13
5-3 MD Bus Data Sources....5-17
5-4 PMCSR Bits....5-21
5-5 ALU Source Code....5-25
5-6 ALU Function Code....5-25
5-7 ALU Destination Code....5-26
5-8 IB DST Code....5-52
5-9 IB SRC Code....5-53
5-10 REG SEL Code....5-56
5-11 Port Clocks....5-63

6-1 CMI Bus Signals....6-2
6-2 Functionn Code....6-7
6-3 MUXA/MUXB SEL <B:A> Code....6-39
6-4 Interrupt Vector Values....6-57
6-5 CI750 I/O Slots....6-60

C-1 CNFGR Bits....C-9

CHAPTER 1
INTRODUCTION

1.1 MANUAL SCOPE

This document provides a technical description of the CI750 computer interconnect hardware. It does not treat the CI750 port architecture or other software applications such as the CI750 port driver, command queues, or the VAX/VMS operating system.

A basic description of the CI750 computer interconnect is given in this chapter. The CI750 contains four extended hex "L" series modules. Chapters 2, 3, 5 and 6 provide a detailed description of each of the four 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 it can be treated 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.

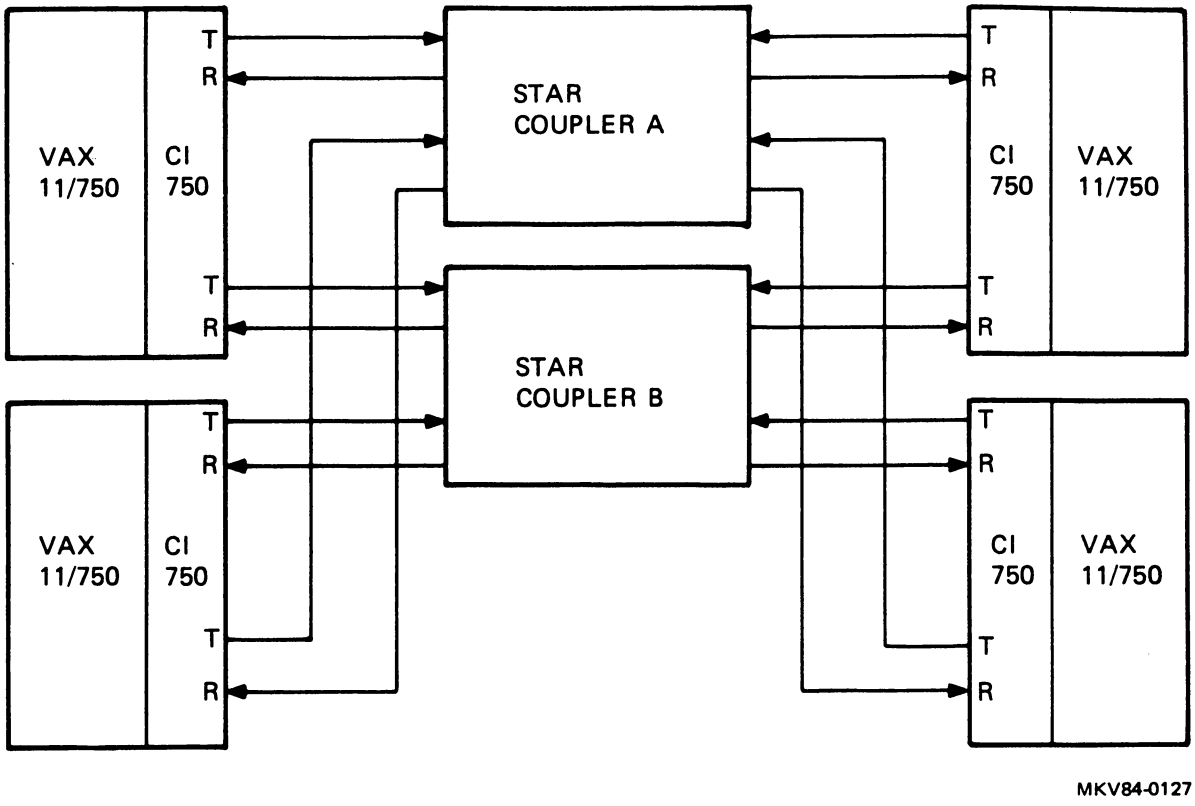


Figure 1-1 Four-Node CI Cluster

Each node within a cluster connects to the computer interconnect via a CI750 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.

1.3 RELATED DOCUMENTS

Table 1-1 is a list of documents providing additional information related to the CI750.

Table 1-1 CI750 Related Documents

Item	Title	Document Number	Contents
1	CI750 User's Guide	EK-CI750-UG	Contains instructions for unpacking, installing, and acceptance testing the CI750. A physical description of the CI750 is also provided. Information is also provided on the CI750 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.

Table 1-1 CI750 Related Documents (Cont)

Item	Title	Document Number	Contents
3	VAX-11/750 Central Processor Unit Technical Description	EK-KA750-TD	Contains a general overall description of the VAX-11/750 plus a detailed discussion of the central processor unit. Included in this discussion is a complete description of the CMI bus including bus signals, timing, CMI protocol, and the VAX-11/750 modules that interface with the CMI.
4	H7202D Power Supply Specification	SP-H7202-D	Contains complete mechanical and electrical specifications for the H7202D power supply. Also included is a general description of the H7202D.
5	H7202B Power System Technical Description	EK-PS730-TD	Contains a physical and functional description of the H7202B power supply.
6	VAX Architecture Handbook	EB-19580-20	Contains a description of the VAX family architecture, including data representations, instructions, registers, and operational modes.
7	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 CI750 INTERFACE

The CI750 is the interface used to connect a VAX-11/750 system to the CI cluster. It connects between the CPU memory interconnect (CMI) of the host system and the CI cluster. Figure 1-2 illustrates the CI750 connection.

The CI750 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 CI750 reduces the amount of overhead software processing required to complete high-level intercomputer communications.

The four modules containing the CI750 logic are listed below.

1. Link Interface Module (ILI) L0100
2. Packet Buffer Module (IPB) L0101
3. Data Path Module (CDP) L0400
4. CMI CIPA Interface Module (CCI) L0009

Figure 1-3 illustrates the configuration of the CI750 modules. The CCI module is installed into one of the three MBA option slots in the VAX-11/750 backplane. The other three modules are housed in a CI750-C Computer Interconnect Port Adapter (CIPA) expander cabinet. The CI750-C expander cabinet is commonly referred to as the CIPA cabinet and will be so referenced throughout this manual. The CIPA cabinet is connected to the host CPU cabinet by a 40-pin CIPA bus cable. As shown in Figure 1-3, the cable actually interconnects the CCI module (in the host CPU cabinet) with the DP module (in the CIPA cabinet).

Figure 1-4 is a block diagram of the CI750. Refer to it in the following discussion of the CI750 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.

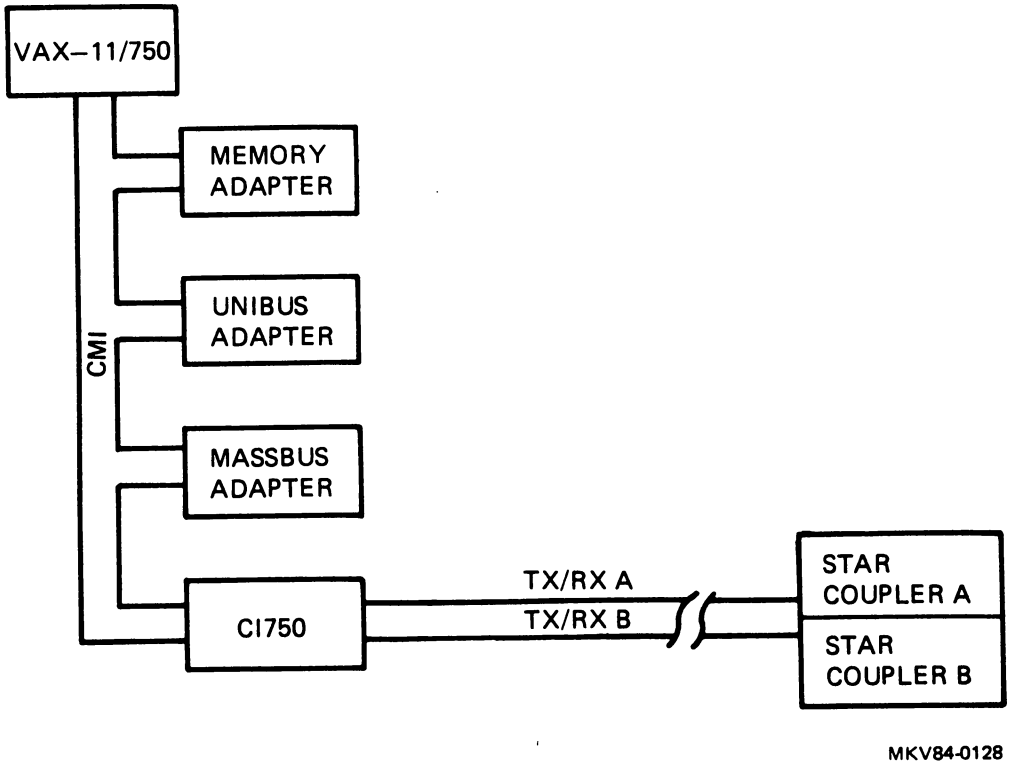


Figure 1-2 CI750 Connection

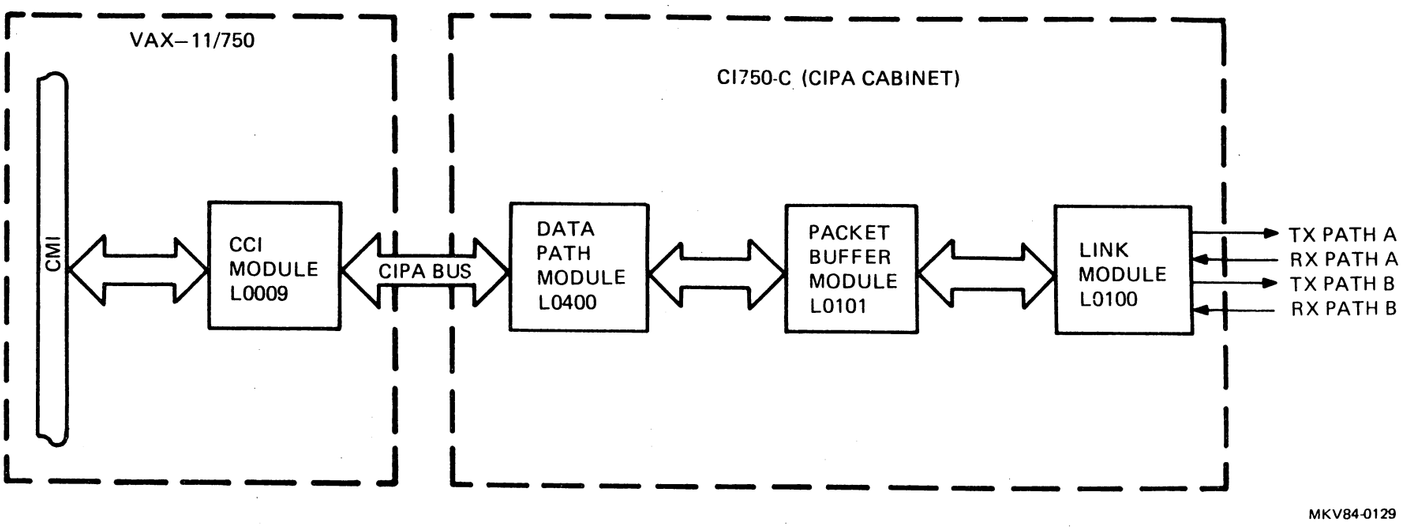


Figure 1-3 CI750 Configuration

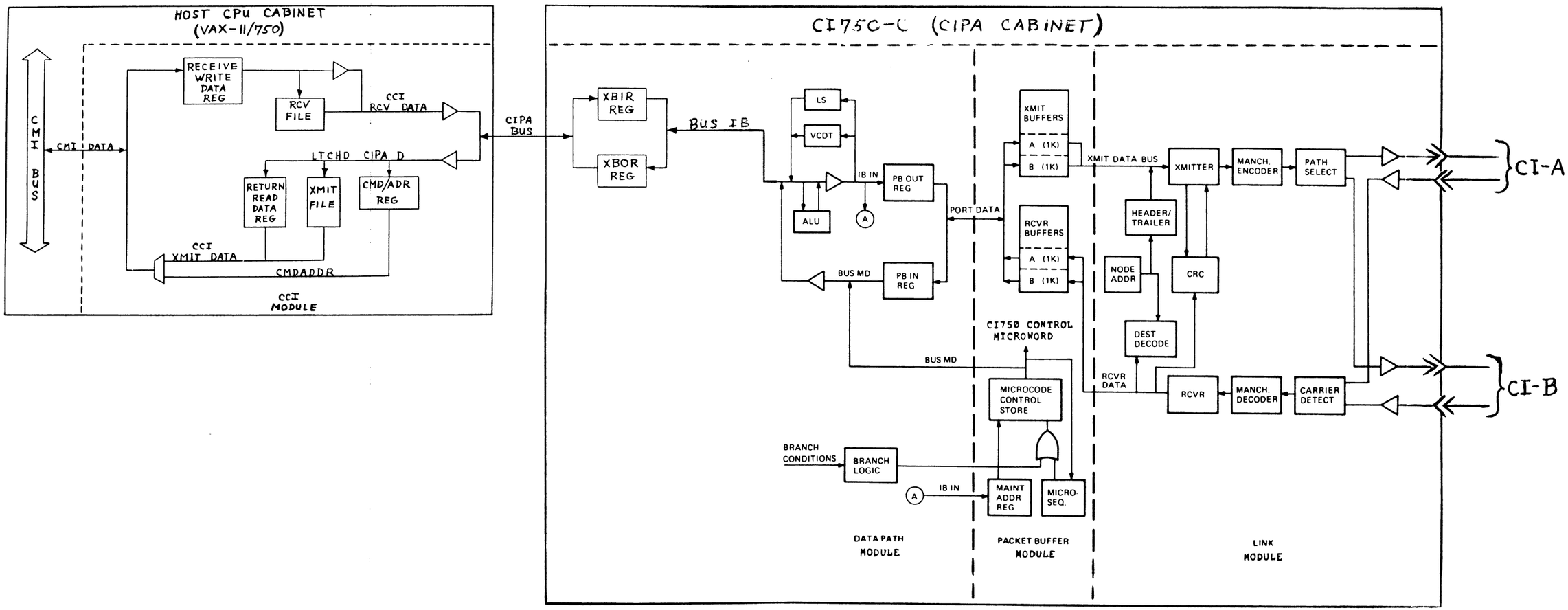


Figure 1-4 CI750 Block Diagram

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 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 CI750. 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 packets are loaded into receive buffer B while the DP is unloading the data from receive buffer A.

The CI750 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 CI750. 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 CI750. The test results are ORed with the microsequencer output to provide branching of the microcode sequences.

The CI750 control microword can be read by the host system via the MD bus 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)

Data flow within the DP is under microcode control. The microcode implements this control by selecting the source and destination for the data on the main DP internal bus (IB). There are several possible data sources and destinations for the IB bus. These are:

1. PB IN and PB OUT registers
2. XBIR (external bus input register) and XBOR (external bus output register)
3. LS (local store)
4. VCDT (virtual circuit descriptor table)
5. ALU (arithmetic logic unit)
6. CS (control store)
7. MD (miscellaneous data)

The PB IN and PB OUT registers interface the DP to the PB via the PORT DATA bus. The PB OUT register can be an IB bus destination (via the IB IN bus) while the PB IN register can be a source for the IB bus (via the MD bus). The three DP buses (IB, IB IN, MD) are 32 bits wide. The PB IN and PB OUT registers accomplish the format conversions necessary to interface with the 8-bit PORT DATA bus.

The XBIR and XBOR registers interface the DP to the CCI module via the CIPA bus. The XBIR register can be a source for the IB bus while the XBOR register can be an IB bus destination. The data on the CIPA bus is in a 16-bit word format. The XBIR and XBOR registers accomplish the format conversions necessary to interface with the 32-bit IB bus.

LS is 256 x 32 of RAM space used to store software status blocks and software registers associated with the C1750 port architecture. LS can be either a destination (via the IB IN bus) or a source for the IB bus.

The VCDT is 256 x 16 of RAM space used to store CI node parameters. The VCDT can be either a destination (via the IB IN bus) or a source for the IB bus.

The ALU is used to perform general purpose arithmetic and logical operations. It interfaces directly with the IB bus where it may serve as either a source or a destination.

The CS in the PB can be read or written from the DP IB bus. The CS can be a data source via the MD bus, or a data destination via the IB bus.

The MD bus can access other miscellaneous data (e.g., selected registers, microword field) which then becomes the data source for the IB bus.

1.4.4 CMI CIPA Interface Module (CCI)

The basic function of the CCI module is to interface the CI750 with the VAX-11/750 CMI bus. All CMI protocol and timing must be followed while transferring data to and from the CMI.

XMIT (transmit) and RCV (receive) files act as isolation buffers for data transferring through the CI750. The CMI side of the files are loaded and unloaded under CMI timing and control while the DP side of the files are loaded and unloaded under CI750 microcode control.

In a port initiated operation (CI750 is CMI bus master), the microcode loads command-address data from the the DP into the CMD/ADR register. The command-address data routes from the CIPA bus to the CMD/ADR register via the LTCHD CIPA D bus.

If the command-address data specified a write operation, the microcode also loads CI write data from the DP into the XMIT file via the same path. Up to four data longwords can be stored in the XMIT file. The microcode then signals the CCI that write data is ready in the XMIT file. Upon being signaled by the microcode, the CCI arbitrates for the CMI bus. When the CCI has won control of the CMI bus, the command-address data is unloaded from the CMD/ADR register onto the CMDADDR bus. The command-address data is selected by a mux and coupled to the CMI DATA lines on the CMI. The XMIT file is then unloaded onto the CCI XMIT DATA bus where it is mux selected for the CMI DATA lines. The data from the XMIT file is written at the CMI address specified in the command-address data. The transfer of data from the CMD/ADR register and the XMIT file to the CMI is controlled and timed from the CMI bus.

If this is a port initiated read operation, the CMD/ADR register is loaded, the microcode signals the CCI that the command-address data is ready, the CCI arbitrates for the CMI bus, and the command-address data is placed onto the CMI. The CCI then takes the read data off the CMI DATA lines and loads it into a Receive Write Data Register and then into the RCV file. Up to four data longwords can be stored in the RCV file. The transfer of the read data from the CMI to the RCV file is controlled and timed from the CMI bus. The microcode is notified that read data is in the RCV file whereupon it proceeds to unload the RCV file onto the CCI RCV DATA bus. The data is then coupled to the DP via the CIPA bus.

Note the reversal in orientation of the "transmit" and "receive" terms from how they were used in the other CI750 modules. Previously, "transmit" had been used in the sense of transmitting data out to the CI bus, and "receive" in the sense of receiving data from the CI bus. In the CCI, "transmit" is used to indicate the transmission of data to the CMI bus, and "receive" is used to indicate the reception of data from the CMI. Hence, the file used to hold data received from the CI is the XMIT file because this data is to be transmitted to the CMI. Likewise, the file used to hold the data to be transmitted to the CI is the RCV file because this data was received from the CMI.

The CCI module provides for CPU access of many CCI and DP registers via unsolicited CMI transfers (CI750 is CMI bus slave). Both reads and writes of the registers can be performed. During unsolicited operations, the Return Read Data Register and the Receive Write Data Register are used instead of the XMIT and RCV files, to transfer the data.

When a DP register is being read, the read data is taken from the CIPA bus and loaded into the Return Read Data Register via the LTCHD CIPA D bus. The read data is then unloaded onto the CCI XMIT DATA bus and then mux selected for transfer to the CMI DATA lines on the CMI bus.

When a DP register is being written, the write data is taken from the CMI DATA lines on the CMI bus and loaded into the Receive Write Data Register. The write data is then unloaded and passed to the CCI RCV DATA bus. From here the write data is coupled to the DP via the CIPA bus.

Another function performed by the CCI module is requesting interrupts of the host CPU when service routines must be run on the CI750.

1.4.5 CI750 Power (Figure 1-5)

Power for the CCI module is obtained from the power system in the host CPU cabinet. The +5.0 V operating voltage and the ground return are obtained from the card cage backplane as is the UBS ACLO and UBS DCLO (see VAX-11/750 documentation listed in Table 1-1). UBS ACLO and UBS DCLO signal a power-up or power-down condition within the CPU cabinet according to power system protocol.

Power within the CIPA cabinet (CI750-C) is supplied from an H7202D Power Supply containing an H7200 +5.0 V Regulator and an H7216 -5.3 V Regulator. The supply receives 120 V, 60 Hz from a switched outlet on a power controller located in the cabinet. The supply provides +5.0 V to the three CI750 modules located in the CIPA cabinet (DP, PB, link) and -5.3 V to the link module. A ground return is provided from each module back to the supply. The supply also provides ACLO and DCLO to the DP module to signal a power-up or a power-down condition within the CIPA cabinet.

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

A description of the H7202D power supply is contained in the engineering specification listed in the table of related documents (Table 1-1). Also listed as a related document is the technical description manual for the H7202B power supply. This document is applicable to the H7202D supply when it is considered that the H7202D is an H7202B with the H7211 communications module removed and the H7213 regulator replaced with the H7216 regulator (the basic difference between the two regulators being their current ratings).

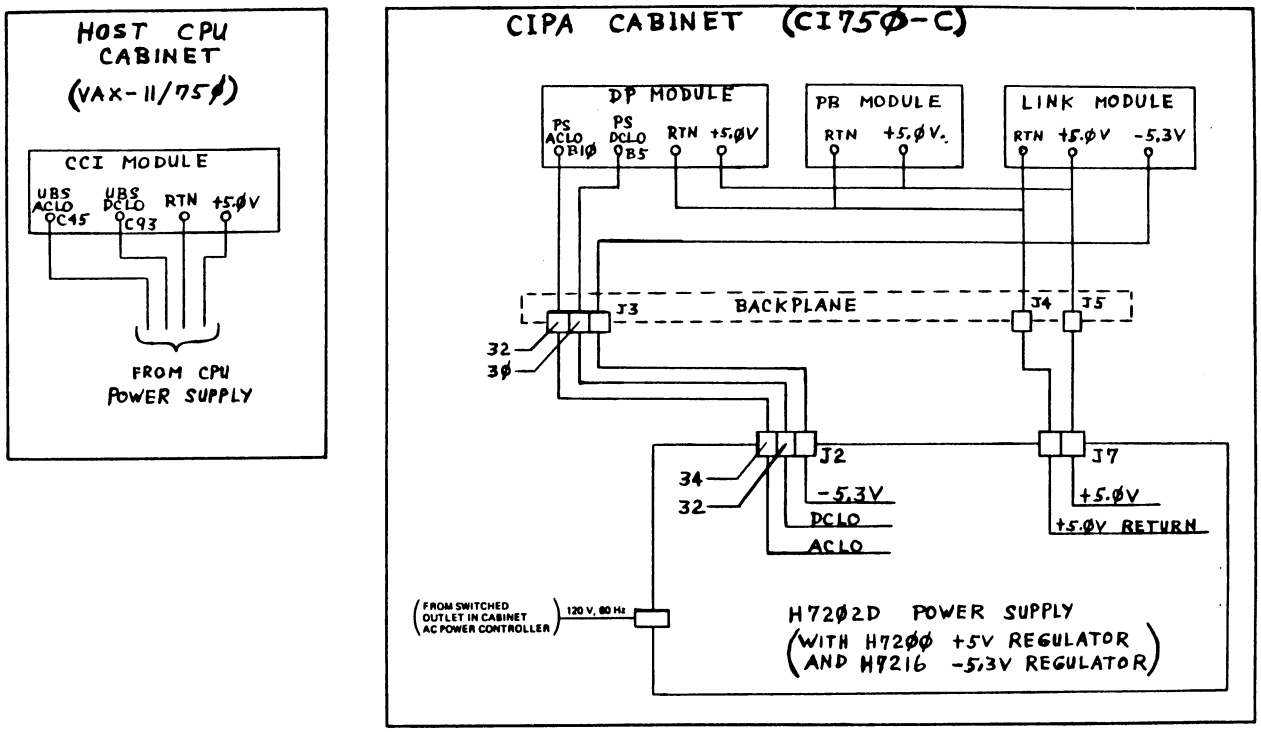


Figure 1-5 CI750 Power Distribution

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.

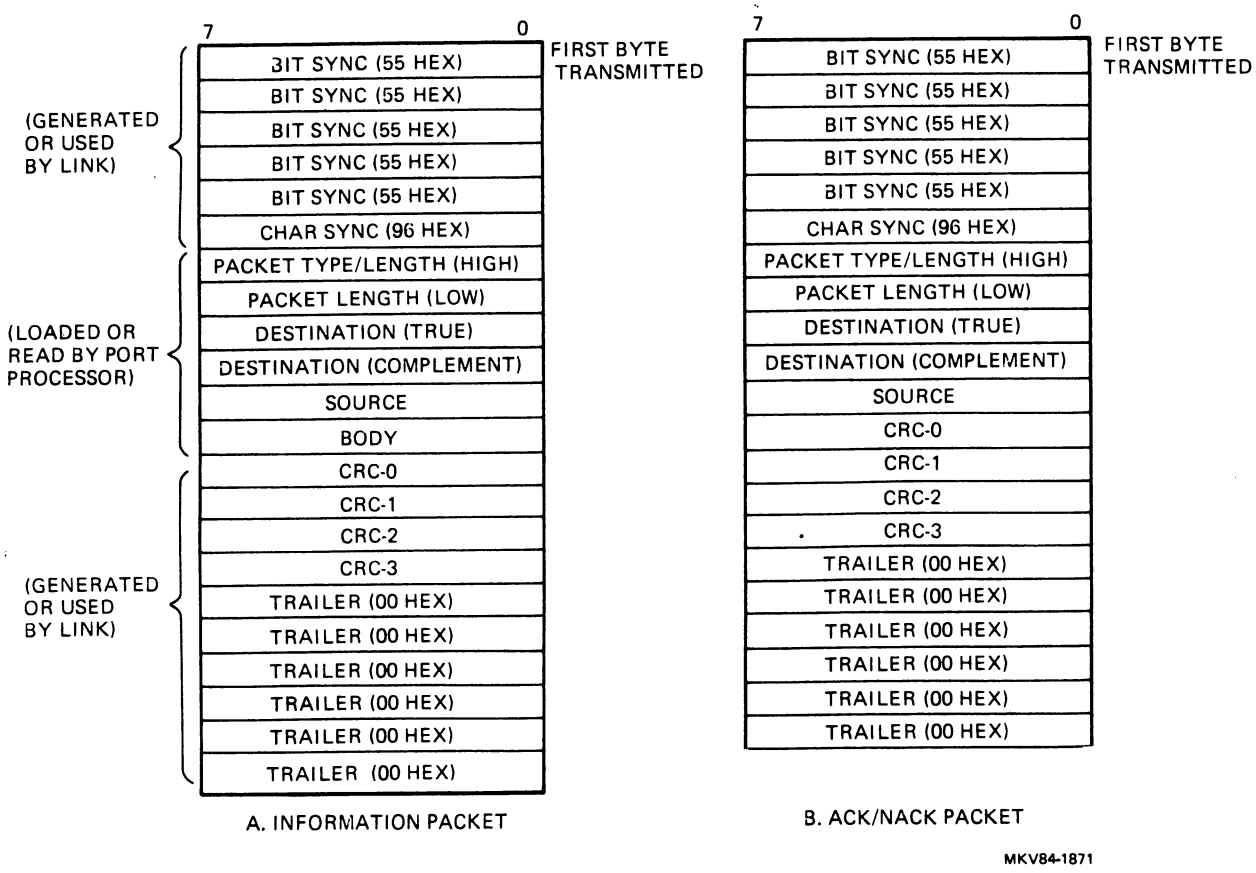


Figure 2-1 Packet Formats

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.

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.

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

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.

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 Ø'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.

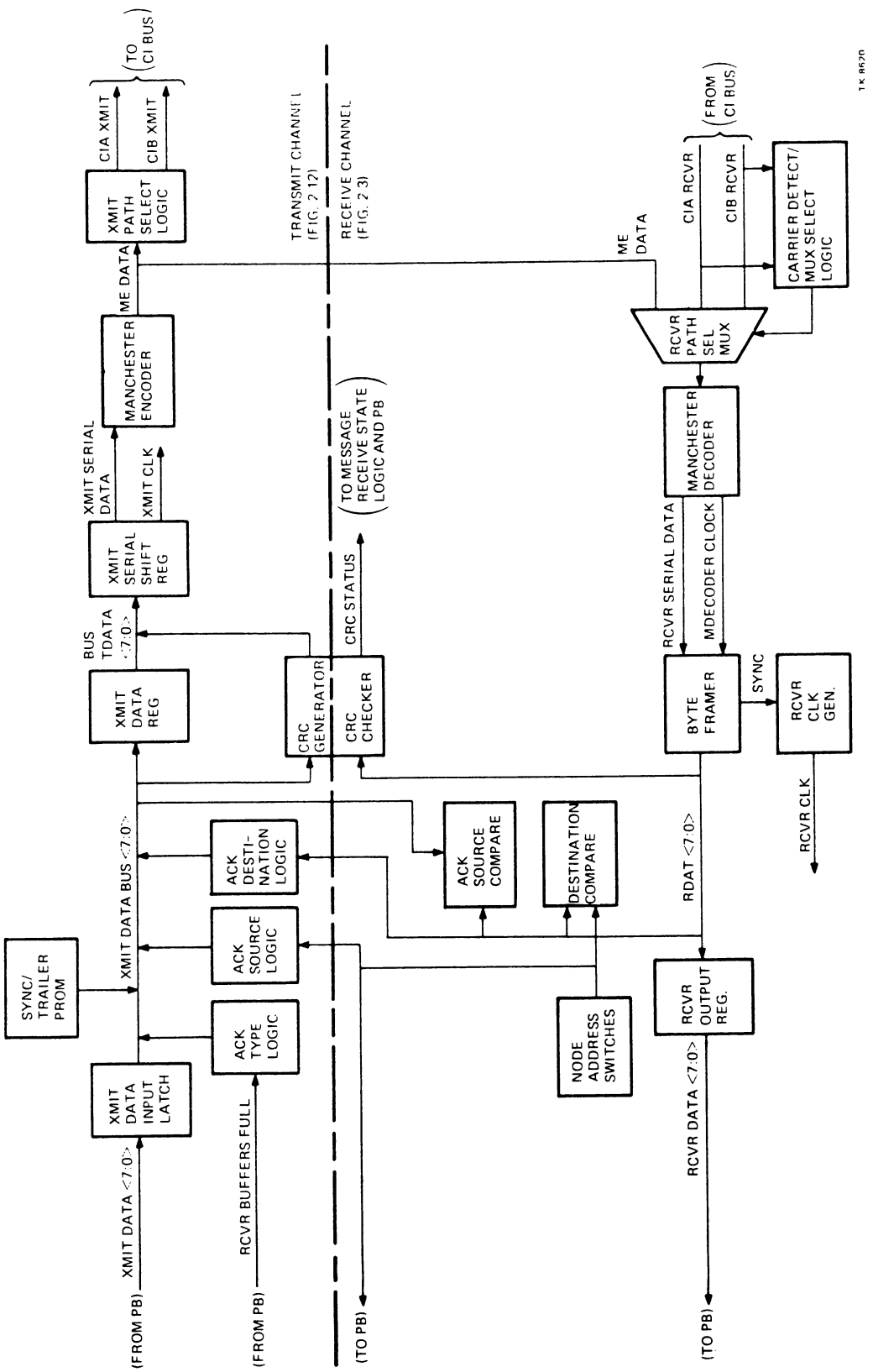


Figure 2-2 Link Simplified Block Diagram

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.

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 7Ø 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.

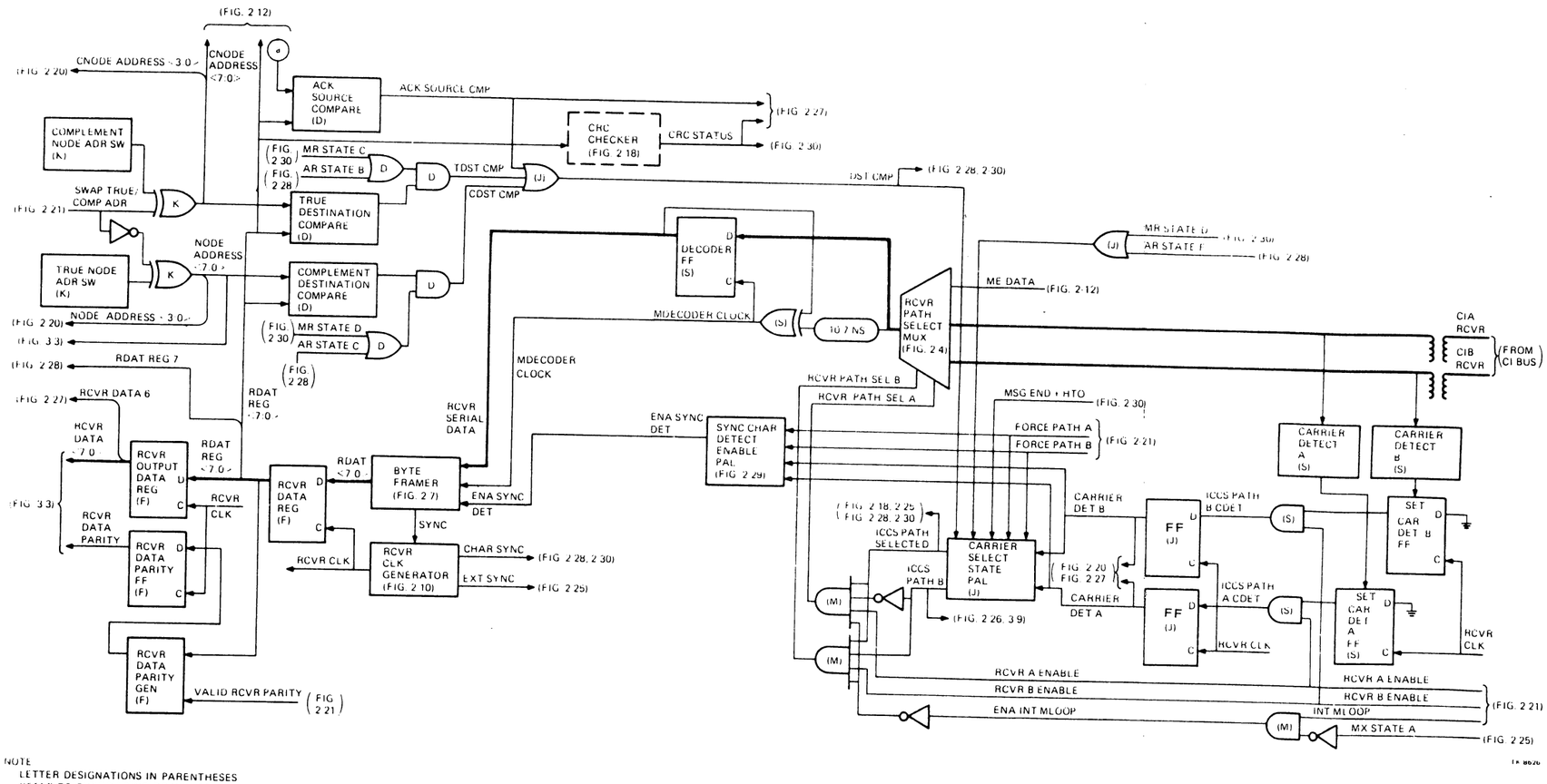
2.4.1.1 Carrier Detect Logic -- Identical and parallel logic monitors paths A and B. If a carrier is present on CI path A, the carrier detect A logic sets the carrier detect A flip-flop. If the port has enabled channel A (RCVR A ENABLE true), ICCS PATH A CDET asserts and causes CARRIER DET A to be asserted 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.



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

Figure 2-3 Receive Channel Block Diagram

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.

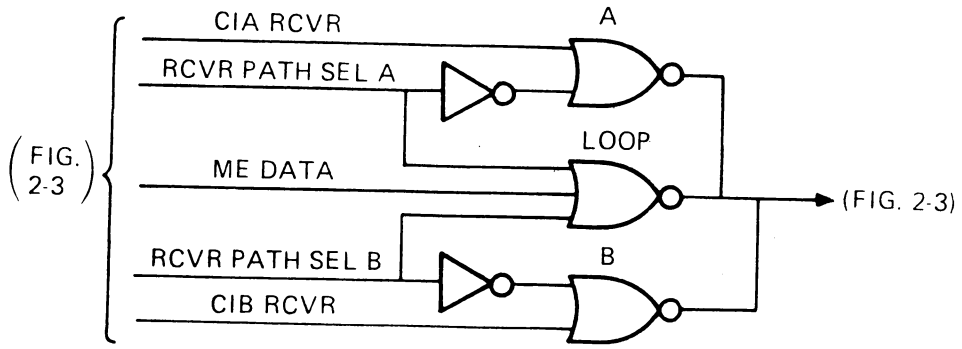
Refer to Figure 2-4. If RCVR PATH SEL A is true, the output of OR gate A can follow the CIA RCVR signal input. The signal RCVR PATH B is false which holds the output of OR gate B low. In ECL logic, a signal low is the non-active state and a high is the active state. Any gate connected to a common line 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.

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).



NOTES:

1. THE LOGIC IN THIS FIGURE IS CONTAINED ON SHEET S OF THE ENGINEERING DRAWINGS.

TK-8614

Figure 2-4 Receive Path Select Mux-ECL Logic

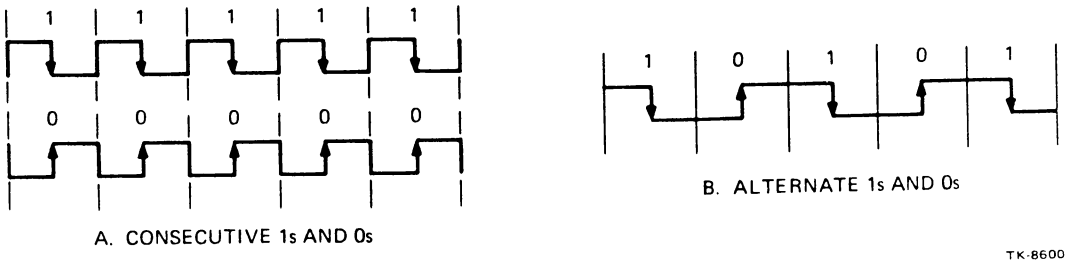


Figure 2-5 PE (Phase Encoded) Data

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 1/4 into each bit cell. This centers the rising edge in the valid strobe area (first half of the bit cell).

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.

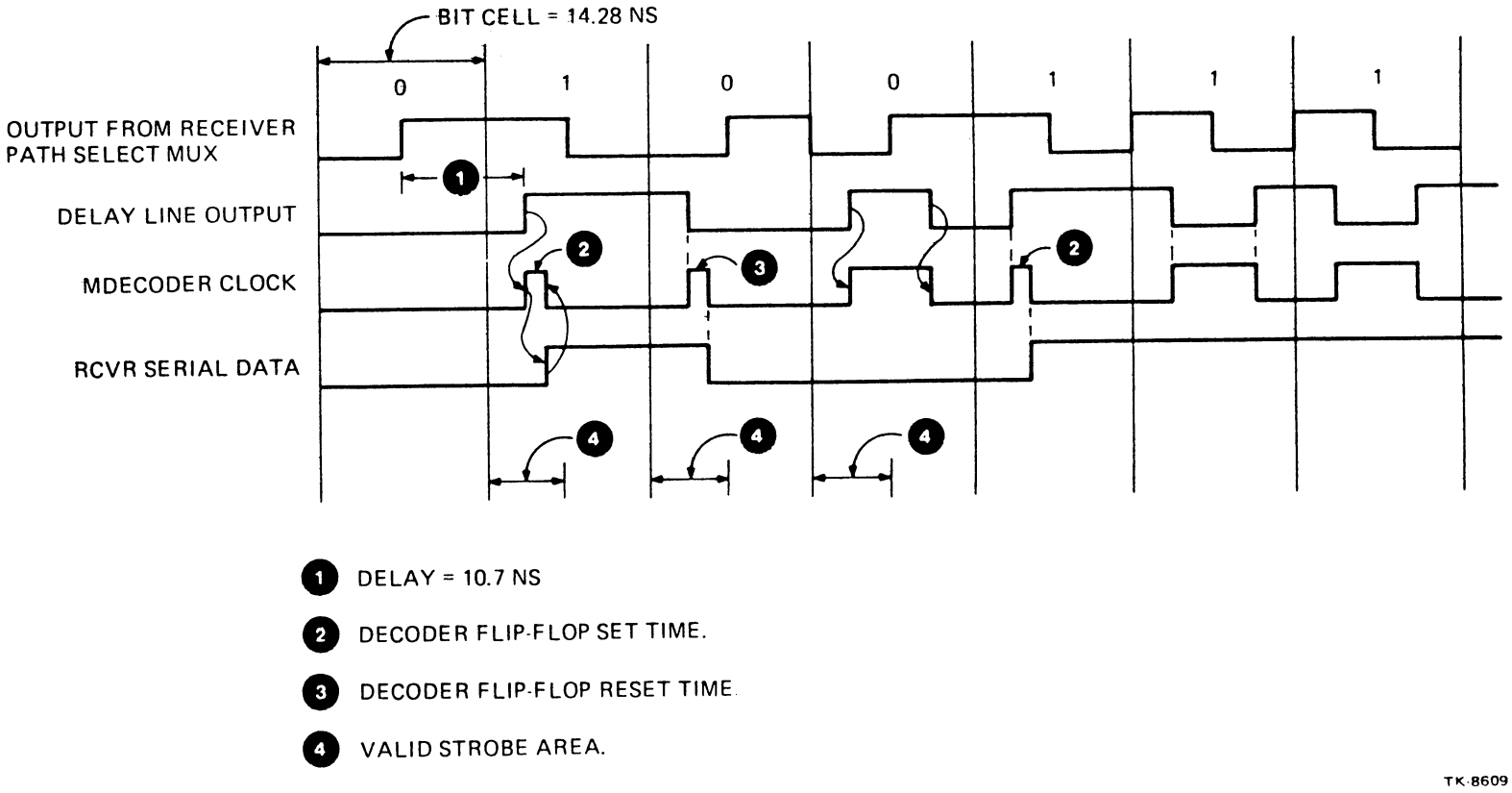


Figure 2-6 Manchester Decoder Timing Diagram

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.

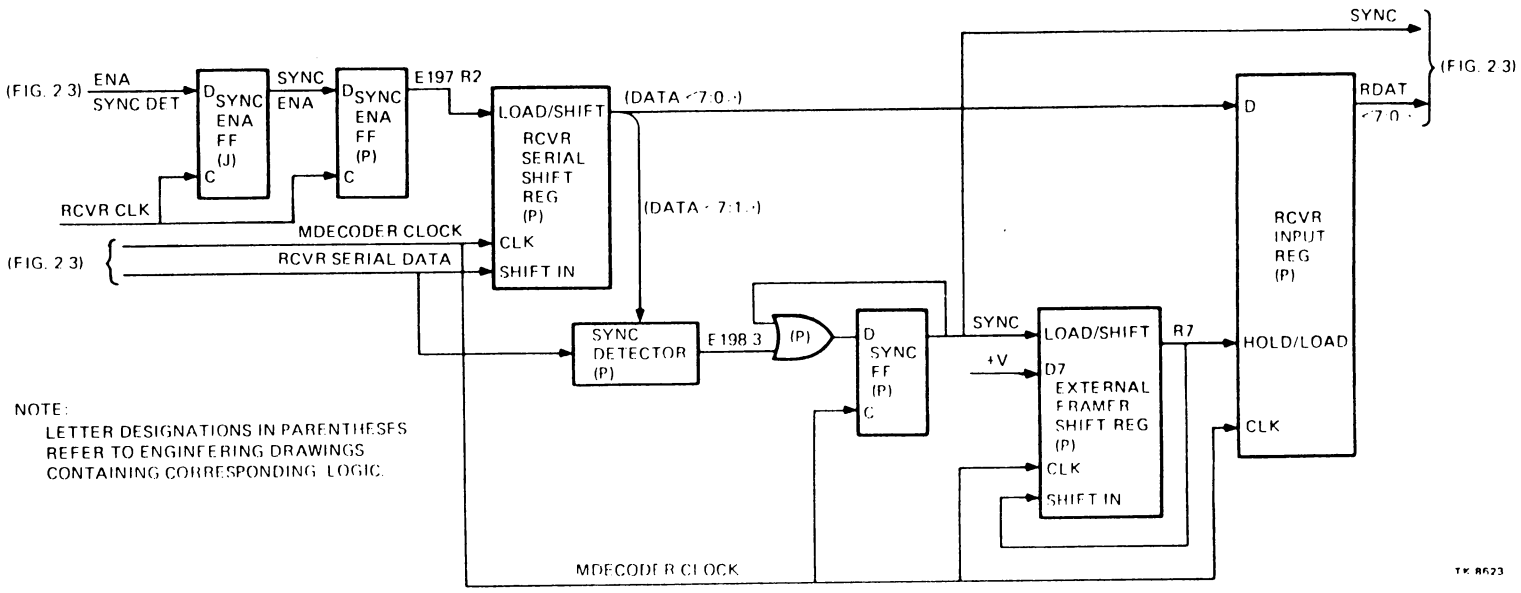


Figure 2-7 Byte Framer Block Diagram

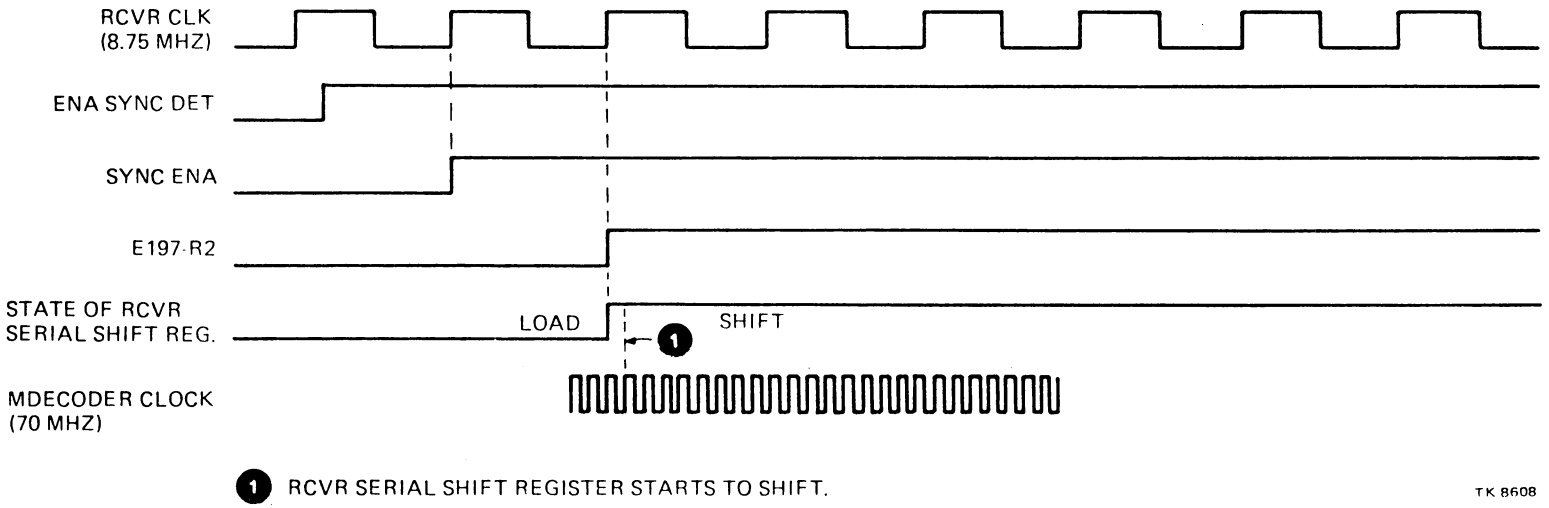


Figure 2-8 Enabling the RCVR Serial Shift Register

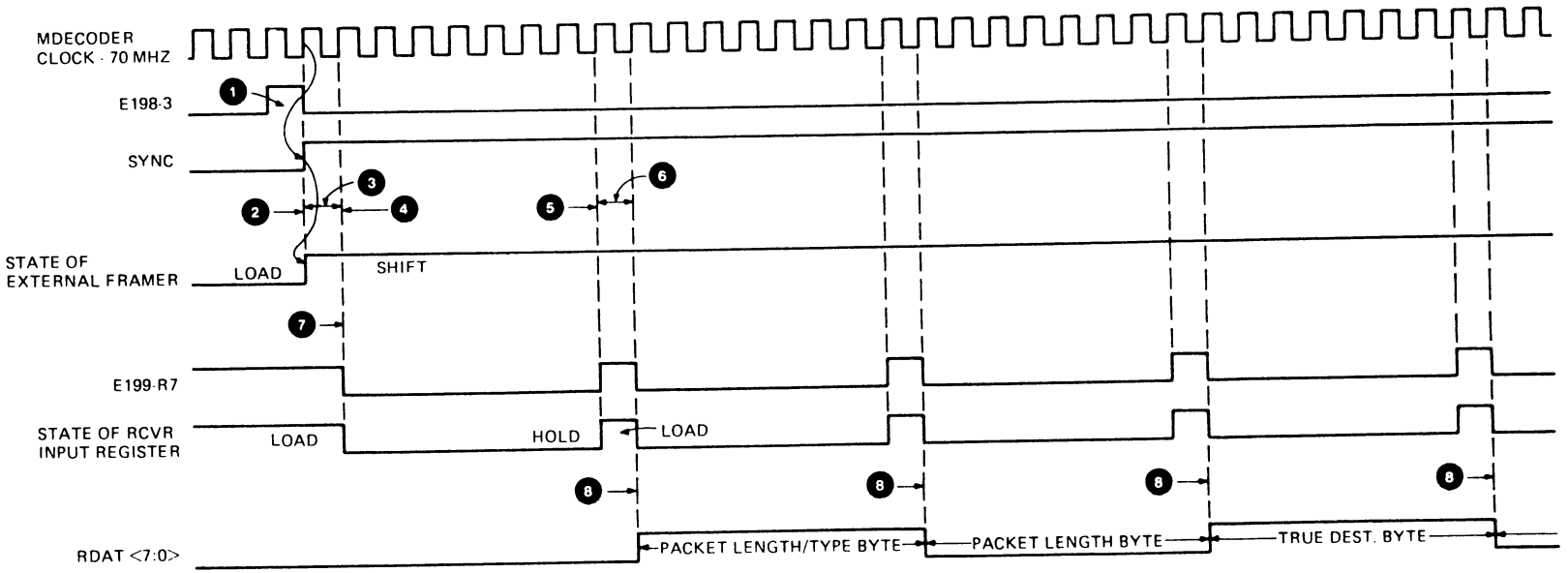


Figure 2-9 Byte Framer Timing Diagram

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.

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).

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

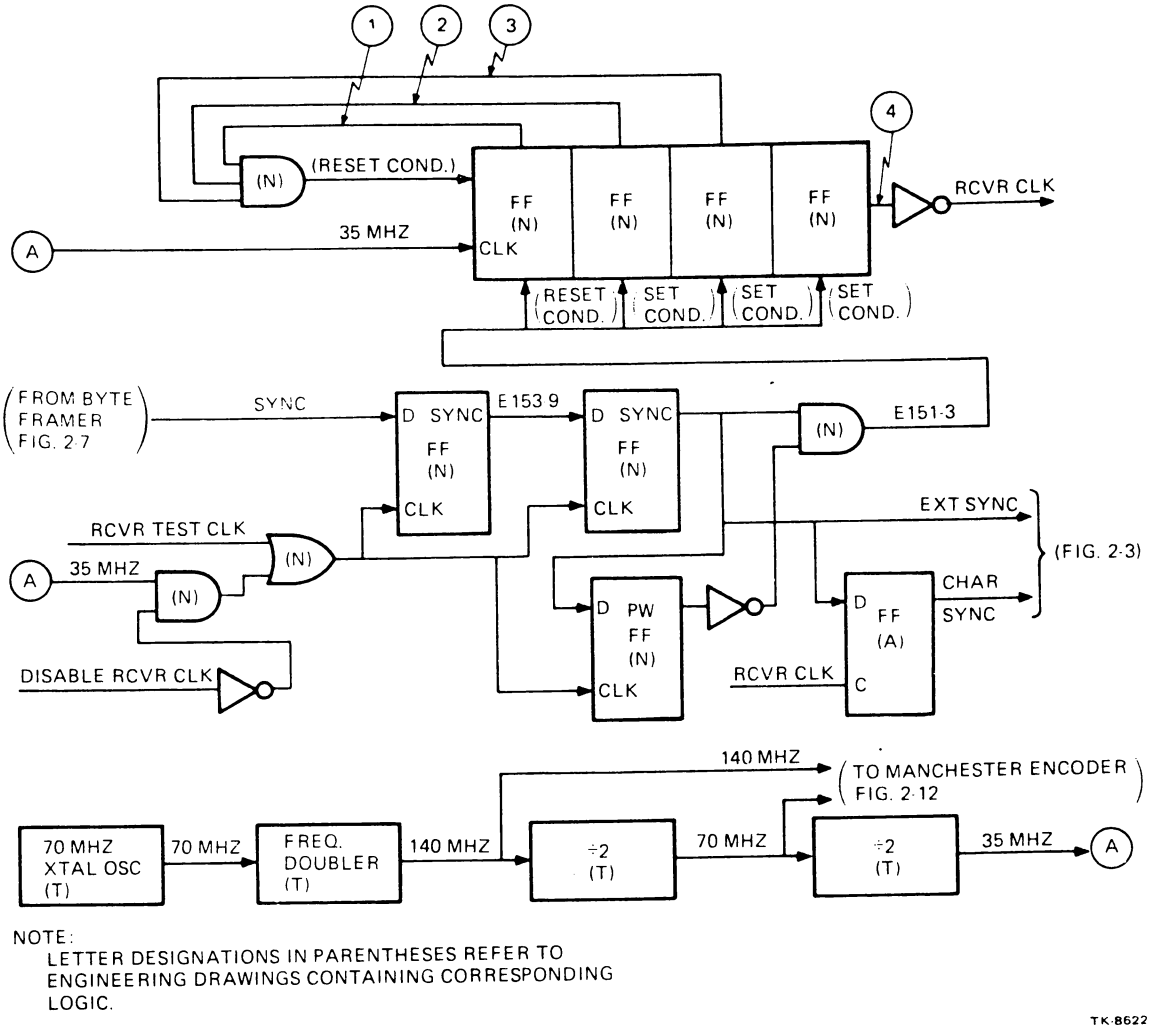


Figure 2-10 RCVR CLK Generator

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 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.

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.

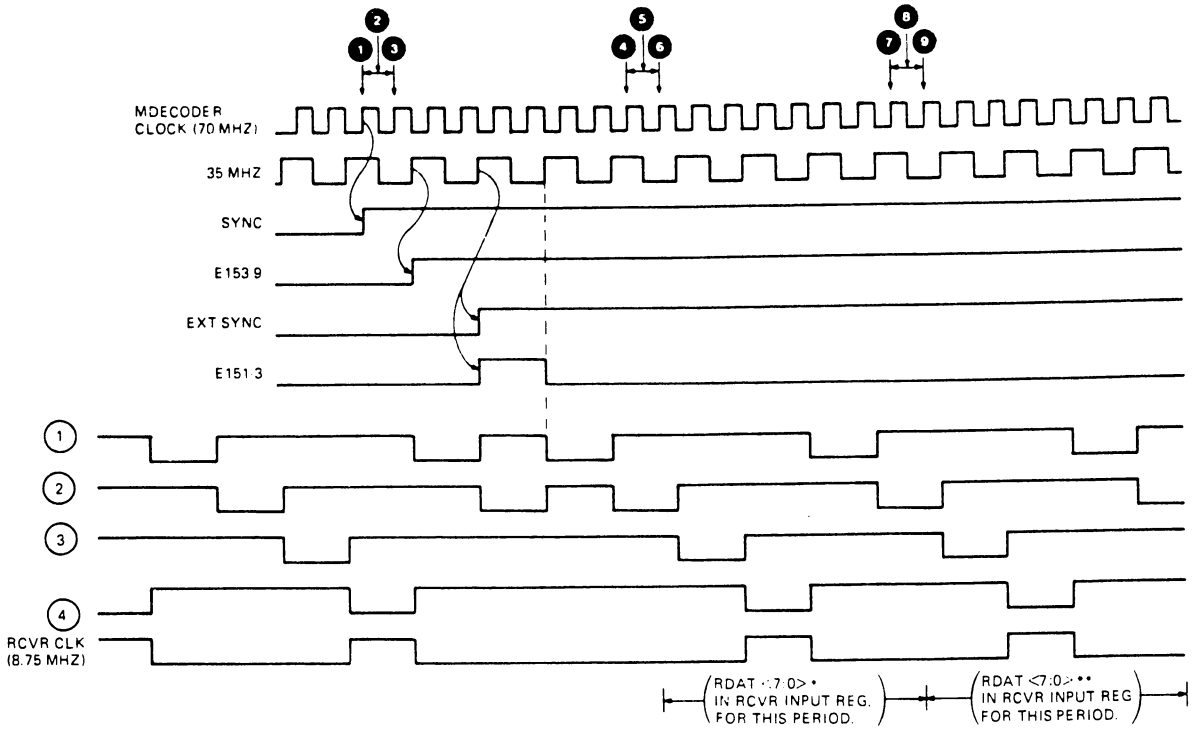


Figure 2-11 RCVR CLK Synchronization

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).

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 x 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.

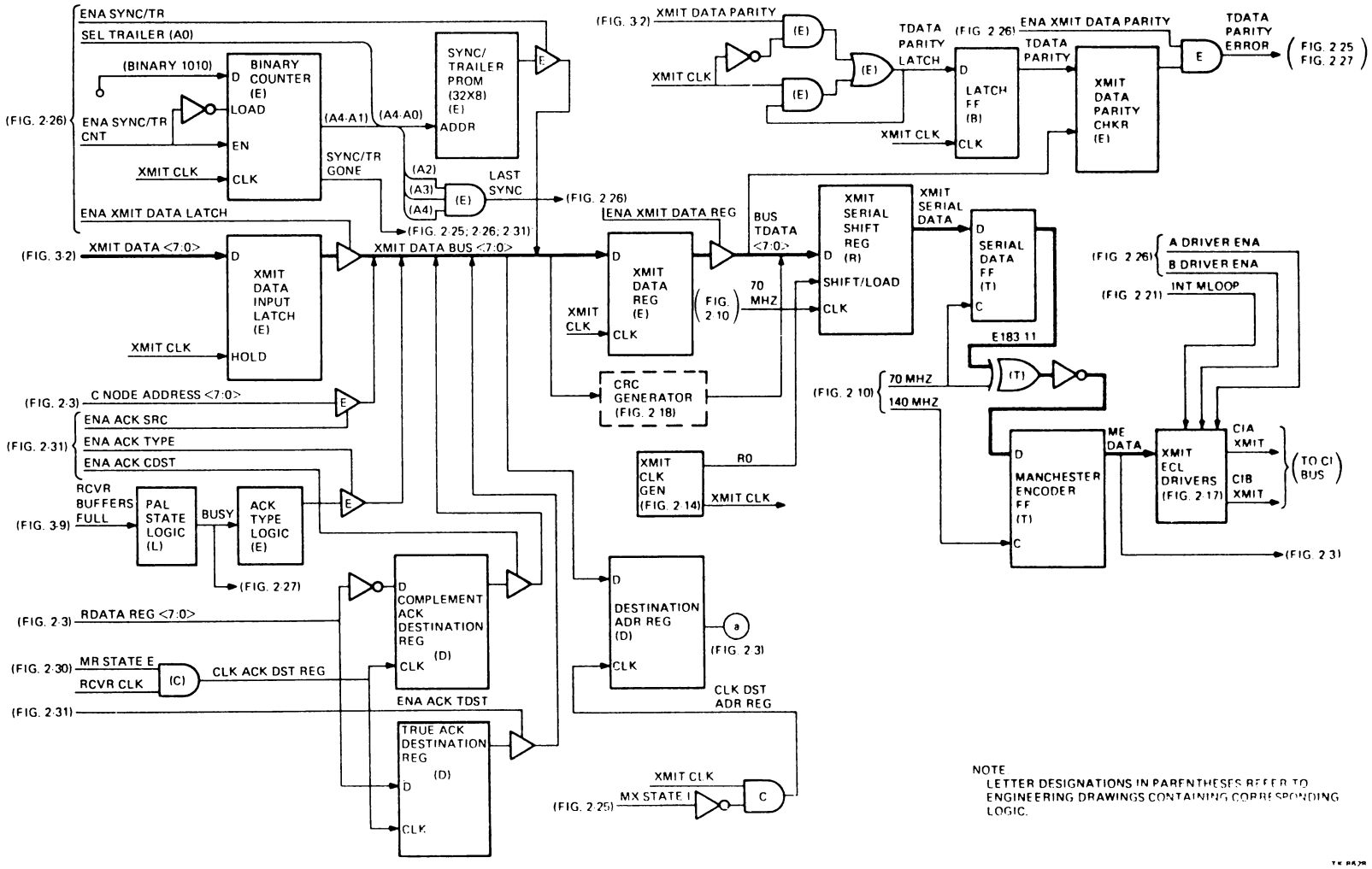


Figure 2-12 Transmit Channel Block Diagram

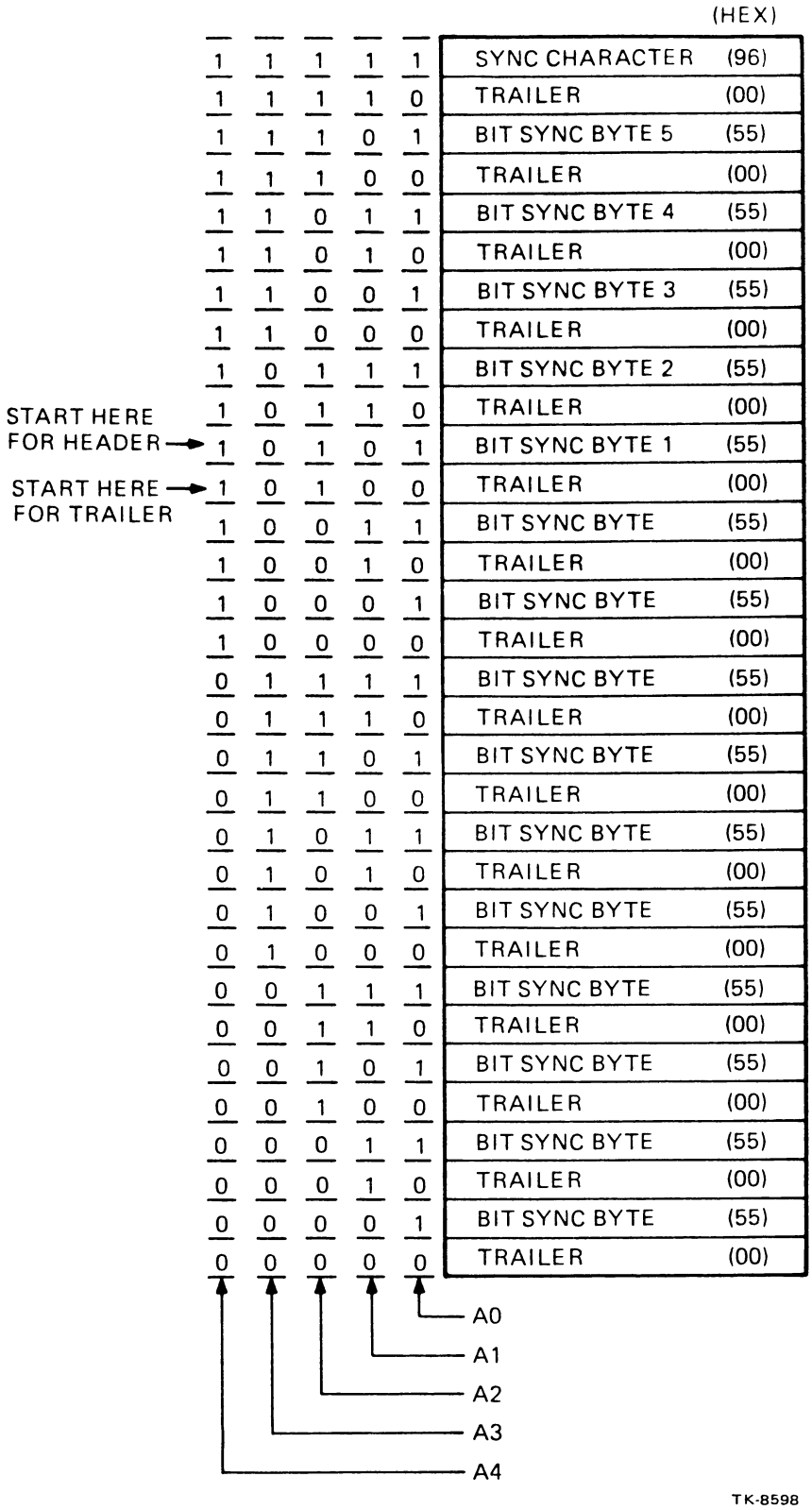


Figure 2-13 Sync/Trailer PROM Space

2.5.3 ACK Packet Inserts

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 the 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.

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 XMIT 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.

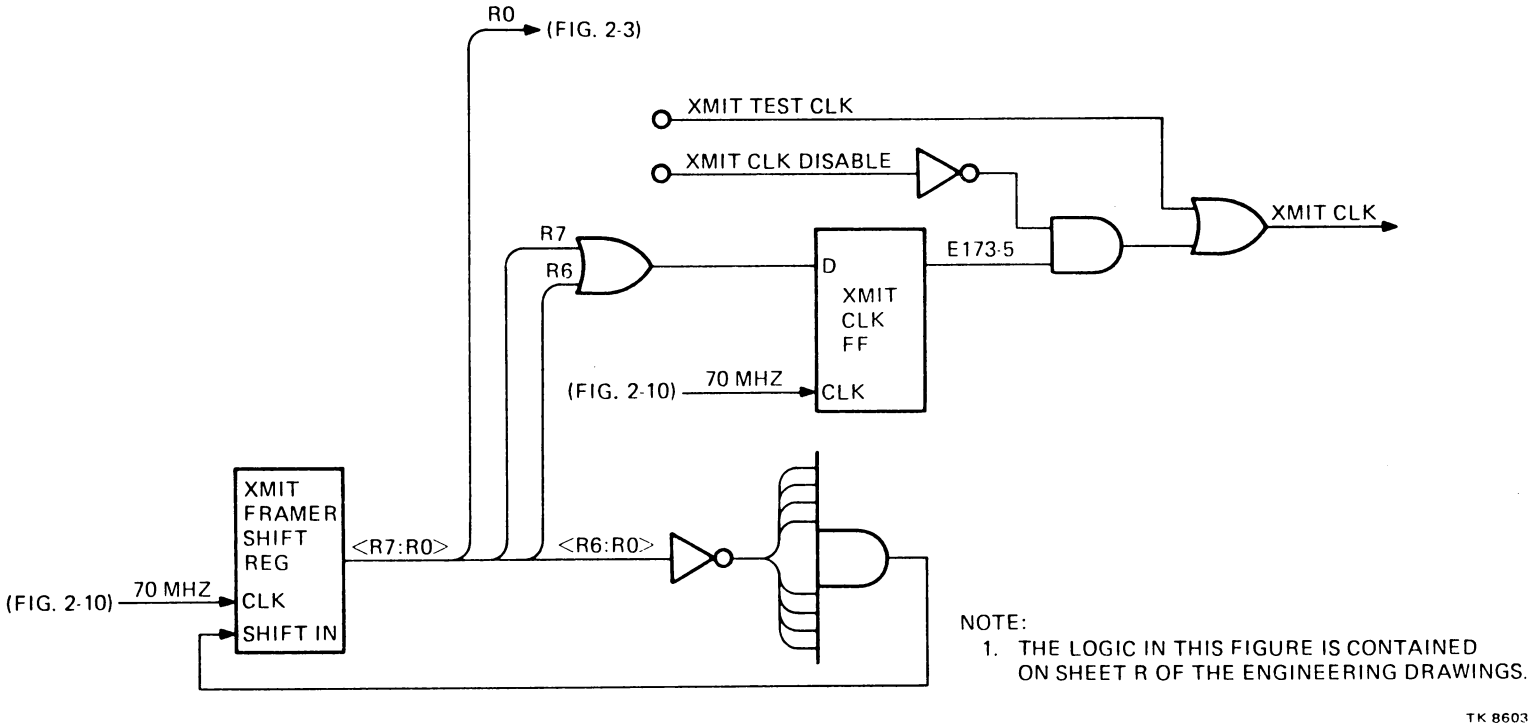


Figure 2-14 XMIT CLK Generator Block Diagram

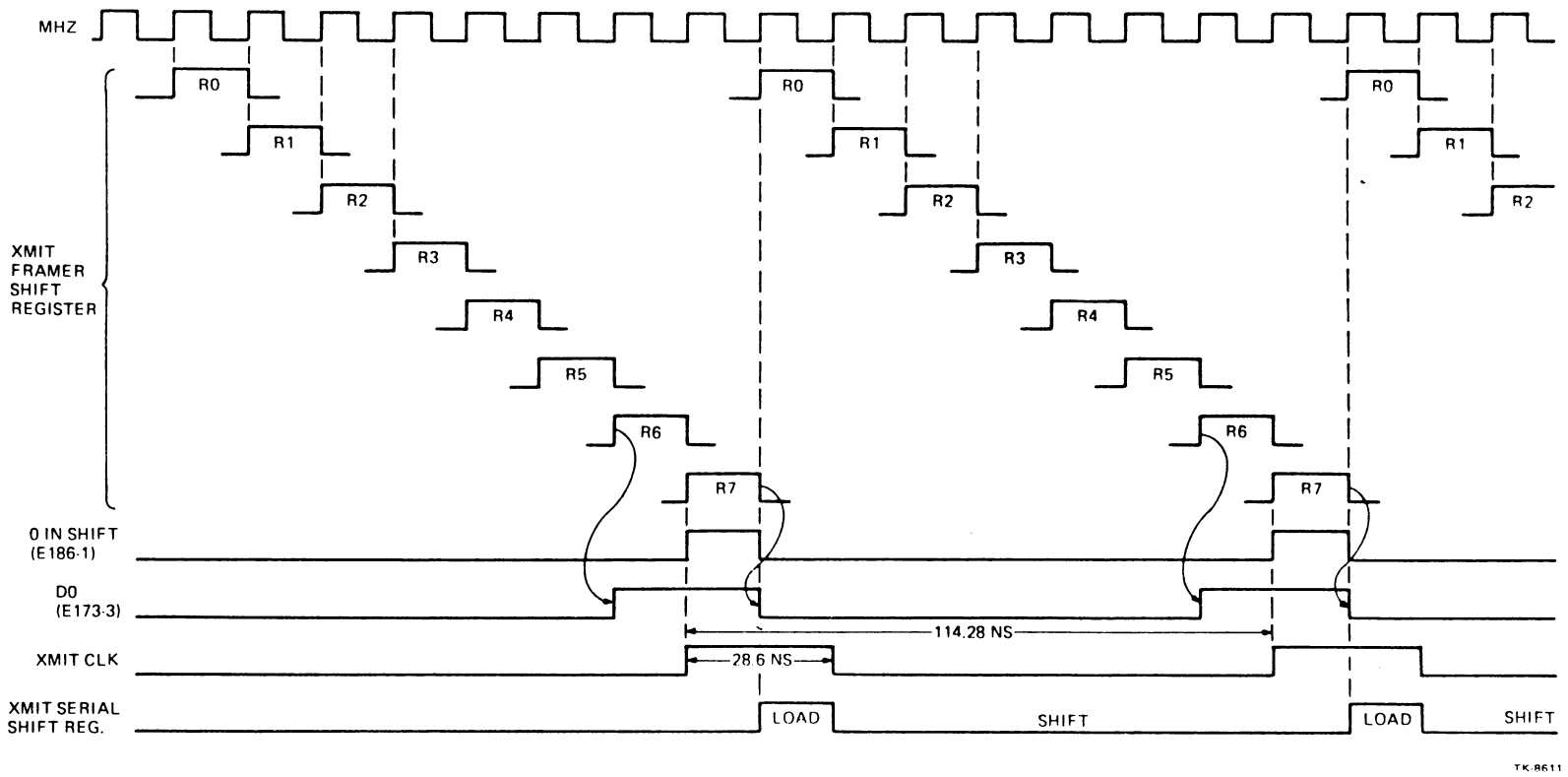


Figure 2-15 XMIT CLK Generator Timing Diagram

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.

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.

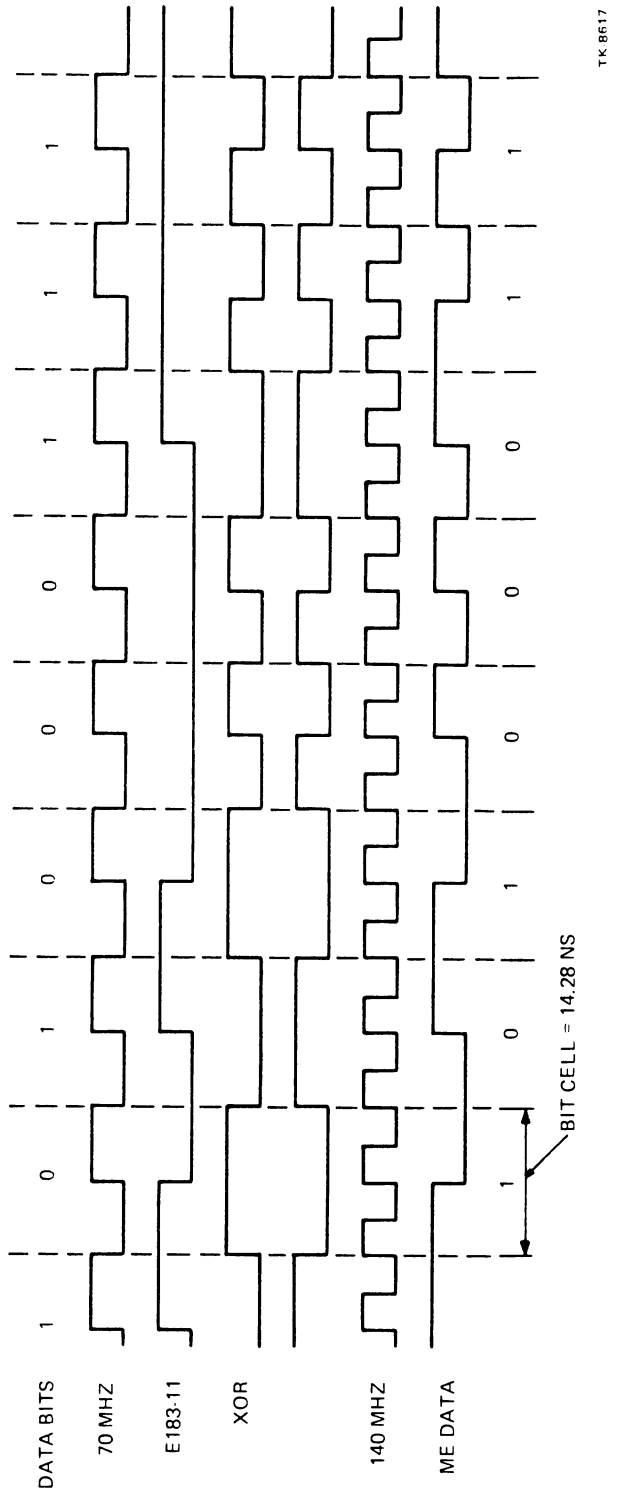


Figure 2-16 Manchester Encoder Timing Diagram

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 (P 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.

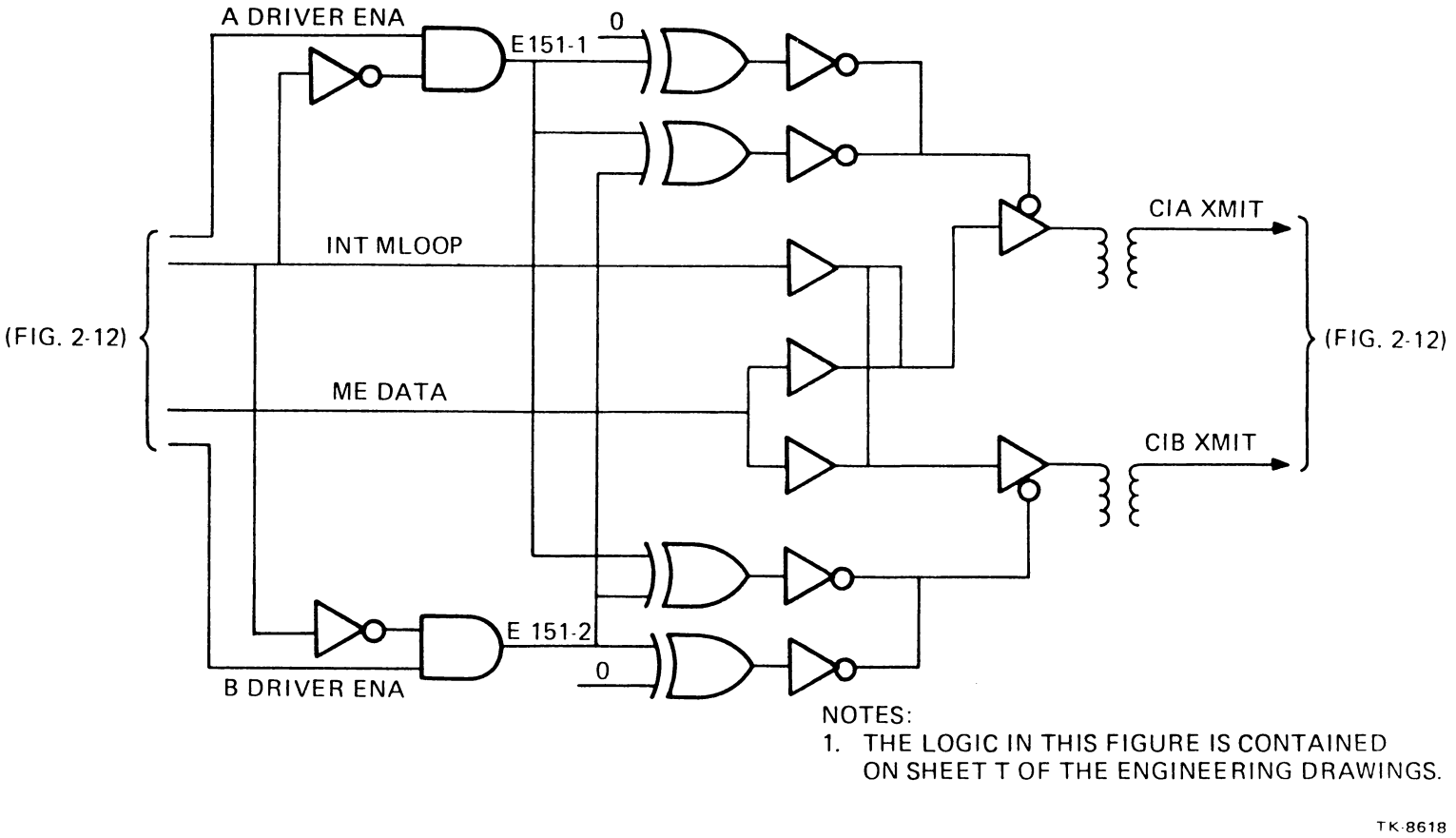


Figure 2-17 XMIT ECL Drivers

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 ensures 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.

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

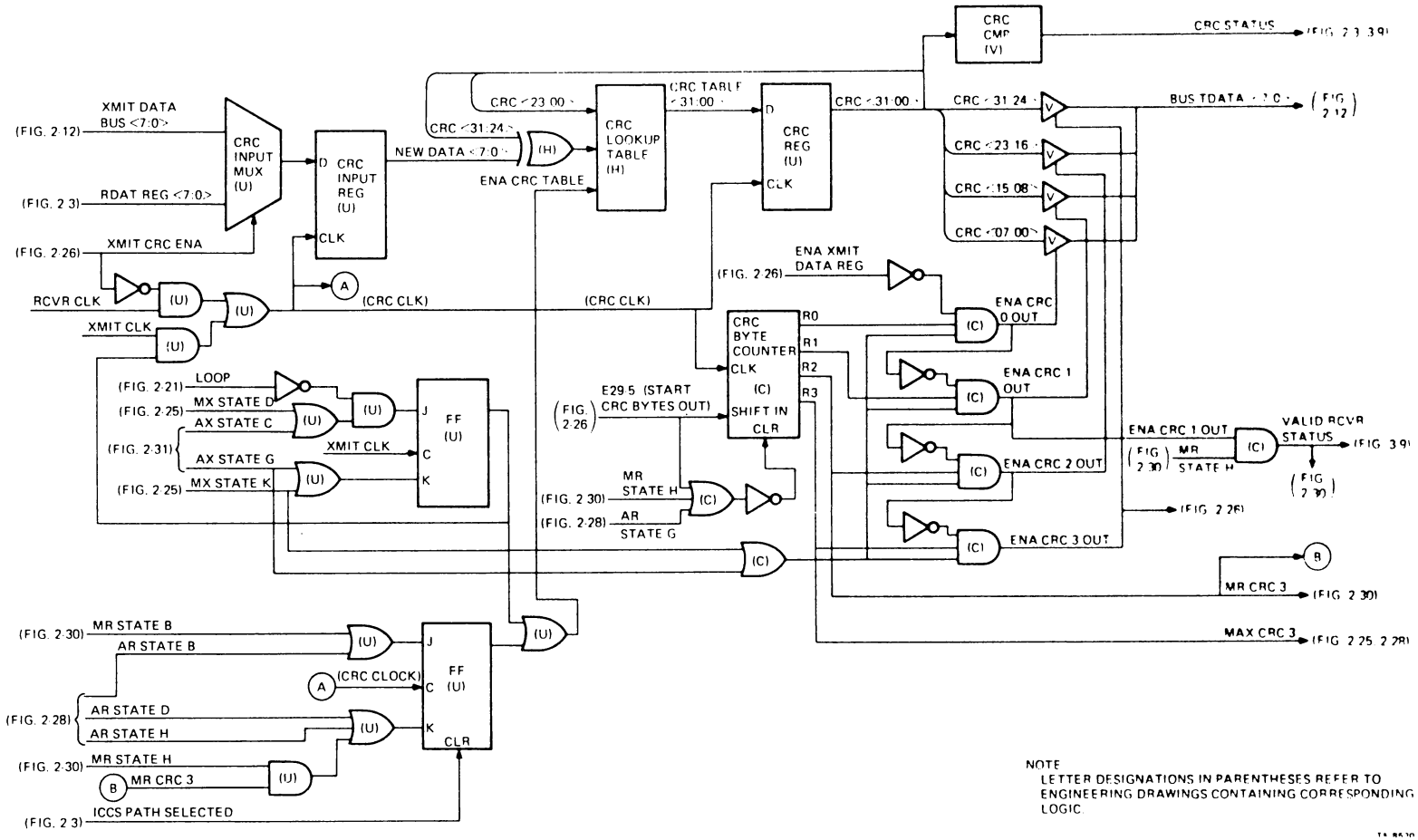


Figure 2-18 CRC Generator/Checker

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.

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.

* 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.

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.

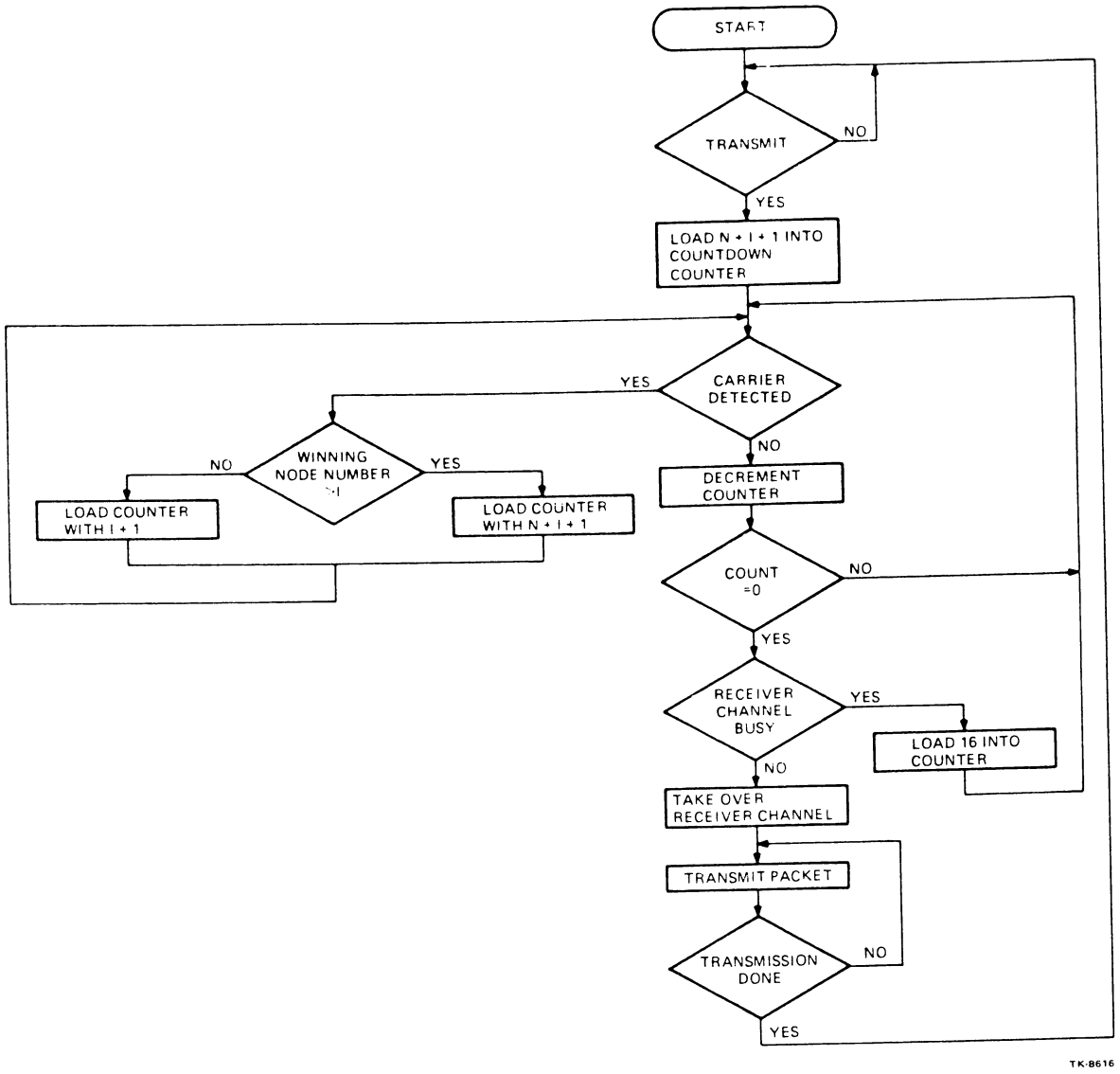


Figure 2-19 Arbitration Flow Diagram

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.

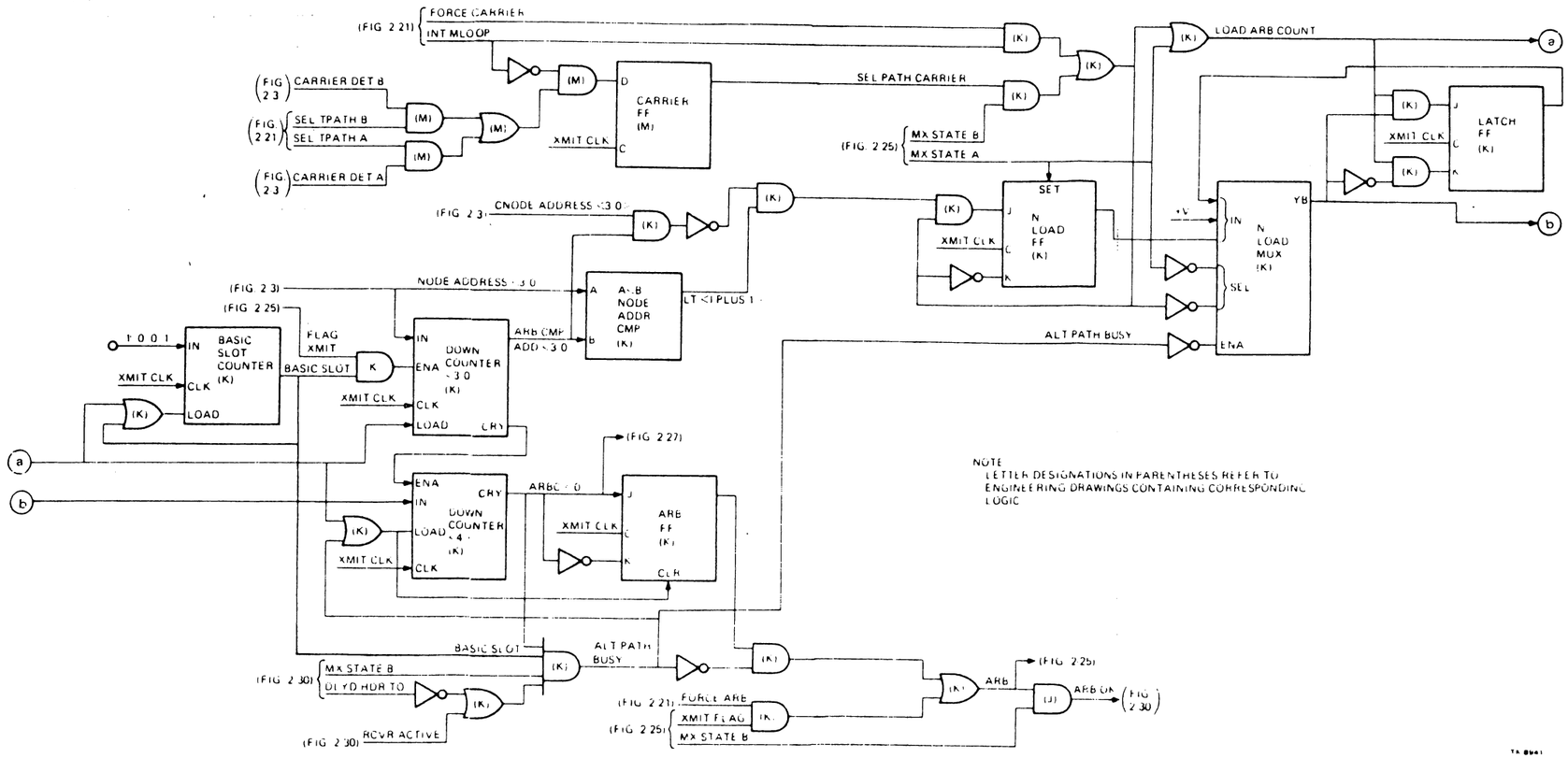
The down counter is in two sections: the lower four bits and the fifth bit. The four-bit section is loaded with the node address (NODE ADDRESS <3:0>). The fifth-bit section is loaded from an N load mux that supplies the N term in the arbitration countdown expression. The mux select inputs are shown in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

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 x 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.



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

Figure 2-20 Arbitration Block Diagram

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

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.

* LT = less than

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.

The true state of RCVR ACTIVE inhibits a successful arbitration by asserting 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.

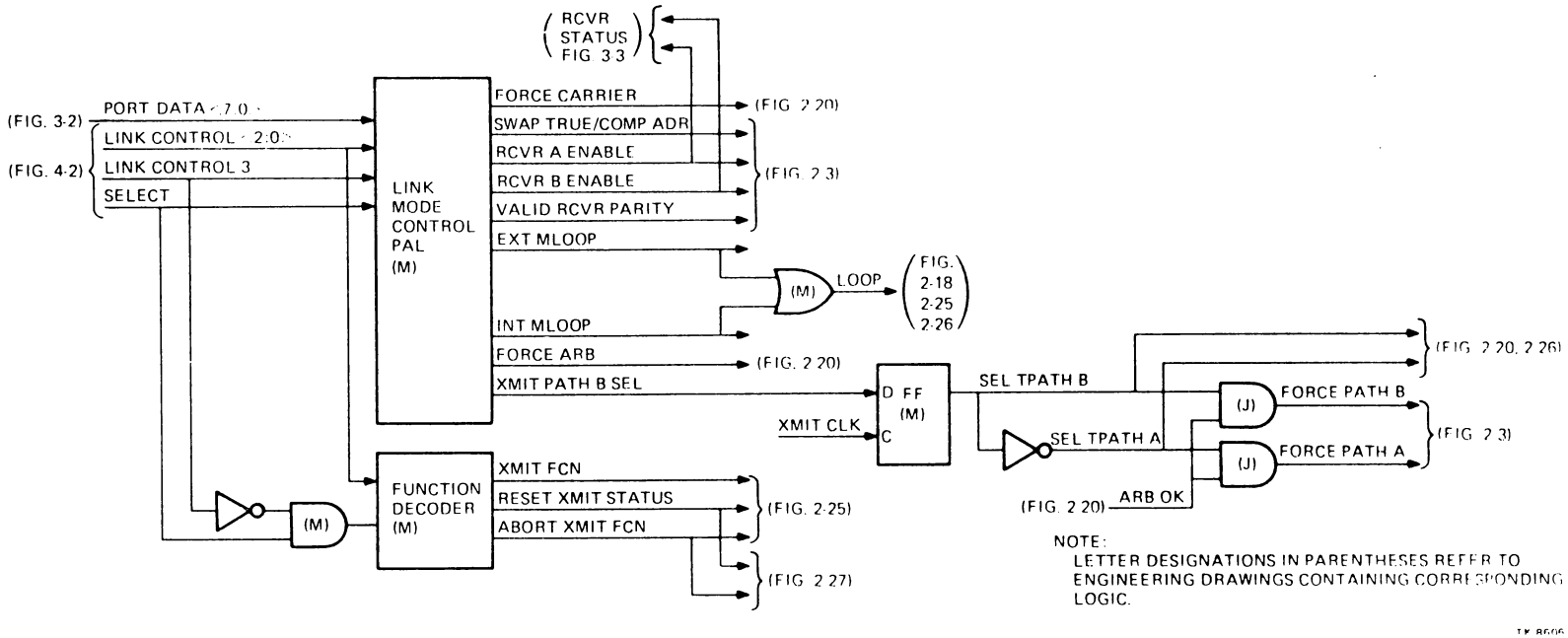


Figure 2-21 Link Functions

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:Ø>), 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:Ø> 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 2.1Ø.

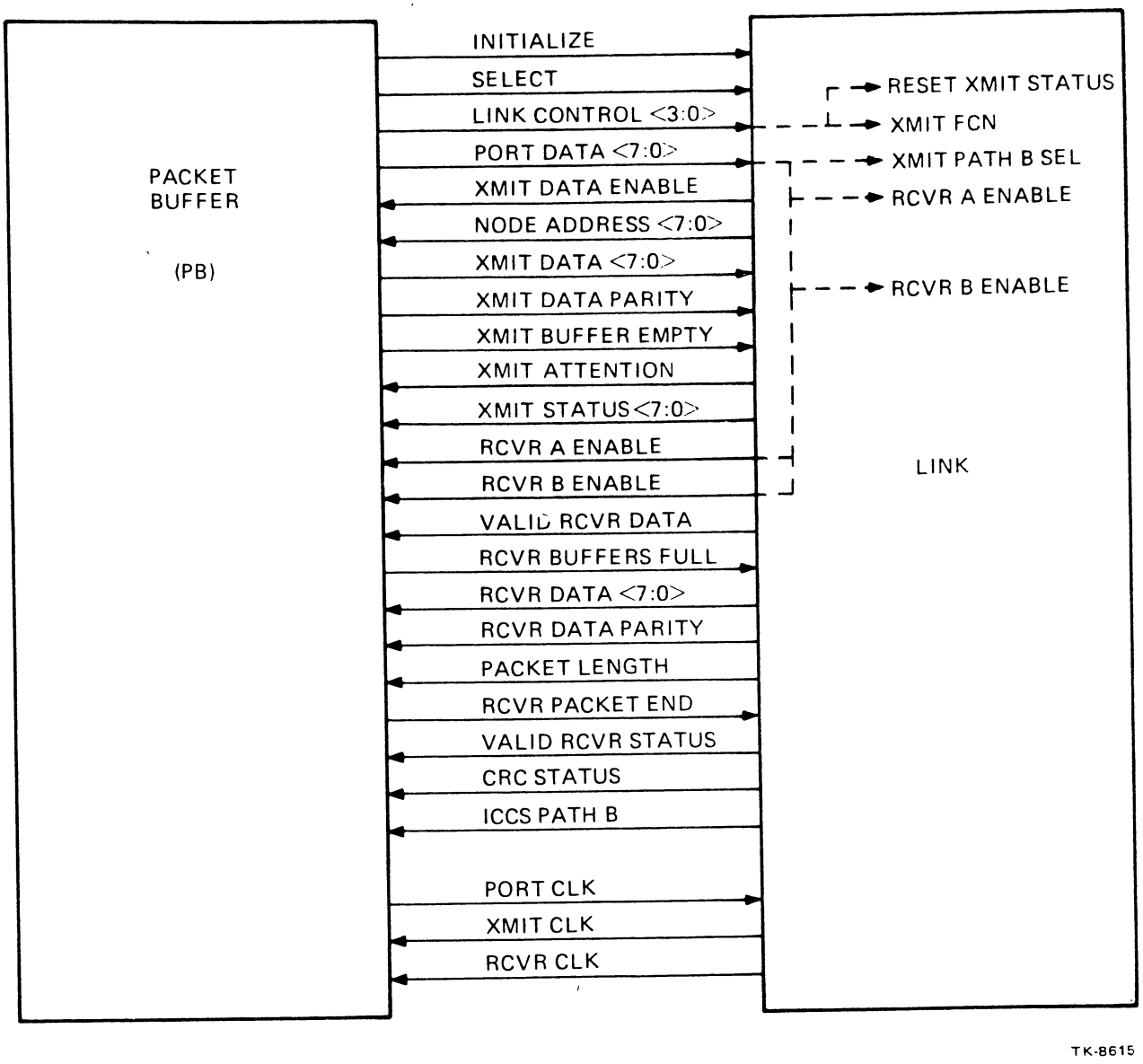


Figure 2-22 Link Interface Signals

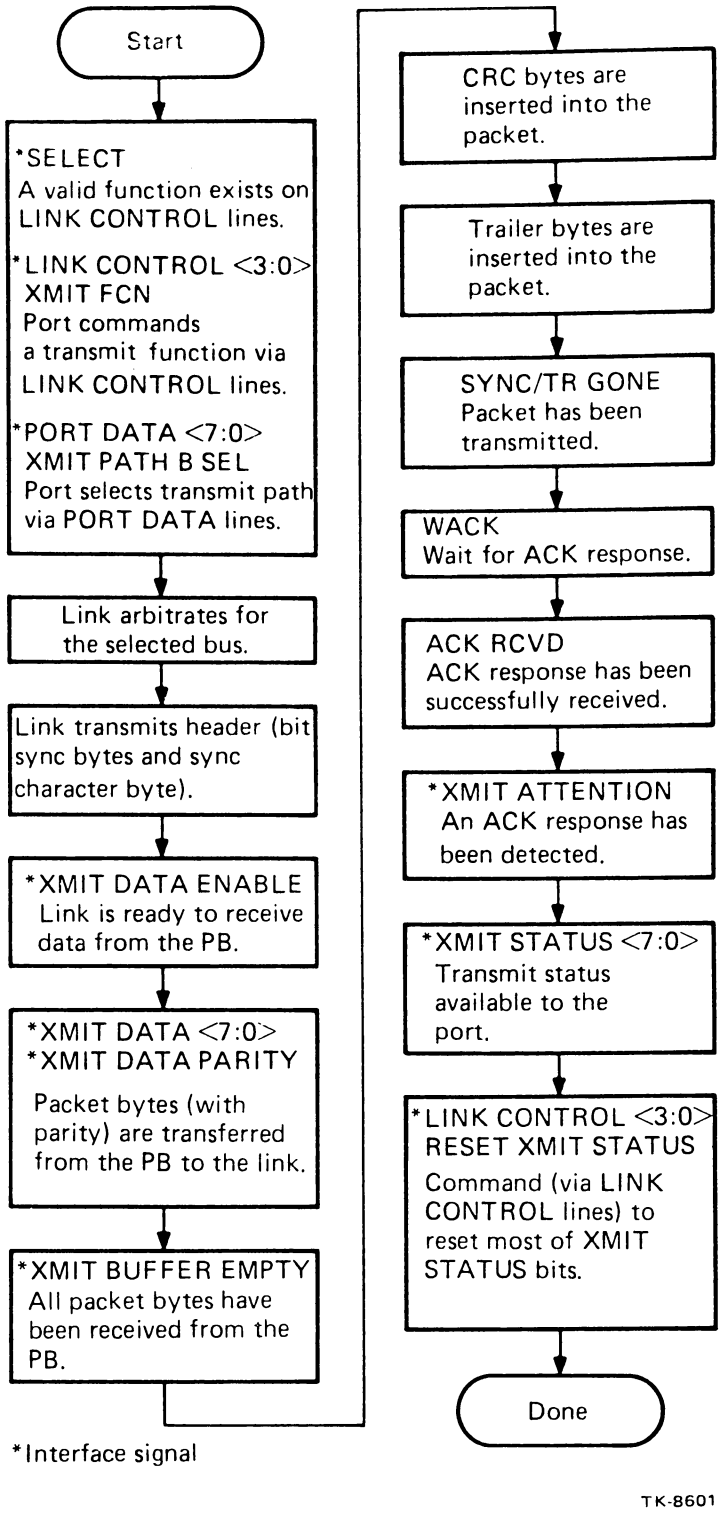


Figure 2-23 Interface Flow Diagram – Transmit Operation

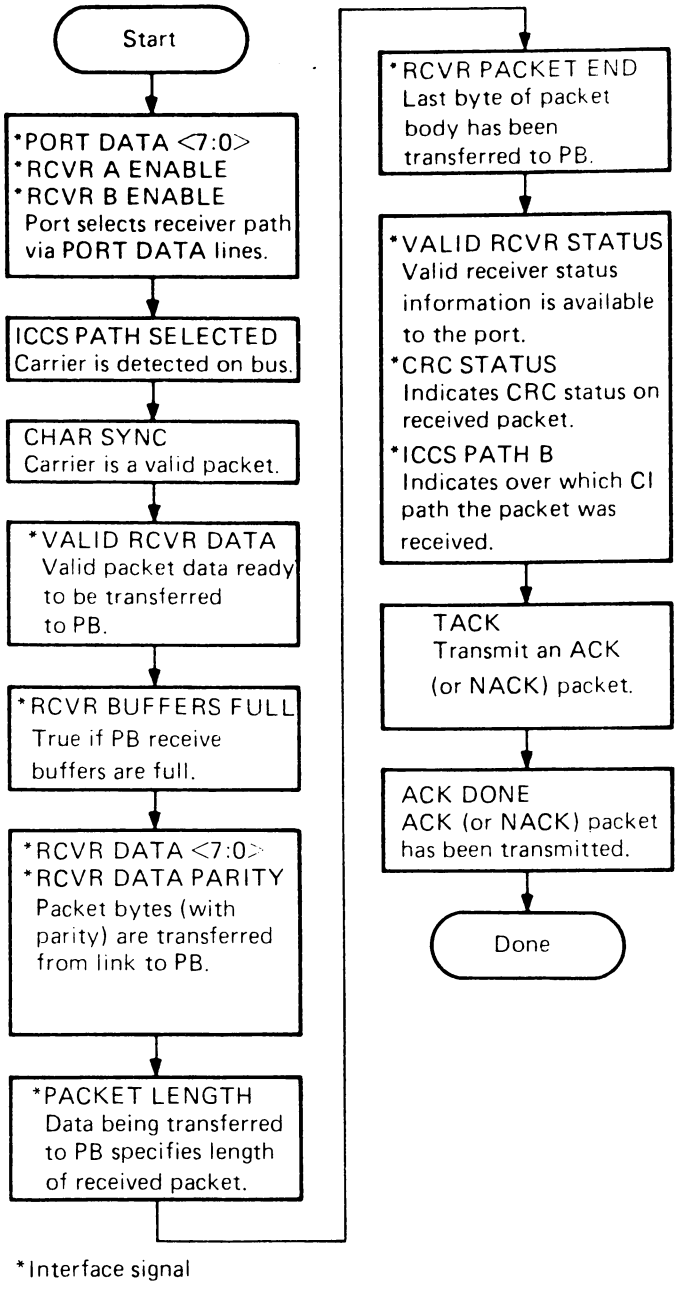


Figure 2-24 Interface Flow Diagram – Receive Operation

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 so long as there is no parity error. If a parity error occurs, PE asserts and transfers the link to state F.

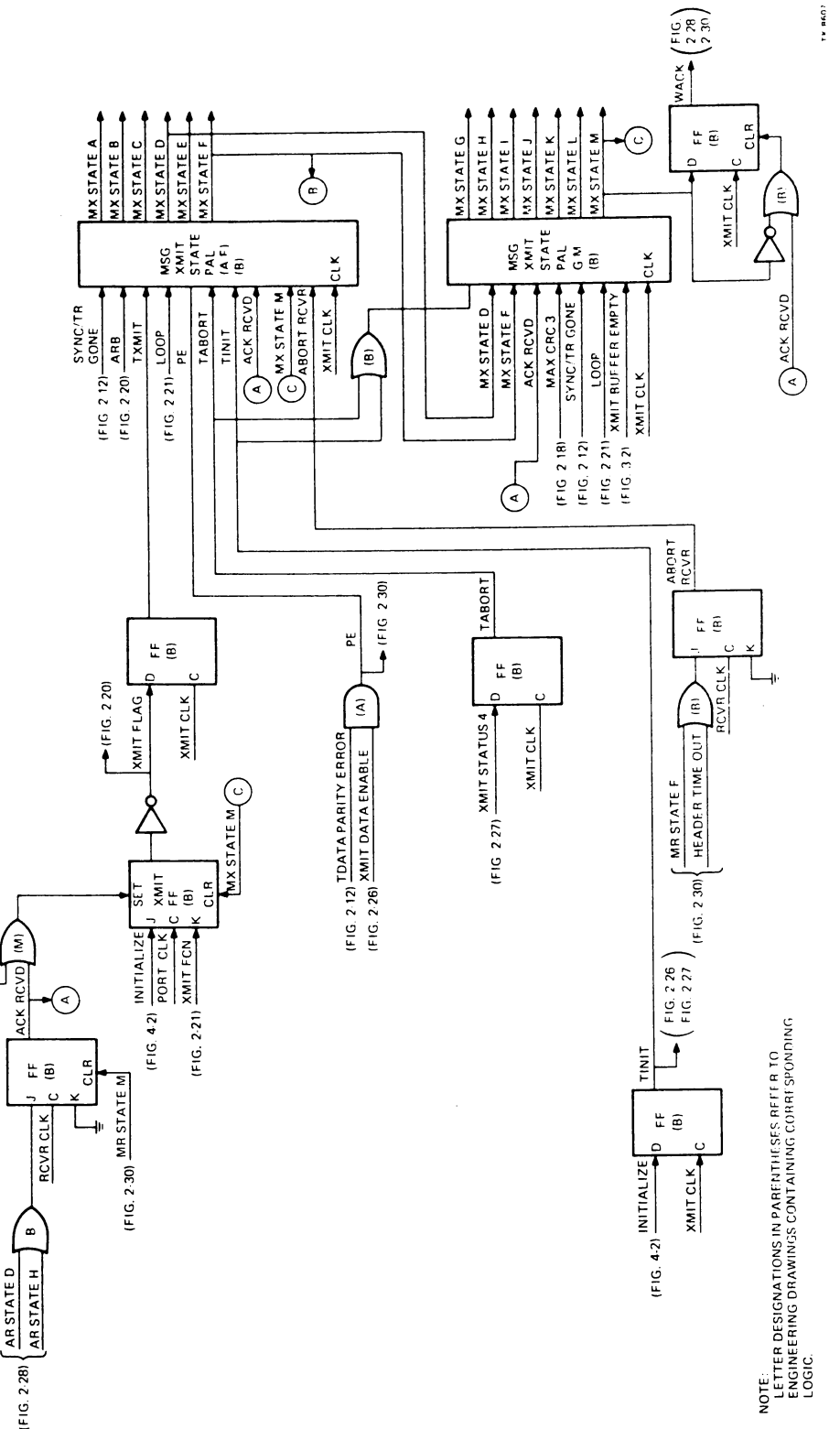


Figure 2-25 Message Transmit State Logic

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.

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 Control 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 recived 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 bus. 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.

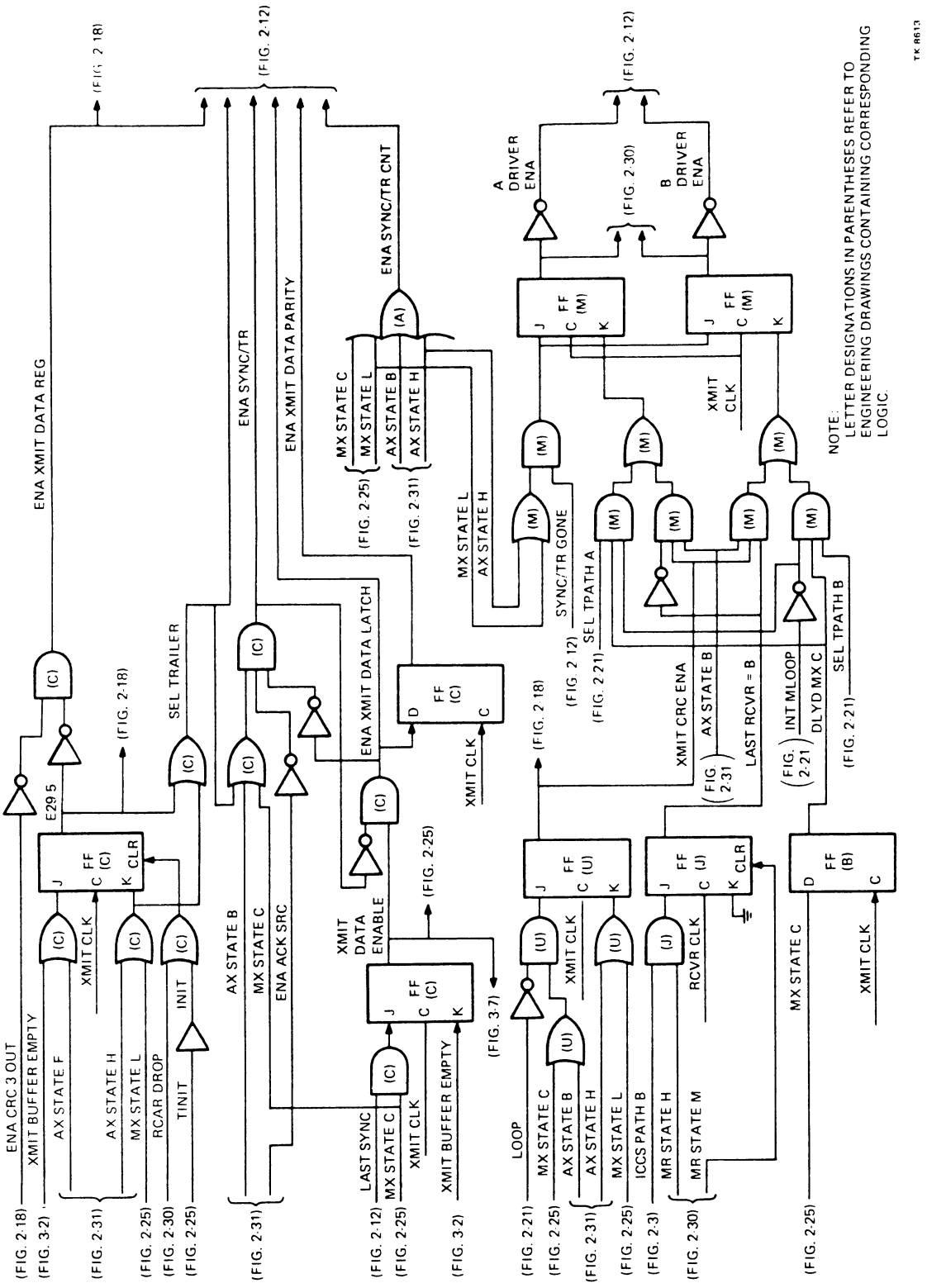


Figure 2-26 Transmit Control Logic

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.

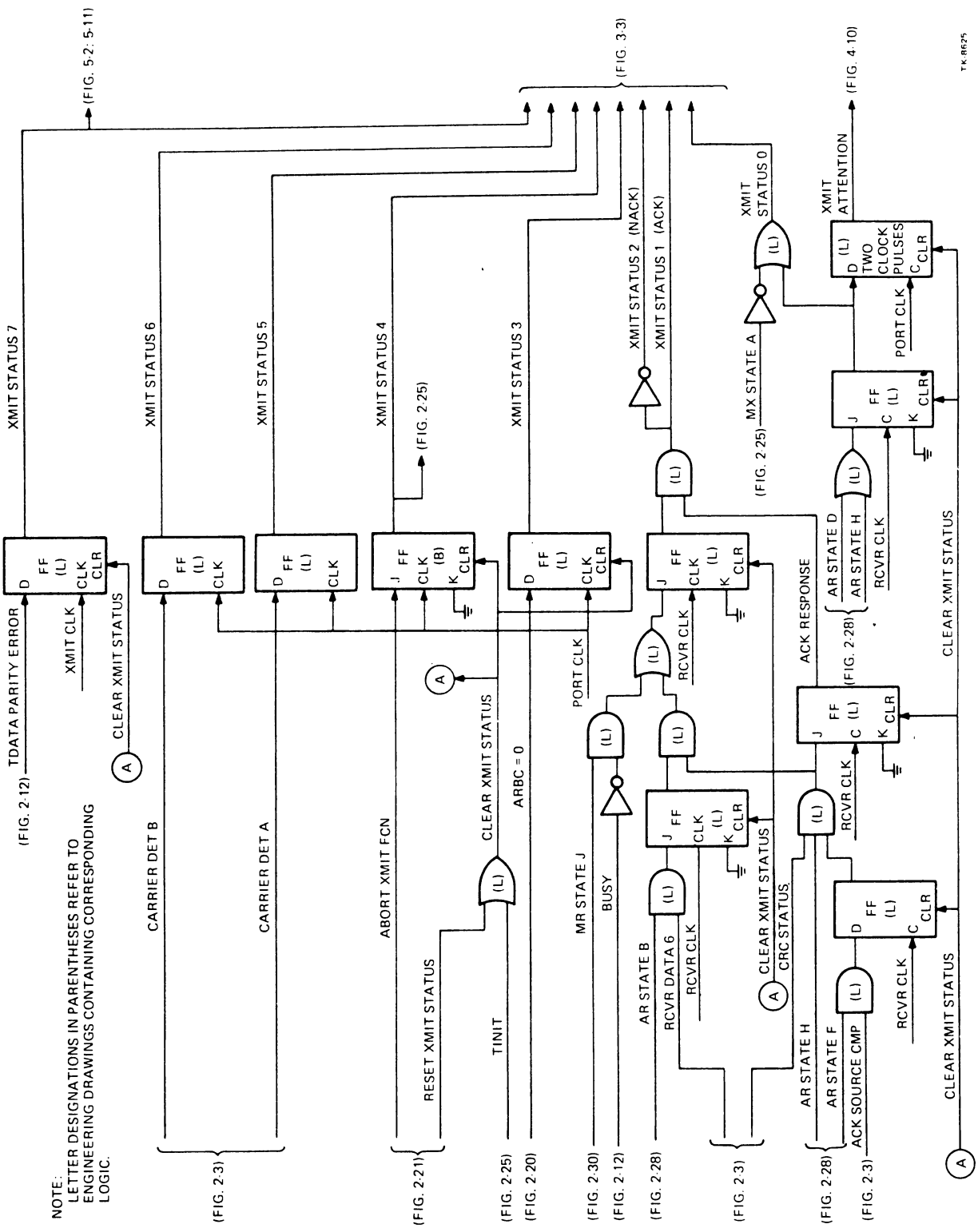


Figure 2-27 Transmit Status

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 microseconds, 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.

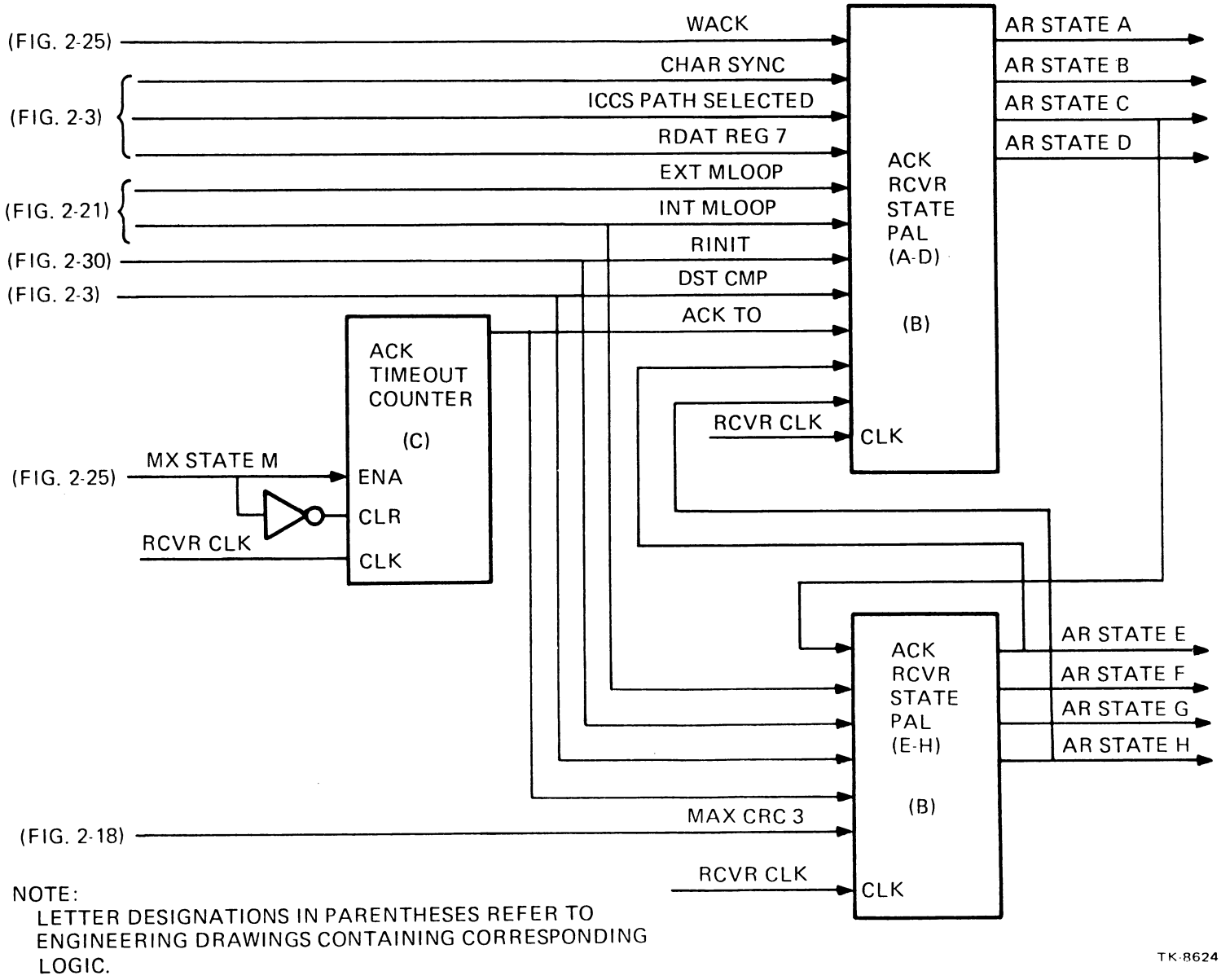


Figure 2-28 ACK Receive State Logic

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.

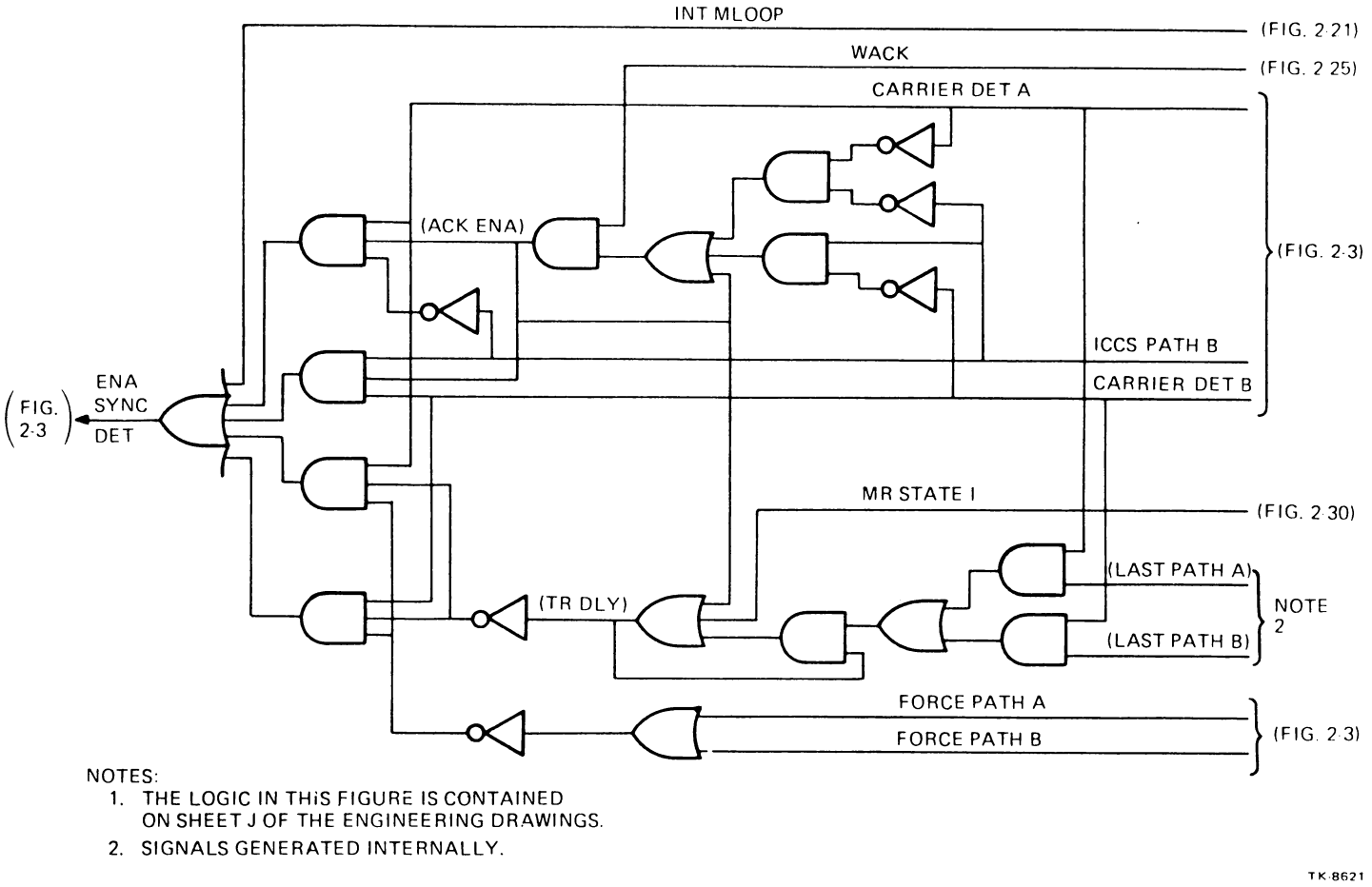


Figure 2-29 Sync Character Detect Enable PAL

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.

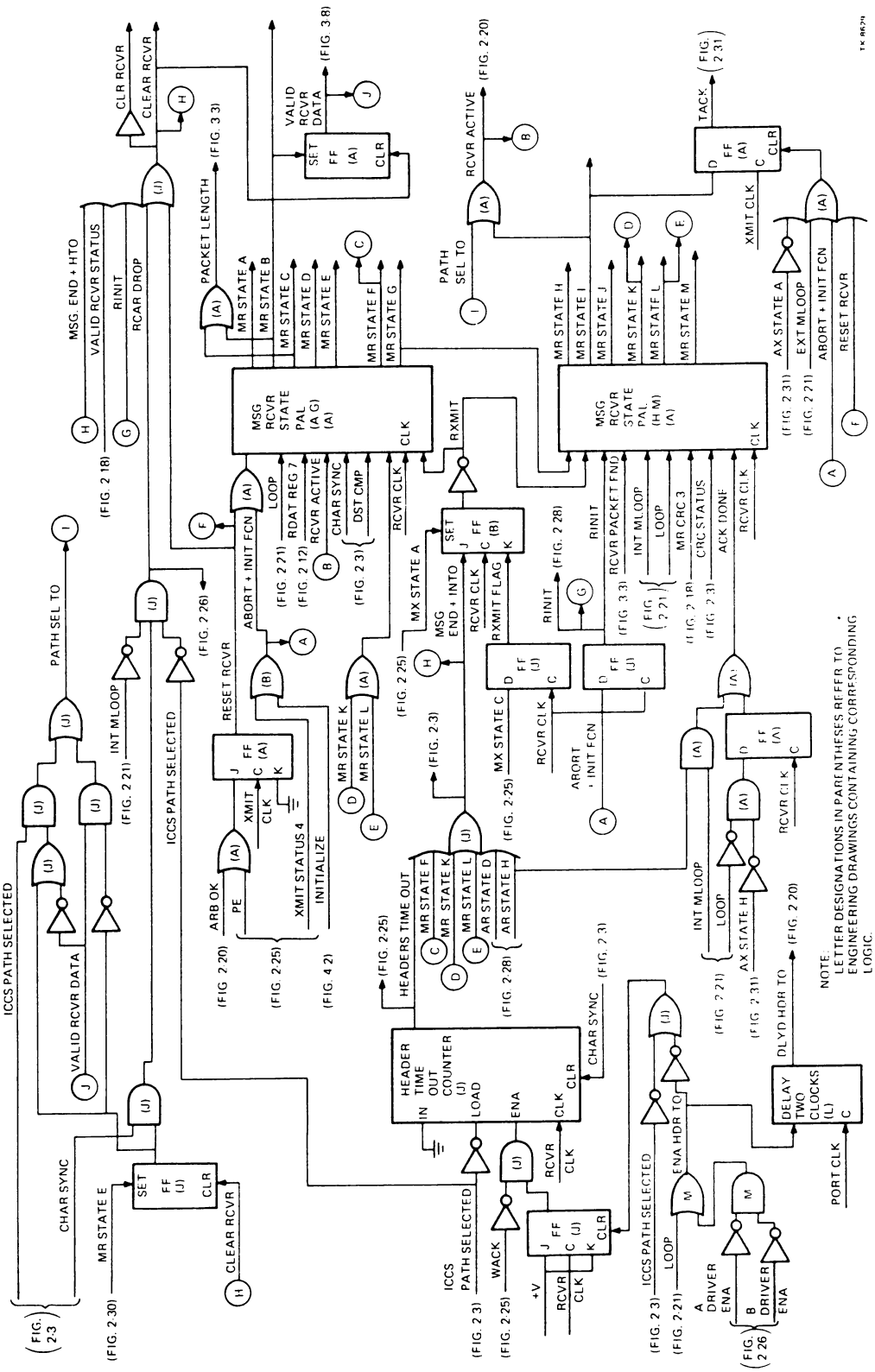


Figure 2-30 Message Receive State Logic

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).

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 microseconds) 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 microsecond timeout period (32 x 114.28 ns = 3.66 microseconds). 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.

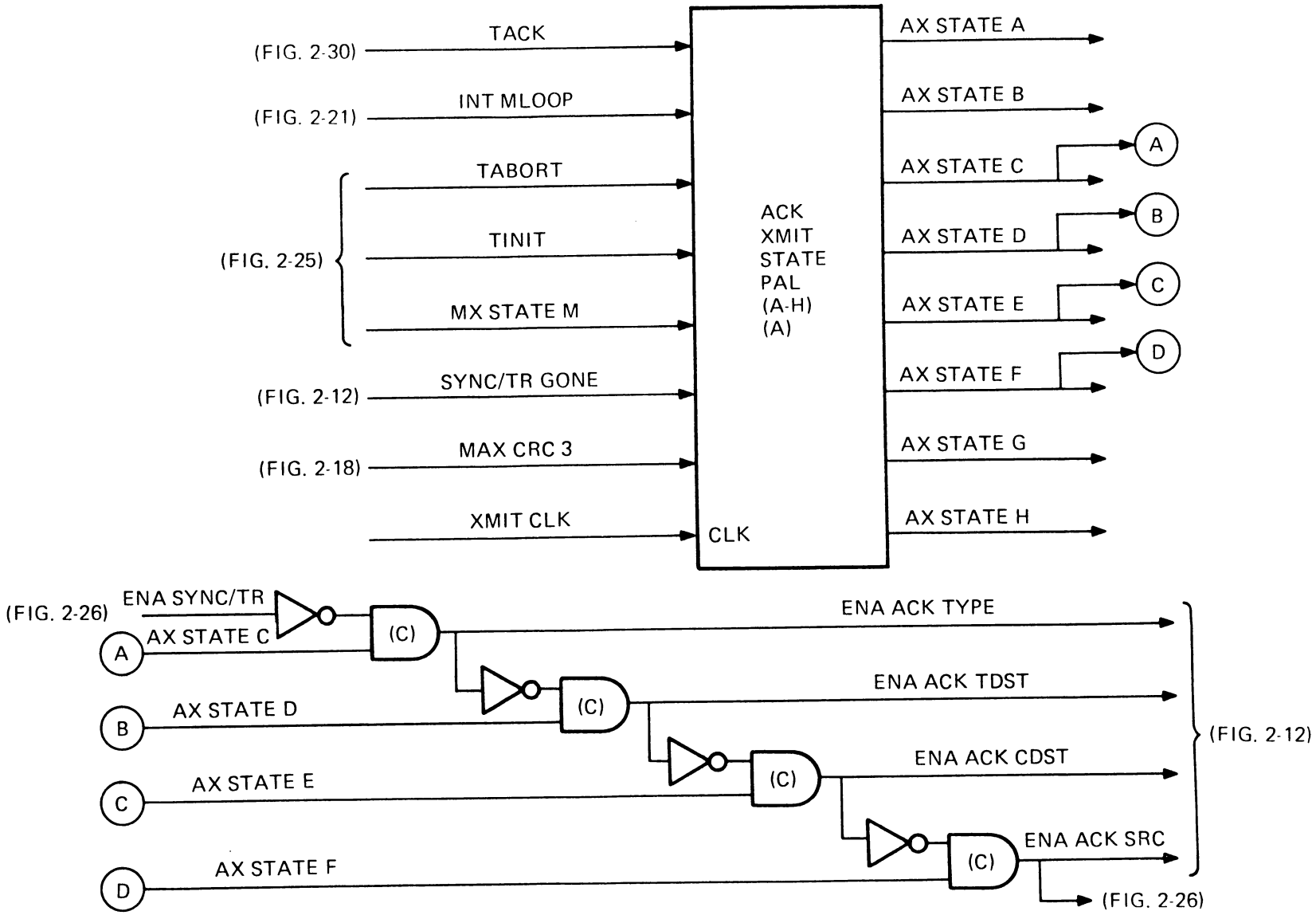
Other signals besides MSG END + HTO assert CLEAR RCVR. One of these is RCAR DROP (receive carrier dropped) which asserts if a carrier is lost during a message reception. ICCS PATH SELECTED asserts before CHAR SYNC asserts and negates after CHAR SYNC negates. If a receive carrier is prematurely lost, 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.



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

TK-8619

Figure 2-31 ACK Transmit State Logic

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

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

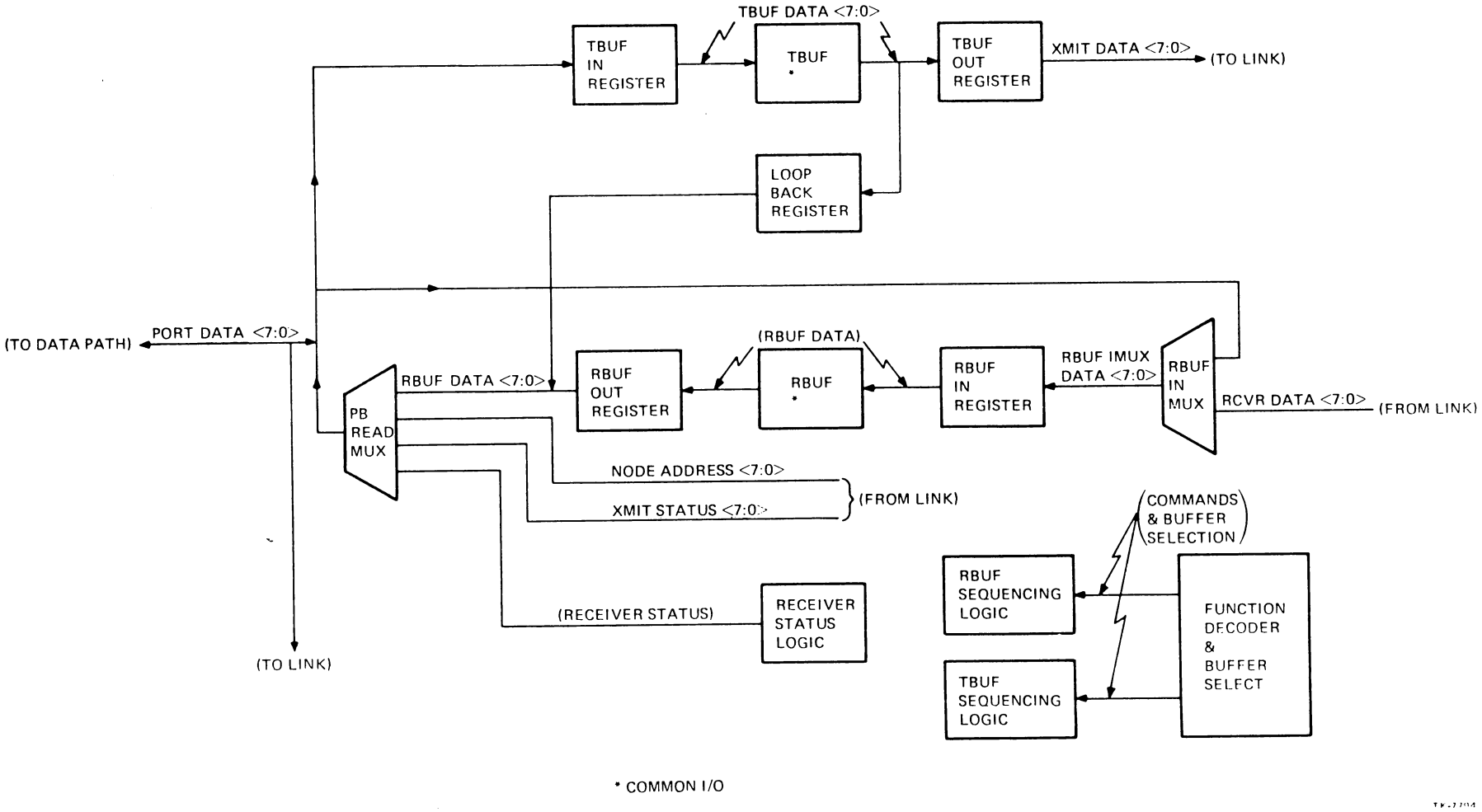


Figure 3-1 Packet Buffer Data Flow

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.

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.

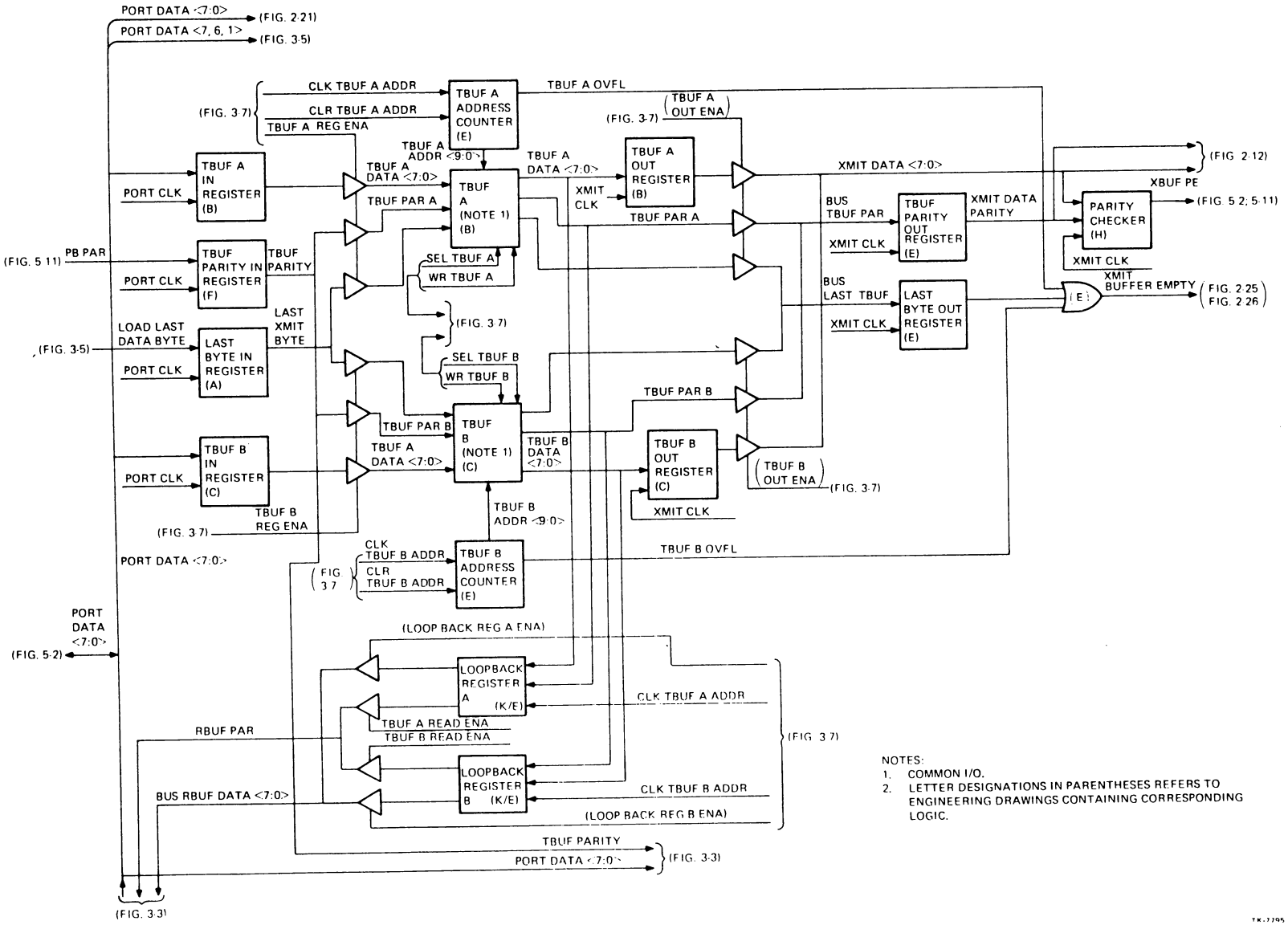


Figure 3-2 TBUF Operations

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.

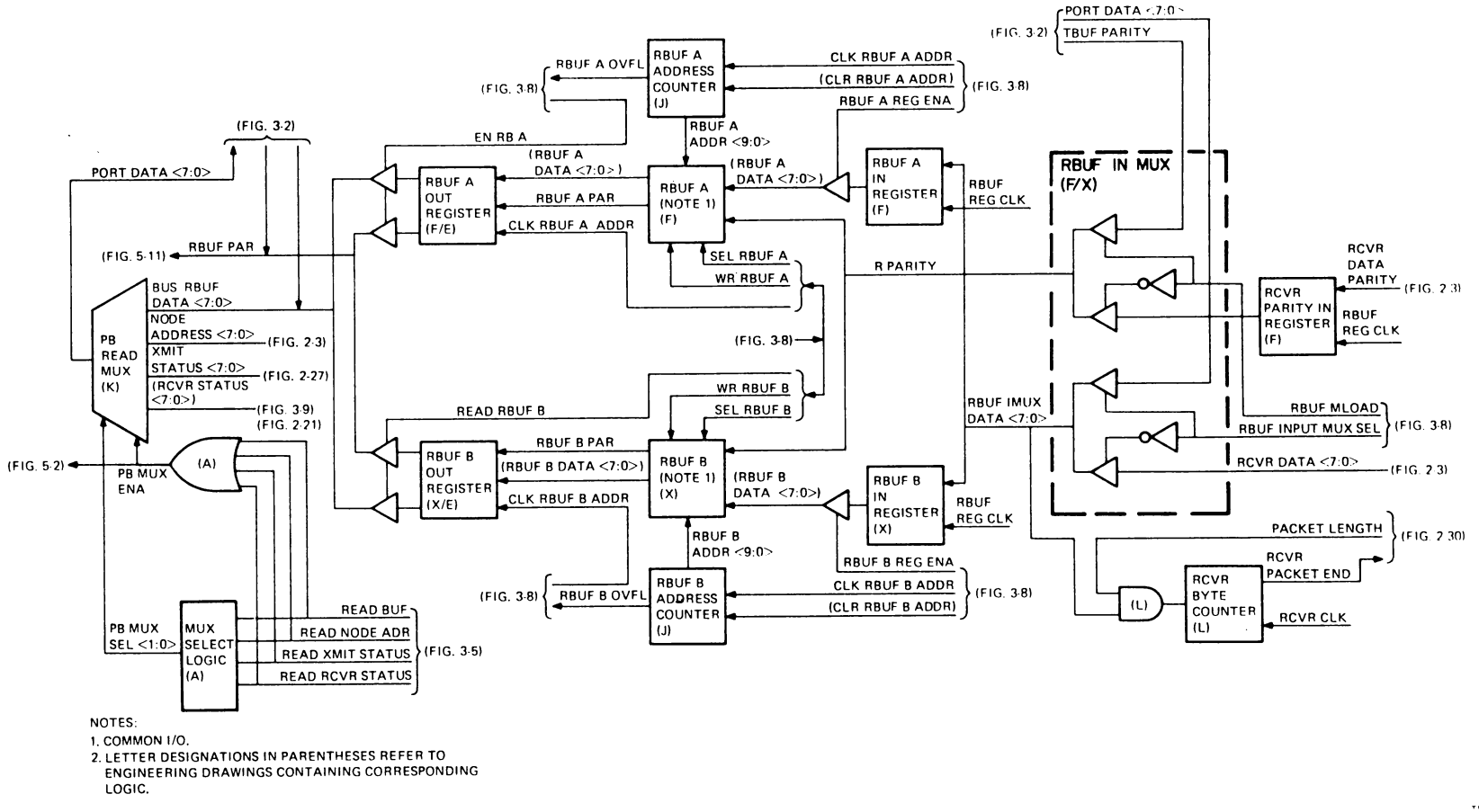
SEL RBUF A enables RBUF A, selecting it for an RBUF A operation. WR RBUF A enables the RBUF A input and disables the output, setting up RBUF A for a write. (RBUF A has a common I/O.)

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.



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

Figure 3-3 RBUF Operations

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.

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 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.

* 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.

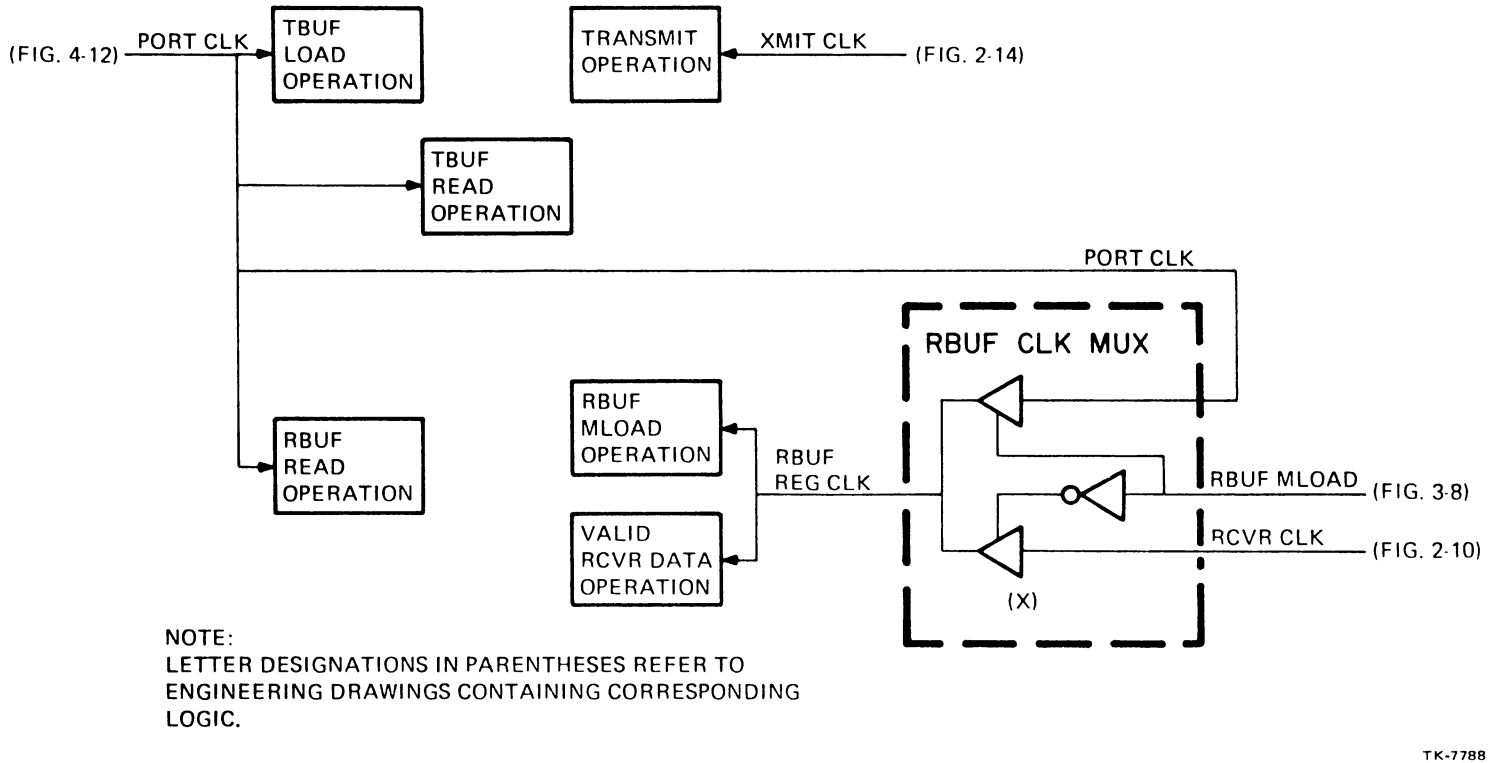


Figure 3-4 Packet Buffer Clocks

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.

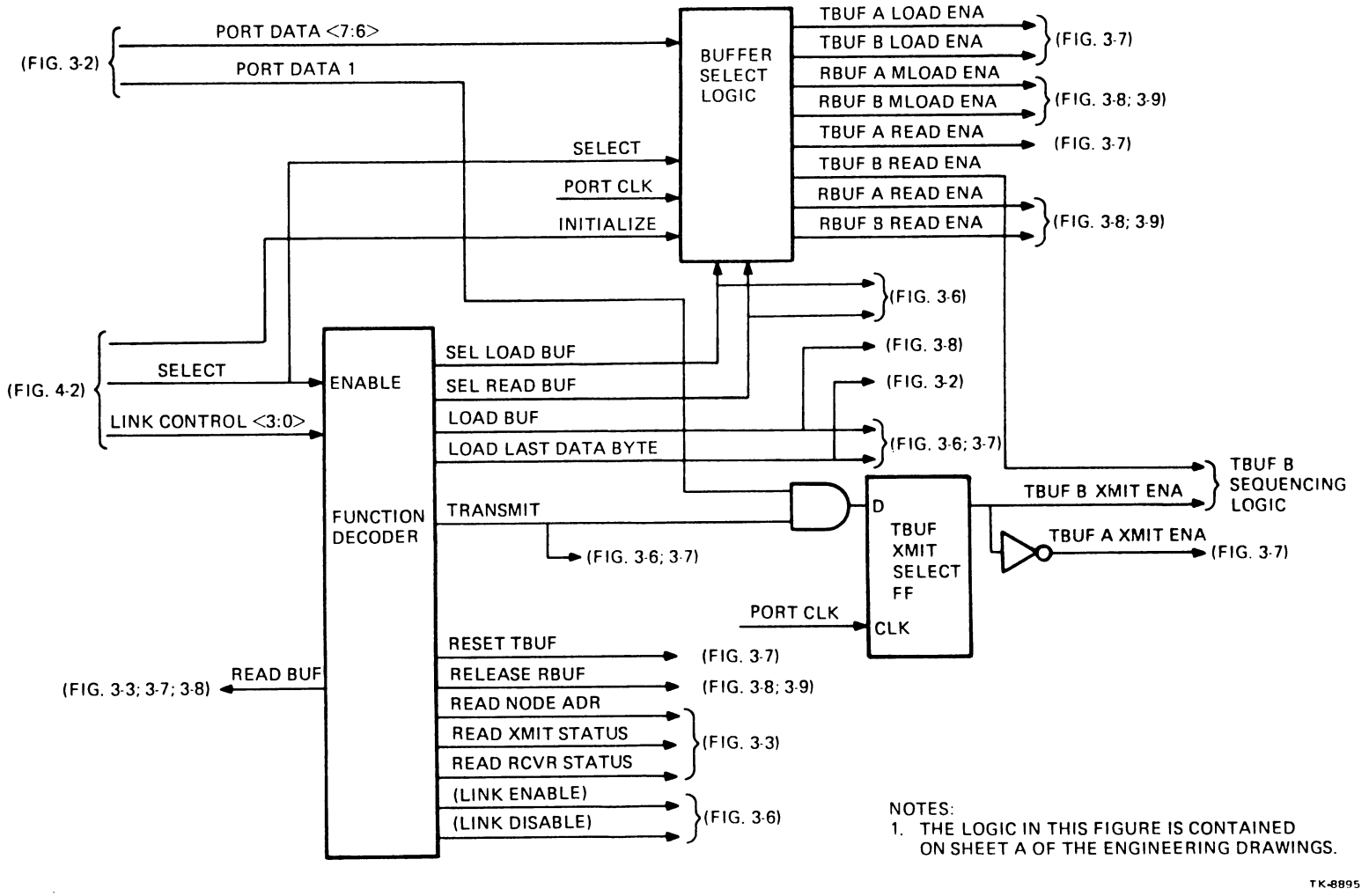


Figure 3-5 Function Decoder And Buffer Select Logic

Table 3-1 Link Control Codes Vs PB Function Commands

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

Table 3-2 Load Buffer Select Code

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

Table 3-3 Read Buffer Select Code

PORT 7	DATA 6	Buffer Selected
Ø	Ø	RBUF A READ ENA
Ø	1	RBUF B READ ENA
1	Ø	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.

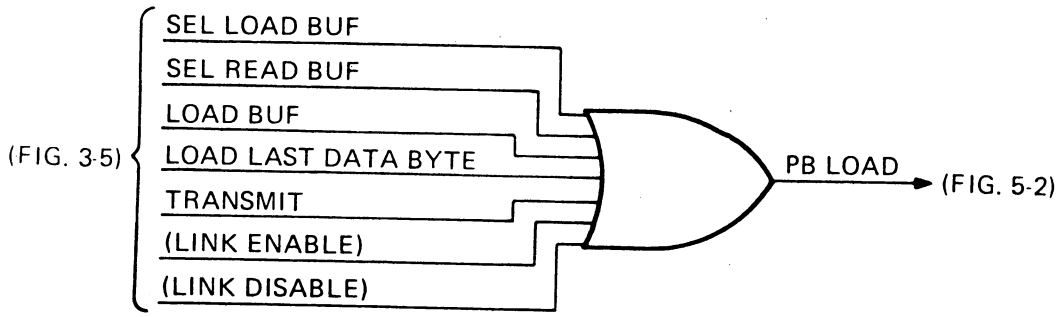
2. LOAD BUF and LOAD LAST DATA BYTE commands obtain the byte to be loaded from the port data bus.

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.

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.



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

TK-7789

Figure 3-6 PB Load Logic

3.7.1 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.

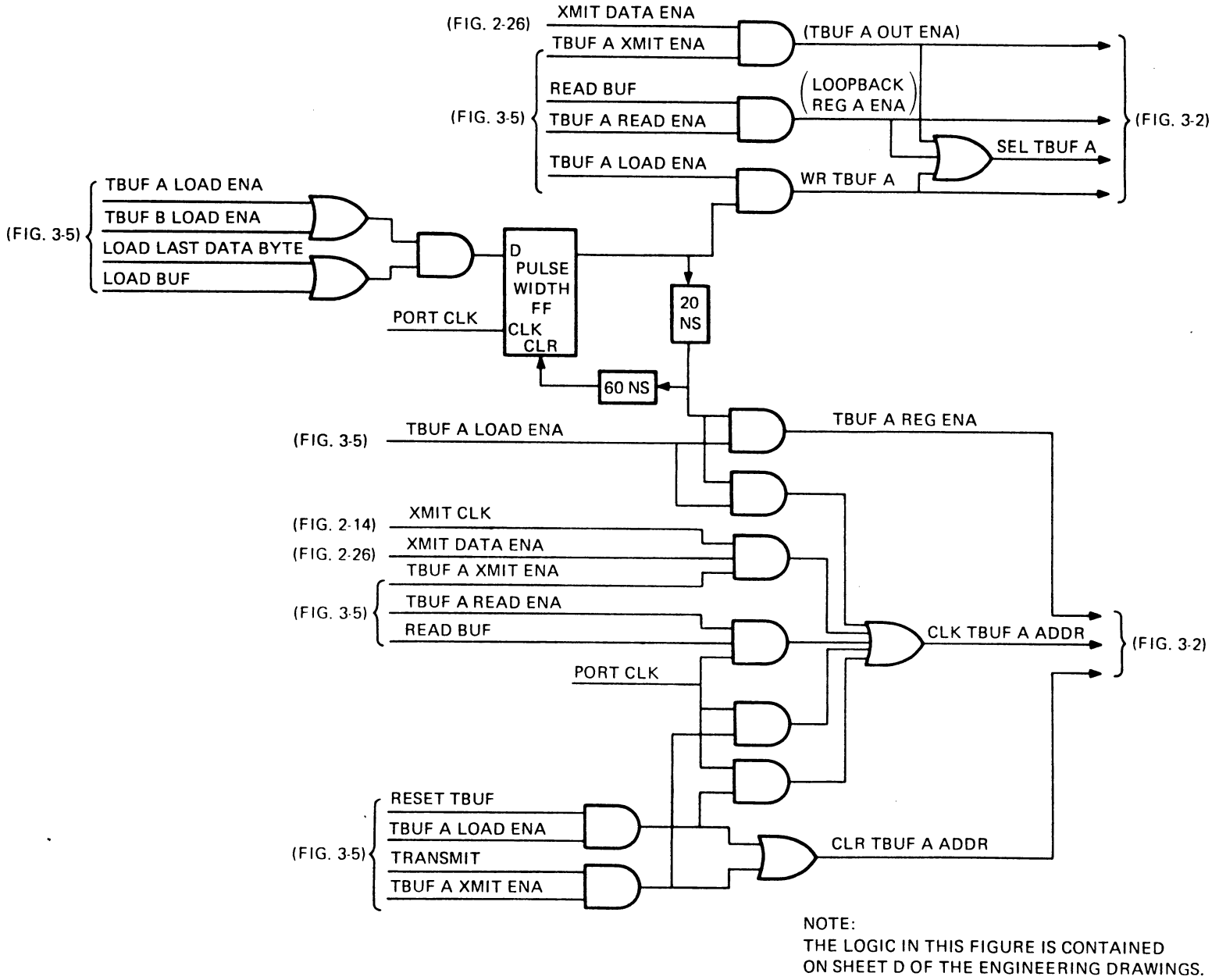


Figure 3-7 TBUF Sequencing Logic

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.

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.

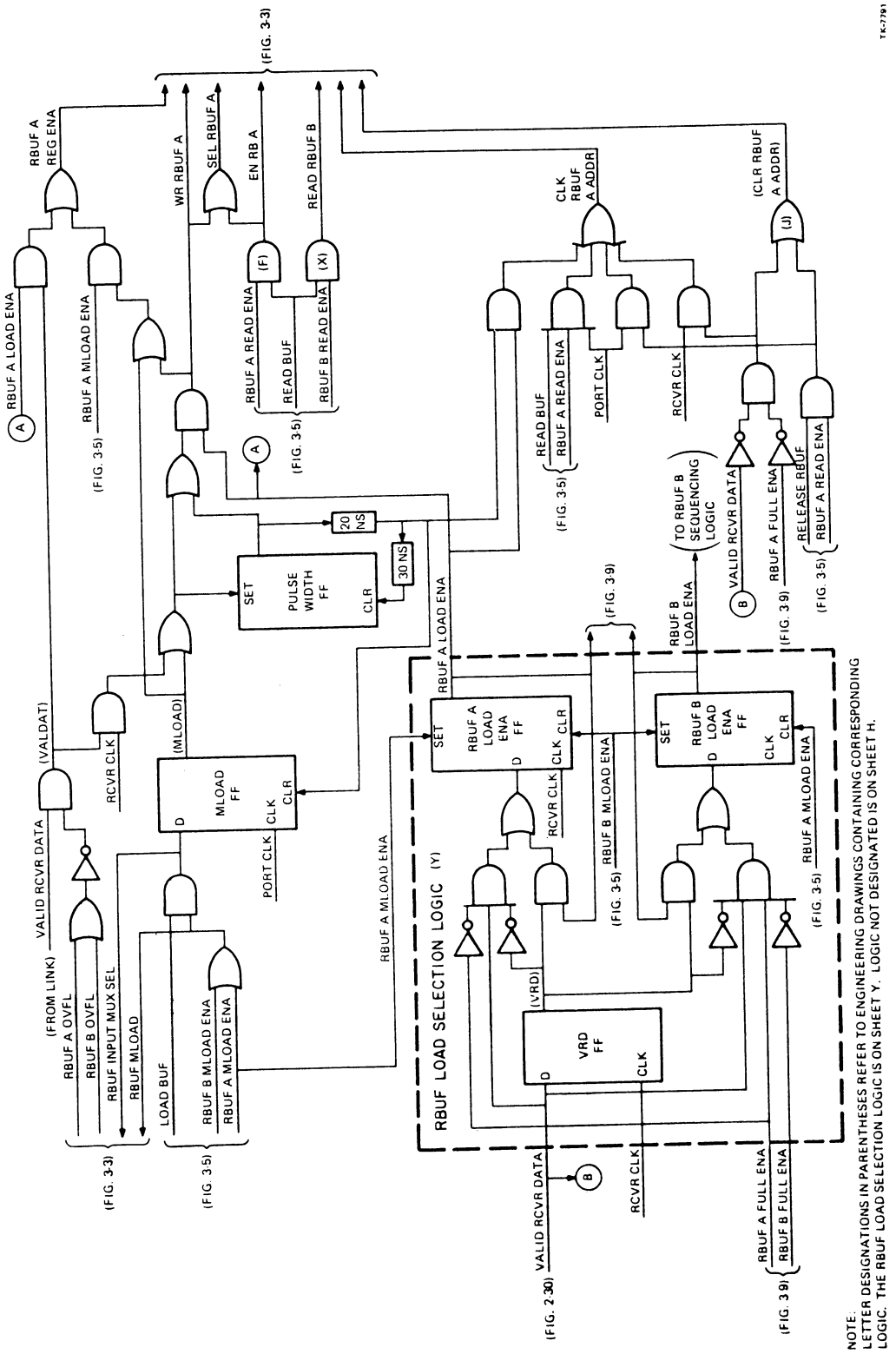


Figure 3-8 RBUF Sequencing Logic

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.

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.

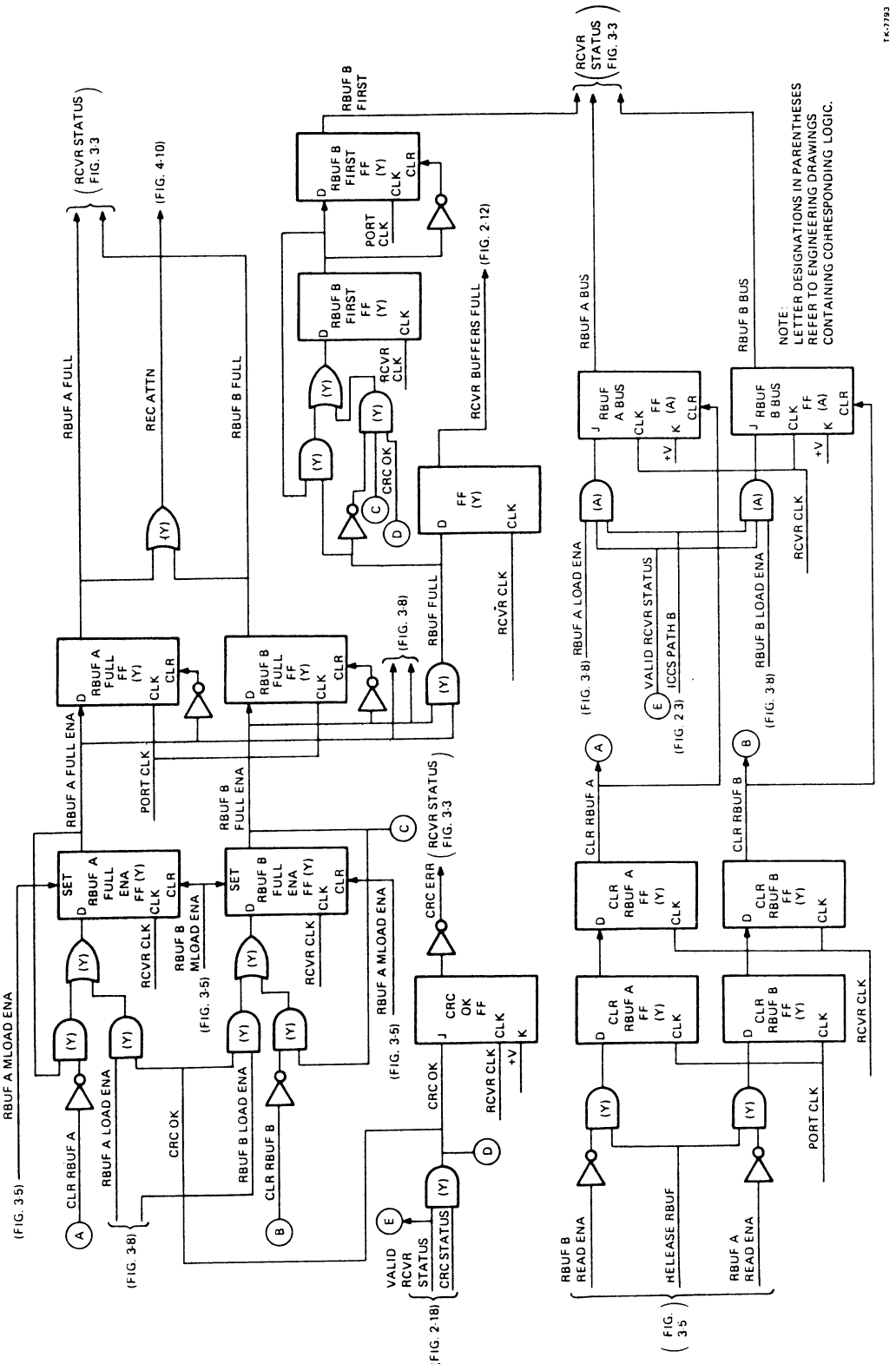


Figure 3-9 RCVR Status Logic

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.

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.

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.

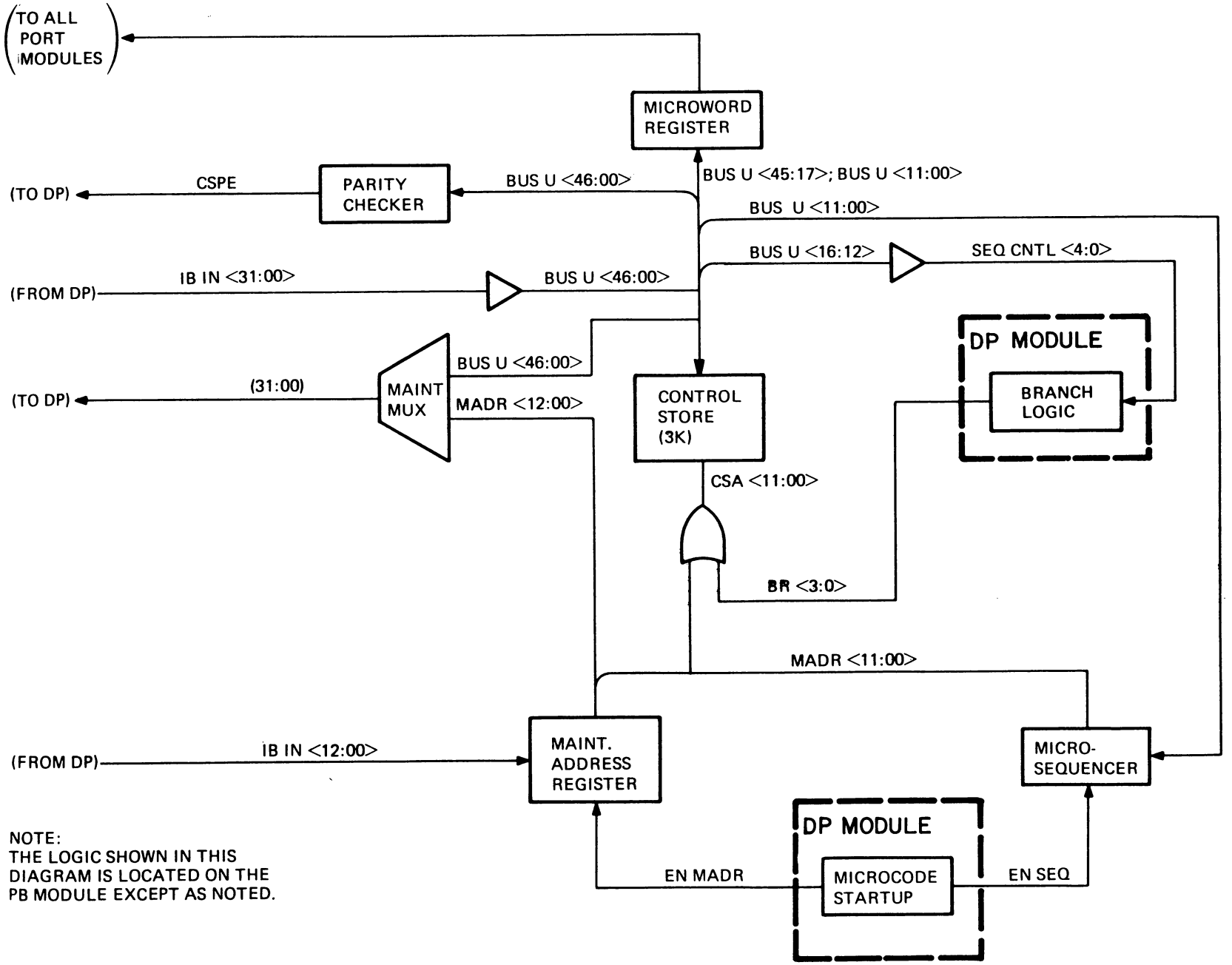


Figure 4-1 Control Store Simplified Block Diagram

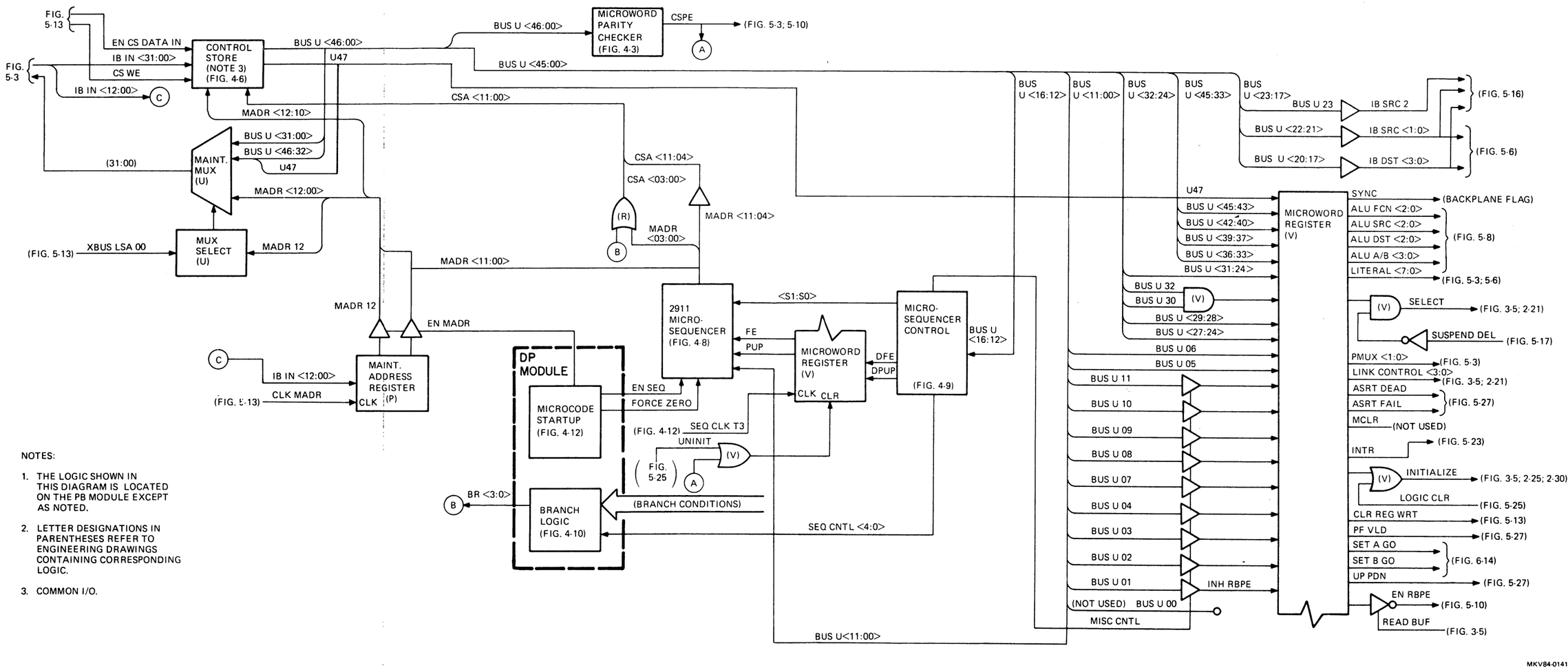
Most of the microword read from the CS is latched into the microword register. The register outputs control signals to all of the port modules.

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 as an input to the DP (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.



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 Plock Diagram

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.

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.

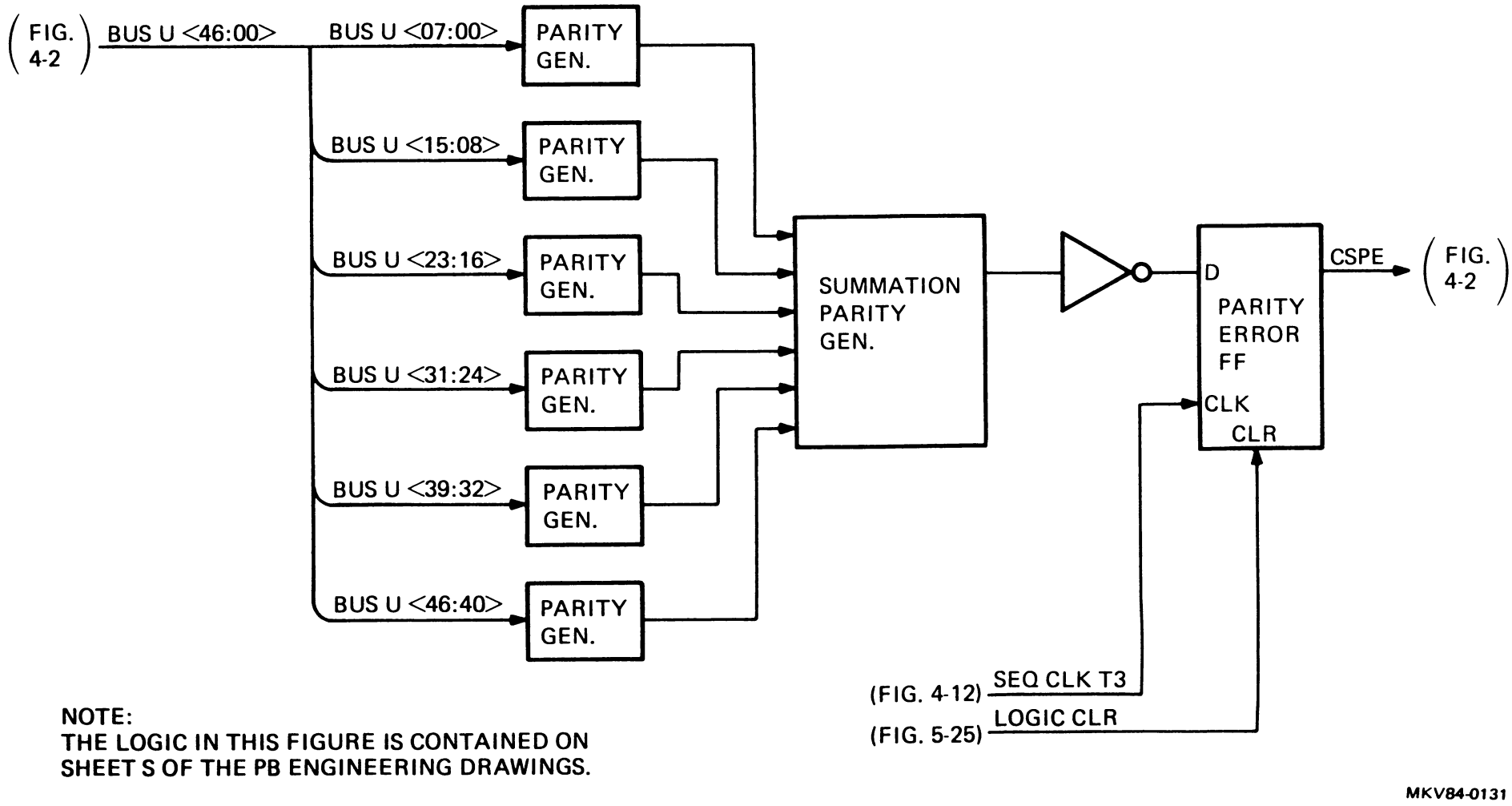


Figure 4-3 Microword Parity Check

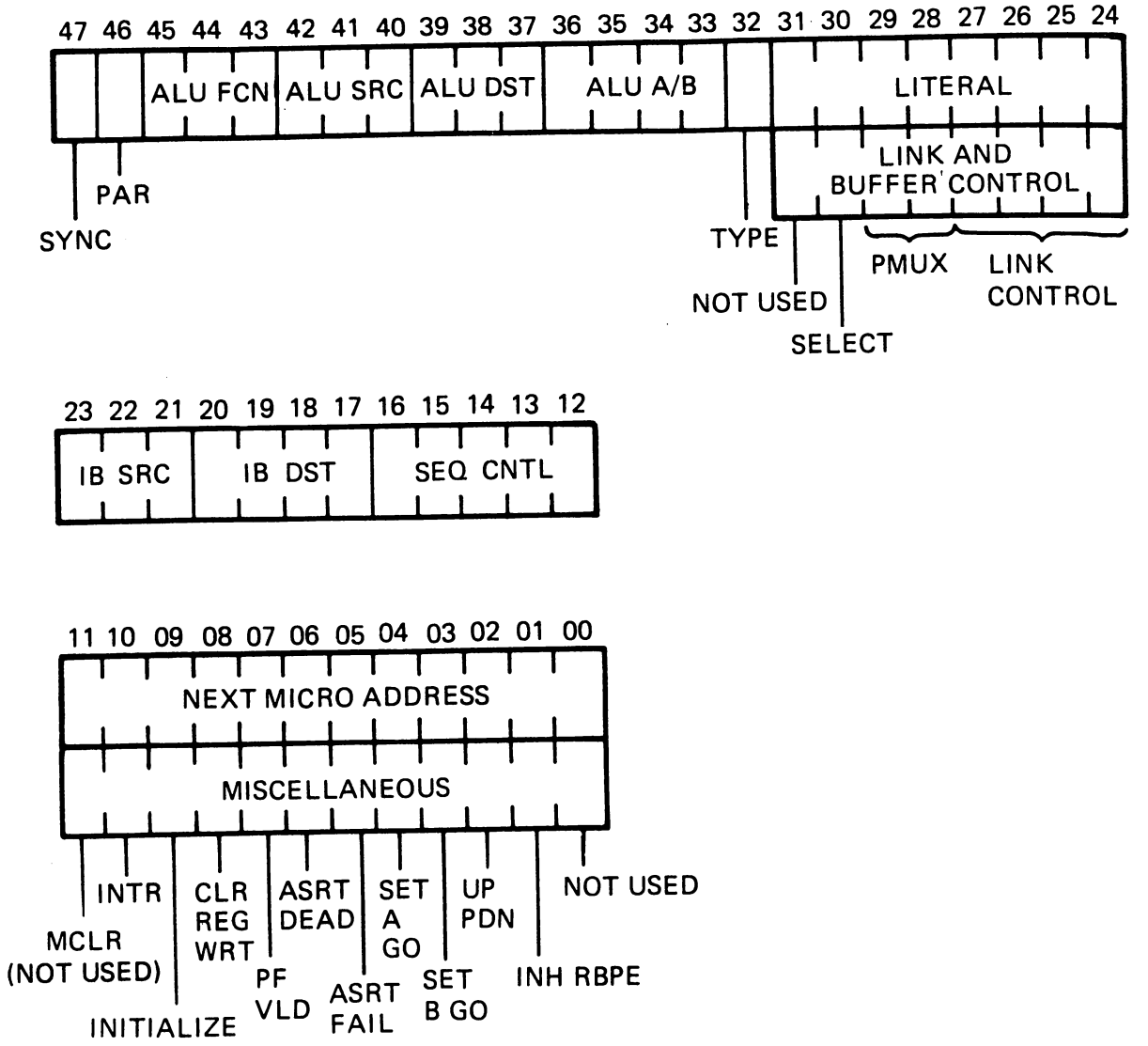


Figure 4-4 Microword Fields

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.

Table 4-1 Microword Fields (Cont)

Bit	Name	Description
<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.
<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	Not used.
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.

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

Table 4-1 Microword Fields (Cont)

Bit	Name	Description
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 MAINTENACE 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 bus (31:00). The microword is applied to the maintenance mux where the mux first selects the lower 32 bits (BUS U<31:00>) 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 maintenance 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.

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 x 47 of PROM and 2K x 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 x 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.

Table 4-2 Maintenance Mux Selection Code

XBUS LSA ØØ	MADR 12	32-Bit Bus (31:ØØ)
Ø	Ø	BUS U<31:ØØ>
Ø	1	BUS U<46:32>; U47
1	X	MADR <12:ØØ>

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

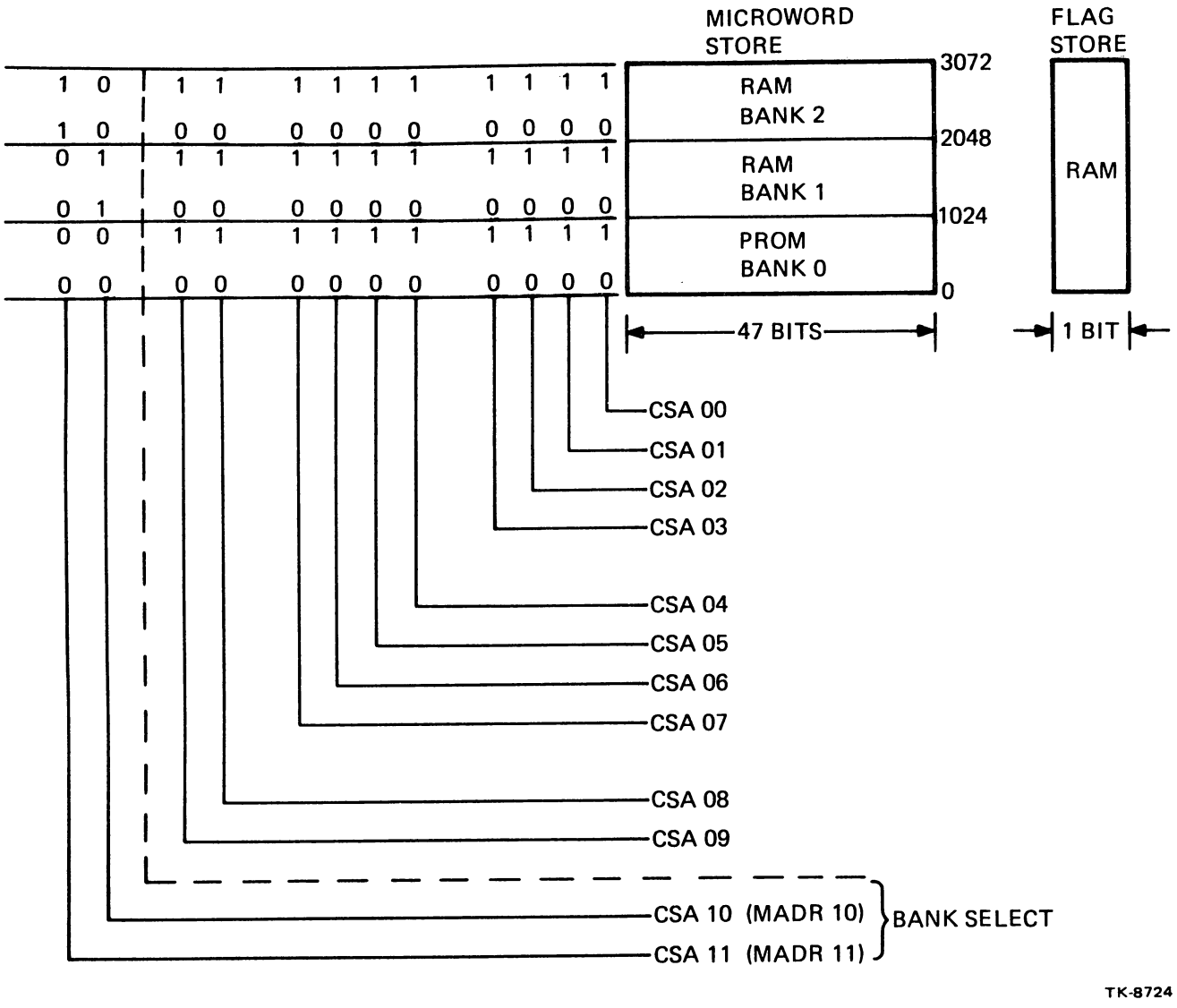


Figure 4-5 Control Store Space

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 x 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 x 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.

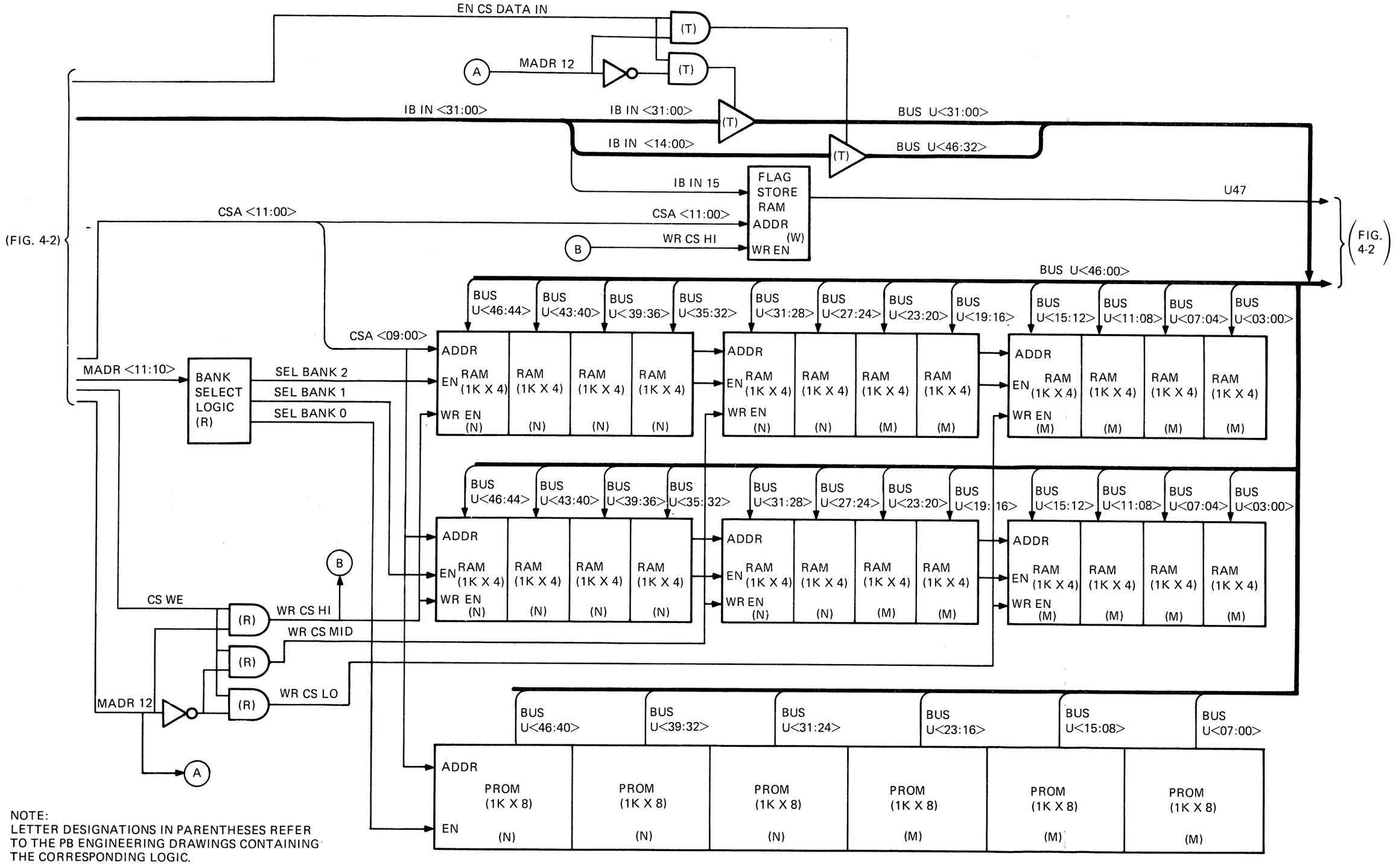
The two writable CS banks are divided into three parts of four RAMs each. The parts are 16 bits each and are designated as LO (BUS U<15:00>), MID (BUS U<31:16>), and HI (BUS U<46:32>). Each part receives a separate write enable signal.

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.



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

TK-8726

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 32-bit bus. MADR 12 is used in the mux select logic to select the high or low portion of the microword for the 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 IN bus. MADR <11:10> is used in the CS logic for CS bank selection.

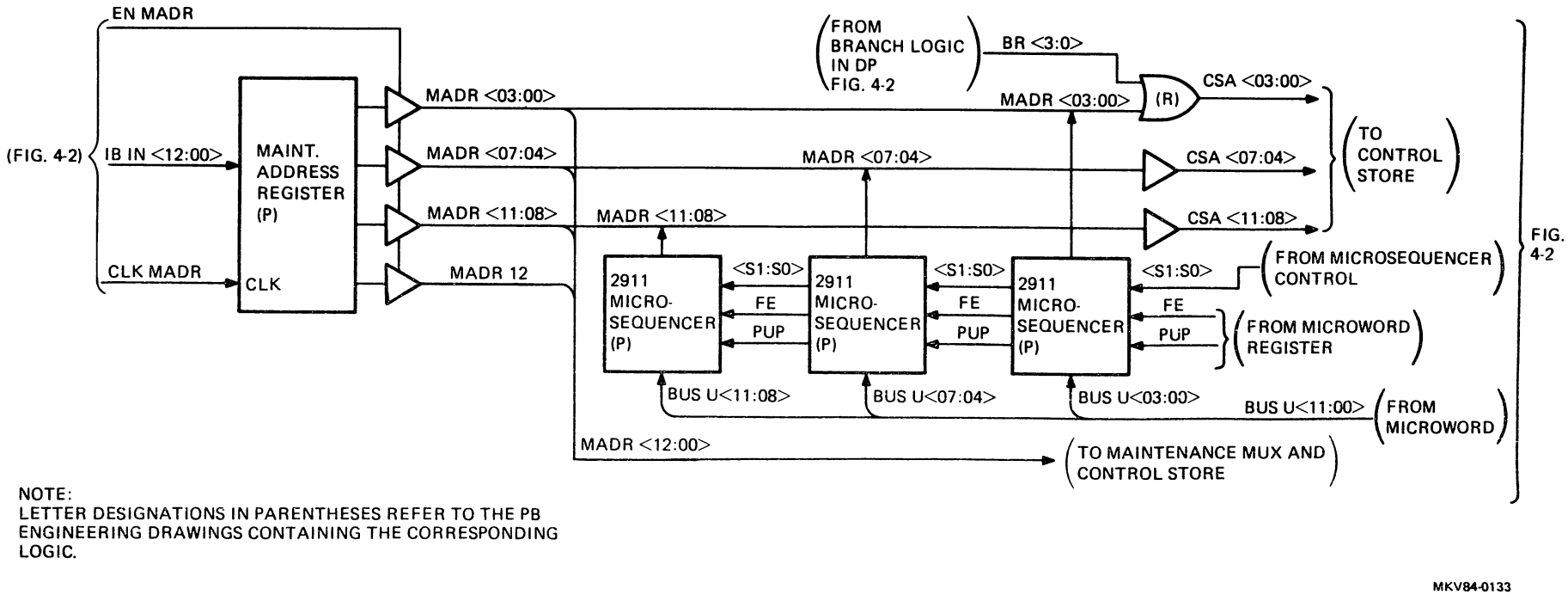
MADR <11:00> is muxed onto common lines with the 12-bit output from the 2911 microsequencer.

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 CRed 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.



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

MKV84-0133

Figure 4-7 Control Store Address Multiplexing

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 x 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.

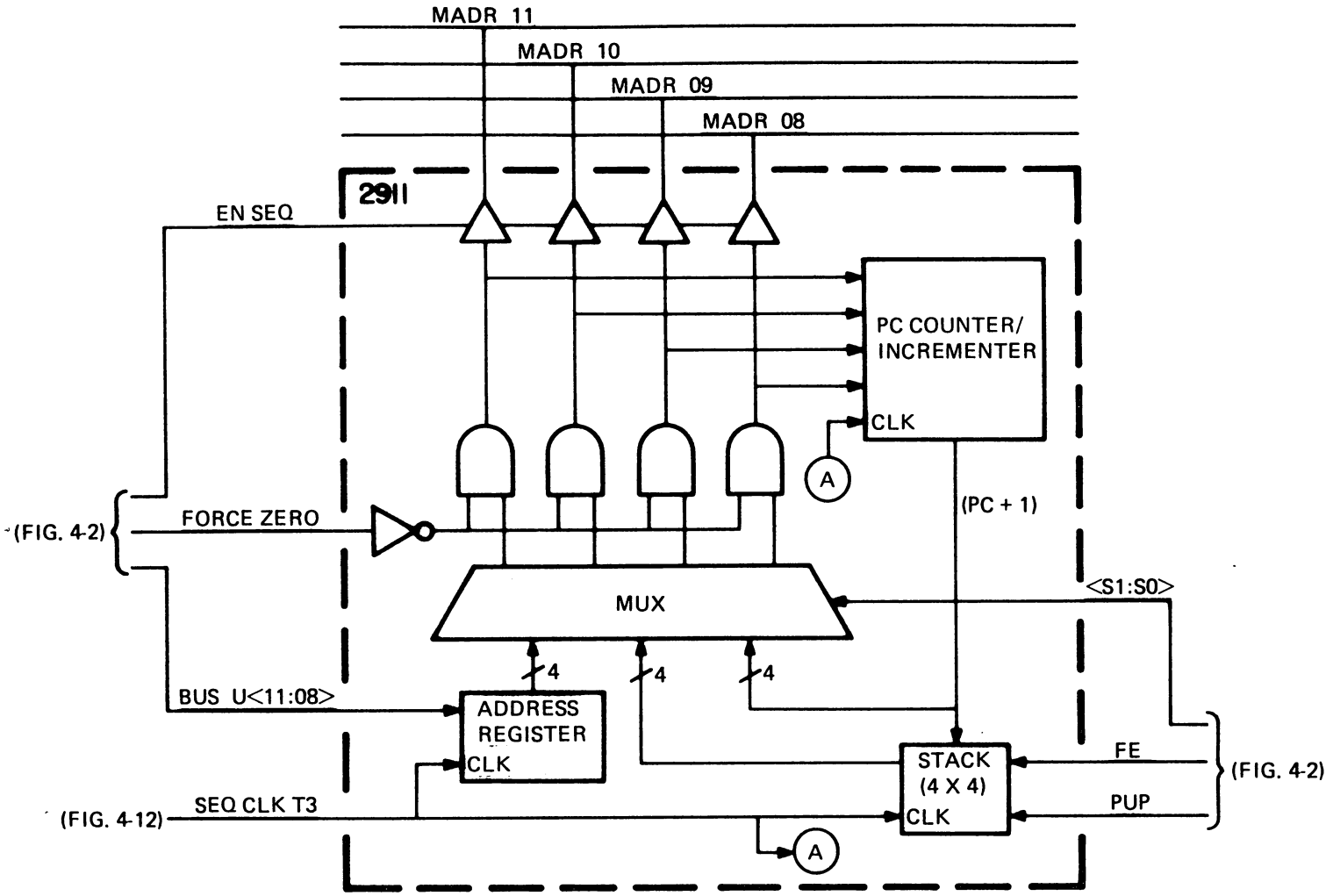
BUS U <16:12> becomes SEQ CNTL <4:0> respectively within the control logic.

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.

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.

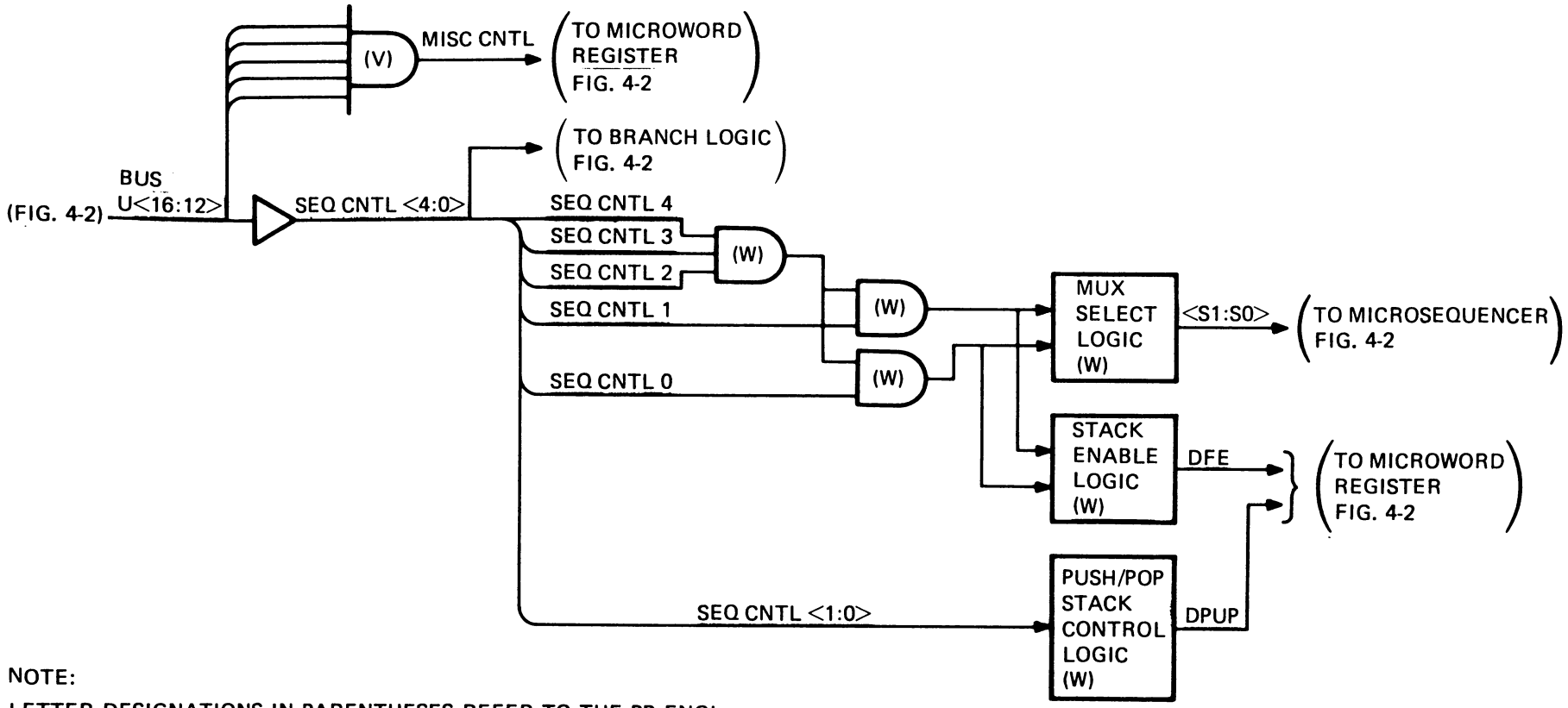
The first 28 counts (bit states) operate the branch logic. During the first four bit states, branches 3, 2, 1, and the A portion of branch 0 are enabled. During the next four bit states, only branch 1 and the A portion of branch 0 are enabled. For the next eight states, the B portion of branch 0 is enabled. 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:
THE LOGIC IN THIS FIGURE IS CONTAINED ON SHEET P OF THE PB ENGINEERING DRAWINGS.

TK-8723

Figure 4-8 2911 Microsequencer



NOTE:

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

Figure 4-9 Microsequencer Control Logic

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

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 SØ control bits from the mux select logic are in the default state (S1 = Ø; SØ = 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:ØØ>) to the MADR <11:ØØ> 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:Ø> are now input to the mux select logic and the stack enable logic. Table 4-3 shows the state of the S1,SØ 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:Ø> are both Ø hence S1 and SØ 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.

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.

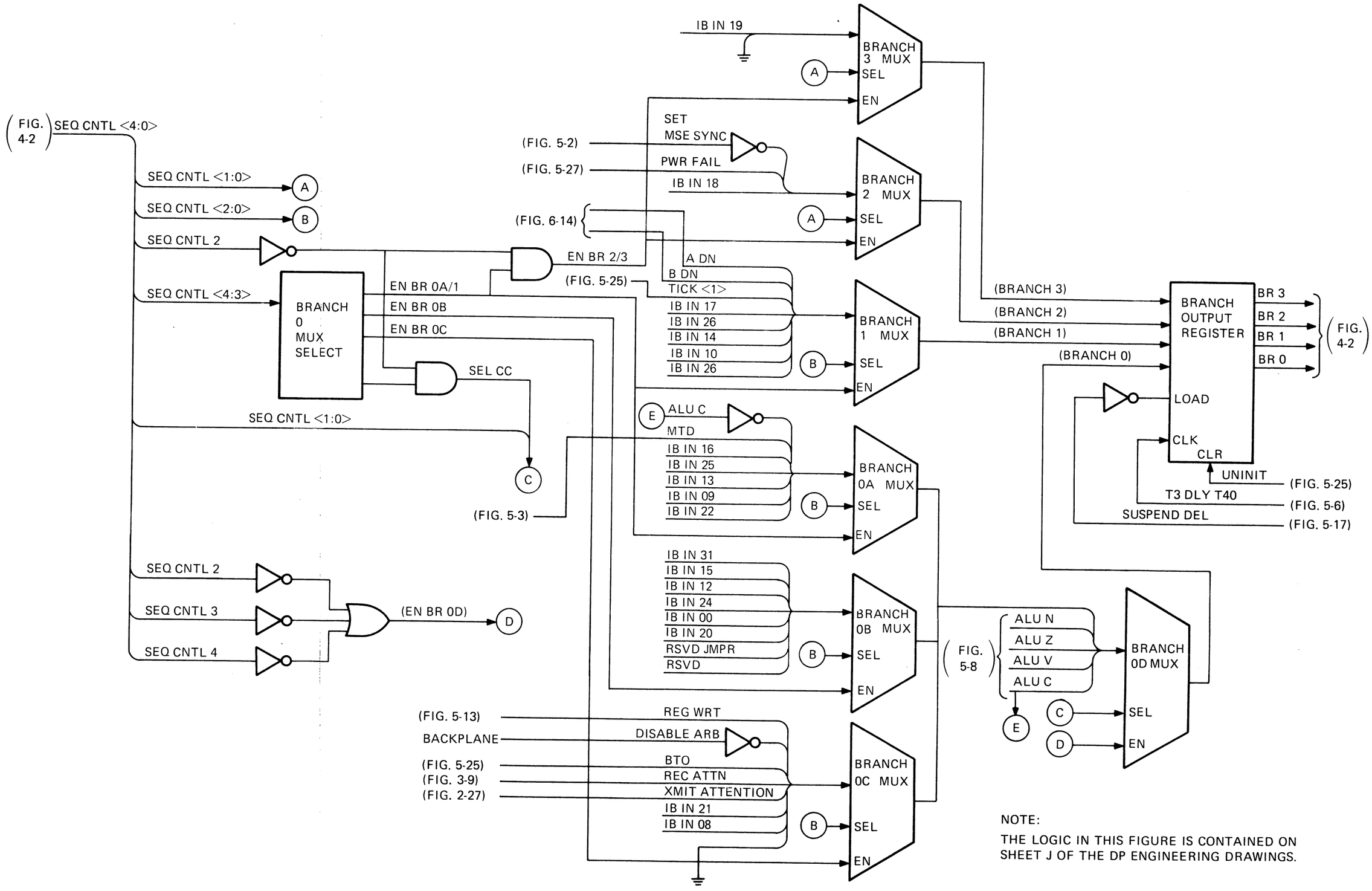


Figure 4-10 Branch Logic

MKV84-0134

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 SET MSE SYNC 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.

Table 4-4 Branch Conditions

Bit State	SEQ CNTL <4:0>	Function	Branch 3	Branch 2	Branch 1	Branch 0
1	00000	Branch	0	$\overline{\text{SET MSE SYNC}}$	A DN	$\overline{\text{ALU C}}$
2	00001	"	0	$\overline{\text{SET MSE SYNC}}$	B DN	$\overline{\text{ALU C}}$
3	00010	"	0	PWR FAIL	TICK <1>	MTD
4	00011	"	IB IN 19	IB IN 18	IB IN 26	IB IN 16
5	00100	"	0	0	IB IN 26	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	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 ATTENTIOI
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 power-up, UNINIT and UNINIT DLY asserts simultaneously in the DP and places the port into the uninitialized state. The assertion of UNINIT (or UNINIT DLY) causes EN MADR to assert to the maintenance address register. Also FORCE ZERO is asserted to the microsequencer in preparation for when the microsequencer will take over the addressing function. In addition, SEQ CLK T3 from the DP is inhibited thereby disabling the microsequencer.

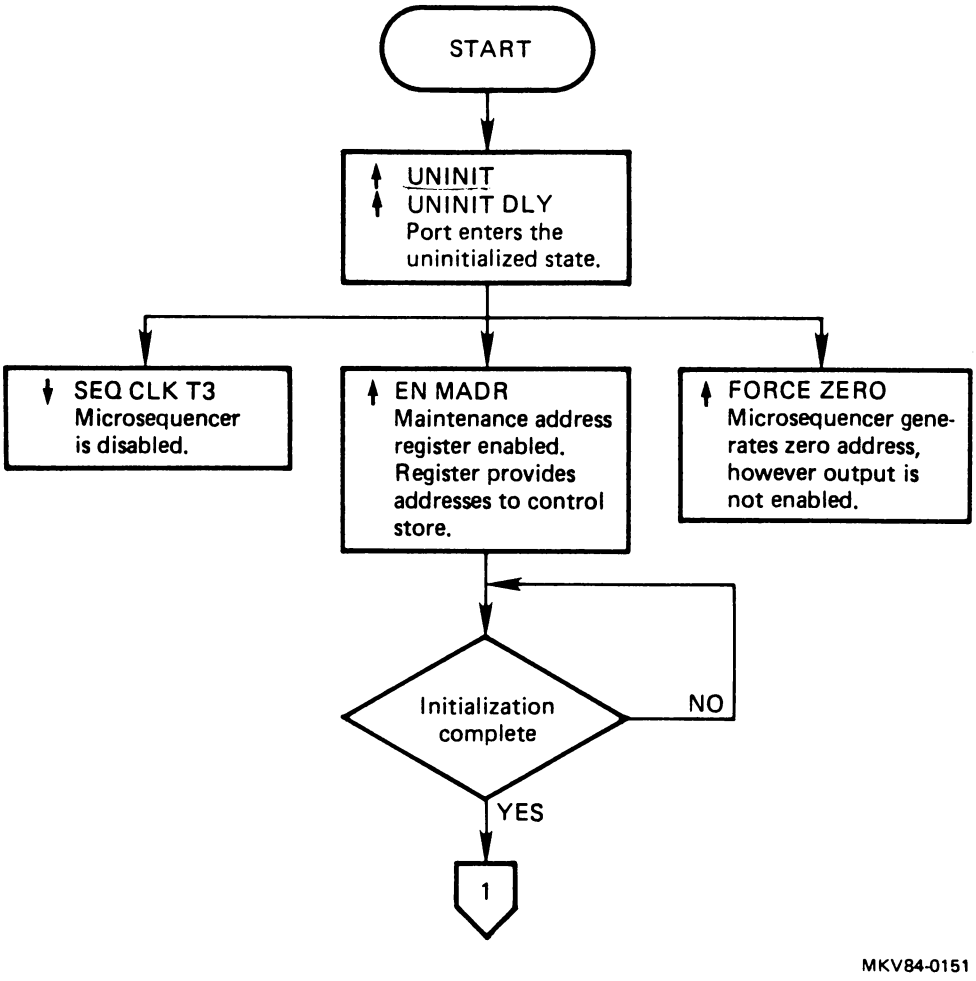


Figure 4-11 Microcode Start-Up Flow Diagram
(Sheet 1 of 2)

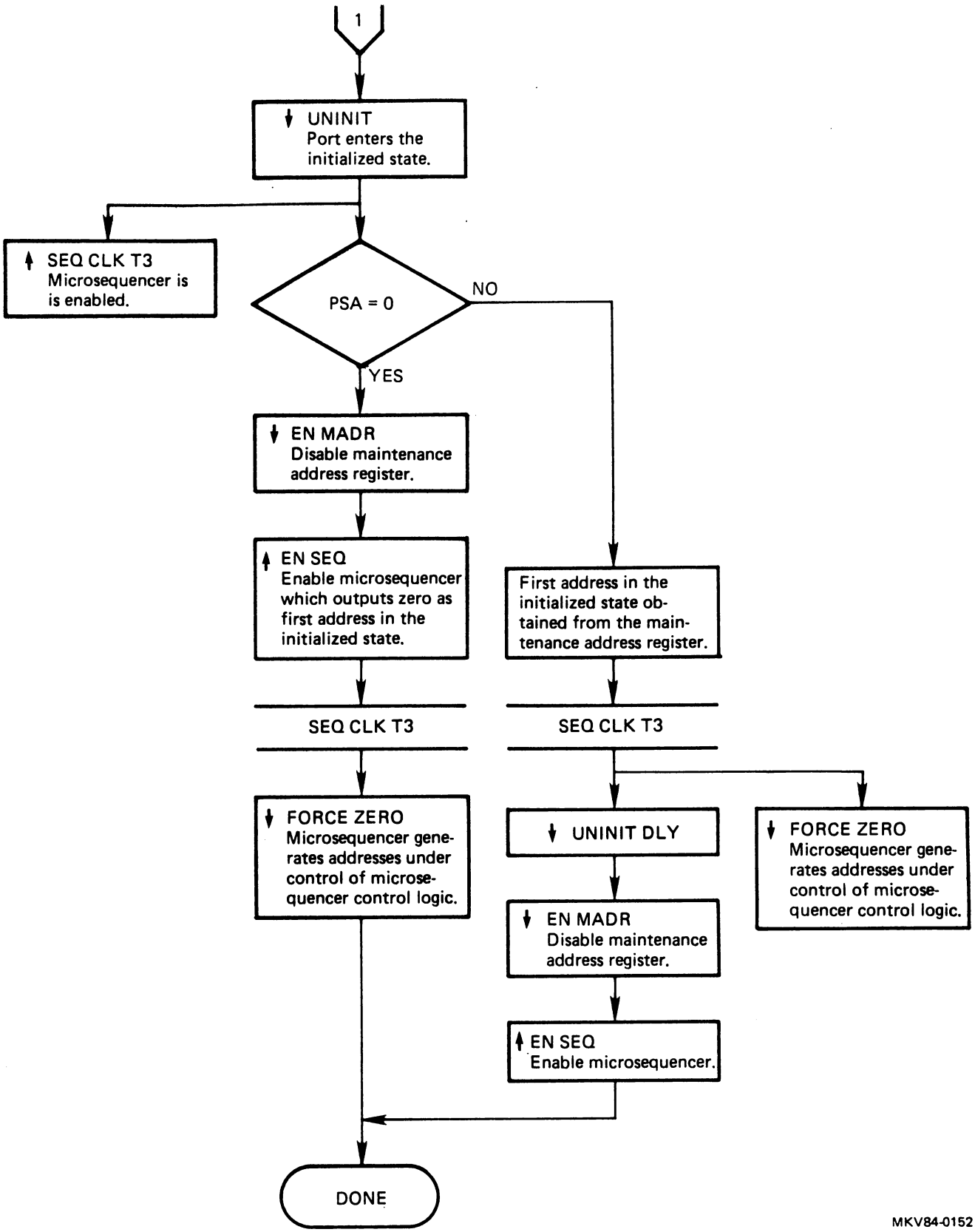
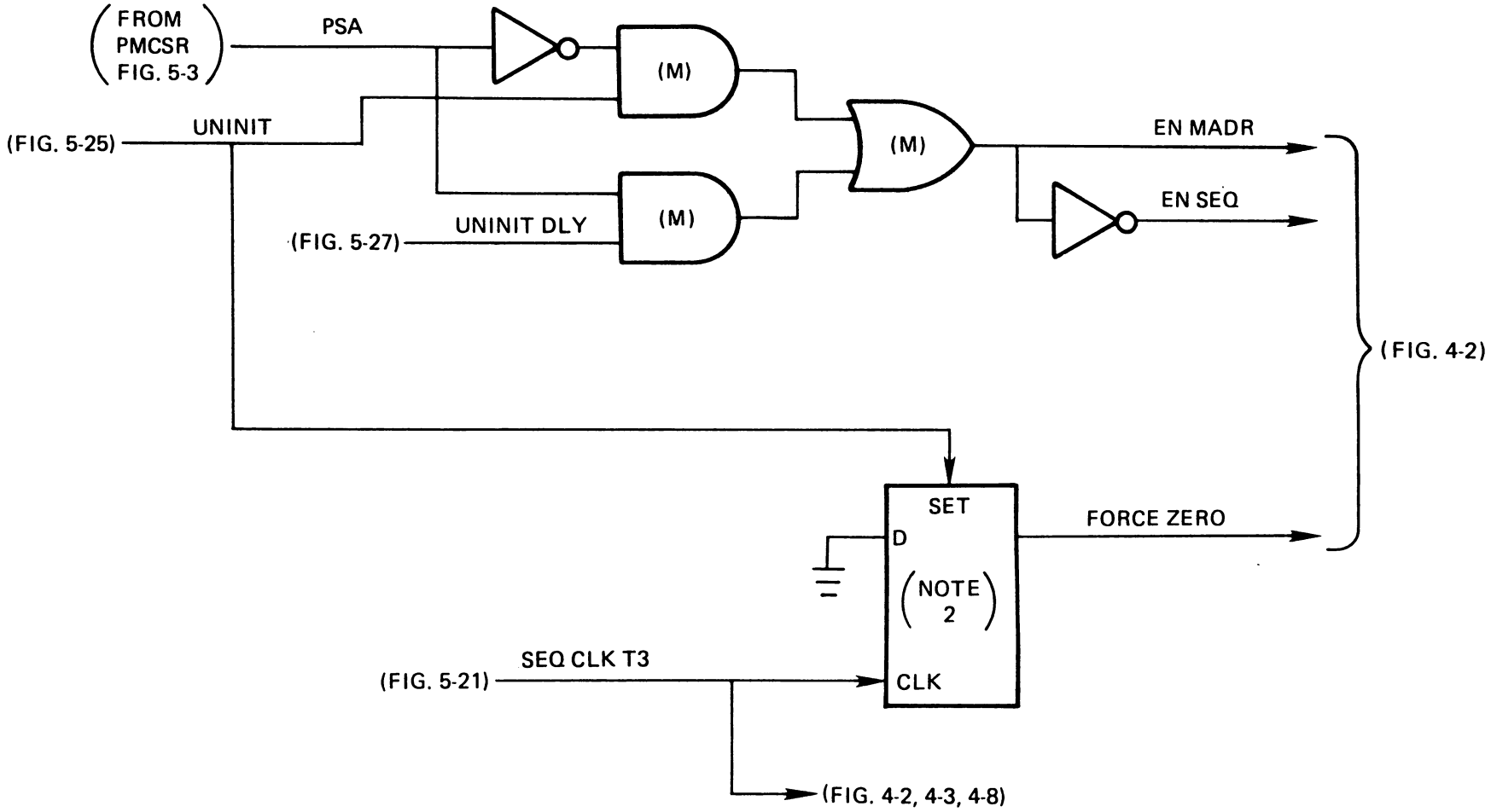


Figure 4-11 Microcode Start-Up Flow Diagram
(Sheet 2 of 2)



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.

MKV84-0137

Figure 4-12 Microcode Start-Up Logic

When initialization is completed, the DP negates UNINIT (UNINIT DLY is not negated until the next clock cycle) and the port goes from the uninitialized to the initialized state. Once in the initialized state, the DP enables SEQ CLK T3 therby enabling 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 PSA (programmable starting address) bit in the PMCSR (port maintenance control/status register). 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 DLY is required to cause EN MADR to negate and EN SEQ to assert. This does not occur until the next clock pulse. 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 next SEQ CLK T3 pulse occurs, UNINIT DLY negates causing EN MADR to negate and EN SEQ to assert. The negation of EN MADR and the assertion of EN SEQ causes the CS address source to shift from the maintenance address register to the microsequencer. The same SEQ CLK T3 pulse resets the FORCE ZERO flip-flop thereby allowing the microsequencer to respond to the next address field of the first microword of the diagnostic routine.

Figure 4-13 is a timing diagram of the microcode start-up sequence. 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.

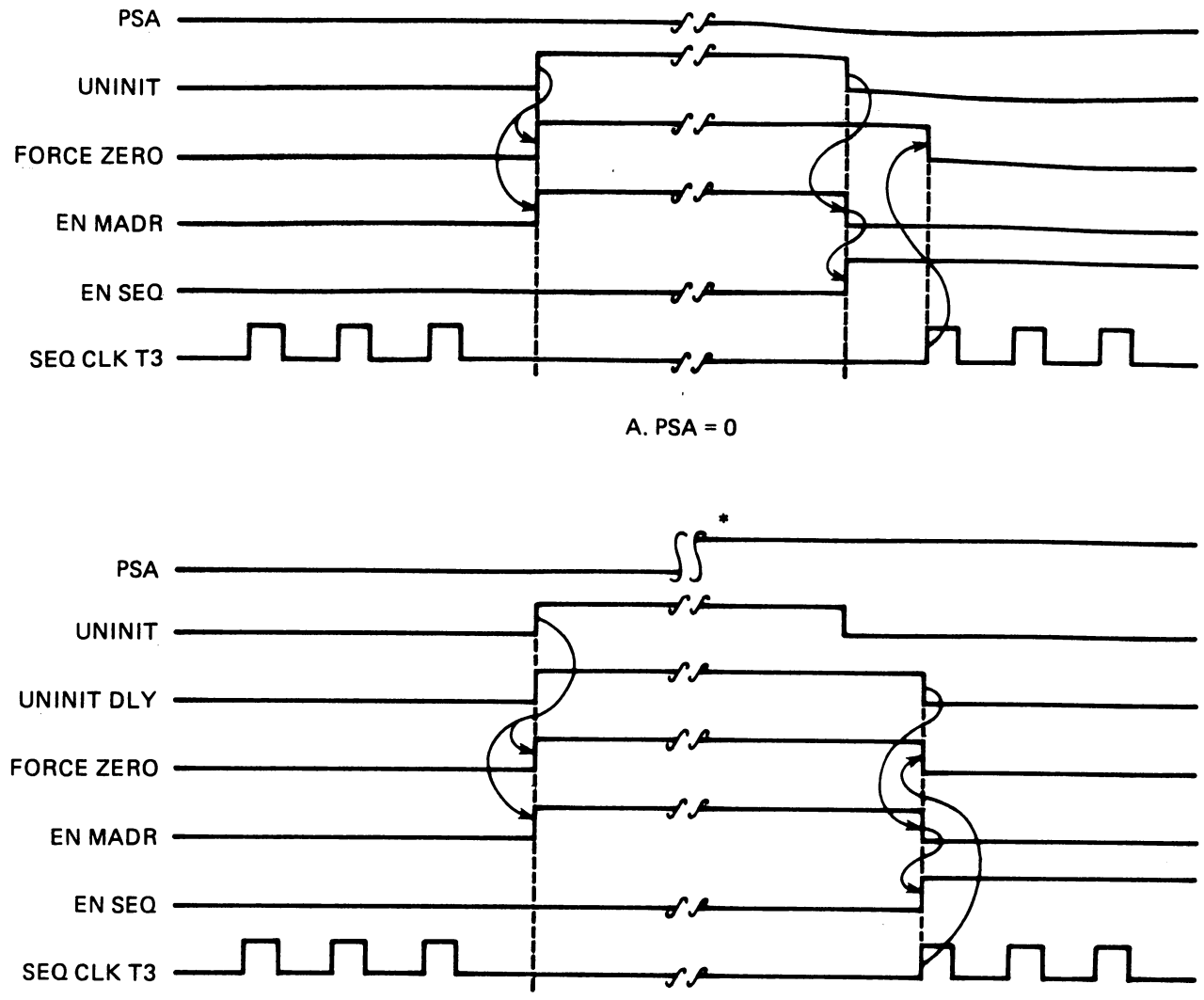


Figure 4-13 Microcode Start-Up Timing

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 GENERAL

Both information data and control data flow within the CI750 Data Path Module (referred to as DP) (Figure 5-1). Data flow may be initiated by the port (port initiated transfer) or by the host CPU (unsolicited CMI transfer). Port initiated transfers are controlled by the port microcode located in the CS (control store). In an unsolicited CMI operation, the port microcode is suspended and the data transfer is controlled by the host CPU via the CCI module and the CIPA bus.

There are three buses within the DP. These are:

1. IB Bus (internal bus)
2. MD Bus (miscellaneous data)
3. IB In Bus

The main bus is the IB bus (internal bus) over which all data flows. All three buses are 32-bits wide while the PORT DATA bus (interfaces with the PB) is 8-bits wide and the CIPA bus (interfaces with the CCI) is 16-bits wide. Hence, data reformatting is required as data flows in and out of the DP.

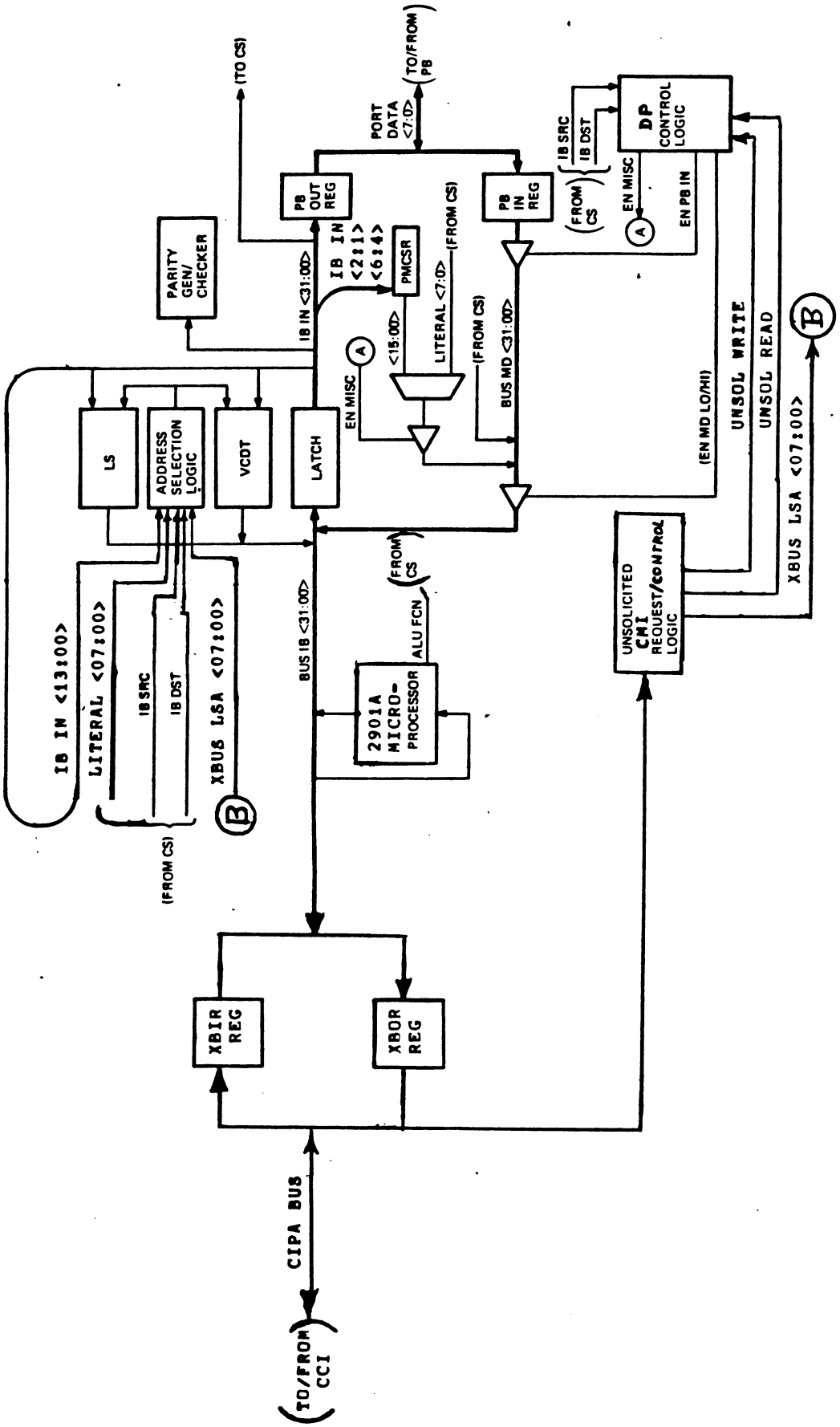


Figure 5-1 Data Path Module Block Diagram

Control of data transfers within the DP involves:

1. Selecting the data source for the IB bus
2. Transferring data from the selected source to the IB bus
3. Selecting the destination for the IB bus
4. Transferring data from the IB bus to the selected destination
5. Reformatting the data as it enters and leaves the DP

Possible data sources for the IB bus are the:

1. LS (local store) RAMs
2. VCDT (virtual circuit descriptor table) RAMs
3. 2901A microprocessor
4. XBIR register (input from CCI)
5. PB IN register (input from PB)*
6. Microword from the CS (control store)*@
7. MADR (maintenance address register) from the CS*@
8. PMCSR (port maintenance control/status register)*@
9. Microword literal field*

* Via the MD bus
@ Only in an unsolicited CMI read operation

Possible destinations for the data on the IB bus are the:

1. LS RAMs#
2. VCDT RAMs#
3. 2901A microprocessor
4. XBOR register (output to the CCI)
5. PB OUT register (output to the PB)#
6. LS/VCDT Address selection logic#
7. Microword CS logic#%
8. MADR#%
9. PMCSR#%

Via the IB IN bus
% Only in an unsolicited CMI write operation

Local store (LS) is a 256 x 32 RAM containing software status blocks and many software registers associated with the port architecture. The VCDT is a 256 x 16 RAM used to store CI node parameters. The LS or VCDT can be selected as a BUS IB source (read the RAM) or a BUS IB destination (write the RAM).

The LS and VCDT are addressed in parallel from the address selection logic. During port initiated operations, the LS/VCDT address may be obtained from the IB bus (via the IB IN bus) or from the microword LITERAL field. If the LS or the VCDT is the IB bus source, the microword IB SRC field selects the LS/VCDT address. If the LS or the VCDT is the IB bus destination, the microword IB DST field selects the LS/VCDT address. During an unsolicited CMI operation, XBUS LSA <07:00> is the LS/VCDT address.

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 an IB bus source or an IB bus destination. The function performed by the 2901A is specified by the ALU FCN field from the microword. The 2901A is not accessed by an unsolicited CMI operation.

The PB IN and PB OUT registers are the data interface between the DP and the PB. The PB IN register functions to convert the data bytes on the PORT DATA bus into longwords for the MD bus. The PB OUT register functions to convert the longwords on the IB IN bus into bytes for the PORT DATA bus.

In a similar manner, the XBOR and XBIR registers are the data interface between the DP and the CCI. The XBOR register functions to convert longwords on the IB bus into 16-bit words for the CIPA bus. The XBIR register functions to convert words on the CIPA bus into longwords for the IB bus.

When enabled, the MD bus carries miscellaneous data to the IB bus. Data carried over the MD bus is the:

1. Output from the PB IN register
2. Microword LITERAL field
3. Output from the PMCSR register
4. Output from the MADR register in the CS
5. Microword from the CS

DP Control Logic controls the flow of data through the DP. The logic enables the selected source and destination for the IB bus and controls the data flow to and from the IB bus. When the port is under microword control, the microword IB DST field and IB SRC field select the IB bus destination and source respectively.

The Unsolicited CMI Request/Control Logic controls data flow between the CCI and the DP. When the port is executing an unsolicited CMI operation, the Unsolicited CMI Request/Control Logic receives commands and control information from the host CPU via the CIPA bus. The Request/Control Logic enables the selected source and destination for the IB bus and then asserts commands to the DP Control Logic to control the data flow to and from the IB bus.

Parity is generated and checked on data flow throughout the DP.

5.2 CIPA BUS

The CIPA (computer interconnect port adapter) bus connects the CCI module in the CPU cabinet with the DP module in the CIPA cabinet.

The bus has 40 signal lines which are divided into groups as shown below.

Data:	17 lines	
Control:	13 lines	
Status:	2 lines	
Power control:	6 lines	
Reserved:	2 lines	(not used)

Figure 5-2 illustrates the CIPA bus and its interface with the CCI and the DP. The figure shows the direction of the signal lines and which lines are bidirectional. Also shown are the mnemonics for the bus signals within the CCI and the DP. This allows identifying what a given CCI signal is called within the DP and vice-versa.

Table 5-1 list the signals by group and gives the function of each. The signals are explained in more detail in the discussion of the functional area to which they pertain.

5.3 DP BUSSES AND INTERFACE

Data transfers throughout the DP undergo reformatting at the DP interfaces. The PB IN and PB OUT registers perform this function at the DP/PB interface. The XBIR and XBOR registers perform this function at the DP/CCI interface.

Refer to Figure 5-3 throughout the following discussion.

5.3.1 PB OUT Register

When the PB OUT register is selected as the IB bus destination, the data on the IB bus (BUS IB <31:00>) inputs 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 8-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 destination for the IB bus.

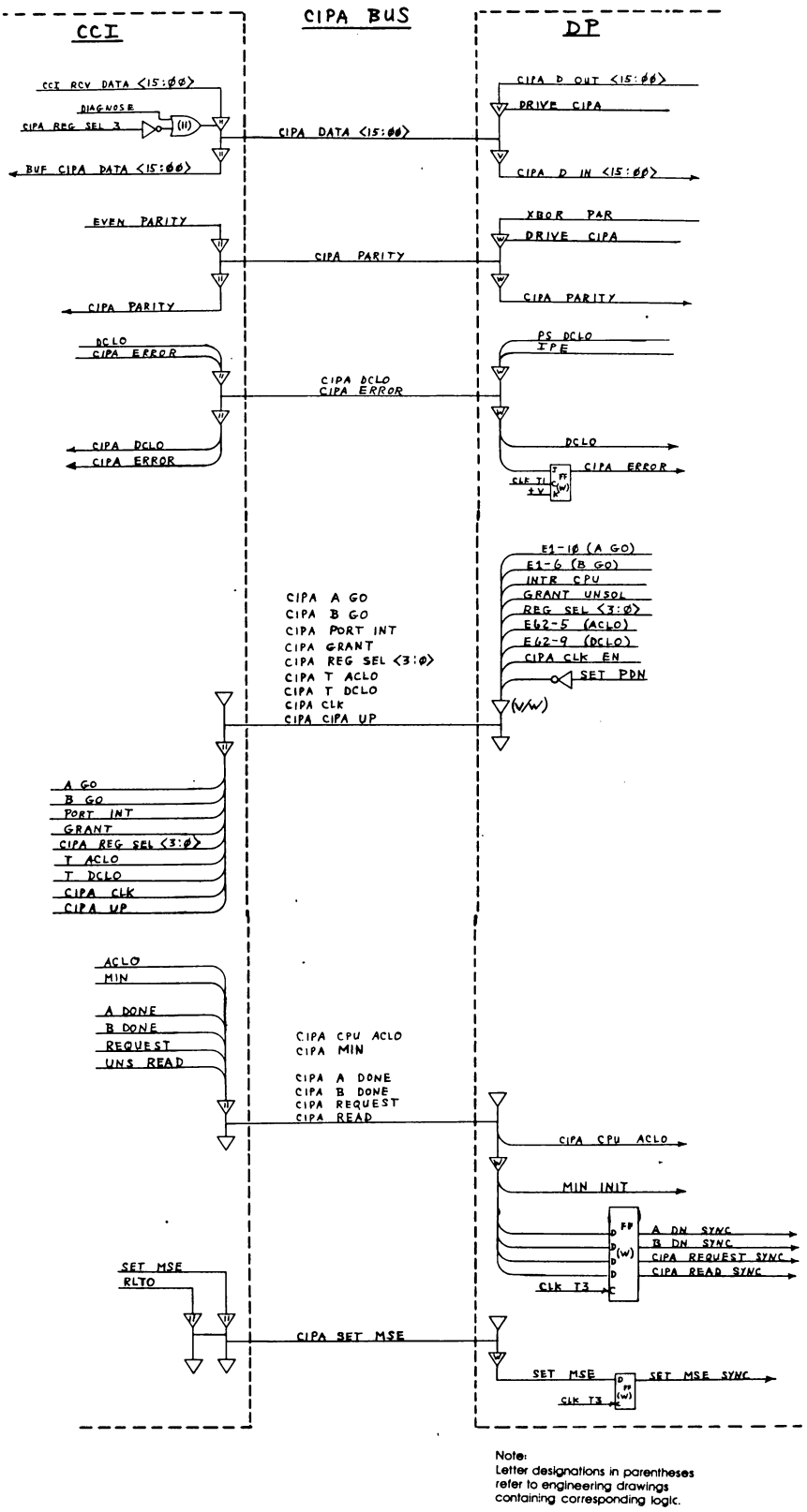


Figure 5-2 CIPA Bus with DP and CCI Interfaces

Table 5-1 CIPA Bus Signals

Group	Mnemonic	Direction	Function
Data	CIPA DATA <15:00>	Bidirectional	Transfers data
	CIPA PARITY	Bidirectional	Odd parity for the data on the CIPA DATA lines
Control	CIPA REG SEL <3:0>	DP to CCI	Selects a CCI register and specifies a write or a read of the register
	CIPA A GO	DP to CCI	Initiates a CMI transfer(s)
	CIPA A DONE	CCI to DP	Indicates CMI transfer(s) initiated by A GO is (are) done
	CIPA B GO	DP to CCI	Initiates a CMI transfer(s)
	CIPA B DONE	CCI to DP	Indicates CMI transfer(s) initiated by a B GO is (are) done
	CIPA REQUEST	CCI to DP	Indicates to the DP that an unsolicited CMI function is pending
	CIPA READ	CCI to DP	Specifies an unsolicited CMI operation as a write or a read
	CIPA GRANT	DP to CCI	Indicates the DP is servicing an unsolicited CMI operation
	CIPA PORT INT	DP to CCI	Initiates an interrupt sequence to the host CPU
	CIPA CLK	DP to CCI	Clocks DP data into CCI and increments read counter for CCI RCV and XMIT files

Table 5-1 CIPA Bus Signals (Cont)

Group	Mnemonic	Direction	Function
Status	CIPA ERROR	Bidirectional	Indicates a parity error on a DP to CCI or CCI to DP data transfer
	CIPA SET MSE	CCI to DP	Indicates either NXM, UCE, or RLTO error in CCI
Power Control	CIPA CIPA UP	DP to CCI	Indicates CIPA cabinet is present, powered-up, and initialized
	CIPA CPU ACLO	CCI to DP	Indicates power going down in CPU cabinet
	CIPA DCLO	Bidirectional	Indicates power is non-operational in either the CPU cabinet or the CIPA cabinet
	CIPA MIN	CCI to DP	Initializes the CIPA
	CIPA T ACLO	DP to CCI	Initiates a power-down of the host system while keeping the CI750 powered-up
	CIPA T DCLO	DP to CCI	Completes the host system power-down initiated by CIPA T ACLO

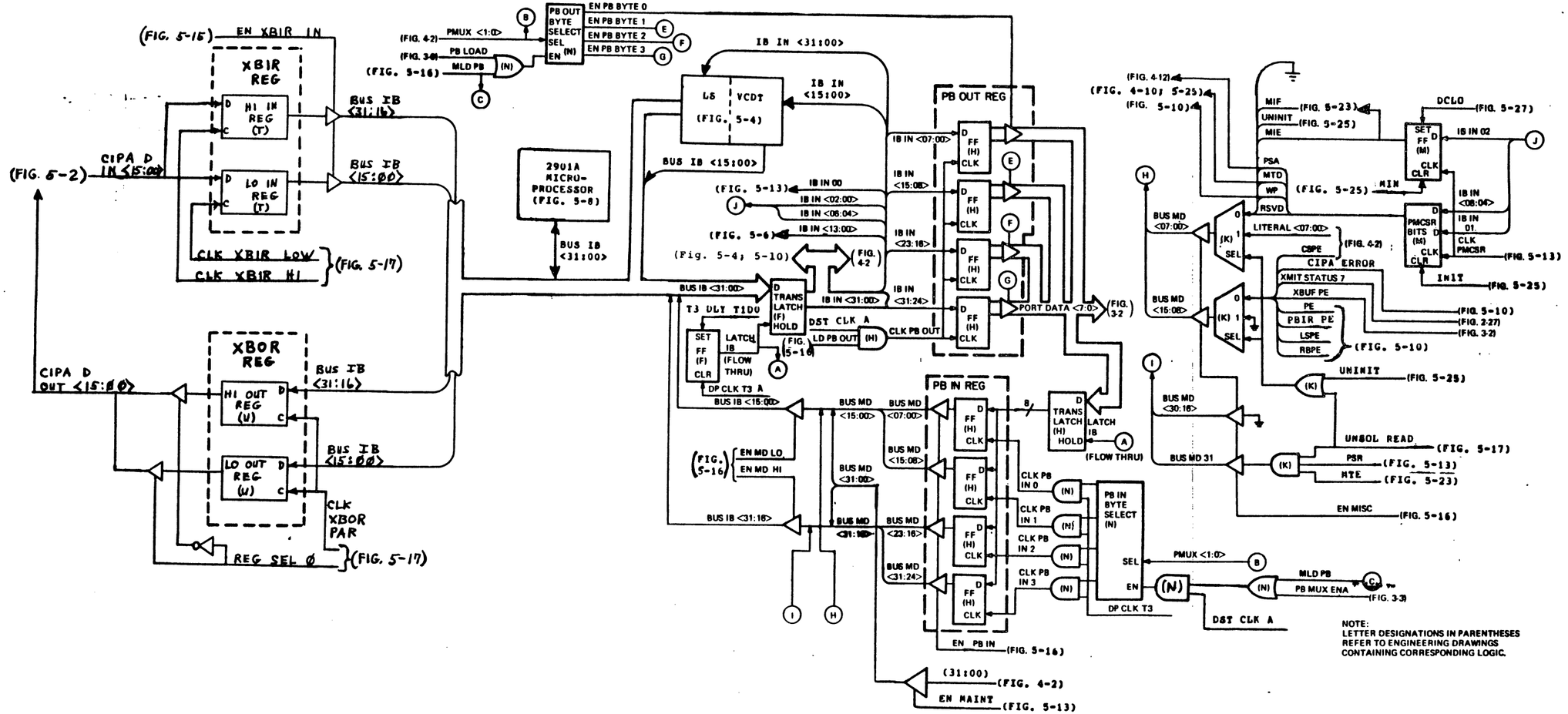


Figure 5-3 DP Buses and Interfaces

The PB OUT register is unloaded under the control of 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 thereby unloading 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.

5.3.2 PB IN Register

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 applied to four sections of the 32-bit PB IN register.

The PB IN register is loaded under the control of the PB. A PB MUX ENA command and a PMUX <1:0> code from the PB control the loading of data into the PB IN register. 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 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 thereby loading up 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 MD BUS as BUS MD <31:00>. EN PB IN is asserted when the PB IN register is selected as the source for the IB bus. EN PB IN then negates while PB MUX ENA asserts to start loading new data bytes into the PB IN register.

The MD bus is gated to the IB bus in two sections. The lower 16 bits are gated by EN MD LO while the upper 16 bits are gated by EN MD HI. EN MD LO and EN MD HI are generated by the DP Control Logic.

5.3.3 XBOR Register

Data on the IB bus (BUS IB <31:00>) is applied to two 16-bit sections of the XBOR register. The 32-bit longword is clocked into the register by CLK XBOR PAR from the CCI/DP Interface Control Logic. CLK XBOR PAR asserts when the XBOR register is selected as the destination for the IB bus.

The XBOR register is unloaded a word at a time by REG SEL 0. When false, REG SEL 0 selects the high word from the XBOR register (BUS IB <31:16>) and outputs the word as CIPA D OUT <15:00>. REG SEL 0 then asserts to select the low word from the XBOR register (BUS IB <15:00>) and outputs it as CIPA D OUT <15:00>. The true state of REG SEL 3 asserts DRIVE CIPA thereby placing the data words from the XBOR register onto the CIPA bus as CIPA DATA <15:00> (Figure 5-2). CLK XBOR PAR then asserts again to load the next longword into the XBOR register.

REG SEL 0 and REG SEL 3 are obtained from the CCI/DP Interface Control Logic.

5.3.4 XBIR Register

The XBIR register is also divided into two sections; a high section and a low section. Input data words from the CCI (CIPA DATA <15:00>) are applied to both sections where they are clocked in by CLK XBIR HI and CLK XBIR LOW from the CCI/DP Interface Control Logic. Longwords transferred from the CCI are transmitted over the CIPA bus a word at a time with the high word being transmitted first. CLK XBIR HI asserts to load the high word on the CIPA bus into the high section of the XBIR register. CLK XBIR LOW then asserts to load the low word into the low word section of the register. The DP Control Logic then asserts EN XBIR IN to gate the longword in the XBIR register onto the IB bus as BUS IB <31:00>. EN XBIR IN then negates while CLK XBIR HI asserts to start loading the next longword into the XBIR register.

5.4 LS AND VCDT (Figure 5-4)

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

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

Figure 5-4 illustrates the LS and VCDT sections and the addressing and enabling signals associated with each. All three sections (LS HI, LS LO, VCDT) are addressed in parallel by LSA <07:00> from an LSA (local store address) mux. Thus access is to the same location in each section.

Data placed into the LS and VCDT is from the IB IN bus. IB IN <31:16> is input into the LS HI section. IB IN <15:00> is input into the LS LO section and the VCDT.

Data out of the LS and VCDT is placed onto the IB bus. The LS HI section outputs onto BUS IB <31:16>. The LS LO section and the VCDT output onto BUS IB <15:00>. When the LS is read out, 32 LS bits are placed onto the IB bus. When the VCDT is read out, the upper 16 bits of the IB bus (BUS IB <31:16>) are zeros supplied from the MD bus (Paragraph 5.8.3.1).

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

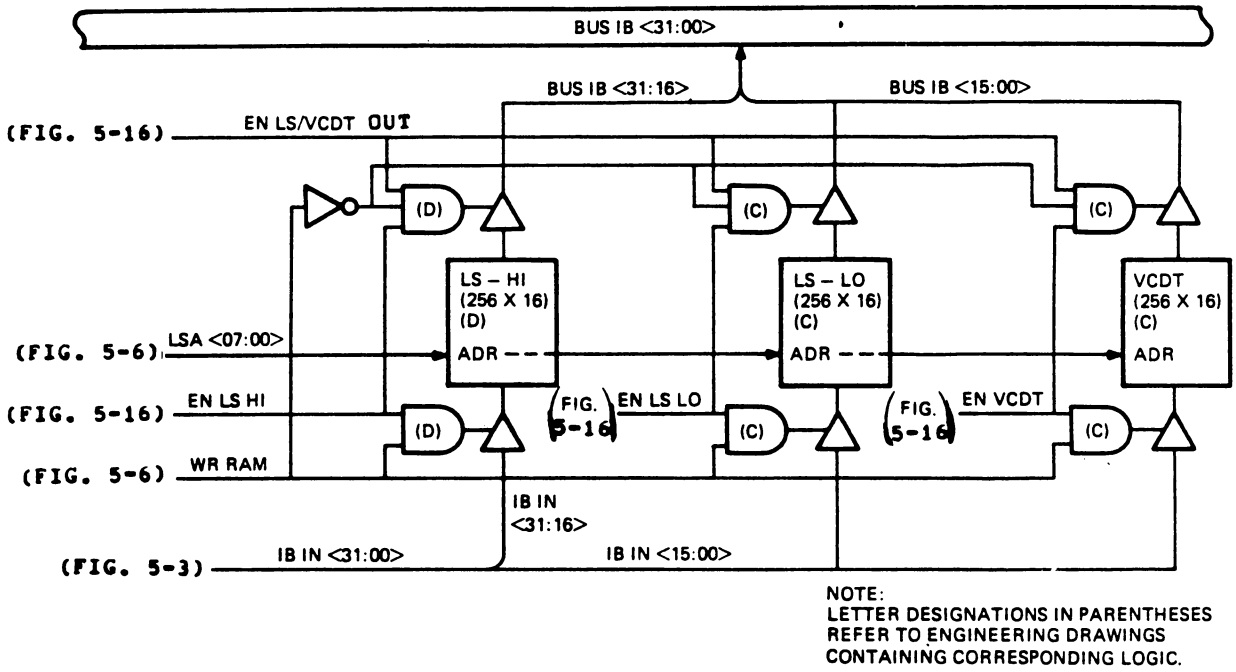


Figure 5-4 LS/VCDT Block Diagram

5.4.1 LS/VCDT Address Selection

Figure 5-5 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 an IB bus destination or a possible IB bus 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 IB bus 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 5-6 is a detailed block diagram of the LS/VCDT address selection function. Refer to it during the following discussion.

The LSA mux has two select inputs (SEL 2, SEL 1) that 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.

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

The SEL 2 and SEL 1 inputs are obtained from two flip-flops. Both flip-flops are set by SUSPEND SEQ when the port goes into the suspend mode (see Paragraph 5.11.2.4) 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. SUSPEND SEQ asserts during an unsolicited CMI request when microcode control of the LS/VCDT address is suspended and the host CPU supplies the LS/VCDT address via the unsolicited CMI request/control logic (see Figure 5-1 and Paragraph 5.8).

When not executing an unsolicited CMI request (SUSPEND SEQ 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 <07:00> input as the LSA address source.

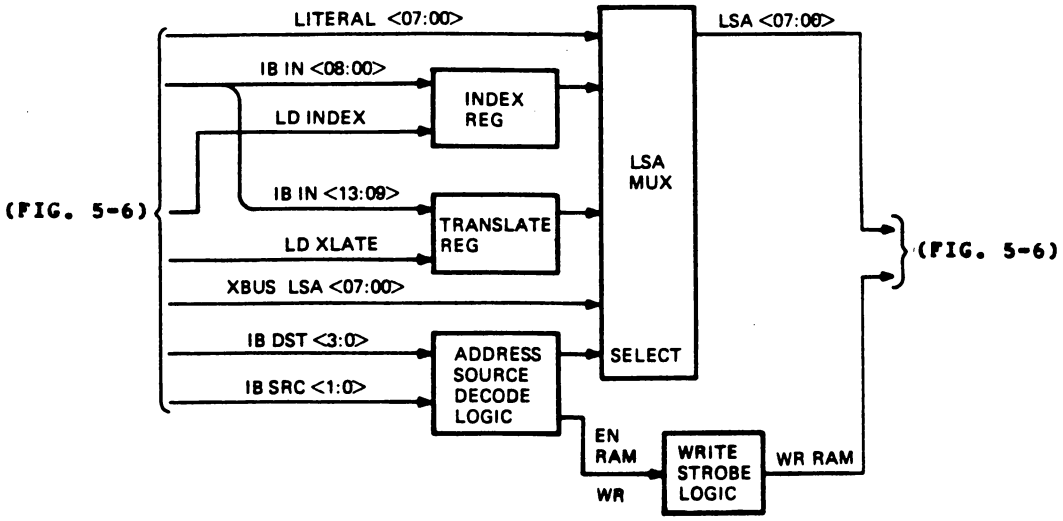


Figure 5-5 LS/VCDT Address Selection
Simplified Block Diagram

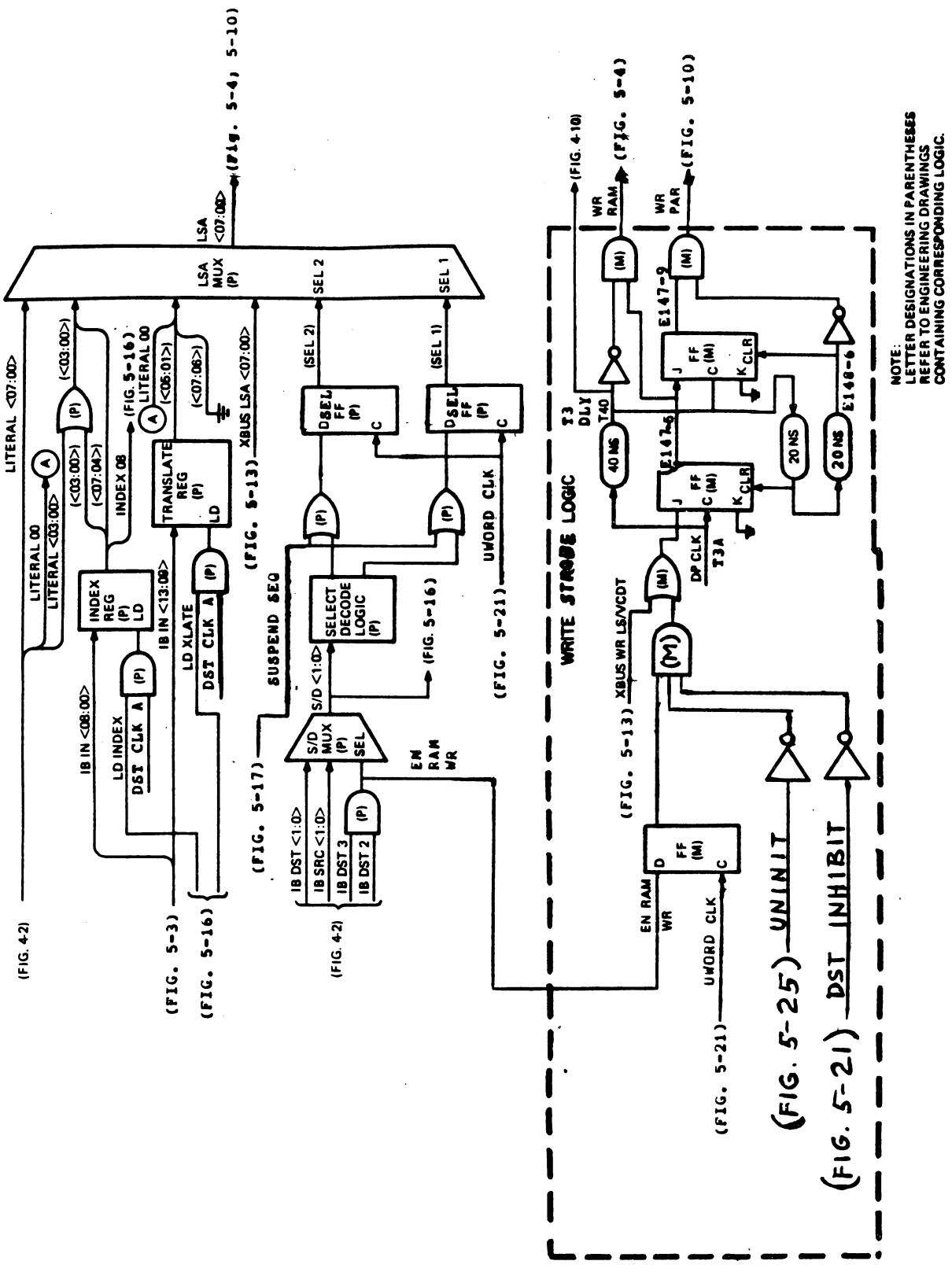


Figure 5-6 LS/VCDT Address Selection Block Diagram

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 (EN RAM WR asserts), the LS or the VCDT is selected as the IB bus destination (see Table 5-8) 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 IB bus. In this case, the mux defaults to IB SRC <1:0> for the S/D <1:0> bits in the event the LS/VCDT is selected as the source for the IB bus.

When the LSA mux SEL bits <2:1> are 0:0, the literal input (LITERAL <07:00>) from the microword 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 allows one bit indexing of the translate LS or VCDT entries.

5.4.2 LS/VCDT Write Strobe Logic

As previously mentioned, when the LS or the VCDT is selected as the destination for the IB bus, IB DST <3:2> are both true thereby asserting EN RAM WR to the write strobe logic. The write strobe logic generates a write strobe for the LS/VCDT RAMs (WR RAM) (Paragraph 5.4) and a write strobe for the LS/VCDT parity RAMs (WR PAR) (Paragraph 5.7.6).

EN RAM WR is applied to a flip-flop. If this is not an unsolicited CMI request (DST INHIBIT false) and the port is not in the uninitialized state (UNINIT false), the flip-flop output is ORed with XBUS WR LS/VCDT from the unsolicited CMI request logic. The OR gate output is applied to 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 and the WR PAR strobe 40 ns wide. The WR PAR strobe begins on the trailing edge of the WR RAM strobe as shown in Figure 5-7.

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 T3 A. 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 T3 DLY T40. T3 DLY T40 is inverted and applied to the WR RAM AND gate causing WR RAM to negate.

The assertion of T3 DLY T40 clocks the second flip-flop. The second flip-flop output is applied to the WR PAR AND gate which then asserts WR PAR. T3 DLY T40 is delayed 40 ns, inverted, and then applied to the WR PAR AND gate causing WR PAR to negate.

5.5 MD BUS

The MD (miscellaneous data) bus carries data from miscellaneous sources to the IB bus. The sources are enabled onto the MD bus one at a time thereby isolating the bus from all sources except the one driving the bus. The enabling signals are supplied by the DP Control Logic and the unsolicited CMI request logic. The data sources and their enabling signals are shown in Table 5-3.

Table 5-3 MD Bus Data Sources

Data Source	Enabling Signal
PB IN register	EN PB IN - from DP control logic
MADR register	EN MAINT - from unsolicited CMI request logic
CS microword (MDATR)	EN MAINT - from unsolicited CMI request logic
PMCSR register	EN MISC - from DP control logic
Microword LITERAL field	EN MISC - from DP control logic

Selection between the MADR register and the CS microword (MDATR) is made by the maintenance mux in the CS. Selection between the PMCSR register and the microword LITERAL field is made by the PMCSR/LITERAL mux.

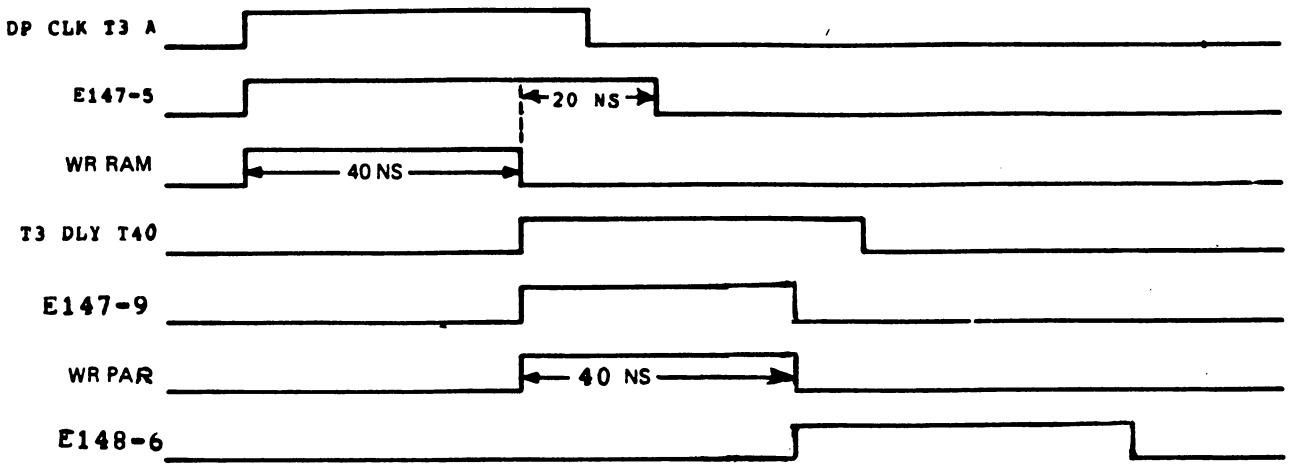


Figure 5-7 Write RAM Timing Diagram

EN MD LO and EN MD HI respectively gate the low word and high word sections of the MD bus to the IB bus. EN MD LO and EN MD HI are supplied from the DP Control Logic.

The data supplied to the IB bus from the MD bus must be in a 32-bit longword format. When a source is read that is not 32 bits, the unused bit locations must be zeros. The MD bus logic supplies the fill-in zeros when necessary.

5.5.1 PB IN Register

EN PB IN gates the 32-bit output of the PB IN register onto the MD bus as discussed in Paragraph 5.3.2.

5.5.2 MADR and MDATR

EN MAINT gates the 32-bit output of the maintenance mux (31:00) onto the MD bus. The maintenance mux is shown in Figure 4-2 and discussed in Paragraph 4.4.

The maintenance mux output is always 32-bits wide. The mux fills in zeros when the selected data source is less than 32 bits.

The maintenance mux selects the maintenance address register (MADR) or the maintenance data register (MDATR)*. The mux selection is accomplished by XBUS LSA 00 from the unsolicited CMI request logic. (Accessing MADR or MDATR is only done via an unsolicited CMI operation.) MADR 12 is used to select the high or low portion of the 48-bit microword for the MD bus.

* Accessing the CS microword is a read (or write) of the maintenance data register (MDATR). MDATR does not exist as a physical register.

When the MADR or the MDATR register is the IB bus destination, the write data is input to the register via the IB IN bus.

5.5.3 PMCSR and Microword LITERAL Field

EN MISC gates the 16-bit output of the PMCSR/LITERAL mux onto the lower half of the MD bus (BUS MD <15:00>). Zeros are gated onto the upper half of the MD bus (BUS MD <31:16>) for all read operations of the PMCSR or the microword LITERAL field. When executing an unsolicited CMI read of the PSR (UNSOL READ and PSR true), bit 31 becomes MTE*.

* The PSR (port status register) is a 32-bit software register located in LS. Bits <30:16> of the register are all zeros. Bit 31 is the MTE (maintenance error) bit. When the PSR is read by an unsolicited CMI operation, the lower 16 bits output from LS onto the lower half of the IB bus. The upper half of the IB bus is supplied from the MD bus by asserting EN MD HI and EN MISC. EN MISC enables the 15 zeros and the MTE bit onto BUS MD <31:16> and EN MD HI gates them to the upper half of the IB bus.

When the port is not in the uninitialized state and not executing an unsolicited CMI read operation (UNINIT and UNSOL READ false), the PMCSR/LITERAL mux selects LITERAL <7:0> for the lower eight bits of the MD BUS (BUS MD <07:00>). The next eight bits (BUS MD <15:08>) are grounded by the mux to supply zeros.

When the port is in the uninitialized state (UNINIT true) or executing an unsolicited CMI read operation (UNSOL READ true), the PMCSR/LITERAL mux selects the 16-bit PMCSR register.

Register bit 00 (MIN) is grounded and therefore always reads as a 0. If the host CPU writes a 1 into PMCSR bit 00, the CCI initializes and asserts CIPA MIN on the CIPA bus. CIPA MIN then functions to initialize the DP (see Paragraph 5.12.2).

An unsolicited CMI write of the PMCSR will assert CLK PMCSR from the unsolicited CMI request logic, to write five PMCSR bits (MIE, PSA, MTD, WP, RSVD).

The PMCSR register bits are described in Table 5-4.

Table 5-4 PMCSR Bits

Bit	Mnemonic	Description
15	PE	Parity Error: PE is the OR of all the port parity error bits. These are PMCSR bits <14:08>. PE is cleared when PMCSR <14:08> are 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. It will not set during an unsolicited CMI operation.
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 CMI operation.
12	RBPE	Receive Buffer Parity Error: Set when a parity error is detected on a data transfer from the PB to the DP.
11	XMIT STATUS 7	Transmit Data Parity Error: Set when a parity error is detected in the link transmit channel.
10	CIPA ERROR	CIPA Error: Set when a parity error is detected on a DP to CCI or CCI to DP data transfer.
09	PBIR PE (OPE)	PB IN Register Parity Error (Output Parity Error): Set when a parity error is detected on a data transfer through the PB IN register to the IB bus.
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: 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 DCLO (during power-up), MIN, or MTE. The microcode is started when UNINIT is cleared by writing a 1 into the PICR or by a boot timeout.

Table 5-4 PMCSR Bits (Cont)

Bit	Mnemonic	Description
Ø6	PSA	Programmable Starting Address: When the PSA bit is set, the port microcode will start running at the address in the MADR register. When the PSA bit is reset the microcode starts at location ØØØ.
Ø5	RSVD	Not used.
Ø4	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. WP is cleared on Initialization.
Ø3	MIF	Maintenance Interrupt Flag: When set, this bit indicates that an interrupt causing condition (DCLO, INTR, MTE) has occurred.
Ø2	MIE	Maintenance Interrupt Enable: When set interrupts are enabled. This bit is set by DCLO during power-up or by writing MIE with a 1. It is cleared during DP initialization or by writing MIE with a Ø.
Ø1	MTD	Maintenance Timer Disable: When set, the boot timer is disabled and cannot cause an interrupt. When reset, the timer is enabled.
ØØ	MIN	Maintenance Initialize: When set, an initialize signal is generated that clears all port errors and leaves the port in the uninitialized state. MIN is write only and always reads as Ø. The MIN bit does not exist in the DP. MIN is written in the CCI module (see Paragraph 5.12.2).

5.6 DP ALU

The DP contains eight 2901A microprocessor chips which constitutes the DP ALU, shown in Figure 5-8. The ALU functions are controlled by the microword from the CS. The following paragraphs discuss the ALU 2901A microprocessor and its operations.

5.6.1 2901A Microprocessor

Eight 2901As are used in parallel to formulate a 32-bit longword input/output to the IB bus. The 2901A contains a 16 x 32 RAM, an ALU (arithmetic logic unit), a Q register, and control circuitry.

The 16-word RAM has two output ports (A and B) and a single input port. The ALU A/B <3:0> address field is used to address the RAM. The A address selects RAM data to be output at port A. The B address selects RAM data to be output at port B. The B address also selects the write location for data input at the input port. The A and B address lines are tied together, hence for a given address, both port A and port B output the same data.

Data is input to the RAM through a RAM shifter. The shifter has three input ports; F, 2F, and F/2. Port F applies the input to the RAM unchanged. Port 2F applies twice the input to the RAM while port F/2 applies 1/2 the input to the RAM. The input port is selected by the ALU DST <2:0> code.

The RAM is 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 muxed into the ALU by the source control signals supplied from the CS microword.

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 port A of the RAM and the IB bus connected as inputs A and D respectively. The ALU S-input mux has both output ports of the RAM and the Q register as inputs A, B, and Q respectively.

The muxes can select various combinations of input pairs among the A, B, D, Q, and zero inputs as source operands to the ALU. ALU SRC <2:0> from the port microword is used to select the ALU source operands. The ALU source code is defined in Table 5-5.

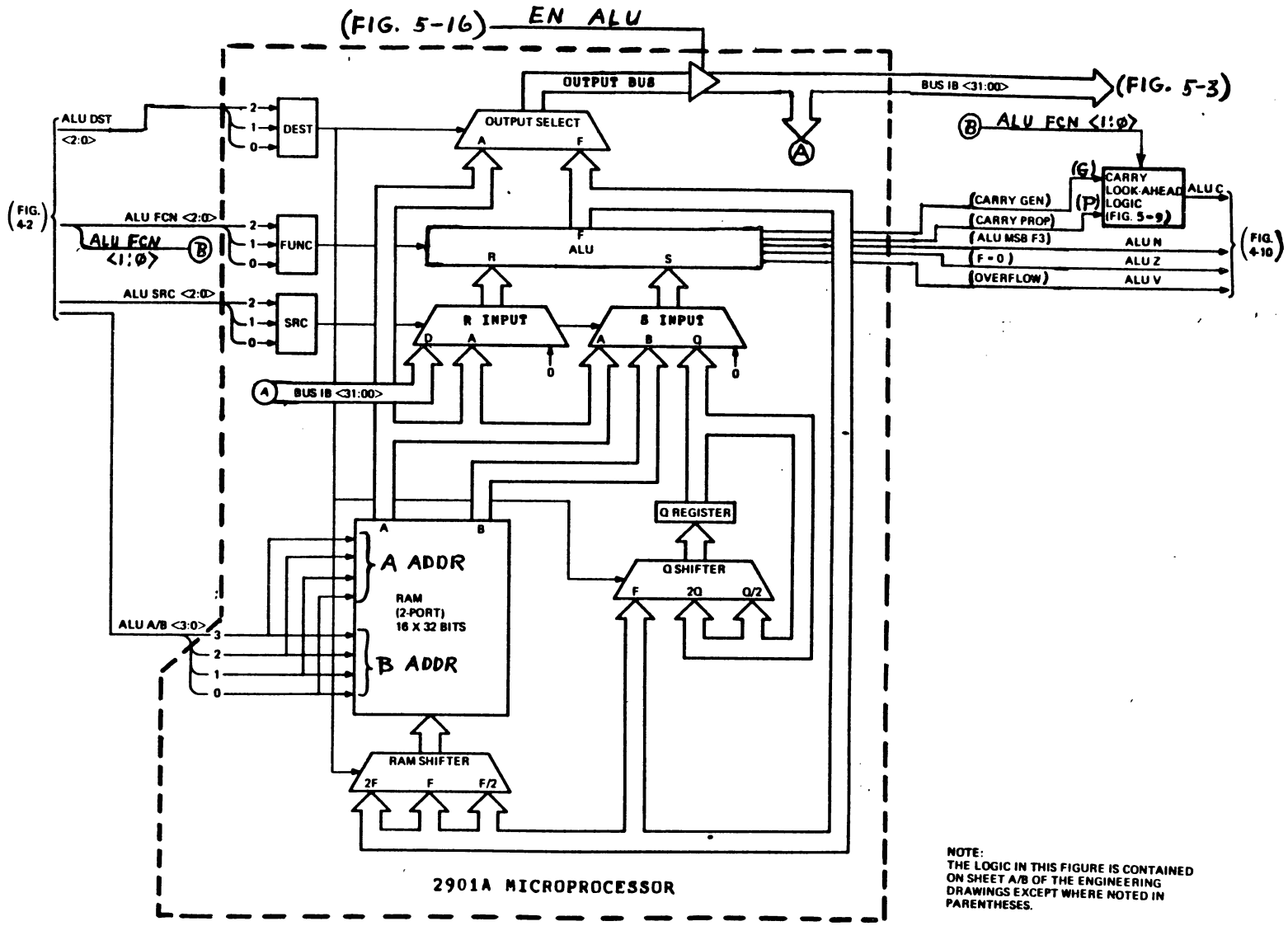


Figure 5-8 ALU Block Diagram

Table 5-5 ALU Source Code

Mnemonic	ALU SRC 2	1	0	Source R	S
AQ	0	0	0	A	Q
AB	0	0	1	A	B
ZQ	0	1	0	0	Q
ZB	0	1	1	0	B
ZA	1	0	0	0	A
DA	1	0	1	D	A
DQ	1	1	0	D	Q
DZ	1	1	1	D	0

The D input to the mux is the direct data input from the IB bus. This port is used to insert all data into the working registers inside the 2901A data path.

The Q input to the mux is from the Q register. The Q register is a separate file used as an accumulator or holding register. It is loaded from the ALU through a Q shifter (input F) or from its own output via feedback loops (inputs 2Q and Q/2). Input 2Q is enabled for multiplication while input Q/2 is enabled for division. Data in the Q shifter is shifted right or left to perform arithmetic operations. Operation of the Q shifter is controlled by the ALU DST <2:0> field.

The ALU functions are selected by ALU FCN <2:0> from the microword. The ALU function code is defined in Table 5-6.

Table 5-6 ALU Function Code

Mnemonic	ALU FCN 2	1	0	Function
ADD	0	0	0	R plus S
SUBR	0	0	1	S minus R
SUBS	0	1	0	R minus S
OR	0	1	1	R OR S
AND	1	0	0	R AND S
NOTRS	1	0	1	Not R AND S
EXOR	1	1	0	R EXOR S
EXNOR	1	1	1	R EXNOR S

The output select mux selects the RAM port A (mux input A) or the ALU (mux input F) for the output bus. The selection is controlled by the ALU DST <2:0> code from the microword.

The ALU DST code also selects the ALU destination by enabling one (or none) of the three inputs to the RAM shifter and the Q shifter. The ALU DST code is defined in Table 5-7.

Table 5-7 ALU Destination Code

ALU DST			ALU Destination		Output Bus
2	1	Ø	RAM	Q Register	
Ø	Ø	Ø	---	F	F
Ø	Ø	1	---	---	F
Ø	1	Ø	F	---	A
Ø	1	1	F	---	F
1	Ø	Ø	F/2	Q/2	F
1	Ø	1	F/2	---	F
1	1	Ø	2F	2Q	F
1	1	1	2F	---	F

Note that although the ALU DST code selects mux inputs A or F for the output bus, data on the output bus is not gated to the IB bus until the DP Control Logic selects the ALU as a source for the IB bus by asserting EN ALU.

The IB bus always inputs to the D input of the R input mux however the ALU is not an IB bus destination unless the D input is selected by the ALU SRC field.

The ALU has four status outputs: carry out (ALU C), sign bit F3 (ALU N), zero bit F=Ø (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 a zero detect. 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.

5.6.2 Data Manipulation

After data is loaded into the microprocessor, the Q register data can be rotated or shifted left or right by the Q shifter. Likewise, the RAM data can be rotated or shifted left or right by the RAM shifter. 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 4-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.

5.6.3 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 DP to function with full 32-bit carry look-ahead generation. Figure 5-9 illustrates this logic.

Each of the 2901A chips generates both a carry generate output (GEN or G) and a carry propagate output (PROP or P). 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 are combined to output ALU C16. ALU C16 goes to another carry skipper and is combined with the GEN and PROP signals from the bit <31:16> 2901As. The outputs of the second carry skipper are combined to output ALU C (carry flag) to the CS branching logic.

5.7 DP PARITY GENERATION AND CHECKING (Figure 5-10)

The DP parity generation and checking logic receives data from the IB IN bus to perform various parity functions. A flow-thru latch links the IB IN bus to the IB bus thereby making IB bus data available to the parity logic. RBPE is the only DP parity signal not using the parity generation and checking logic (see RBPE; Paragraph 5.7.2).

The only parity check made during an unsolicited CMI operation is on the data transferred over the CIPA bus (CIPA ERROR check). Even this check is not made when the offset address is transferred over the bus (see Paragraph 5.8.1).

5.7.1 Parity Generation and Checking Logic

Data on the IB IN bus (IB IN <31:00>) is divided into bytes and applied to four parity generators. Each generator outputs an odd parity bit (BYTE <3:0> PAR) generated on the associated input byte. BYTE 0 PAR is odd parity for IB IN <07:00>, BYTE 1 PAR is odd parity for IB IN <15:08>, etc.

The parity bits for the two lower bytes (BYTE <1:0> PAR) are XORed to generate a parity bit (LO WD PARITY) for the low word on the IB IN bus (IB IN <15:00>). In a similar manner, the parity bits for the two upper bytes (BYTE <3:2> PAR) are XORed to generate a parity bit (HI WD PARITY) for the high word on the IB IN bus (IB IN <31:16>).

The byte parity bits and word parity bits generated on the IB IN data, are used to perform various parity generation and checking functions as discussed in paragraphs 5.7.3 through 5.7.6.

A wrong parity (WP) bit from the PMCSR is input to the low byte parity generator. The WP bit is used to insert a parity error into the parity logic for maintenance testing purposes. When WP is asserted, the low byte parity generator produces BYTE 0 PAR as an even parity bit instead of odd. The even parity carries through to the LO WD PARITY bit and therefore is effective in all parity word checks.

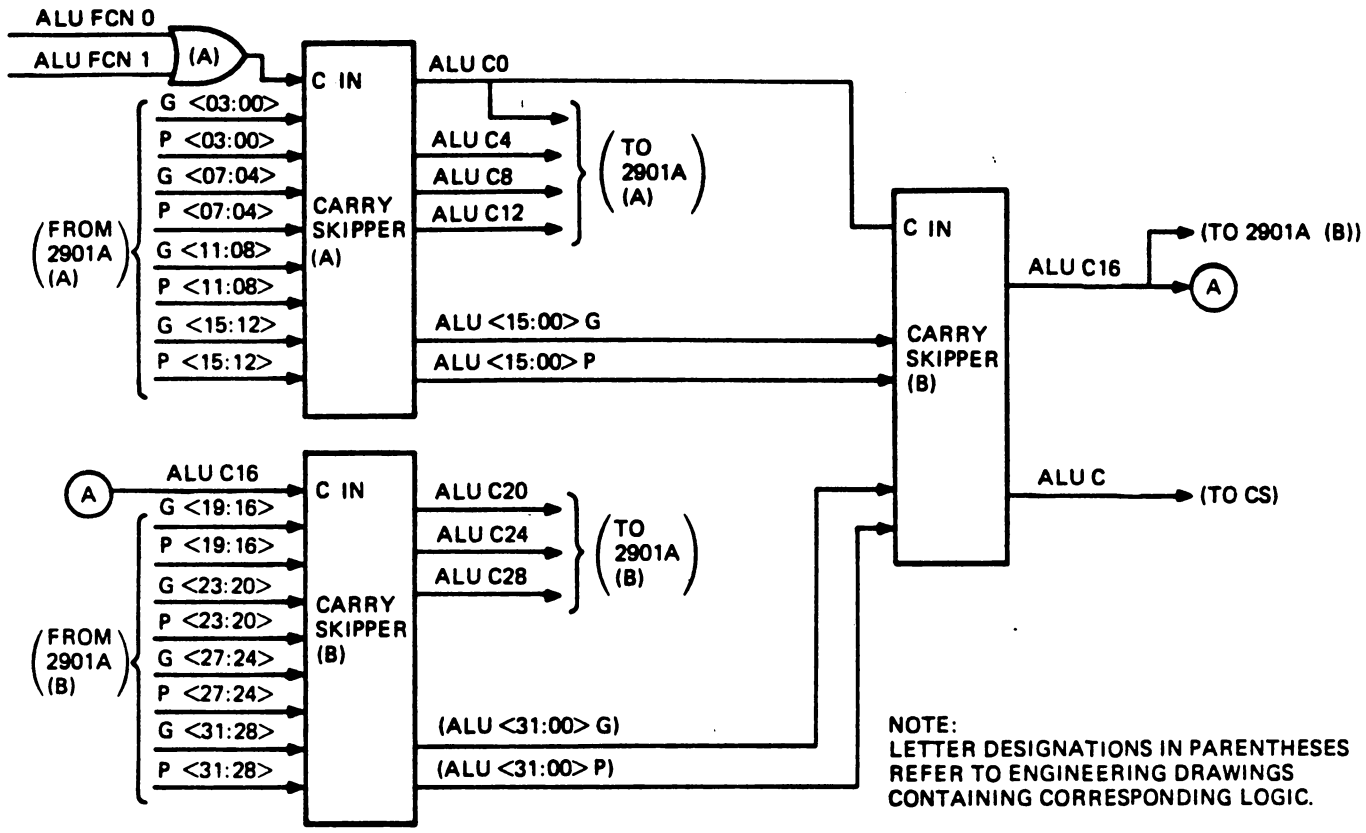


Figure 5-9 Carry Look-Ahead Logic

For byte parity checks, LD PB OUT is asserted (to load the PB OUT register) and gates WP to the other three byte parity generators. This results in even parity being generated on each data byte transferred from the PB OUT register to the PB over the PORT DATA bus.

5.7.2 Receive Buffer Parity Error (RBPE)

RBPE indicates a parity error on data transferred from the PB into the DP. It is the only parity check that does not involve the parity generation and checking logic.

Data bytes from the PB are received over the PORT DATA bus, coupled through a latch, and applied to the PB IN register. The data bytes are also applied to an even parity generator where an RB PAR parity bit is generated for each byte.

An even parity bit (RBUF PAR) is received from the PB along with each input byte. The RBUF PAR bits are compared with the generated RB PAR bits in an XOR gate. If a match is not obtained, a parity error has occurred.

If the data byte was valid data (not undefined residue left in the PB), EN RBPE from the CS will be true. With EN RBPE true, an error output from the XOR gate will assert RBPE.

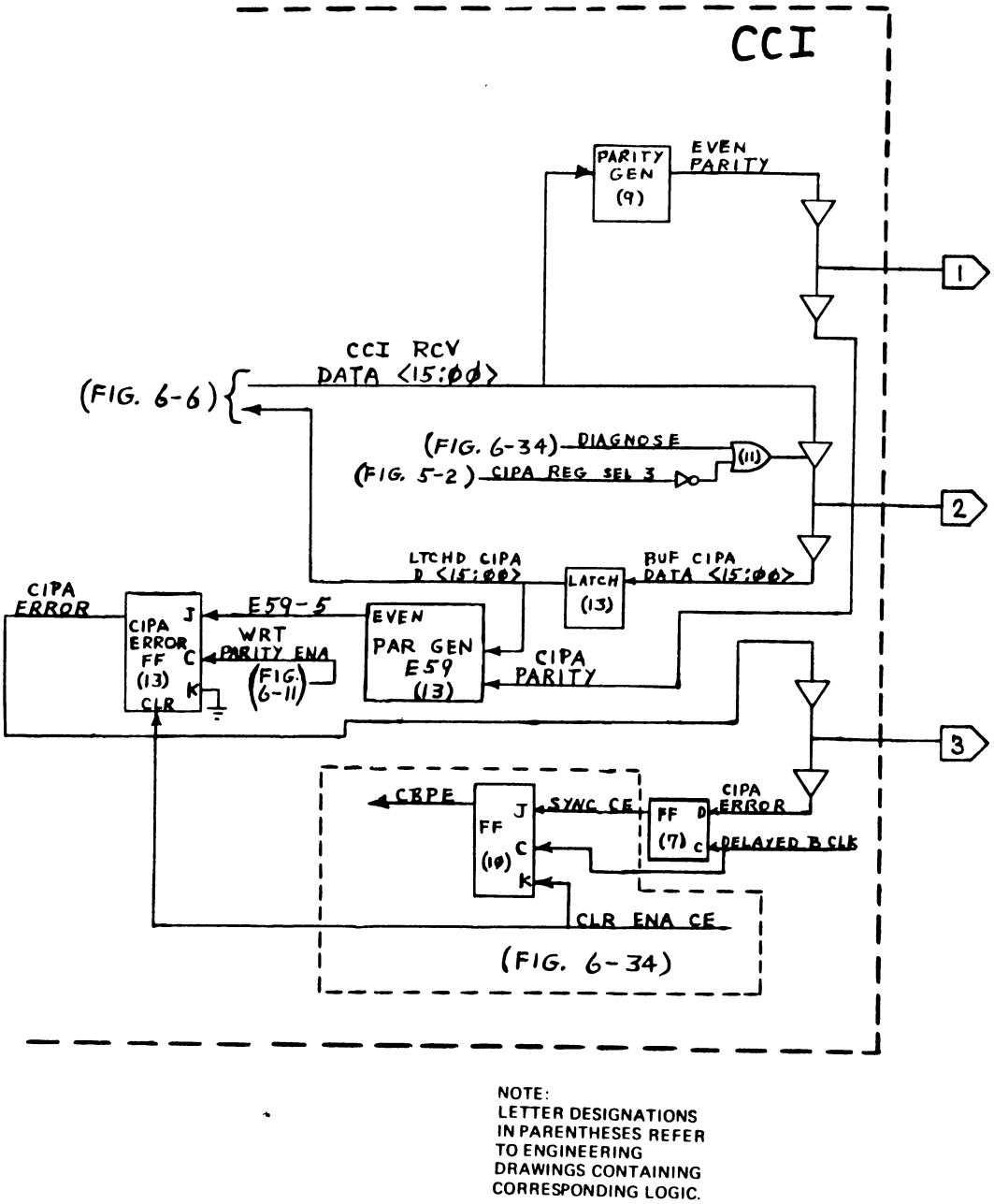
RBPE is applied to a PE OR gate and to the PMCSR register.

5.7.3 PB IN Register Parity Error (PBIR PE) [Output Parity Error (OPE)]

PBIR PE indicates a parity error on data transferred through the PB IN register to the IB bus.

Data bytes from the PORT DATA bus are coupled through a latch and applied to the 32-bit PB IN register. The register is enabled by PB MUX ENA. The PMUX <1:0> code places the input bytes into their proper position within the register.

The RB PAR parity bits (generated on the data bytes input to the PB IN register) are applied to a four-bit parity latch. The latch is enabled by PB MUX EN each time a data byte is input from the PB. The PMUX <1:0> code latches one of the parity bits in each of the four output positions. The four bits latched are the parity bits for the four bytes loaded into the PB IN register. The parity bits are applied from the parity latch to the A inputs of a parity comparator.



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

Figure 5-10 Parity Generation and Checking
(Sheet 1 of 2)

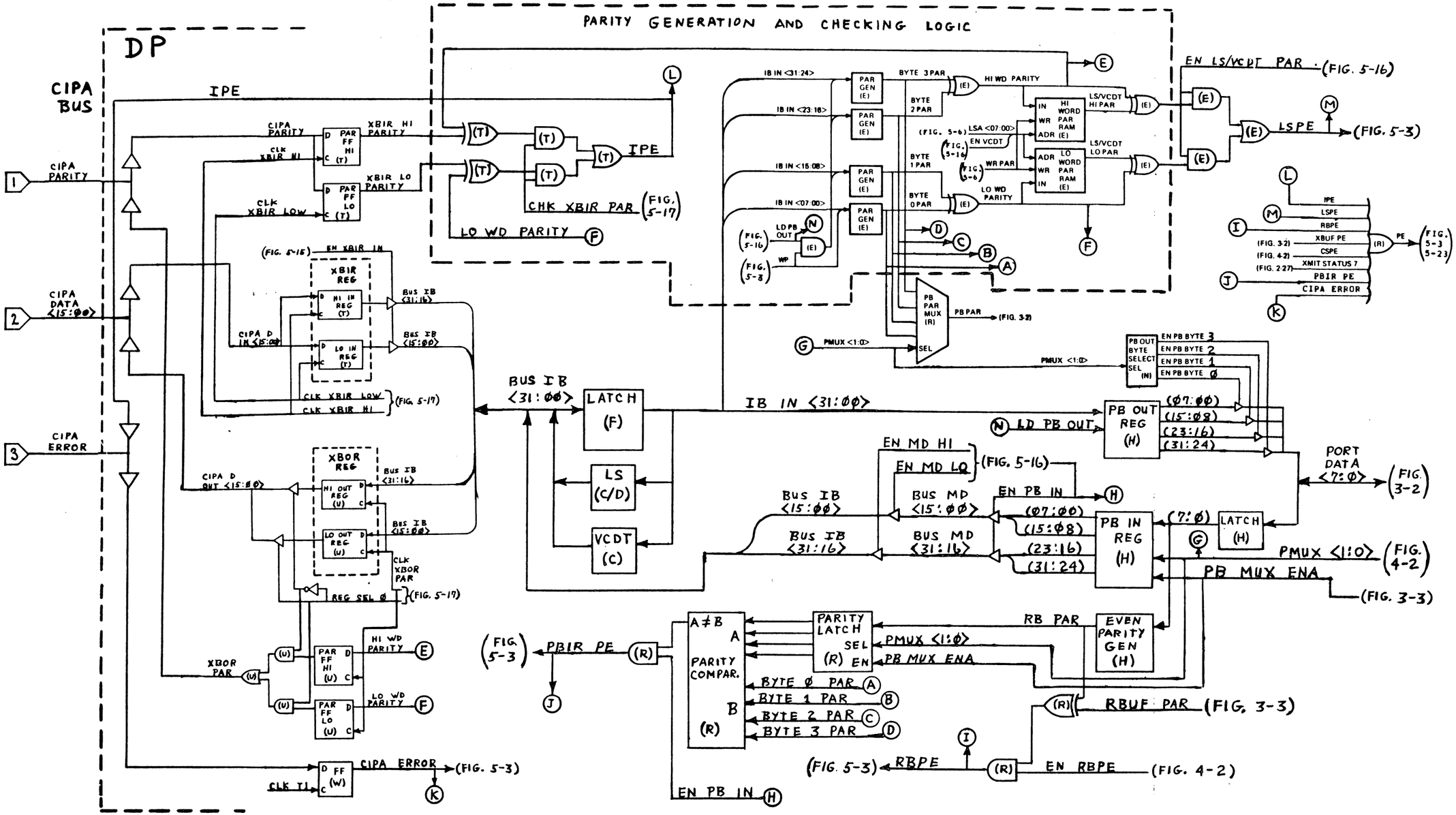


Figure 5-10 Parity Generation and Checking
(Sheet 2 of 2)

If the PB IN register is selected as the source for the IB bus, EN PB IN asserts and gates the data from the PB IN register to the MD bus. From the MD bus, the data is transferred to the IB bus by EN MD LO and EN MD HI. For parity checking purposes, the data is passed from the IB bus to the IB IN bus (through the flow-thru latch) and then to the parity generation and checking logic where byte and word parity bits are generated on the data. The byte parity bits generated (BYTE <3:0> PAR) are applied to the B inputs of the parity comparator.

If the comparator detects a mismatch between the A and the B inputs, the A≠ B output is asserted causing PBIR PE to assert (EN PB IN is true).

PBIR PE is applied to the PE OR gate and the PMCSR register.

5.7.4 CIPA ERROR

CIPA ERROR indicates a parity error on data transferred over the CIPA bus. The data transfer can be in either direction; from the DP to the CCI or from the CCI to the DP. This is the only parity check made during an unsolicited CMI function.

CIPA ERROR is applied to the PE OR gate and the PMCSR register.

A CBPE (CIPA bus parity error) error bit in the CCI configuration register (CNFGR) is also set when a parity error is detected on a CIPA bus data transfer. Thus both modules are warned of a transfer error in either direction.

Parity checking for both directions of data flow is discussed below.

5.7.4.1 DP to CCI Parity Check

The DP to CCI parity check is performed on data transferred from the IB bus, through the XBOR register, over the CIPA bus, and through the CCI input latch. The source of the data on the IB bus makes no difference. Parity will have already been checked on the data from its source to the IB bus.

Data on the IB bus is applied to the parity generation and checking logic (via the latch and the IB IN bus) where high word parity and low word parity bits (HI WD PARITY, LO WD PARITY) are generated. This process is discussed in Paragraph 5.7.1.

The high word and low word parity bits are clocked into high and low parity flip-flops by CLK XBOR PAR. The flip-flop outputs are gated out by REG SEL 0 to become XBOR PAR and then CIPA PARITY on the CIPA bus. REG SEL 0 is initially false to gate out the high word parity bit. It then asserts to gate out the low word parity bit.

The IB bus data (BUS IB <31:00>) is clocked into the XBOR register by CLK XBOR PAR, and then gated out onto the CIPA bus by REG SEL 0. The initially false state of REG SEL 0 gates the high word onto the CIPA bus. When REG SEL 0 asserts, the low word is gated out to the CIPA bus.

Thus, the data high word and its associated high word parity bit input together into the CCI, followed by the data low word and its associated low word parity bit. The data words are passed through an input latch and applied to a parity generator where it is combined with its associated parity bit from the CIPA bus. An error free data transfer results in odd parity being generated. If a data error occurred, the generator produces even parity resulting in the assertion of E59-5 to a CIPA ERROR flip-flop. The CIPA ERROR flip-flop is clocked by WRT PARITY ENA which asserts for every DP to CCI data transfer. Thus the true state of E59-5 causes the assertion of CIPA ERROR. CIPA ERROR returns to the DP over the CIPA ERROR line of the CIPA bus where it sets a CIPA ERROR flip-flop in the DP.

CIPA ERROR on the CIPA bus is also looped back into the CCI where it asserts SYNC CE. SYNC CE sets a CBPE flip-flop which asserts the CBPE bit in the CNFGR register.

5.7.4.2 CCI to DP Parity Check (IPE)

The CCI to DP parity check is performed on data transferred through the CCI output drivers, over the CIPA bus, through the XBIR register to the IB bus, and then through the flow-thru latch to the IB IN bus.

Data on the CCI RCV DATA bus in the CCI is in word format (CCI RCV DATA <15:00>). The data is transferred through the CCI output drivers, over the CIPA bus (CIPA DATA <15:00>), and applied to the XBIR register as CIPA D IN <15:00>).

The data on the CCI RCV DATA bus is also applied to a parity generator where odd parity is generated. The generated parity bits (EVEN PARITY*) are transferred to the DP over the CIPA bus (CIPA PARITY) and applied to high and low parity flip-flops in the DP.

* Odd parity is used. The mnemonic relates to the output pin of the generator chip.

In a data transfer, the first word applied to the XBIR register is a high word. Its associated parity bit is applied to the high word parity flip-flop. CLK XBIR HI loads the high word into the high portion of the XBIR register. CLK XBIR HI also loads the high word parity bit into the high word parity flip-flop asserting XBIR HI PARITY.

The next data transfer over the CIPA bus is the low word and its associated parity bit. CLK XBIR LOW asserts to clock the low word into the low portion of the XBIR register. CLK XBIR LOW also clocks the low word parity bit into the low word parity flip-flop asserting XBIR LO PARITY.

The data longword in the XBIR register is gated out to the IB bus by EN XBIR IN. From the IB bus, the longword is transferred through the flow-thru latch to the IB IN bus and then to the parity generation and checking logic. In the parity generation and checking logic, high word and low word parity bits are generated (HI WD PARITY, LO WD PARITY) and compared (XORed) with the corresponding parity bits from the XBIR parity flip-flops. If the corresponding bits do not match, an error has occurred during the data transfer. In this case, IPE (input parity error) will assert when the CCI/DP Interface Control Logic checks XBIR parity (by asserting CHK XBIR PAR).

IPE is coupled back to the CIPA bus where it asserts CIPA ERROR. CIPA ERROR loops back into the DP where it asserts CIPA ERROR to the PE OR gate and the PMCSR register. IPE is also applied to the PE OR gate, however it is not applied to the PMCSR register.

CIPA ERROR (on the CIPA bus) also sets the CBPE error bit in the CCI configuration register as discussed in Paragraph 5.7.4.1.

5.7.5 Packet Buffer Parity (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 longwords on the IB IN bus are input to the parity generation and checking logic where byte parity bits are generated (four for each longword). The four parity bits (BYTE <3:0> PAR) are applied to a PB parity mux. The mux select code (PMUX <1:0>) selects the output parity bit 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. Hence the parity bit on the PB PAR line is for the data byte on the PORT DATA bus.

5.7.6 Local Store Parity Error (LSPE)

LSPE indicates a parity error on data written into LS or the VCDT from the IB IN bus, or read out of LS or the VCDT onto the IB bus.

A data longword being written into LS or the VCDT* from the IB IN bus, is also input into the parity generation and checking logic where high word and low word parity bits are generated. The high word and low word parity bits (HI WD PARITY, LO WD PARITY) are respectively written into a high word parity RAM and a low word parity RAM. The two RAMs are addressed by LSA <07:00> from the LS/VCDT address selection logic thereby writing the parity bits at the same address as the data being written into LS or the VCDT. EN VCDT is applied to the parity RAMs as the most significant address bit. If the VCDT is being written, EN VCDT is true thereby writing the VCDT parity bits in a location within the RAMs separate from the LS parity bits. The RAM write strobe (WR PAR) is obtained from the write strobe logic in the LS/VCDT address selection logic (Figure 5-6).

* The VCDT is only 16 bits wide. It receives inputs from the low word half of the IB IN bus and outputs to the low word half of the IB bus. When writing the VCDT, the high word on the IB IN bus is all zeros. When reading out the VCDT, zeros are placed on the high word of the IB bus. Thus the high word parity function operates normally with parity being generated and checked on an all zero 16-bit word.

When the LS or the VCDT is read, the data is output onto the IB bus. The flow-thru latch couples the data from the IB bus to the IB IN bus where it inputs into the parity generation and checking logic. The logic generates high word and low word parity bits on the input data.

The LS/VCDT address (LSA <7:0>) associated with the read operation, addresses the high word parity RAM and the low word parity RAM thereby accessing the parity bits stored when the LS/VCDT data (now being read) was written. Write strobe WR PAR is false thereby enabling parity RAM outputs LS/VCDT HI PAR and LS/VCDT LO PAR.

The high word and low word parity bits (HI WD PARITY, LO WD PARITY) generated from the read data, are compared respectively with LS/VCDT HI PAR and LS/VCDT LO PAR from the parity RAMs. If the compared bits do not match, a data error occurred during the writing or reading of the LS/VCDT RAMs. In this case LSPE will assert when the DP Control Logic checks the LS/VCDT parity (by asserting EN LS/VCDT PAR).

LSPE is applied to the PE OR gate and the PMCSR register.

5.7.7 Parity Error (PE)

PE is an OR function of eight port parity error bits. The bits comprise the five parity error bits discussed in this section (RBPE, PBIR PE, CIPA ERROR, IPE, LSPE), two from the PB, and one from the link module. XBUF PE from the PB transmit channel and CSPE from the control store RAMs are received from the PB module. XMIT STATUS 7 (TDATA PARITY ERROR) is received from the link transmit channel.

PE and seven of the eight parity error signals that assert PE are applied to the PMCSR register (see Figure 5-3 and Table 5-4). The eighth parity error signal (IPE) is not part of the PMCSR register as an IPE parity error sets the CIPA ERROR bit (Paragraph 5.7.4.2).

PE is also applied to the error/interrupt logic where it initiates an interrupt to the CPU.

5.8 UNSOLICITED CMI REQUESTS

An unsolicited CMI request is a read or a write of a port register that was initiated by the host CPU and not by the port microcode. The register may be located in the CCI or the DP. The configuration register (CNFGR) is the only CCI register accessed by an unsolicited request. All other registers accessed by unsolicited requests are located in the DP.

The DP is in the suspend mode of operation while the unsolicited request is being executed (see Paragraph 5.11.2.4). In the suspend mode of operation, the microsequencer clock is stopped and the microcode branch flags and status information are saved. Thus the port can resume microcode operation after the unsolicited operation is completed.

A brief overview of unsolicited CMI operations is shown in Figure 5-11. The host CPU issues a request to the DP via the CCI. The host then loads the address of the target register into the address offset register in the CCI. If a write operation is being requested, the write data is loaded into the Receive Write Data Register.

When the DP receives the unsolicited request, it suspends microcode operation. The DP then reads the target address from the address offset register in the CCI and loads it into the XBUS LSA register in the DP.

If a write function was requested, the DP reads the XBUS LSA register, decodes the output to select the target register, and enables the selected register for a write operation. The DP then reads the write data from the Receive Write Data Register in the CCI and loads the data into the selected register. Control of the port is then returned to the port microcode which resumes operation at the address frozen during the suspend mode.

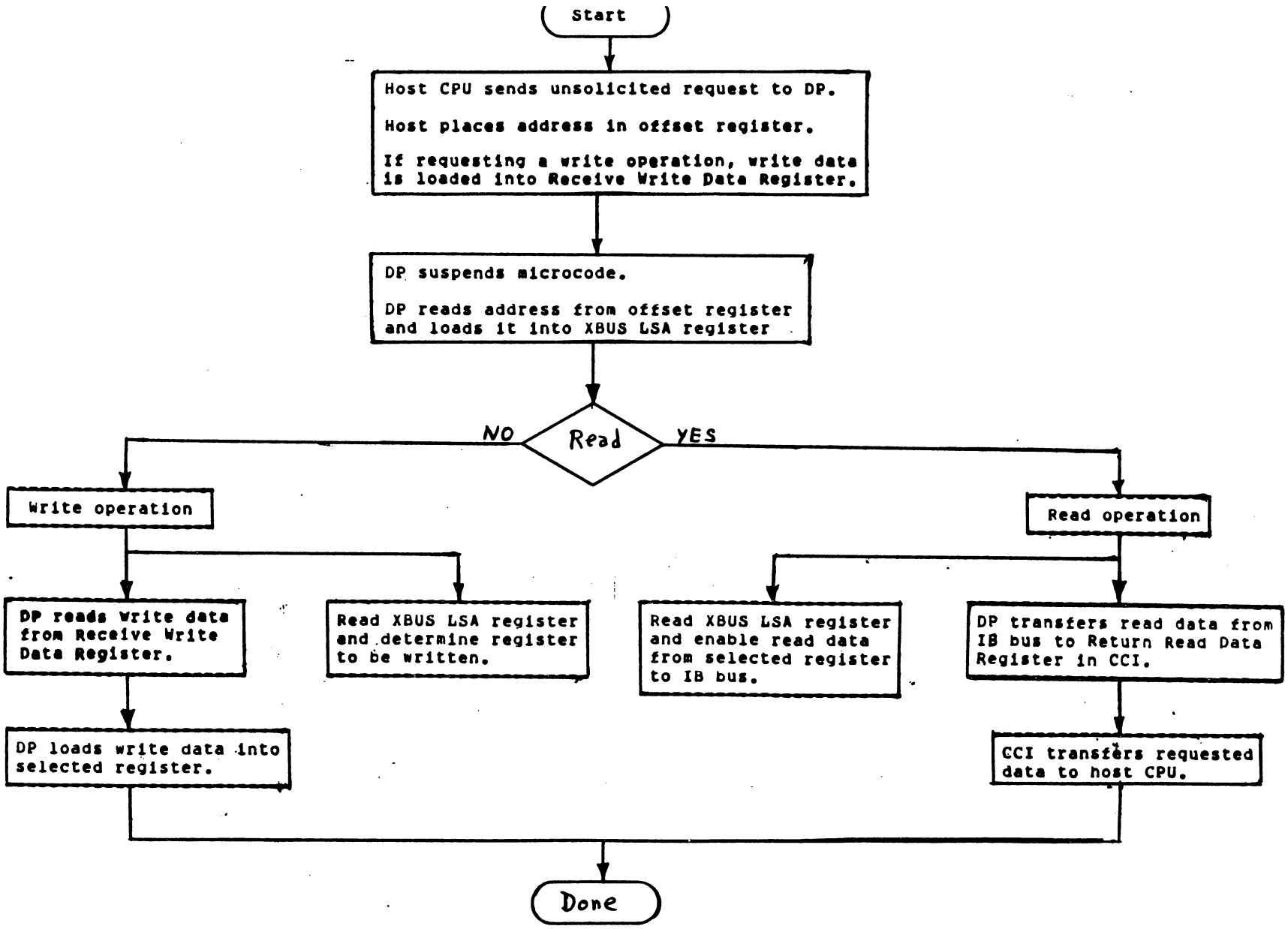


Figure 5-11 Unsolicited CMI Operations -- Simplified Flow Diagram

If a read function was requested, the DP reads the XBUS LSA register, decodes the output to select the target register, and gates the data from the selected register to the IB bus. The DP then transfers the read data from the IB bus to the CCI and loads the data into the Return Read Data Register. The host CPU takes the read data from the Return Read Data Register via the CMI. Control of the port is then returned to the port microcode which resumes operation at the address frozen during the suspend mode.

5.8.1 Starting An Unsolicited Sequence (Figure 5-12)

An unsolicited CMI operation is requested by the host CPU via the CMI bus and the CCI. CIPA REQUEST SYNC and CIPA READ SYNC are asserted to the CCI/DP Interface Control Logic from the CCI. CIPA REQUEST SYNC initiates the unsolicited sequence. CIPA READ SYNC specifies the read operation necessary to read the CCI offset register to determine which register is to be accessed.

The CCI/DP Interface Control Logic asserts GRANT UNSOL causing CIPA GRANT to assert to the CCI. CIPA GRANT indicates to the CCI that the unsolicited request is being serviced.

The CCI/DP Interface Control Logic also asserts SUSPEND SEQ to stop the port microcode from running and place the port into the suspend mode of operation. SUSPEND SEQ also forces the LSA mux to select XBUS LSA <07:00> for the LS/VCDT address (Paragraph 5.4.1).

In addition, the Control Logic asserts the REG SEL <3:0> code to the CCI specifying the address offset register as the register to be read.

In the CCI, the REG SEL <3:0> code is decoded to assert RD ADDR OFFSET. RD ADDR OFFSET enables the data out of the address offset register. If the register to be accessed is in the DP, the contents of the offset register is returned to the DP over the CIPA DATA lines on the CIPA bus.

Twelve bits (CIPA D IN <11:00>) of the 16-bit data word received from the CCI specifies the address of the register to be accessed for the unsolicited operation. The address is clocked into the XBUS LSA register (see Figure 5-13) by LD XBUS LSA from the CCI/DP Interface Control Logic.

In passing from the CCI to the XBUS LSA register in the DP, the offset address does not route through the XBIR register. Hence the offset address is not checked for parity.

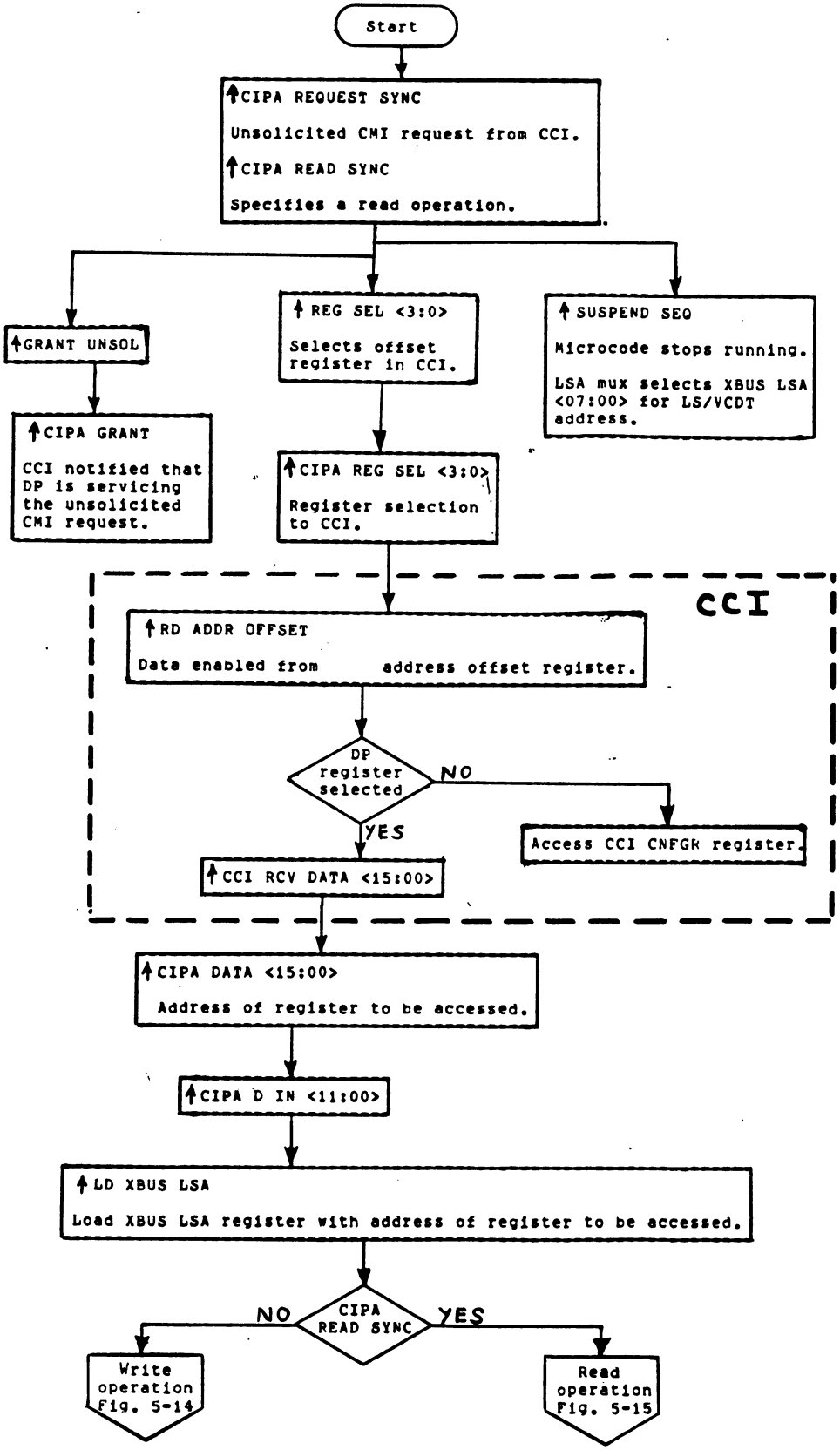
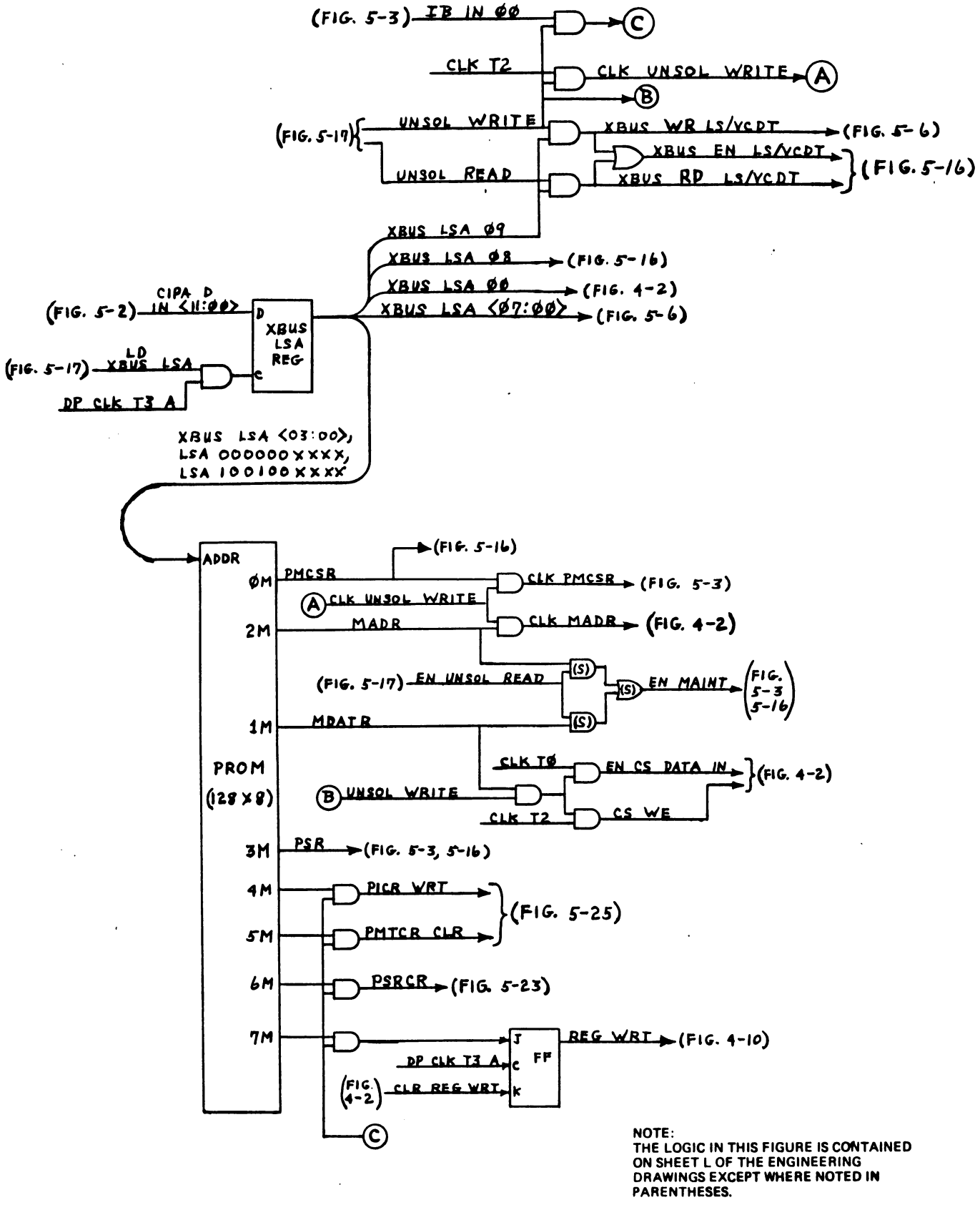


Figure 5-12 Starting an Unsolicited Operation



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

Figure 5-13 Unsolicited CMI Request Logic

The CCI/DP Interface Control Logic now checks the CIPA READ SYNC line from the CCI to determine if the unsolicited request is a read or a write. If the request is for a write of a DP location (CIPA READ SYNC false), continue with Paragraph 5.8.2 and Figure 5-14. If the request is for a read of a DP location (CIPA READ SYNC true), continue with Paragraph 5.8.3 and Figure 5-15. Figure 5-13 illustrates the logic associated with the write and read sequences and should be referenced throughout both discussions.

5.8.2 Unsolicited Write Sequence (Figures 5-13 and 5-14)

An unsolicited write sequence involves obtaining the write data from the CCI, selecting the register to be written, and generating the clock to load the write data into the selected register.

5.8.2.1 Obtaining the Write Data -- The CCI/DP Interface Control Logic asserts the REG SEL <3:Ø> code to the CCI to access the Receive Write Data Register. The code is coupled to the CCI over the CIPA bus (CIPA REG SEL <3:Ø>). In the CCI the REG SEL input is decoded to assert RD RCV WD REG HI. RD RCV WD REG HI enables the high word from the Receive Write Data Register onto the CCI RCV DATA <15:ØØ> bus and then onto the CIPA bus (CIPA DATA <15:ØØ>). CLK XBIR HI is asserted by the CCI/DP Interface Control Logic to load the data high word into the high word portion of the XBIR register. The high word portion is selected by the false state of REG SEL Ø (see Paragraph 5.3.4).

The CCI/DP Interface Control Logic then asserts REG SEL Ø. The assertion of REG SEL Ø alters the REG SEL code so as to select the low word from the Receive Write Data Register. The REG SEL <3:Ø> code is again dispatched to the CCI where it asserts RD RCV WD REG LO to retrieve the write data low word. The write data low word is coupled from the Receive Write Data Register, over the CIPA bus, and clocked into the low word portion of the XBIR register by CLK XBIR LOW.

The CCI/DP Interface Control Logic then asserts EN UNSOL WRITE to apply the write data to the selected register and to generate the register clock pulse.

EN UNSOL WRITE asserts EN XBIR IN (Figure 5-16) to gate the write data from the XBIR register to the IB bus. The write data is coupled from the IB bus, through the flow-thru latch to the IB IN bus where it is available to the selected register.

The next DP CLK T3 A after the assertion of EN UNSOL WRITE, causes UNSOL WRITE to go true (Figure 5-17). UNSOL WRITE enables the register write clock if the selected register is the PMTCR, PSRCR, PICR, REG WRT, or the LS/VCDT area. This is discussed below in the register selection discussion.

The next CLK T2 pulse after the assertion of UNSOL WRITE, causes CLK UNSOL WRITE to go true. CLK UNSOL WRITE enables the register write clock if the selected register is the PMCSR or the MADR. This is discussed below in the register selection discussion.

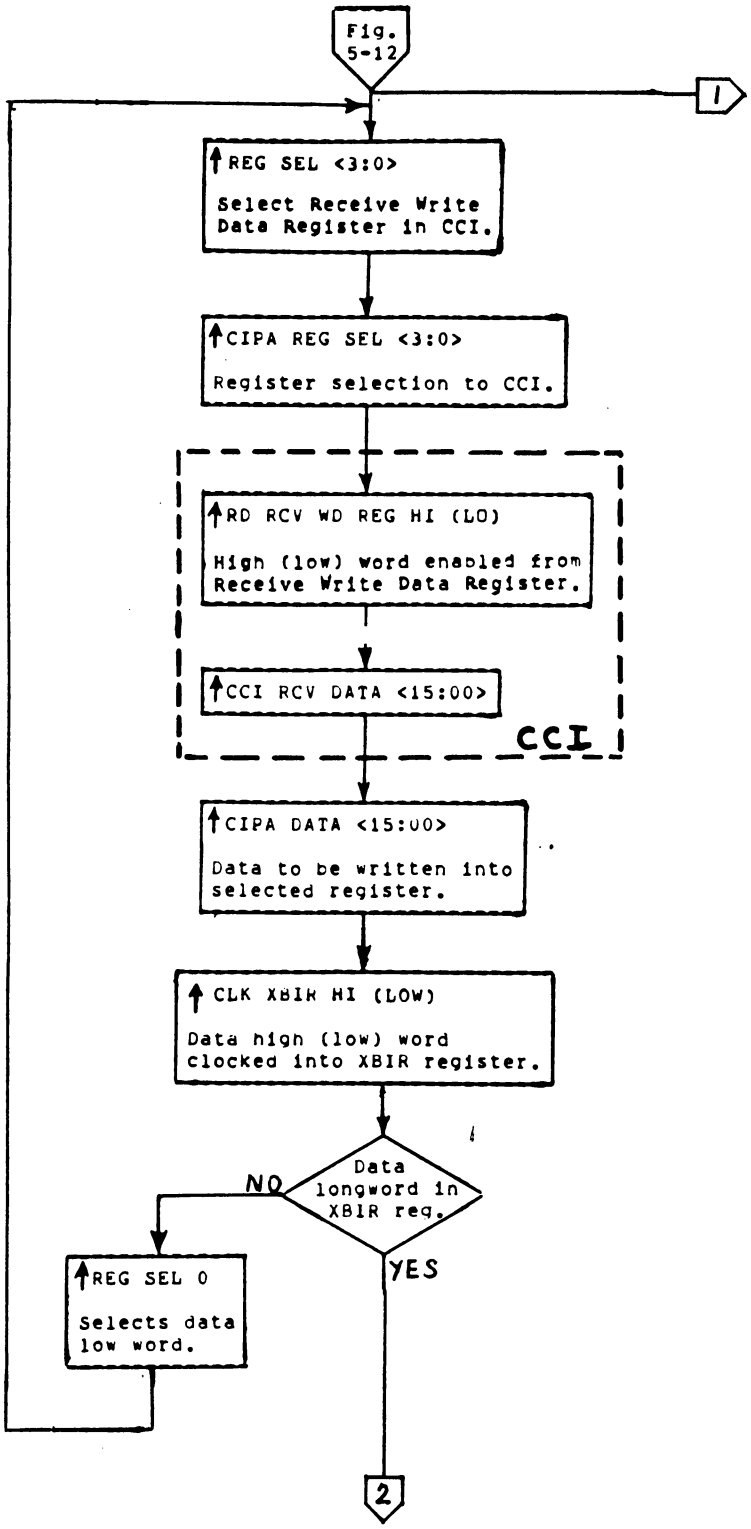


Figure 5-14 Unsolicited CMI Write Operation Flow Diagram (Sheet 1 of 2)

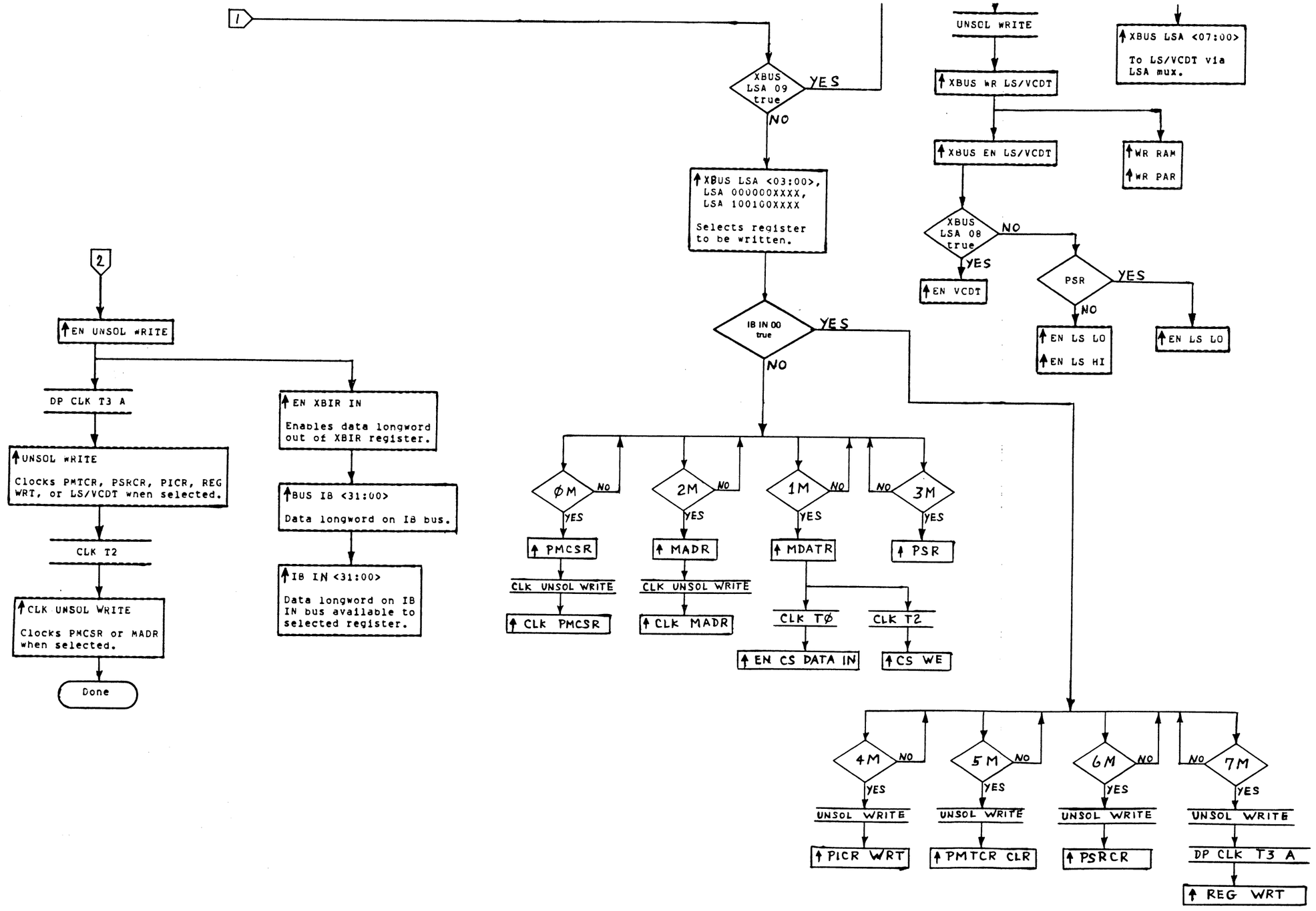


Figure 5-14 Unsolicited CMI Write Operation Flow Diagram (Sheet 2 of 2)

5.8.2.2 Register Selection -- The XBUS LSA register contains the address of the selected register to be written (Paragraph 5.8.1).

If XBUS LSA Ø9 from the register is true, the LS/VCDT is selected as the area to be written.

In this case, XBUS LSA <Ø7:ØØ> from the register is applied to the LS/VCDT address selection mux where it is selected as the LS/VCDT address (Paragraph 5.4.1).

When UNSOL WRITE asserts, XBUS WR LS/VCDT asserts to the LS/VCDT write strobe logic where the WR RAM and the WR PAR write strobes are generated (Paragraph 5.4.2).

XBUS WR LS/VCDT also asserts XBUS EN LS/VCDT to the DP Control Logic where XBUS LSA Ø8 from the XBUS LSA register is examined. The true state of XBUS LSA Ø8 causes EN VCDT to assert thereby specifying the VCDT as the selected area to be written. If XBUS LSA Ø8 is false, the LS is the selected area to be written. In this case EN LS HI and EN LS LO assert to enable the LS low and LS high sections respectively.

Note that if the LS access is to the PSR*, only EN LS LO asserts. The upper portion of the PSR is all zeros except bit 31 (MTE) which is read only (see Paragraph 5.5.3).

* Port Status Register; a software register located in LS.

If XBUS LSA Ø9 is false, the LS/VCDT is not selected as the area to be written. The selection is made by a 128 x 8 PROM which outputs the enabling signal for the selected area. The PROM is addressed by XBUS LSA <Ø3:ØØ>, LSA ØØØØØØXXXX, and LSA 1ØØ1ØØXXXX from the XBUS LSA register. PROM outputs are available at terminals ØM through 7M inclusive.

The state of write bit IB IN ØØ determines which of the PROM outputs are to be used. IF IB IN ØØ is false, PROM outputs ØM, 1M, 2M, or 3M will designate the area to be written. If IB IN ØØ is true, PROM output 4M, 5M, 6M, or 7M will designate the area.

The case of IB IN ØØ false is considered first.

When CLK UNSOL WRITE asserts (Paragraph 5.8.2.1), it samples PROM outputs ØM and 2M. If PROM output ØM (PMCSR) is true, the PMCSR is selected for the write operation and CLK PMCSR asserts to load data into the PMCSR from the IB IN bus (Figure 5-3).

If PROM output 2M (MADR) is true, the MADR is selected for the write operation and CLK MADR asserts to the CS where write data is loaded into the MADR from the IB IN bus.

If PROM output 01 (MDATR) is true, the MDATR register is selected for the write operation. In this case, EN CS DATA IN and CS WE are asserted by CLK T0 and CLK T1 respectively. EN CS DATA IN gates the write data into the CS from the IB IN bus while CS WE strobes the write data into the CS.

PROM output 3M is asserted when the PSR is selected. The PSR is located in the LS area, therefore the LS/VCDT has actually been selected as the destination for the write data (XBUS LSA 09 true). The assertion of PSR from the PROM supplements the LS/VCDT selection by specifying the PSR. With PSR true, only the low section of the LS is enabled as discussed earlier in this paragraph.

If IB IN 00 is true, PROM outputs 4M, 5M, 6M, and 7M are sampled by UNSOL WRITE. When UNSOL WRITE asserts it clocks the asserted PROM output to perform the function described below.

If output 4M is true, PICR WRT (port initialize control register) asserts thereby negating UNINIT and taking the port out of the uninitialized state (Paragraph 5.12.2.2).

If output 5M is true, PMTCR CLR (port maintenance timer control register) is asserted to reset the boot timer logic thereby extending the boot timeout period (Paragraph 5.12.2.3).

If output 6M is true, PSRCR (port status release control register) is asserted to the interrupt logic where it resets the maintenance interrupt flag (MIF) in the PMCSR (Paragraph 5.12.1).

If output 7M is true, REG WRT asserts (after the next DP CLK T3 A pulse) 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.

5.8.3 Unsolicited Read Sequence (Figures 5-13 and 5-15)

An unsolicited read sequence involves selecting the register to be read, reading the register out to the IB bus, and transferring the read data from the IB bus to the CCI.

The CCI/DP Interface Control Logic asserts EN UNSOL READ which enables the read data out of the selected register if the selected register is the PMCSR, the MDATR, or the MADR. This is discussed below in the register selection discussion.

The assertion of EN UNSOL READ causes UNSOL READ to assert (Figure 5-17). UNSOL READ enables the read data out of the selected register if the selected register is in the LS/VCDT area. This is discussed below in the register selection discussion.

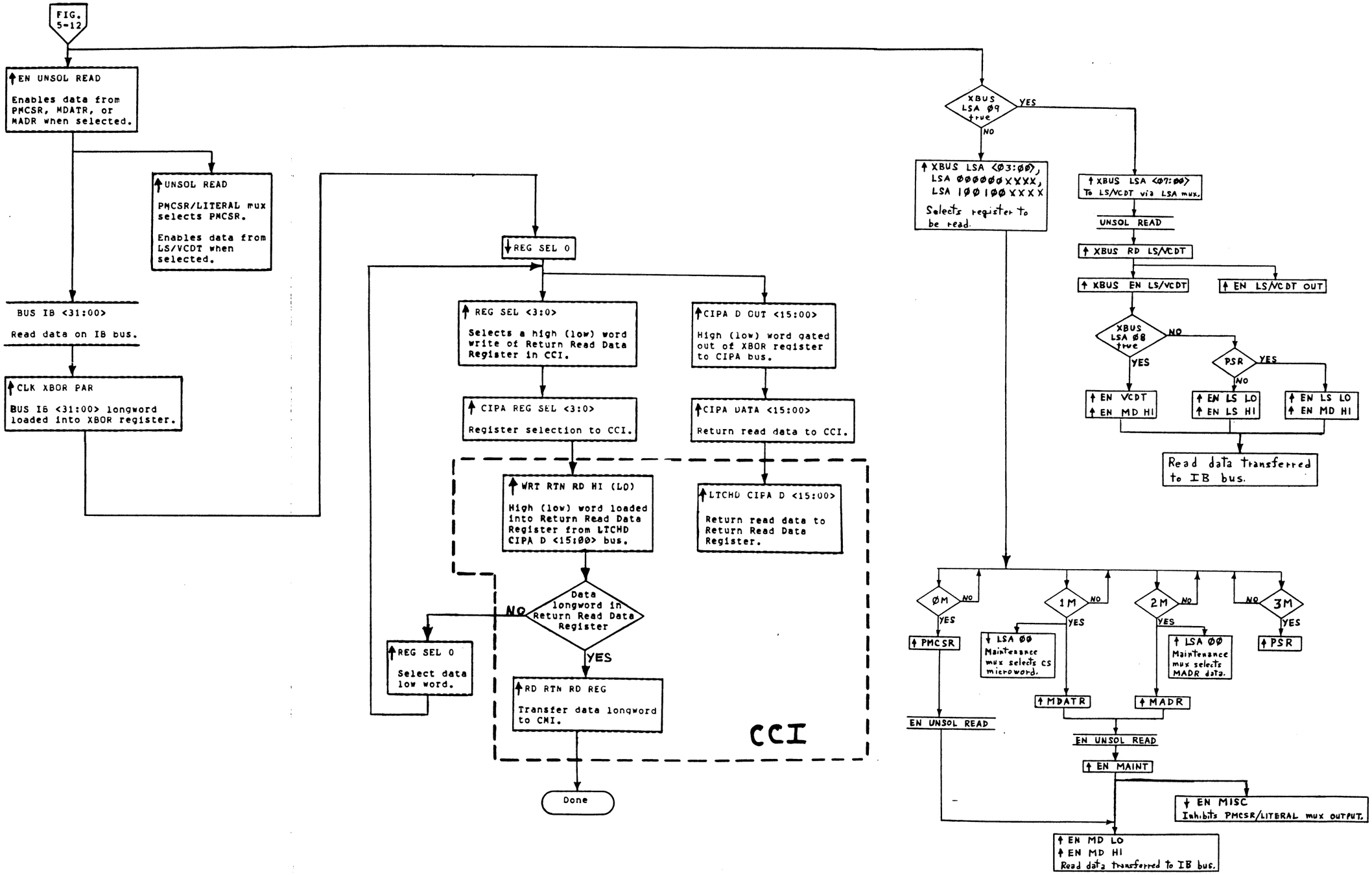


Figure 5-15 Unsolicited CMI Read Operation Flow Diagram

5.8.3.1 Register Selection -- The XBUS LSA register contains the address of the selected register to be read (Paragraph 5.8.1).

If XBUS LSA 09 from the register is true, the LS/VCDT is selected as the area to be read.

In this case, XBUS LSA <07:00> from the register is applied to the LS/VCDT address selection mux where it is selected as the LS/VCDT address (Paragraph 5.4.1).

When UNSOL READ asserts, XBUS RD LS/VCDT goes true causing EN LS/VCDT OUT to assert. EN LS/VCDT OUT gates the output from the VCDT and from both sections of the LS onto the IB bus when the VCDT or the LS are enabled.

The assertion of XBUS RD LS/VCDT also causes XBUS EN LS/VCDT to assert to the DP Control Logic where it enables either the VCDT or the LS depending on the state of XBUS LSA 08 from the XBUS LSA register.

If XBUS LSA 08 is true, EN VCDT asserts to enable the VCDT. The VCDT RAM is only 16-bits wide and outputs onto the low word portion of the IB bus. Hence, EN MD HI asserts (along with EN VCDT) to enable all zeros onto the high word portion of the IB bus via the MD bus (Paragraph 5.5).

If XBUS LSA 08 is false, EN LS HI and EN LS LO assert to enable both the high and low sections of LS onto the IB bus. Note that if the LS access is to the PSR, EN MD HI asserts in place of EN LS HI. As discussed in Paragraph 5.8.2.2, only the lower 16 bits of the PSR are written into LS. When the PSR is read, the 16 bits are read out of the low section of LS onto the low word portion of the IB bus. The upper 16 bits (15 zeros and the MTE bit) are supplied to the IB bus from the MD bus (Paragraph 5.5.3).

If XBUS LSA Ø9 is false, the LS/VCDT is not selected as the area to be read. Instead, the selection is made by a 128 x 8 PROM whose output enables the read data from the selected register. The PROM is addressed by XBUS LSA <Ø3:ØØ>, LSA ØØØØØØXXXX, and LSA 1ØØ1ØØXXXX from the XBUS LSA register. PROM outputs used in accessing read locations are at PROM terminals ØM, 1M, 2M, and 3M.

If output 0M is true, PMCSR asserts to the DP Control Logic. The assertion of EN UNSOL READ causes PMCSR to assert EN MD HI and EN MD LO to gate the PMCSR data to the IB bus.

If output 1M is true, MDATR asserts indicating a read of the maintenance data register. XBUS LSA 00 from the XBUS LSA register is negated and applied to the maintenance mux in the CS causing it to select the microword from the CS RAMs.

When EN UNSOL READ asserts, the true state of MDATR causes EN MAINT to assert and EN MISC to negate. EN MAINT gates the output from the maintenance mux onto the MD bus while the negation of EN MISC isolates the output of the PMCSR/LITERAL mux from the MD bus (Figure 5-3). EN MAINT also causes the DP Control Logic to assert EN MD HI and EN MD LO to gate the MDATR data (the microword) from the MD bus to the IB bus.

If output 2M is true, MADR asserts indicating a read of the maintenance address register. XBUS LSA 00 from the XBUS LSA register is asserted causing the maintenance mux in the CS to select the data from the MADR register.

When EN UNSOL READ asserts, the true state of MADR causes EN MAINT to assert and EN MISC to negate. EN MAINT gates the output from the maintenance mux onto the MD bus while the negation of EN MISC isolates the output of the PMCSR/LITERAL mux from the MD bus. EN MAINT also causes the DP Control Logic to assert EN MD HI and EN MD LO to gate the MADR data from the MD bus to the IB bus.

If output 3M is true, PSR asserts indicating a read of the PSR register in the LS. In this case the LS/VCDT has actually been selected as the area to be read (XBUS LSA 09 true). The assertion of PSR from the PROM, supplements the LS/VCDT selection by specifying the PSR. A read of the PSR was described earlier in this paragraph when the LS/VCDT area was discussed.

5.8.3.2 Transferring the Read Data to the CCI -- The read data taken from the selected register is now on the IB bus as BUS IB <31:00>). The CCI/DP Interface Control Logic will initiate a transfer of the read data from the IB bus to the Return Read Data Register in the CCI (and then to the CMI).

The CCI/DP Interface Control Logic asserts CLK XBOR PAR to load the data longword from the IB bus into the XBOR register. The negated state of REG SEL 0 gates the high word from the XBOR register out to the CIPA BUS as CIPA DATA <15:00>. In the CCI the data high word is applied to the Return Read Data Register as LTCHD CIPA D <15:00>.

The CCI/DP Interface Control Logic also asserts the REG SEL <3:0> code for a high word write of the CCI Return Read Data Register. The register select code is transferred to the CCI over the CIPA bus as CIPA REG SEL <3:0>.

The register select code is input to the CCI and applied to a write decoder. In response to the input REG SEL code, the write decoder outputs WRT RTN RD HI to load the read data (LTCHD CIPA D <15:00>) into the high word section of the Return Read Data Register.

The CCI/DP Interface Control Logic then asserts REG SEL 0 which gates the low word from the XBOR register onto the CIPA bus. The low word is transferred to the CCI over the CIPA bus where it is applied to the Return Read Data Register as LTCHD CIPA D <15:00>.

Asserting REG SEL 0 altered the REG SEL code to specify a low word write of the Return Read Data Register. The REG SEL code is transferred to the CCI where it is applied to the write decoder. The write decoder responds to the altered REG SEL code by outputing WRT RTN RD LO to the Return Read Data Register where it loads the read data into the low word section of the register.

With the requested read data in the Return Read Data Register, the CCI asserts RD RTN RD REG to initiate a data transfer of the requested data to the CMI.

5.9 DP CONTROL LOGIC (Figure 5-16)

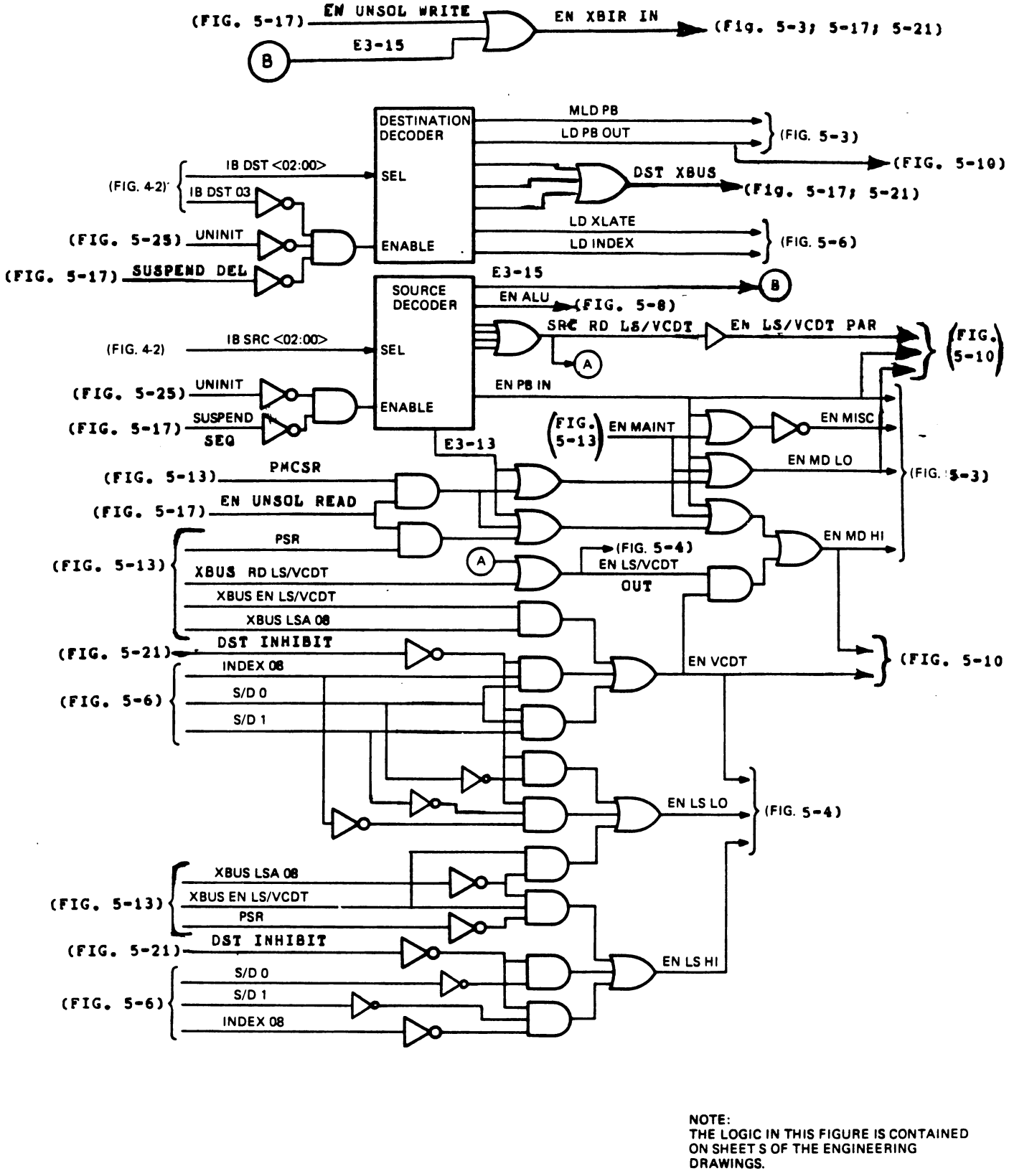
The DP Control Logic generates the commands and enabling signals controlling the data flow within the DP. The control logic receives its commands from the CS microword except during unsolicited CMI operations when the microcode is suspended and control shifts to the host CPU. The commands received from the microword specify the source and destination for the IB bus.

5.9.1 IB Bus Destination

The IB DST <03:00> field from the microword selects the destination for the data on the IB bus. Table 5-8 lists the IB DST code and the destinations it selects.

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 5-8). The remaining three bits (IB DST <02:00>) are decoded to select the destinations shown in the table. Note that if the port is in the uninitialized state (UNINIT true), or an unsolicited CMI operation is in progress (SUSPEND DEL true), the decoder is disabled and the destination of the IB bus is selected by the host via the CCI module and the XBUS LSA register (Paragraph 5.8).

Decoder outputs LD INDEX or LD XLATE are asserted when the LS or the VCDT is to be the IB bus destination. The selection process takes two cycles of microcode. During the first microcycle (when IB DST 03 = 0), LD INDEX (or LD XLATE) loads the index register (or the translate register) with the LS or VCDT address (Figure 5-6). The selection between the LS or the VCDT is made during the second microcycle when IB DST 03 = 1.



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

Figure 5-16 DP Control Logic

Table 5-8 IB DST Code

IB DST				Destination	S/D		LS/VCDT Address Source
3	2	1	Ø		1	Ø	
Ø	Ø	Ø	Ø	No operation	-	-	-------
Ø	Ø	Ø	1	LD INDEX	-	-	-------
Ø	Ø	1	Ø	LD XLATE	-	-	-------
Ø	Ø	1	1	DST XBUS (XMIT file)	-	-	-------
Ø	1	Ø	Ø	DST XBUS (byte mask register)	-	-	-------
Ø	1	Ø	1	DST XBUS (command/ address reg.)	-	-	-------
Ø	1	1	Ø	LD PB OUT	-	-	-------
Ø	1	1	1	MLD PB	-	-	-------
1	Ø	Ø	Ø	Not used	-	-	-------
1	Ø	Ø	1	" "	-	-	-------
1	Ø	1	Ø	" "	-	-	-------
1	Ø	1	1	" "	-	-	-------
1	1	Ø	Ø	LS	Ø	Ø	LITERAL
1	1	Ø	1	LS or VCDT*	Ø	1	Index Register
1	1	1	Ø	LS	1	Ø	Translate Register
1	1	1	1	VCDT	1	1	LITERAL

*Depending on INDEX Ø8; INDEX Ø8 = Ø, LS selected
INDEX Ø8 = 1, VCDT selected

The next three codes from the destination decoder assert DST XBUS. This output is asserted when the CCI module is selected as the IB bus destination. The three possible destinations within the CCI are the XMIT file, the byte mask register, and the command/address registers. The DST XBUS output is applied to the CCI/DP Interface Control Logic along with the DST <1 · bits. The CCI/DP Interface Control Logic responds to the DST <1:Ø> code by asserting the REG SEL <3:Ø> code specifying the selected destination (Paragraph 5.1Ø.1).

Decoder output LD PB OUT is asserted when the PB OUT register is selected as the IB bus destination. LD PB OUT loads the PB OUT register (via the IB IN bus) for output onto the PORT DATA bus (Figure 5-3).

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 <Ø3:Ø2> = 1:1, the LS/VCDT area is selected as the IB bus destination. The IB DST code is converted into an S/D (source/destination) code generated to control the LS/VCDT address selection in the LS/VCDT address selection logic (Paragraph 5.4.1). The S/D code is listed in Table 5-8 along with the address sources that it selects. The S/D code also enables the LS or VCDT as an IB bus destination as shown in Table 5-8 and described in Paragraph 5.9.3.

5.9.2 IB Bus Source

The IB SRC <Ø2:ØØ> field from the microword selects the data source for the IB bus. Table 5-9 lists the IB SRC code and the sources it selects.

Table 5-9 IB SRC Code

IB SRC			IB Source	S/D		LS/VCDT Address Source
2	1	Ø		1	Ø	
Ø	Ø	Ø	LS	Ø	Ø	LITERAL
Ø	Ø	1	LS or VCDT*	Ø	1	Index Register
Ø	1	Ø	LS	1	Ø	Translate Register
Ø	1	1	VCDT	1	1	LITERAL
1	Ø	Ø	EN PB IN	-	-	----
1	Ø	1	E3-13 (LITERAL)	-	-	----
1	1	Ø	EN ALU	-	-	----
1	1	1	E3-15 (EN XBIR IN)	-	-	----

* Depending on INDEX Ø8; Ø = LS, 1 = VCDT

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 CMI operation is in progress (SUSPEND SEQ true), the decoder is disabled and the source for the IB bus is selected by the host via the CCI module and the XBUS LSA register (Paragraph 5.8).

The first four codes listed in Table 5-9 assert SRC RD LS/VCDT thereby selecting the LS or the VCDT as the IB bus source. The IB SRC code is converted into a S/D (source/destination) code generated to control the LS/VCDT address selection in the LS/VCDT address selection logic (Paragraph 5.4.1). The S/D code is listed in Table 5-9 along with the address sources that it selects. The S/D code also enables the LS or VCDT as a source for the IB bus as shown in Table 5-9 and described in Paragraph 5.9.3.

SRC RD LS/VCDT asserts EN LS/VCDT PAR to the DP parity logic to enable a parity check on the data read out of the LS or the VCDT.

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

The true state of EN LS/VCDT OUT causes EN MD HI to assert if the VCDT has been selected (EN VCDT true). Thus the all-zeros high word from the MD bus is placed onto the IB bus along with the low word from the VCDT (see Paragraphs 5.4 and 5.8.3.1).

Decoder output EN PB IN is asserted when the PB IN register is selected as the source for the IB bus. As shown in Figure 5-3, EN PB IN gates the data from the PB IN register onto the MD bus. The data is then gated to the IB bus by EN MD HI and EN MD LO. Note in Figure 5-16 that EN PB IN asserts EN MD HI and EN MD LO, and negates EN MISC. The assertion of EN MD HI and EN MD LO effects the data transfer from the MD bus to the IB bus. The negation of EN MISC isolates the PMCSR/LITERAL mux output from the MD bus while the PB IN register data is being transferred.

Decoder output E3-13 is asserted when the microword LITERAL field is selected as the source for the IB bus. The true state of EN MISC gates the LITERAL data from the PMCSR/LITERAL mux onto the MD bus. The E3-13 decoder output asserts EN MD HI and EN MD LO to transfer the LITERAL data (and the all-zero high word) from the MD bus to the IB bus.

Decoder output EN ALU is asserted when the ALU is selected as the data source for the IB bus. EN ALU enables the ALU logic.

Decoder output E3-15 is asserted when the XBIR register is selected as the source for the IB bus. E3-15 asserts EN XBIR IN via an OR gate. EN XBIR IN gates the data in the XBIR register onto the IB bus.

EN XBIR IN is also applied to the CCI/DP Interface Control Logic where it asserts CHK XBIR PAR. CHK XBIR PAR is applied to the DP parity logic to enable an IPE parity check on data transferred from the CCI to the IB bus via the XBIR register. An IPE parity error results in the assertion of CIPA ERROR in the DP (PMCSR) and in the assertion of CBPE in the CCI (CNFGR) (Paragraph 5.7.4.2).

5.9.3 Control Signals

This paragraph discusses the gating signals generated by the DP Control Logic and under what conditions they are asserted.

EN MISC is used in the MD bus logic (Figure 5-3) to gate the output of the PMCSR/LITERAL mux onto the MD bus. EN MISC is always asserted except when the MD bus is being used to transfer data from the PB IN register to the IB bus (EN PB IN true), or to transfer data from the MDATR or the MADR registers (in the CS) to the IB bus (EN MAINT true).

EN MD HI and EN MD LO gate the MD bus high word and the MD bus low word respectively onto the IB bus. Both signals are asserted by EN PB IN thus gating the assembled longword from the PB IN register to the IB bus. Both signals are also asserted by EN MAINT thus gating the MADR and MDATR data from the CS onto the IB bus. Both signals are again asserted by the E3-13 (LITERAL) output of the source decoder thus gating the LITERAL data to the IB bus (via the PMCSR/LITERAL mux).

When executing an unsolicited CMI read operation (EN UNSOL READ true), and if the PMCSR is specified (PMCSR true), both EN MD HI and EN MD LO are again asserted. If the unsolicited read operation specifies the PSR (PSR true), only EN MD HI asserts (Paragraph 5.8.3.1).

EN MD HI is also asserted by the ANDing of EN LS/VCDT OUT and EN VCDT. EN LS/VCDT OUT is asserted by an unsolicited CMI read request of the LS or the VCDT (XBUS RD LS/VCDT true), or by a port initiated read request of the LS or the VCDT (SRC RD LS/VCDT asserted by the source decoder). When the VCDT is the selected source (EN VCDT true), EN MD HI is asserted to supply the all-zero high word onto the IB bus.

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.

During an unsolicited CMI request, DST INHIBIT asserts and disables two of the three AND gates in each signal area. The third AND gate is used to enable the signal during the unsolicited operation. XBUS EN LS/VCDT from the unsolicited CMI request logic is applied to the three active AND gates along with XBUS LSA 08. XBUX LSA 08 selects between the LS and the VCDT. If XBUS LSA 08 is true, the VCDT is enabled (EN VCDT asserts). If XBUS LSA 08 is false, the high and low section of LS are enabled (EN LS HI and EN LS LO assert). If the LS access is to the PSR, only EN LS LO asserts. However, in this case EN MD HI asserts to gate the high word portion of the PSR from the MD bus to the IB bus.

When the port is under microcode control (DST INHIBIT 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 three 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 or the VCDT. Using the S/D code from Table 5-8 or 5-9 and following the logic of Figure 5-16, it can be seen that the LS or the VCDT is enabled as shown in the tables. Note that INDEX 08 is used to select between the LS and the VCDT when necessary.

5.10 CCI/DP INTERFACE CONTROL LOGIC

The CCI/DP Interface Control Logic is shown in Figure 5-17. The logic consists of a PAL (programmable array logic) and two PROMs. The logic senses input commands from the CCI and the port microword, and outputs the required signals to execute the commanded operations.

Included in the output is a four-bit register select code (REG SEL <3:0>) which is sent to the CCI where it is decoded to select the CCI register or file to be accessed. The register select code is given in Table 5-10. Note that REG SEL 3 of the code specifies whether the operation is a read or a write (false = read; true = write). Hence when REG SEL 3 is true, DRIVE CIPA asserts to the interface logic (Figure 5-2) to enable write data out of the DP to the CCI.

Table 5-10 REG SEL Code

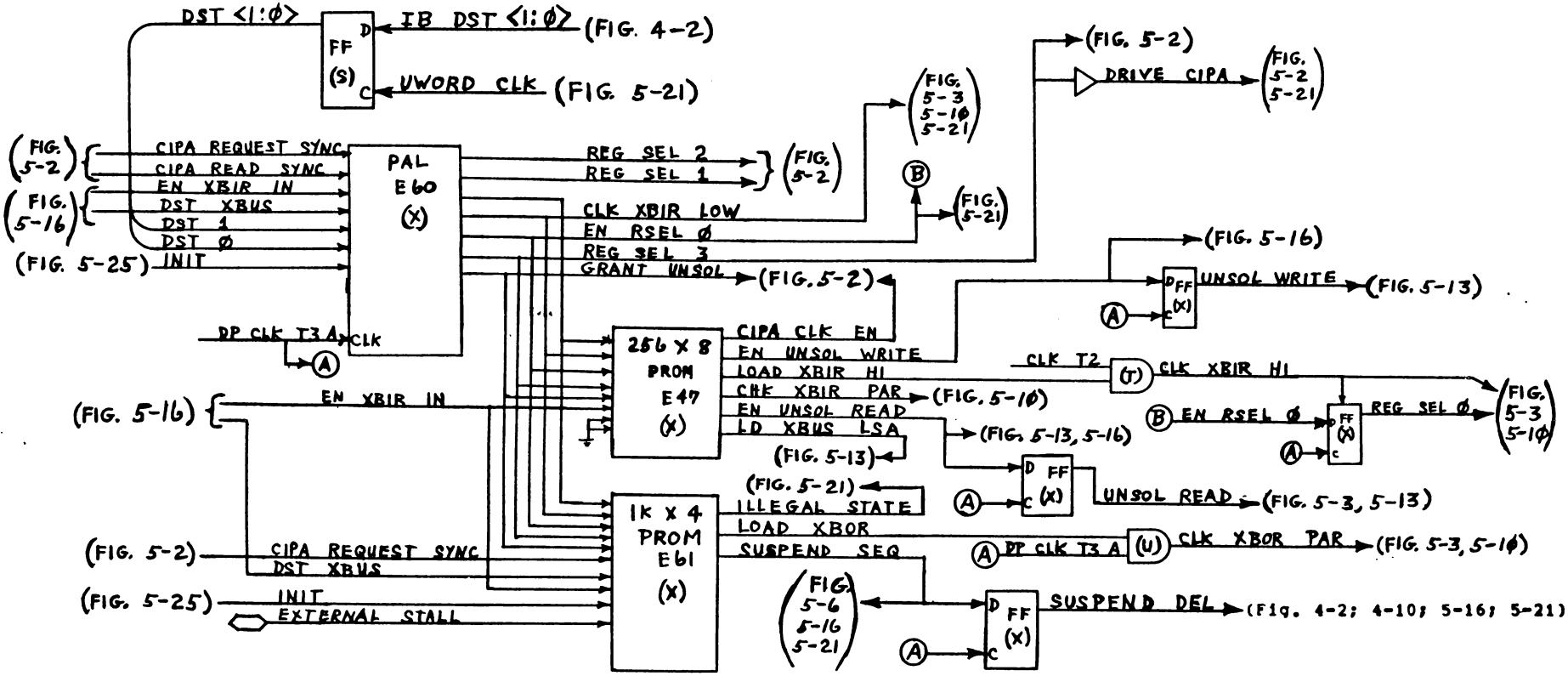
REG SEL 3 2 1 0	Decoded Mnemonic (Read)	REG SEL 3 2 1 0	Decoded Mnemonic (Write)
0 0 0 0	RD ADDR OFFSET	1 0 0 0	LD CMD/ADDR HI
0 0 0 1	" " "	1 0 0 1	LD ADDR LO
0 0 1 0	No Op	1 0 1 0	No Op
0 0 1 1	" "	1 0 1 1	LD BYTE MASK
0 1 0 0	RD RCV REG HI	1 1 0 0	LD XMIT FILE HI
0 1 0 1	RD RCV REG LO	1 1 0 1	LD XMIT FILE LO
0 1 1 0	RD RCV FILE HI	1 1 1 0	LD RTN RD REG HI
0 1 1 1	RD RCV FILE LO	1 1 1 1	LD RTN RD REG LO

The functioning of the CCI/DP Interface Control Logic is explained in terms of a port initiated read and write of the CCI (from the DP), and an unsolicited read and write of the DP. These operations have already been discussed in detail in other sections of this chapter. Only the commands issued by the CCI/DP Interface Control Logic are discussed here.

The PAL contained in the Interface Control Logic (E60) functions as a sequencer. DP CLK T3 A clocks the PAL causing it to go through a sequence of states as shown on the CI750 state drawings.* If a location is addressed in the Interface Control Logic that does not correspond to a valid state, the logic asserts ILLEGAL STATE to the port clock logic (Figure 5-21) to stop the DP clock (DP CLK T3).

* The state drawings are part of the CI750 Engineering Drawing Set.

INIT initializes PAL E60 placing it into the idle state.



Note:
Letter designations in parentheses refer to engineering drawings containing corresponding logic.

Figure 5-17 CCI/DP Interface Control Logic

5.10.1 Port Initiated Write of CCI

Figure 5-18 illustrates the CCI/DP Interface Control Logic signals involved in a port initiated write operation.

When the DP has data to be transferred to the CCI, DST XBUS is asserted by the DP Control Logic. The CCI/DP Interface Control Logic responds by outputting LOAD XBOR to load the XBOR register with the data to be transferred.

The DST <1:0> code from the microword specifies where in the CCI the data is to be written. The destination for the data could be the command/address registers, the byte mask register, or the XMIT file. The Interface Control Logic outputs the appropriate REG SEL code according to the DST <1:0> input.

CIPA CLK EN is output to the CCI to clock the data in from the DP.

If the CCI destination was the byte mask register, the operation is complete as only one transfer is necessary. If the CCI destination was the XMIT file or the command/address high register, another transfer is required to write the low word into the XMIT file or the low address into the low address register.

If another transfer is required, the DST <1:0> code specifies the destination to the CCI/DP Interface Control Logic which outputs the appropriate REG SEL code along with the CIPA CLK EN signal to clock the data into the CCI.

NOTE

A write operation of the CCI always takes 800 ns (two 400 ns cycles). When writing the byte mask register only one transfer is required, however the microcode executes a no-op cycle resulting in the transfer time for the byte mask register also being 800 ns.

5.10.2 Port Initiated Read Of CCI

Figure 5-19 illustrates the CCI/DP Interface Control Logic signals involved in a port initiated read operation.

When data is in the CCI RCV file ready to be transferred to the DP, the DP Control Logic specifies a read of the CCI by asserting EN XBIR IN. The CCI/DP Interface Control Logic outputs the REG SEL code for a high word read of the CCI RCV file. The RCV file is the only CCI location that can be read by the DP in a port initiated transfer.

The logic then asserts LOAD XBIR HI to load the high word from the CCI into the high section of the XBIR register.

The logic also outputs the REG SEL code for a low word read of the RCV file. CIPA CLK EN is output to the CCI to increment the RCV file read pointer.

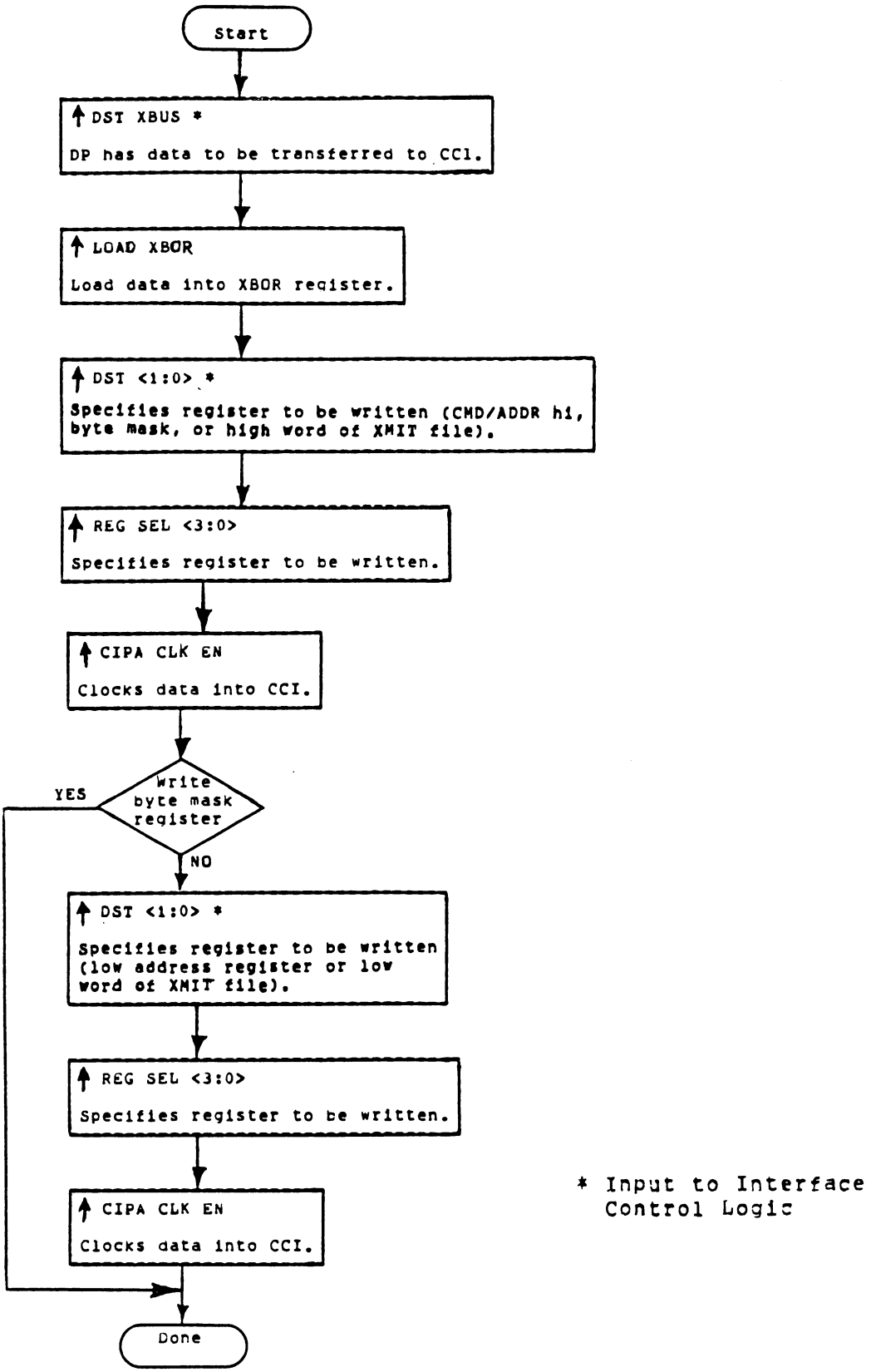


Figure 5-18 CCI/DP Interface Control Logic -- Port Initiated Write of CCI

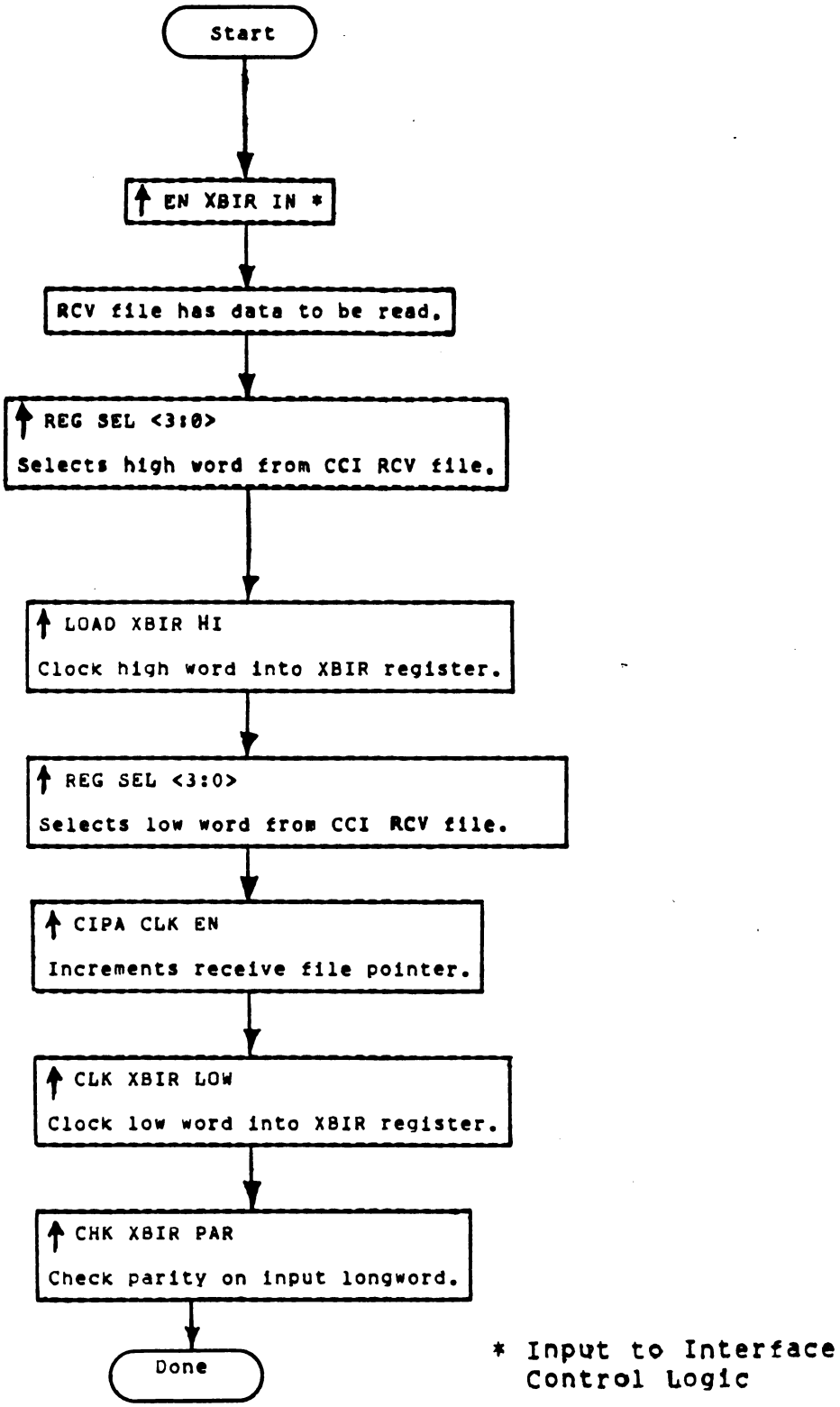


Figure 5-19 CCI/DP Interface Control Logic -- Port Initiated Read of CCI

The logic then asserts CLK XBIR LOW to load the low word from the CCI into the low section of the XBIR register.

Finally CHK XBIR PAR asserts to enable a parity check on the data longword transferred in from the CCI.

5.10.3 Unsolicited Request Operations

Figure 5-20 illustrates the CCI/DP Interface Control Logic signals involved in an unsolicited request operation.

When an unsolicited request operation is initiated, CIPA REQUEST SYNC and CIPA READ SYNC are asserted to the Interface Control Logic from the CCI.

The logic responds to CIPA REQUEST SYNC by asserting GRANT UNSOL and SUSPEND SEQ. GRANT UNSOL is sent to the CCI to indicate that an unsolicited operation is in progress. SUSPEND SEQ is applied to the clock logic to stop the port microcode from running (Paragraph 5.11.2).

The logic responds to CIPA READ SYNC by asserting the REG SEL code for a read of the CCI address offset register. All unsolicited operations start by a read of the address offset register to determine the address of the register to be accessed by the operation. The address passes from the CCI, over the CIPA bus, and then to the LSA address register in the DP. The address does not pass through the XBIR register, hence does not undergo parity checking.

The Interface Control Logic then outputs LD XBUS LSA to clock the register address into the LSA address register.

At this point, the CIPA READ SYNC input is checked to determine if the operation is a read or a write. If CIPA READ SYNC is true, the Interface Control Logic outputs EN UNSOL READ indicating an unsolicited read operation is in progress.

The logic then outputs LOAD XBOR to load the data read from the selected register, into the XBOR register for transfer to the CCI.

The logic also outputs the REG SEL code for writing the data high word into the Return Read Data Register. The Return Read Data Register is the only CCI location that is written from the DP in an unsolicited read operation.

The Interface Control Logic increments the REG SEL code to write the data low word into the Return Read Data Register. This completes the unsolicited read operation.

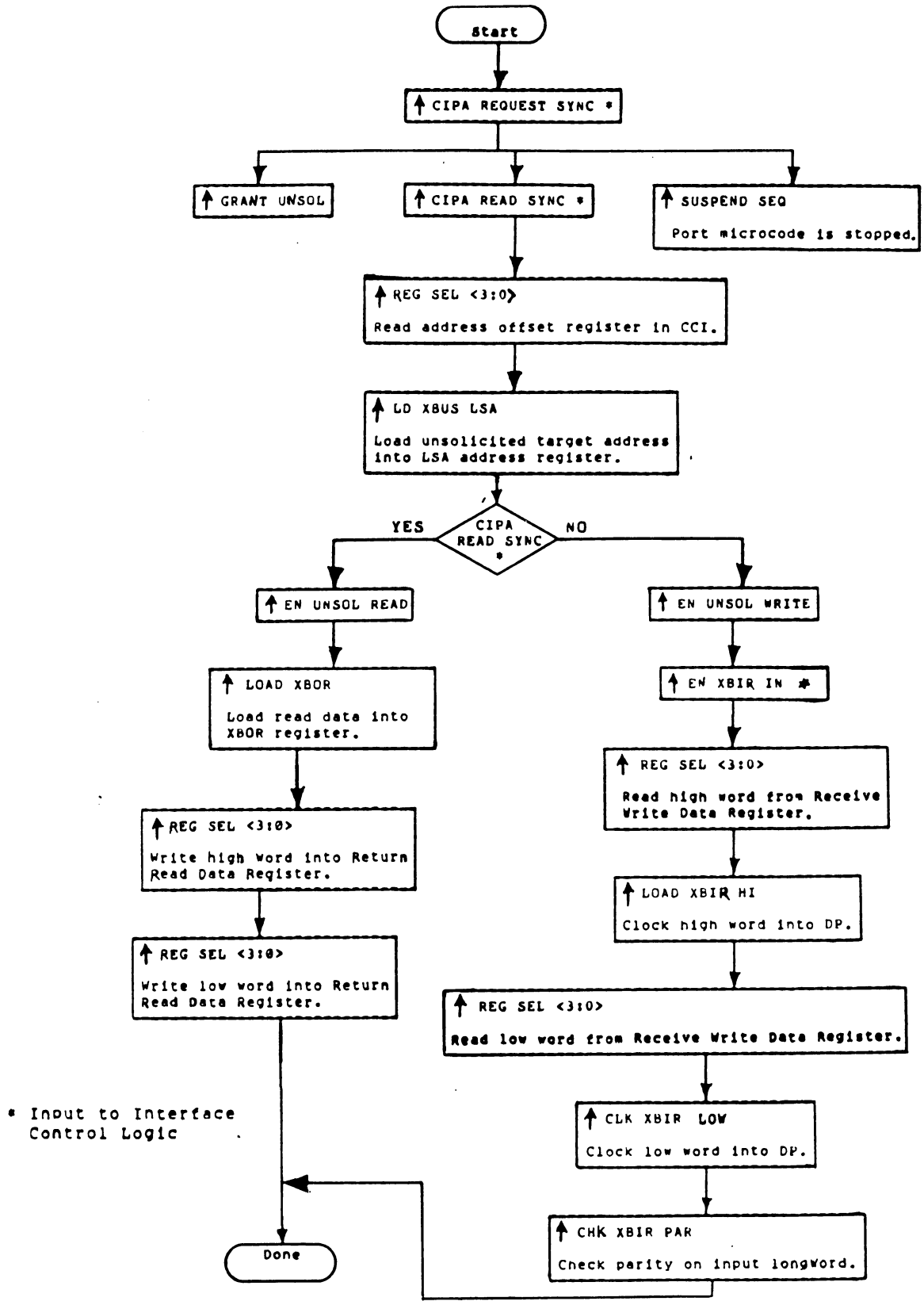


Figure 5-20 CCCI/DP Interface Control Logic -- Unsolicited Request Operations

If CIPA READ SYNC checked false when the type of operation was being determined, the Interface Control Logic would output EN UNSOL WRITE indicating an unsolicited write operation is in progress.

The DP Control Logic specifies a read of the CCI by asserting EN XBIR IN to the CCI/DP Interface Control Logic.

The CCI/DP Interface Control Logic responds by outputting the REG SEL code for reading a high word from the Receive Write Data Register in the CCI. The Receive Write Data Register is the only CCI location that is read from the DP in an unsolicited write operation.

The logic outputs LOAD XBIR HI to clock the high word into the DP.

The logic then increments the REG SEL code and outputs it to the CCI to read the data low word from the Receive Write Data Register.

CLK XBIR LOW asserts from the logic to clock the data low word into the DP.

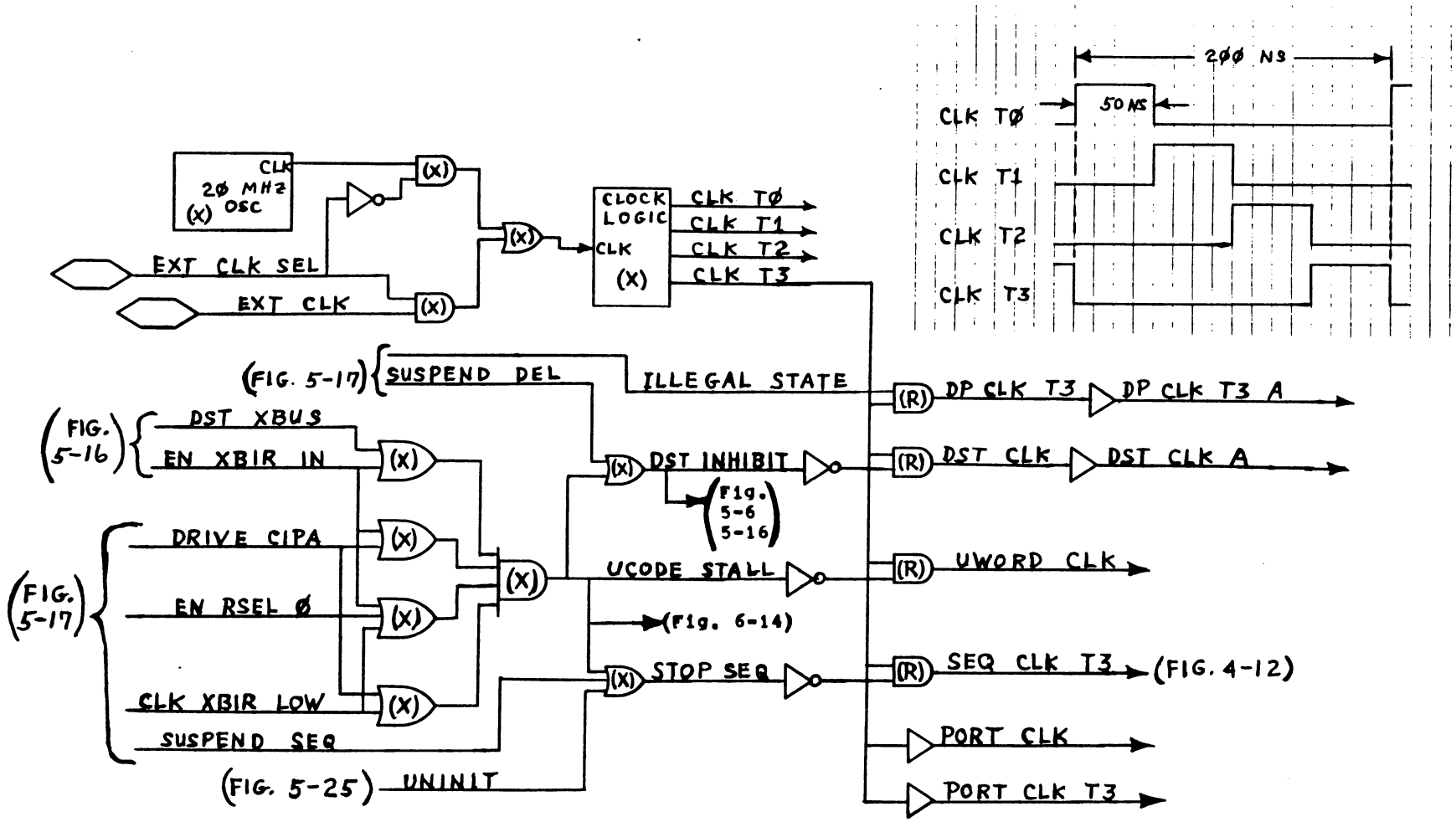
Finally, CHK XBIR PAR asserts to enable a parity check on the data longword transferred in from the CCI.

5.11 Port clocks And Operating Modes

5.11.1 Port Clocks

The CMI has a bus cycle of 160 ns. The bus cycle is established by the CMI bus clock (CMI B CLK) which is a 6.25 MHz clock with a period of 160 ns. The CI750 port uses a 5 MHz clock with a time period of 200 ns. Data transfers at the port-to-CMI interface use the 160 ns bus clock. Timing throughout the rest of the port is done with the 200 ns clock.

As seen in Figure 5-21, the output of a 20 MHz crystal oscillator is applied to clock logic where the 20 MHz is divided down to 5 MHz (200 ns) and phase shifted to produce four 200 ns clocks 90° out of phase with each other. An external clock can be used in place of the 20 MHz oscillator by asserting EXT CLK SEL and inputting the external clock as EXT CLK. The four clocks output from the clock logic are designated as CLK T0, CLK T1, CLK T2, and CLK T3.



Note:
Letter designations in parentheses refer to engineering drawings containing corresponding logic.

Figure 5-21 Port Clocks

CLK T3 is the base clock used to generate all the port critical clocks. The clocks derived from CLK T3 are listed in Table 5-11. The table also lists where the clocks are used and the modes in which they are enabled (indicated by an X).

Table 5-11 Port Clocks

Clock	Where Used	Modes			
		Run	Uninitialized	Stall	Suspend
DP CLK T3 A	DP	X	X	X	X
SEQ CLK T3	PB	X	-	-	-
UWORD CLK	DP	X	X	-	X
DST CLK A	DP	X	X	-	-
PORT CLK & PORT CLK T3	PB@	X	X	X	X

@ PORT CLK is also used in the link module.

DP CLK T3 A is used to clock control signals and some destination registers. It is running at all times except when ILLEGAL STATE is asserted by the CCN/DP Interface Control Logic. ILLEGAL STATE indicates a fatal error condition wherein the error hardware may not be reliable. By gating off DP CLK T3 A, the port error condition is preserved for maintenance testing.

SEQ CLK T3 is used to clock the CS microsequencer that runs the port microcode. It is used mainly to control microsequencing and to load microcode registers.

UWORD CLK is used to clock microword logic whose inputs are functions of the port microword on the CS bus.

DST CLK A is used to load most of the destination registers controlled by the microcode (e.g. the PB OUT and PB IN registers).

PORT CLK and PORT CLK T3 are used to control operations within the PB and the link modules. The two clocks are functionally identical to CLK T3 and are always running.

5.11.2 Operating Modes

The DP operates in one of the following four modes:

1. Run Mode
2. Uninitialized Mode
3. Stall Mode
4. Suspend Mode

5.11.2.1 Run Mode -- This is the normal operating mode. Microinstructions are executed every 200 ns. All clocks are operational.

5.11.2.2 Uninitialized Mode -- This mode is entered when the port is powered up or an error condition is detected. When UNINIT comes true, STOP SEQ asserts and gates off SEQ CLK T3 thereby stopping the port microcode. The CS microsequencer in the PB stops at address zero.

5.11.2.3 Stall Mode -- There are occasions when a microinstruction cannot be completed within the basic 200 ns clock cycle, for example when the data to be transferred is not yet available. Under such conditions, the microcode is stalled thereby stretching out the microcycle in multiples of 200 ns.

As seen in Figure 5-21, UCODE STALL may assert when data is being transferred over the CIPA bus in either direction. When data is to be transferred in to the DP from the CCI (EN XBIR IN asserts) but:

* the low word input cycle of the preceding input transfer is not complete (CLK XBIR LOW still true)

  or

* the preceding transfer from the DP to the CCI is still in progress (DRIVE CIPA true),

then UCODE STALL asserts.

When data is to be transferred out of the DP to the CCI (DST XBUS and DRIVE CIPA assert) but:

* the low word cycle of the preceding transfer (in either direction) has not completed (EN RSEL 0 still true),

then UCODE STALL asserts.

The assertion of UCODE STALL gates off UWORD CLK to prevent clocking of the microword logic. UCODE STALL also causes STOP SEQ and DST INHIBIT to assert. STOP SEQ gates off SEQ CLK T3 which stops the port microcode from running. DST INHIBIT gates off DST CLK A to prevent loading of the destination registers. In addition, UCODE STALL inhibits the assertion of GO in the CCI thereby delaying the start of any new transfers.

Figure 5-21A illustrates port action during the stall mode. In time period 1, microword X is executed with all clocks operational. In time period 2, microword X+1 executes but cannot finish resulting in the stall condition and the assertion of UCODE STALL. UCODE STALL gates off clocks SEQ CLK T3, UWORD CLK, and DST CLK A. In time period 3 and 4 the stall condition remains and the three clocks remain gated off while microword X+1 continues to execute. In time period 5 the stall condition is removed, UCODE STALL negates, and the three clocks are gated on. The microword logic and destination registers are clocked by UWORD CLK and DST CLK A respectively, thereby completing the execution of microword X+1. SEQ CLK T3 clocks the next microword (X+2) from the CS. In time period 6 the port returns to the run mode and the X+2 microword executes.

5.11.2.4 Suspend Mode -- While the port is in the run mode, the host CPU may want to access a port register via an unsolicited CMI request. To make the port data paths available for the unsolicited CMI access, the microcode is temporarily stopped (suspended) until the unsolicited CMI access is completed. In this case the port is said to be in the suspend mode.

In the suspend mode SUSPEND SEQ and SUSPEND DEL are asserted by the CCI/DP Interface Control Logic. SUSPEND SEQ asserts STOP SEQ which gates off SEQ CLK T3 thereby stopping the port microcode. The DP is now controlled by the host CPU.

SUSPEND DEL asserts DST INHIBIT which inhibits DST CLK A thereby preventing the destination registers from being loaded. Note in Figure 5-17 that SUSPEND DEL is asserted one cycle after STOP SEQ asserts. Thus during the cycle in which SUSPEND SEQ asserts, DST CLK A is still active to load the destination registers. Hence, execution of the current microword is completed (not stalled). This makes the cycle in which SUSPEND SEQ asserts, the last cycle of the run mode.

Figure 5-21B illustrates port action during the suspend mode. In time period 1 microword X is executed with all clocks operational. In time period 2 microword X+1 is executing when the unsolicited CMI request appears and causes SUSPEND SEQ to assert. SUSPEND SEQ gates off SEQ CLK T3 so that the microcode does not advance to the next microword. UWORD CLK and DST CLK A respectively clock the microword logic and the destination registers thereby completing execution of microword X+1. SUSPEND DEL asserts on the next DP CLK T3 A pulse and gates off DST CLK A.

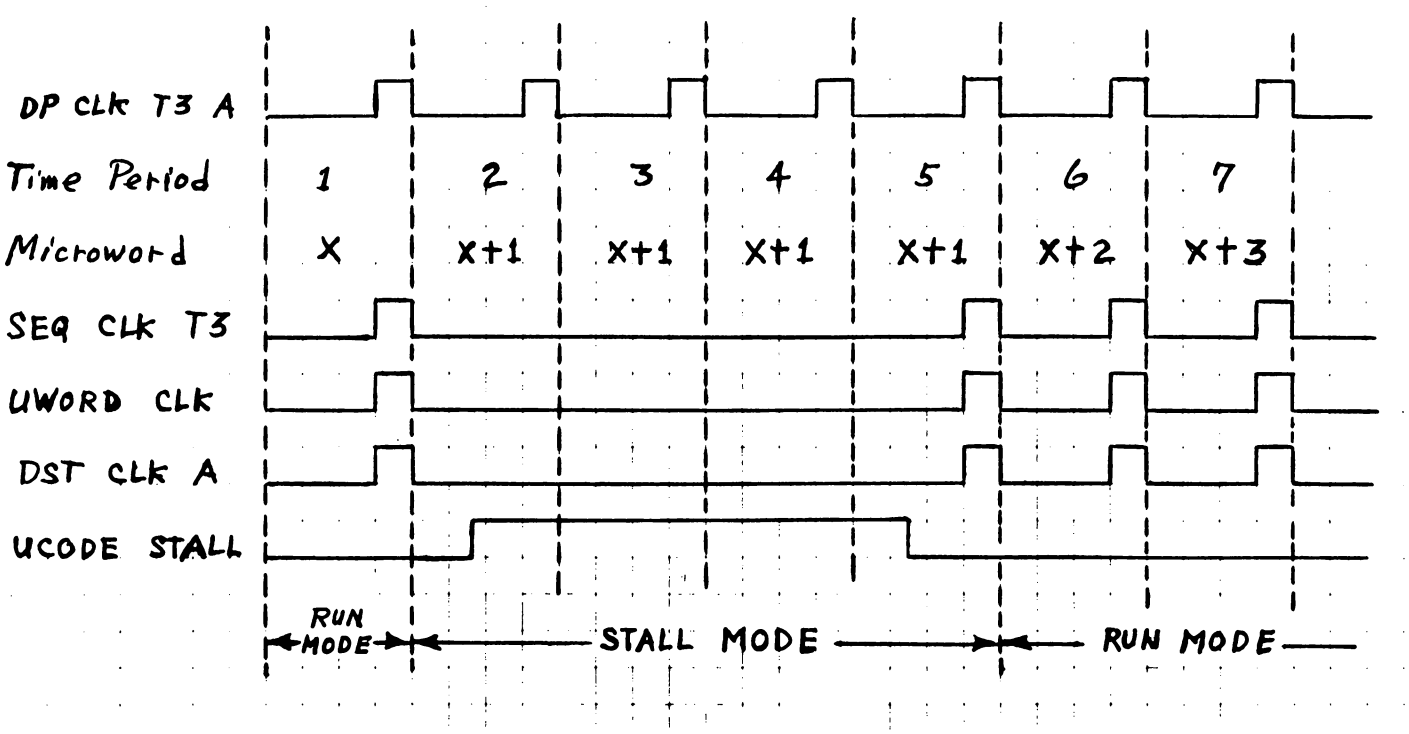


Figure 5-21A Stall Mode Timing

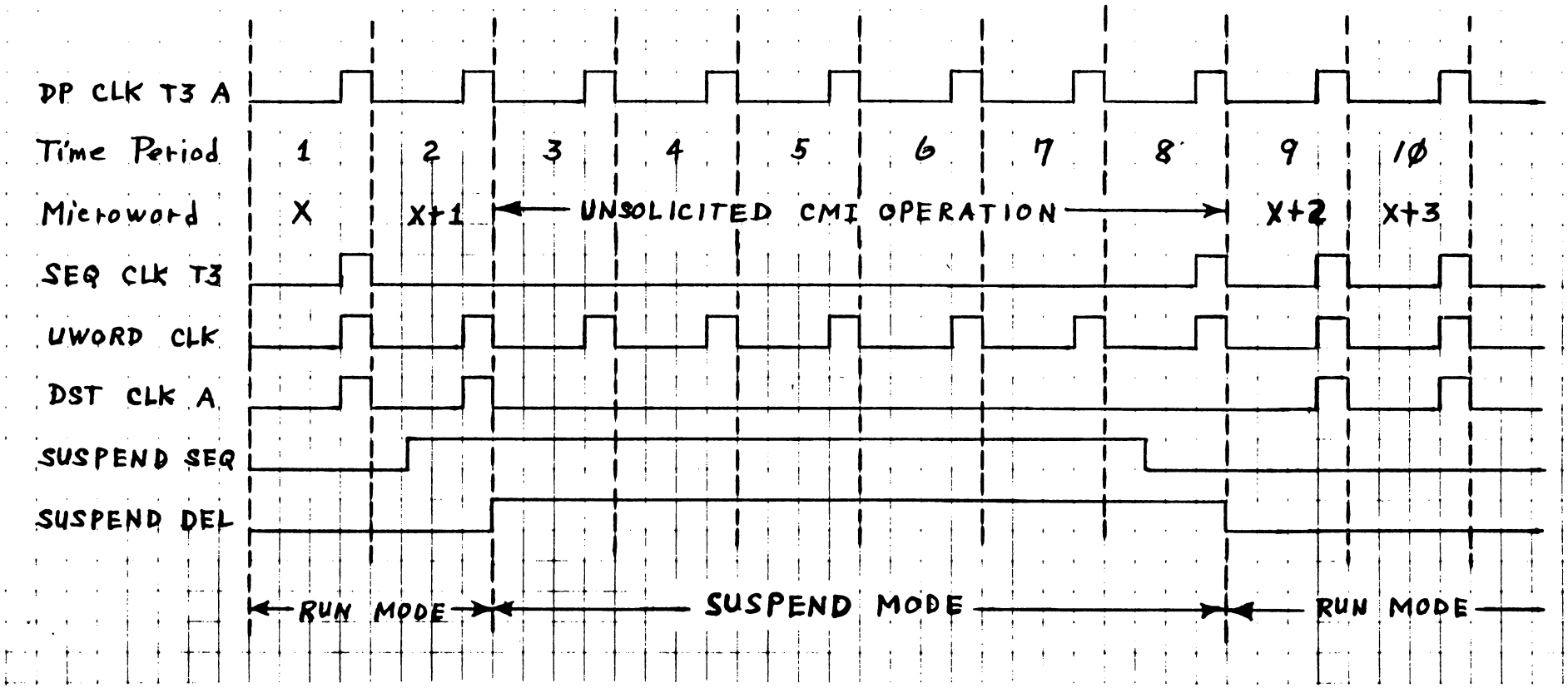


Figure 5-21B Suspend Mode Timing

During time periods 3 through 8 the unsolicited CMI function is executing. UWORD CLK is active during the suspend mode because portions of the microword logic are used for the unsolicited CMI operation. In time period 8 the unsolicited operation is completed and SUSPEND SEQ negates. The negation of SUSPEND SEQ enables SEQ CLK T3 which clocks the next microword (X+2). DST CLK A remains gated off (due to the one cycle delay of SUSPEND DEL) thereby preventing data from the unsolicited CMI operation from being clocked into the destination registers. In time period 9 the run mode is resumed with all clocks operational to execute microword X+2.

5.11.2.5 Differences Between Stall Mode and Suspend Mode -- The major difference between the stall mode and the suspend mode is that in the stall mode, destination registers are inhibited from loading until the final cycle of the stall condition. Whereas in the suspend mode, destination registers are enabled in the first suspend cycle and inhibited from loading throughout the remaining suspend cycles. It is necessary to complete execution of the microword in the first cycle of the suspend mode (as opposed to deferring it until the final cycle as is the case in the stall mode). The reason for this is that some microword logic registers are used to service unsolicited CMI functions. By the time an unsolicited CMI function is completed, these registers will no longer contain the original microword contents.

Note in Figure 5-21A that microword X+1 is stalled by being stretched out over four DP CLK T3 A time periods. This is in contrast to Figure 5-21B where microword X+1 is completed in time period 2. There is no microword execution in the next 6 time periods while the unsolicited CMI function operates. Microword X+2 executes in the time period following the completion of the unsolicited CMI function (time period 9).

5.12 INTERRUPT, INITIALIZE, AND POWER CONTROL FUNCTIONS

Figure 5-22 is a block diagram of the interrupt, initialize, and power control functions.

The interrupt logic asserts INTR CPU to the CCI which in turn generates an interrupt to the host CPU. Interrupts are generated by:

1. SET MIF (maintenance interrupt flag) on a port power-up
2. SET PDN (power-down) during a power failure in the host CPU cabinet or the CIPA cabinet
3. PE when a port parity error has occurred
4. INTR from the port microword

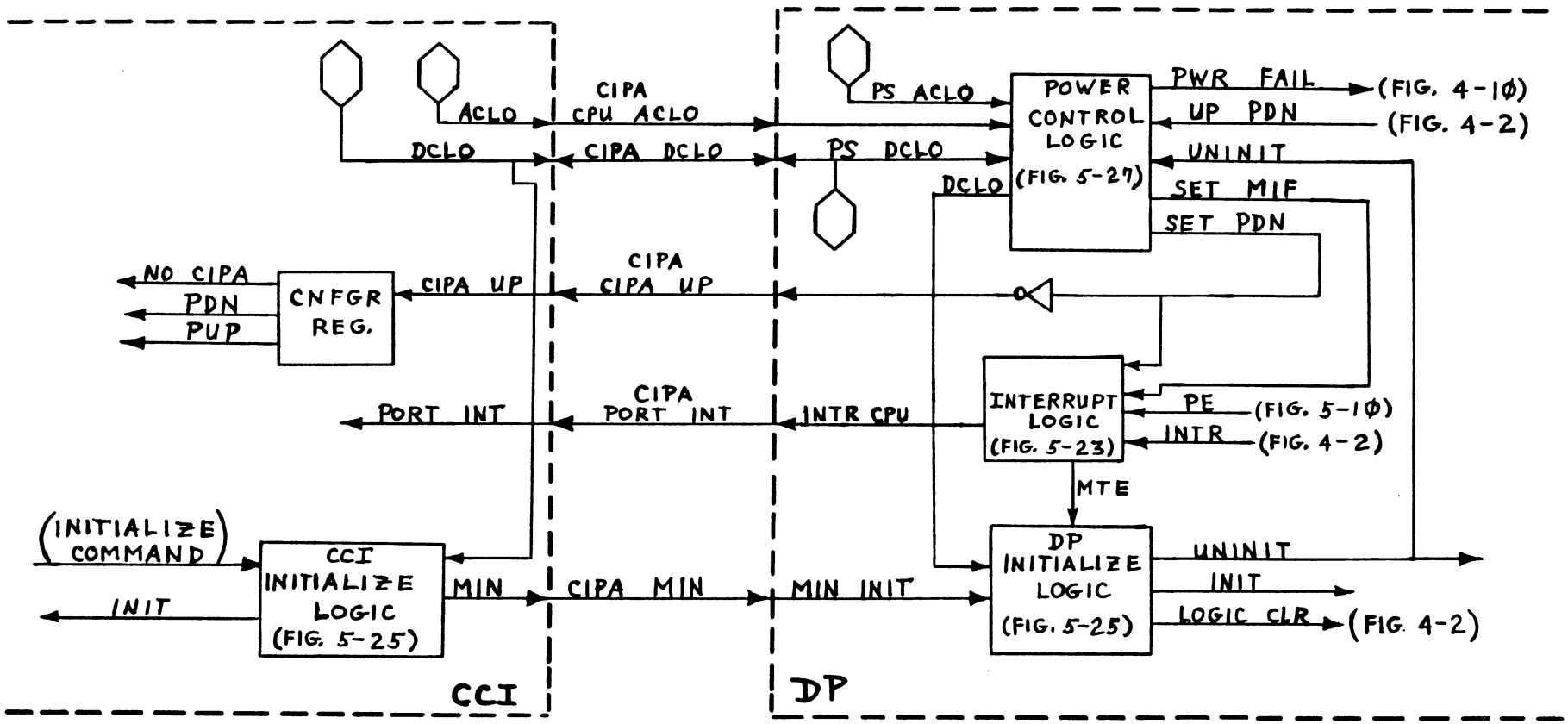


Figure 5-22 Interrupt, Initialize, and Power Control Block Diagram

Interrupts caused by port failures (PE or SET PDN) assert MTE (maintenance error) to the initialize logic to place the port into the uninitialized state.

The initialize logic is divided between the CCI and the DP. A port initialize command may be issued by the host CPU to the CCI initialize logic. The CCI initialize logic responds by asserting INIT to the CCI logic and MIN (maintenance initialize) to the DP. INIT clears the CCI logic to its reset state. MIN is applied to the DP initialize logic as MIN INIT. MIN INIT causes the DP initialize logic to output INIT to clear the DP logic, and LOGIC CLR to clear the PB logic. The DP initialize logic also asserts UNINIT to place the port into the uninitialized state.

During port power-up, DCLO asserts to the initialize logic to reset the DP and CCI logic circuits and place the port into the uninitialized state. MTE from the interrupt logic also places the port into the uninitialized state but does not reset the DP.

The power control logic functions to initialize the port during a power-up, and generate an interrupt as the power-up sequence is completed. During power-up, DCLO in the DP asserts to the initialize logic to initialize the CCI. As the power-up sequence completes, the power control logic asserts SET MIF to the interrupt logic where it generates an interrupt to the host CPU.

The power control logic also functions to perform a power fail sequence when a power failure occurs. During port operation, the power control logic monitors ac and dc power in the CIPA cabinet and in the host CPU cabinet. If power fails in either cabinet, an ACLO signal is asserted to the power control logic which asserts PWR FAIL to the microcode branching logic. If the port is in the initialized state (UNINIT false), the microcode suspends operation at a logical break point and returns UP PDN to the power control logic. UP PDN causes the power control logic to assert SET PDN indicating that the port is powering down. If the port is in the uninitialized state (UNINIT true), PWR FAIL directly asserts SET PDN via the power control hardware. SET PDN is inverted and transferred to the CCI as a negated CIPA UP. The negated CIPA UP is applied to the configuration register where it sets bits NO CIPA and PDN, and resets the PUP bit. SET PDN is also applied to the interrupt logic where it causes an interrupt to the host CPU.

When ac power resumes, SET PDN negates and CIPA UP asserts. The assertion of CIPA UP causes the PUP bit to assert, and the PDN and NO CIPA bits to negate. In addition, SET MIF is asserted to the interrupt logic to generate an interrupt to the host CPU.

5.12.1 Interrupt Function

Figure 5-23 illustrates the interrupt logic. Figure 5-24 is a flow diagram of the interrupt sequence. Refer to them throughout the following discussion.

An interrupt to the host CPU can be initiated by:

1. a port power failure (SET PDN)
2. a port parity error (PE)
3. the port microcode (INTR)
4. the port powering up (SET MIF)

The assertion of SET PDN causes SET MTE to assert until the next DP CLK T3 A clock pulse (the next DP CLK T3 A pulse resets the SET MTE flip-flop negating SET MTE). The assertion of SET MTE causes MTE to assert. The assertion of PE also causes MTE to assert. MTE is applied to the initialization logic where it asserts SET UNINIT thereby causing the port to enter the uninitialized state (Paragraph 5.12.2.2).

MTE is a bit in the PSR register (Paragraph 5.5.3).

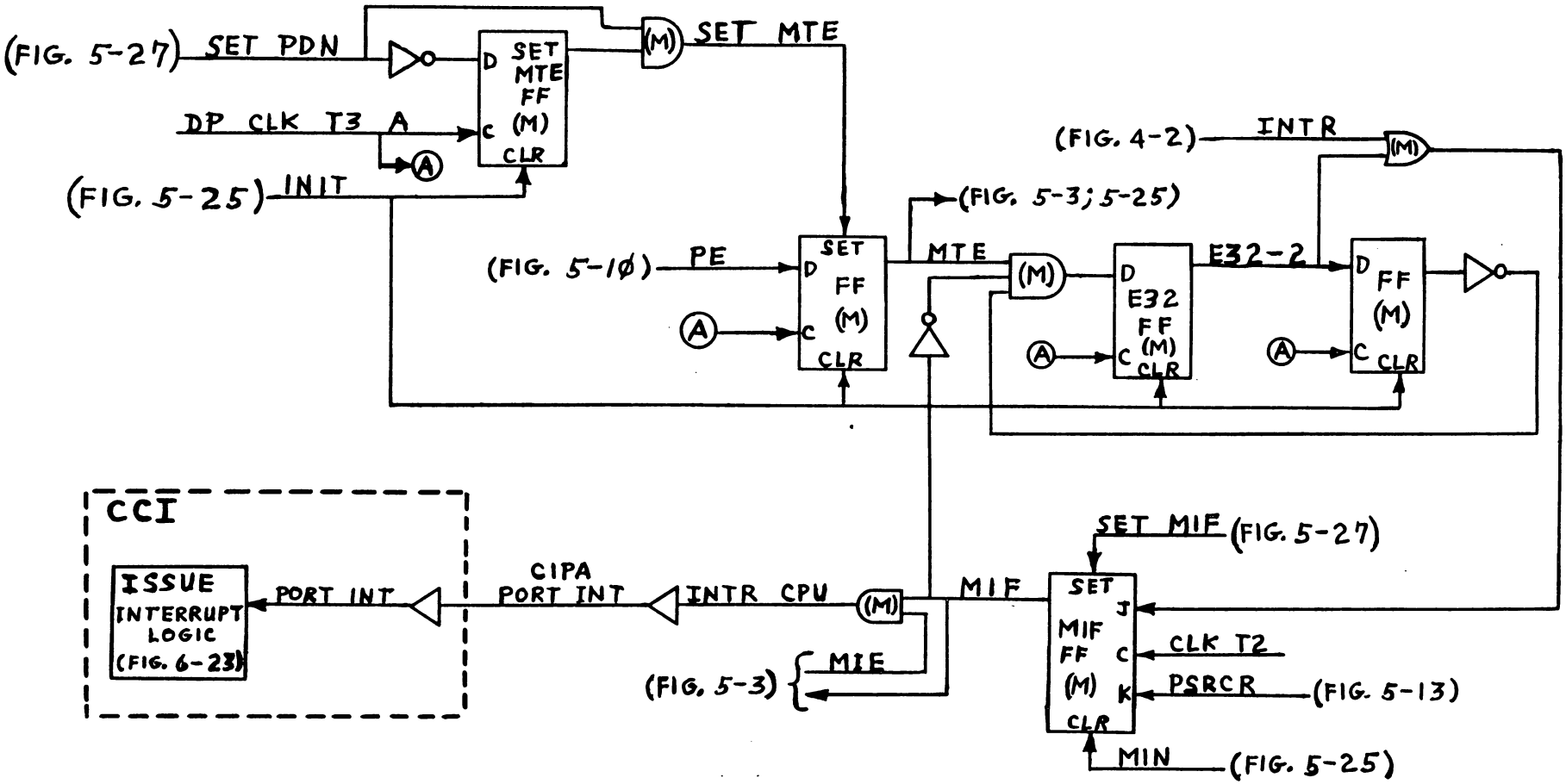
MTE is also applied to an AND gate where the maintenance interrupt flag (MIF) is sampled. If MIF is true, another interrupt is being serviced and the AND gate is disabled. The interrupt sequence must be completed and MIF negated before the current error can generate an interrupt. If MIF is false, the AND gate is enabled resulting in flip-flop E32 being set on the next DP CLK T3 A clock pulse and asserting E32-2.

The assertion of E32-2 sets another flip-flop whose inverted output is fed back to the E32 input. The negative feedback path results in the negation of E32 after two DP CLK T3 A pulses. Thus the logic resets itself in preparation for another interrupt.

E32-2 is ORed with INTR from the CS microword. The assertion of either E32-2 or INTR sets a MIF flip-flop on the next CLK T2 pulse, causing MIF to assert. MIF, in turn, asserts INTR CPU due to the true state of MIE. (MIE is asserted on power-up by DCLO; see Paragraph 5.12.3.1. It is negated only for maintenance testing of the interrupt logic.)

During a system power-up, MIF is directly set by the assertion of SET MIF from the power control logic.

MIF is a bit in the PMCSR register.



Note:
Letter designations in parentheses refer to engineering drawings containing corresponding logic.

Figure 5-23 Interrupt Logic

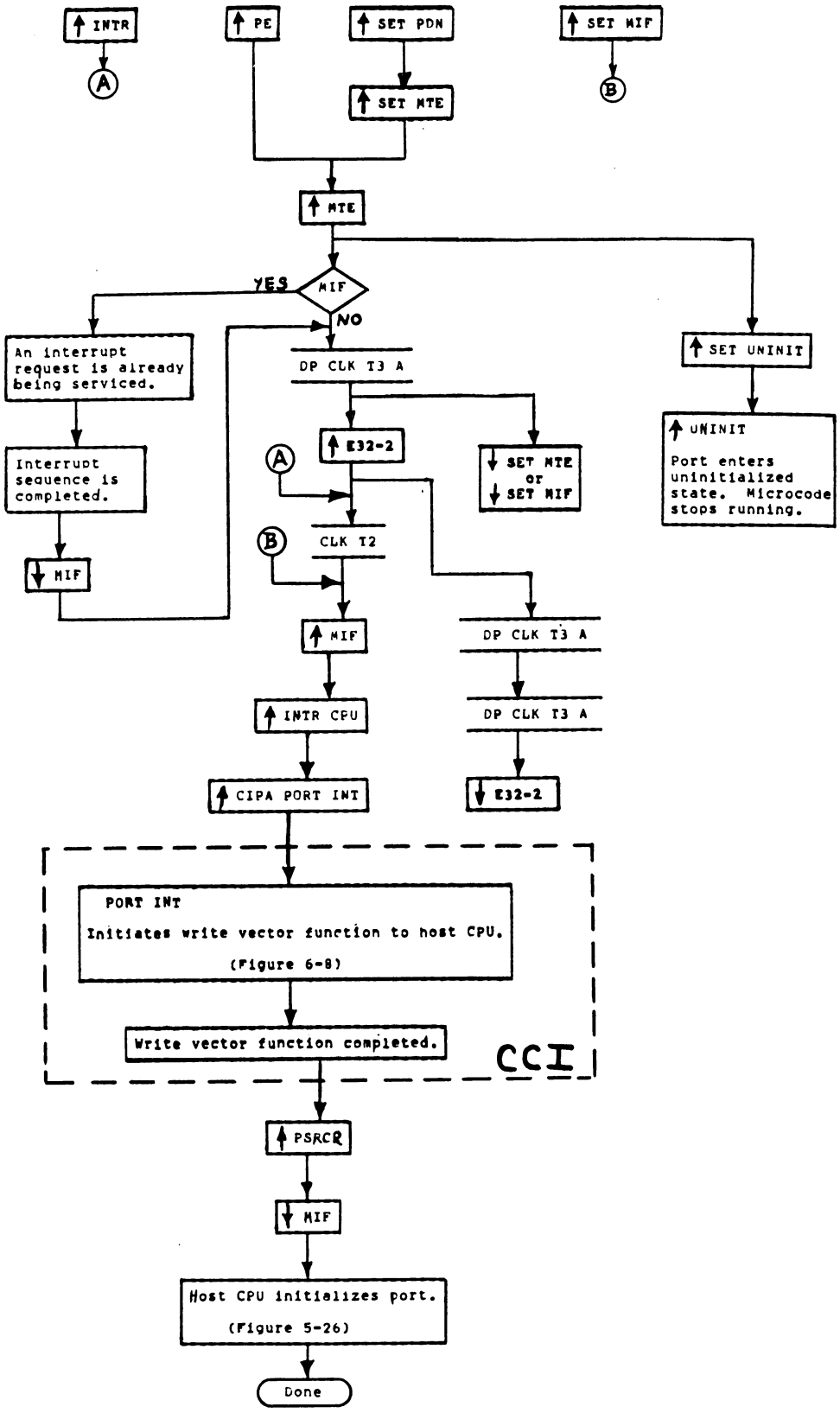


Figure 5-24 Interrupt Sequence

INTR CPU is transferred over the CIPA bus (as CIPA PORT INT) to the CCI where it asserts PORT INT to the CCI issue interrupt logic. The issue interrupt logic initiates a write vector function to the host CPU. When the write vector function is complete, the host CPU resets the MIF flip-flop by asserting PSRCR via an unsolicited CMI write sequence. The host CPU then initializes the port by issuing a MIN command as described in Paragraph 5.12.2.

5.12.2 Initialize Function

The initialize function is divided between the CCI module and the DP module. Both the CCI and the DP have initialize logic which are described in Paragraphs 5.12.2.1 and 5.12.2.2.

Figure 5-25 illustrates the initialize logic. Figure 5-26 is a flow diagram of the initialize sequence. Refer to them throughout the following discussion.

5.12.2.1 CCI Initialize Logic -- The CI750 port can be initialized by a maintenance initialize (MIN) command from the host CPU.

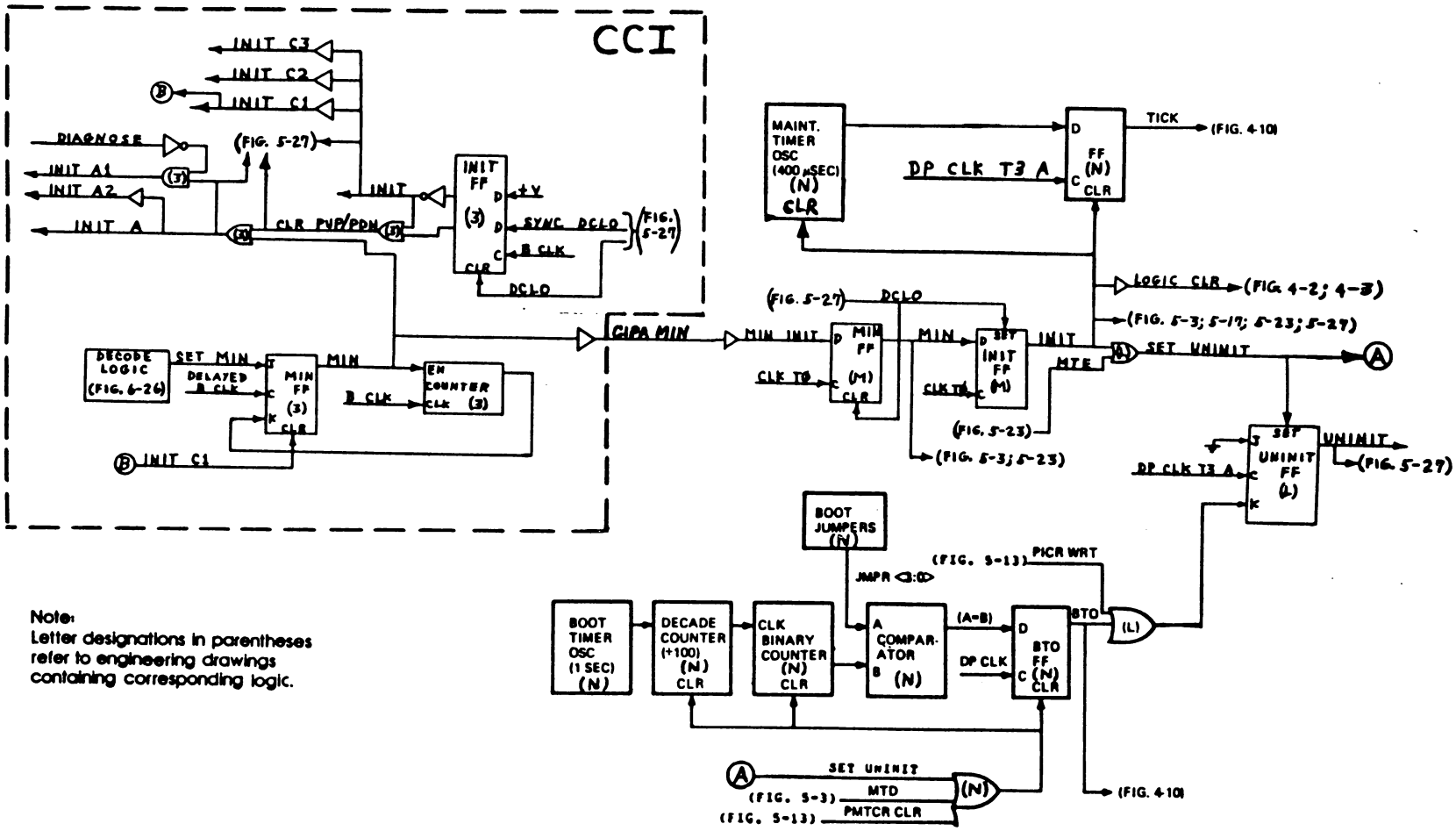
When the CCI decode logic senses an initialize command, it asserts SET MIN to a MIN flip-flop causing it to set. When the MIN flip-flop sets, it asserts MIN to the CCI initialize logic causing INIT A, INIT A1, and INIT A2 to assert. INIT A, INIT A1, and INIT A2 function to clear the error bits in the CNFGR register and reset the CCI logic circuits except for the FPLA logic array circuitry.

MIN is also applied to the CIPA bus as CIPA MIN and then to the DP.

The output of the MIN flip-flop enables a four-bit binary counter clocked by B CLK. After five B CLK cycles, the counter output goes true and resets the MIN flip-flop negating MIN.

During power-up the CCI initialize logic receives DCLO and SYNC DCLO from the power control logic. As DCLO and SYNC DCLO are power control signals, the response of the CCI initialize logic to these signals is covered in the description of the Power-Up Sequence (Paragraph 5.12.3.1).

5.12.2.2 DP Initialize Logic -- CIPA MIN is received from the CCI via the CIPA bus, and becomes MIN INIT in the DP. MIN INIT asserts MIN which in turn resets the MIE and MIF bits in the PMCSR register. In addition, MIN sets an INIT flip-flop causing INIT to assert.



Note:
Letter designations in parentheses refer to engineering drawings containing corresponding logic.

Figure 5-25 Initialize Logic

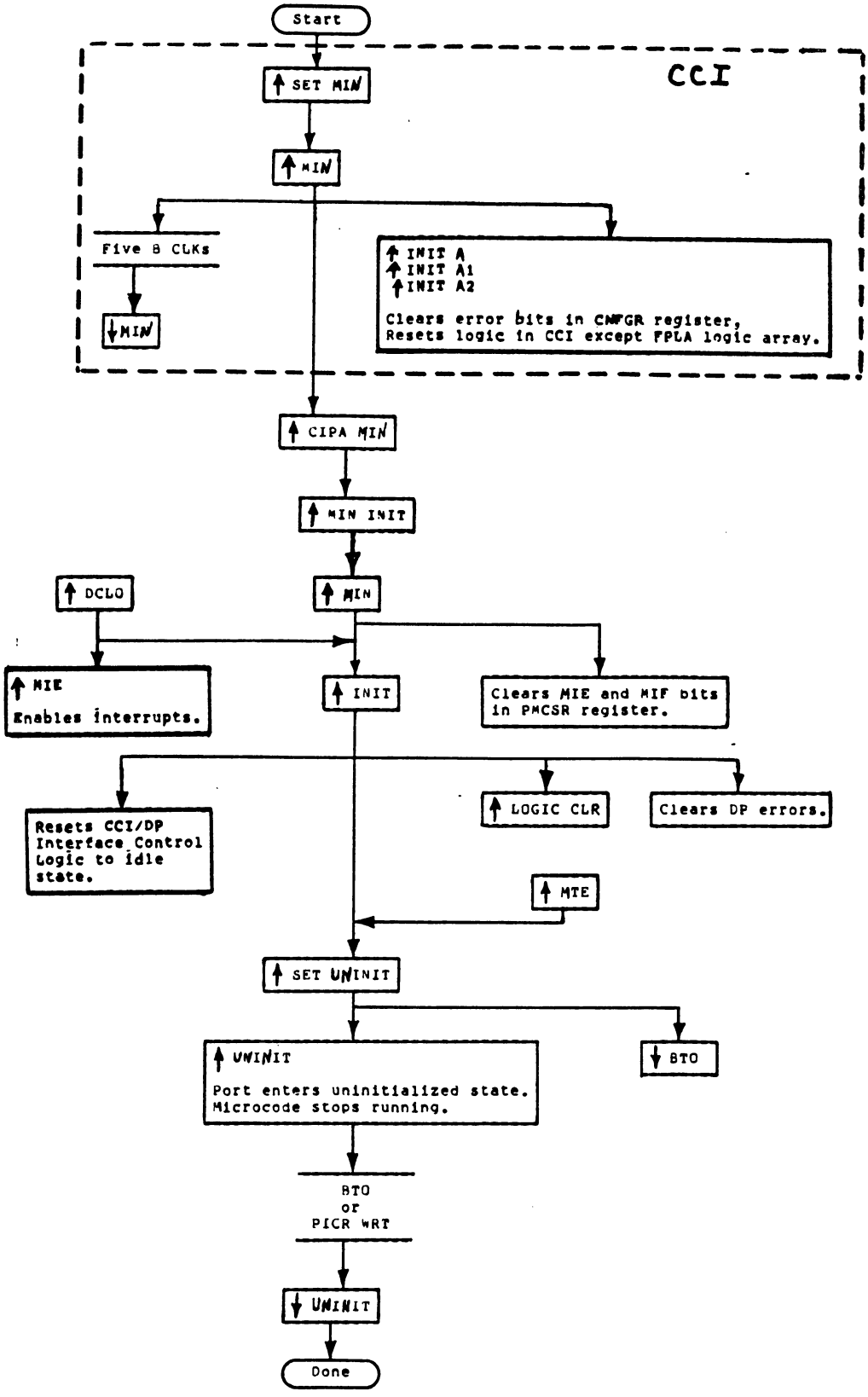


Figure 5-26 Initialize Sequence

The assertion of INIT accomplishes the following functions:

1. Clears DP errors

2. Asserts LOGIC CLR to initialize the PB module

3. Resets the CCI/DP Interface Control Logic thereby placing the DP into the idle state

4. Asserts SET UNINIT to place the port into the uninitialized state

5. Resets the boot timer via the assertion of SET UNINIT (Paragraph 5.12.2.3)

6. Resets maintenance timer

The assertion of SET UNINIT sets the UNINIT flip-flop asserting UNINIT and placing the port into the uninitialized state.

Note in the flow diagram of Figure 5-26 that the assertion of MTE (maintenance error) from the interrupt logic also asserts SET UNINIT and places the port into the uninitialized state.

A third way the port is placed into the uninitialized state is by a system power-up. During system power-up, DCLO asserts and directly sets the INIT flip-flop causing INIT and SET UNINIT to assert.*

* System power protocol requires that +5 V be up and operational before ACLO and DCLO assert.

The UNINIT flip-flop holds the port in the uninitialized state until it is reset by the assertion of BTO from the boot timer or PICR WRT from the unsolicited CMI request logic.

PICR WRT asserts when the CPU performs an unsolicited CMI write to the port initialization control register (PICR). Thus the host can start up the microcode before the assertion of BTO (before the boot timeout period has expired).

5.12.2.3 Boot Timer and Maintenance Timer -- The boot timer logic is used to delay starting the CS microcode by temporarily 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 allow time for the cluster to boot and to load the microcode.

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 (A=B), the BTO (boot timeout) flip-flop sets and asserts BTO to the UNINIT flip-flop via an OR gate. BTO causes the UNINIT flip-flop to reset and negate UNINIT. The negation of UNINIT takes the port out of the uninitialized state and starts the microcode running.

SET UNINIT clears the BTO decade counter and holds the BTO flip-flop reset. Hence the BTO counter is not enabled until the condition causing SET UNINIT to be true, is cleared.

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

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.

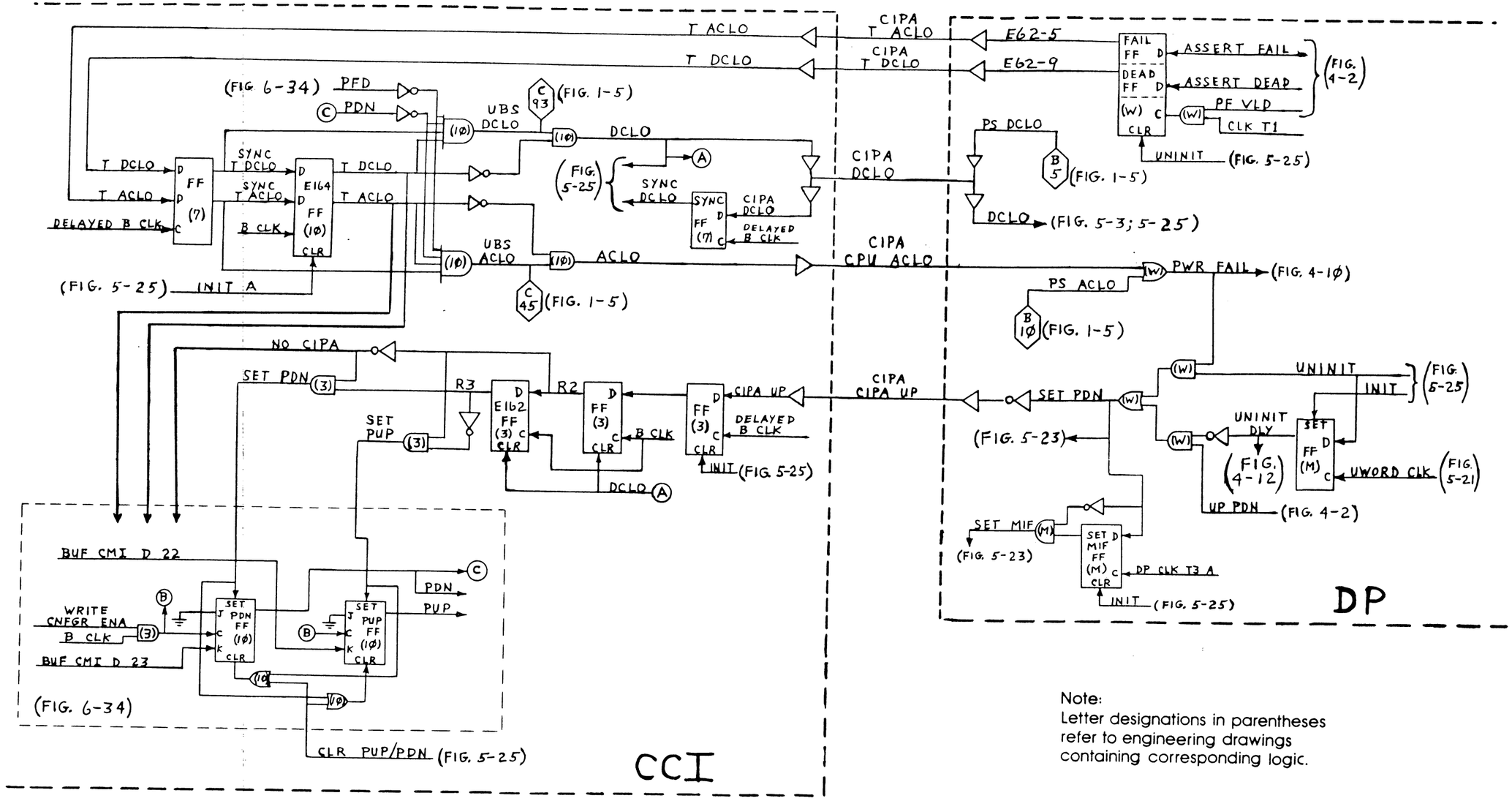
5.12.3 Power Control Function

The power control function causes the CIPA cabinet and the CCI module in the host CPU cabinet to be initialized on system power-up, and shuts down the CI750 when a power failure occurs within the CIPA or the host CPU cabinet. It also generates interrupts to the interrupt logic for both system power-ups and power failures.

Figure 5-27 illustrates the power control logic. Figures 5-28 and 5-29 show the power-up and power fail sequences respectively. Refer to Figure 5-27 throughout Section 5.12.3.

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

5.12.3.1 Power-up Sequence -- When system power is applied, both the CCI in the host CPU cabinet and the DP in the CIPA cabinet undergo a power-up sequence. Both sequences are illustrated in Figure 5-28. The power-up sequence in the host CPU cabinet is considered first.



Note:
Letter designations in parentheses refer to engineering drawings containing corresponding logic.

Figure 5-27 Power Control Logic

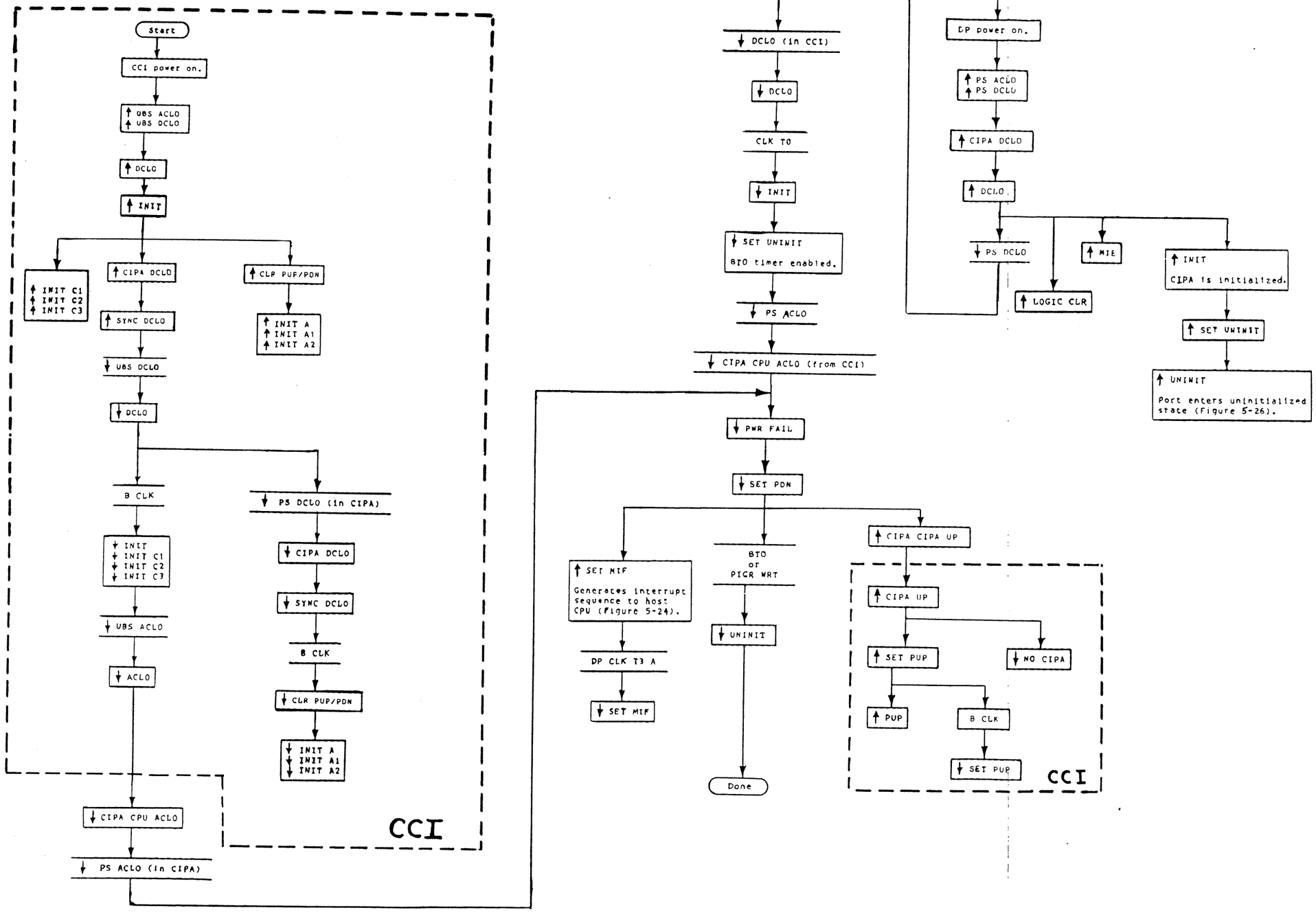


Figure 5-28 Power-up Sequence

The power-up sequence in the host CPU cabinet resets the CCI hardware registers and logic circuits. When power is applied to the CCI, the +5 V becomes operational after which UBS ACLO and UBS DCLO come true.* UBS ACLO is obtained from terminal C45 while UBS DCLO is obtained from terminal C93. UBS DCLO asserts DCLO which in turn resets the INIT flip-flop in the CCI initialize logic (Figure 5-25) thereby asserting INIT. The assertion of INIT causes INIT C1, INIT C2, INIT C3, and CLR PUP/PDN to assert. When CLR PUP/PDN comes true, it causes INIT A, INIT A1, and INIT A2 to assert. DCLO also asserts CIPA DCLO causing SYNC DCLO to come true on the next DELAYED B CLK pulse. SYNC DCLO is applied to the INIT flip-flop which is presently being held reset by DCLO.

* System power protocol requires that +5 V be up and operational before ACLO and DCLO assert.

When UBS DCLO negates in the power-up sequence, DCLO also negates. The negation of DCLO removes the clear signal from the INIT flip-flop conditioning both halves of the flip-flop to set (SYNC DCLO true). The next B CLK pulse sets the INIT flip-flop thereby negating INIT, INIT C1, INIT C2, and INIT C3.

The negation of DCLO also causes CIPA DCLO to negate when the the corresponding signal in the CIPA cabinet (PS DCLO) negates.* When this occurs, SYNC DCLO negates. With SYNC DCLO false, the next B CLK pulse resets the lower half of the INIT flip-flop thereby negating CLR PUP/PDN, INIT A, INIT A1, and INIT A2.

* The purpose of interactive DCLO signals between the CIPA and the CPU cabinet will be seen in the power fail sequence described in Paragraph 5.12.3.2.

UBS ACLO and ACLO negate to complete the CCI power-up. The negation of ACLO negates CIPA CPU ACLO on the CIPA bus.

The power-up sequence in the CIPA cabinet resets the CIPA hardware and logic circuits. When power is applied to the DP, the +5 V becomes operational after which PS ACLO and PS DCLO come true.* PS ACLO is obtained from terminal B10 while PS DCLO is obtained from terminal B5. PS DCLO asserts CIPA DCLO and then DCLO. DCLO sets the MIE bit (maintenance interrupt enable) in the PMCSR thereby enabling interrupts to the host CPU.

* System power protocol requires that +5 V be up and operational before ACLO and DCLO assert.

In addition, DCLO is applied to the DP initialize logic where it asserts INIT and LOGIC CLR. INIT initializes the DP logic and asserts SET UNINIT and UNINIT placing the port into the uninitialized state (see Paragraph 5.12.2.2). LOGIC CLR initializes the PB module.

As the sequence proceeds, PS DCLO negates and when the corresponding signal in the CPU cabinet (DCLO) goes false, DCLO will negate. With DCLO false, the next CLK TØ pulse will reset the INIT flip-flop (Figure 5-25) causing INIT and SET UNINIT to negate. The negation of SET UNINIT enables the BTO timer which then begins counting down the boot time-out period.

The next step in the sequence is the negation of PS ACLO. If CIPA CPU ACLO from the CCI is false (indicating that the host CPU cabinet is powered-up and initialized), then the negation of PS ACLO causes PWR FAIL and then SET PDN to negate.

The negation of SET PDN causes SET MIF to assert until the next DP CLK T3 A clock pulse (the next DP CLK T3 A pulse resets the SET MIF flip-flop negating SET MIF). The SET MIF pulse is applied to the interrupt logic to generate an interrupt sequence to the host CPU (Paragraph 5.12.1).

The negation of SET PDN also causes CIPA CIPA UP to assert on the CIPA bus and then CIPA UP to assert in the CCI. The true state of CIPA UP causes the NO CIPA bit in the configuration register to reset and SET PUP to assert. SET PUP directly sets the PUP flip-flop thereby asserting the PUP bit in the configuration register. The AND gate that generates SET PUP receives two input signals. One is the R2 input to flip-flop E162; the other is the inverse of the R3 output of E162. Thus SET PUP is negated one B CLK after it is asserted, forming a 200 ns pulse.

When the boot timer startup delay has timed out (BTO asserts) or the host asserts PICR WRT via an unsolicited CMI write operation, UNINIT negates and the port leaves the uninitialized state (Paragraph 5.12.2.2).

Note in Figure 5-28 that the negation of PWR FAIL (and then SET PDN) indicates the completion of a successful power-up sequence. Only then is the PUP bit set and the CPU interrupted. For PWR FAIL to negate, both the CPU cabinet and the CIPA must have completed good power-ups as indicated by the false state of CIPA CPU ACLO from the CPU cabinet and PS ACLO in the DP.

5.12.3.2 Power Fail Sequence (Figure 5-29) -- When a power failure occurs within the CIPA cabinet, PS ACLO comes true on terminal B1Ø in the DP and asserts PWR FAIL. PWR FAIL is also asserted by a power failure within the host CPU cabinet. A power failure in the CPU cabinet asserts UBS ACLO on terminal C45 in the CCI. With T ACLO false (discussed in Paragraph 5.12.3.3), ACLO asserts thereby causing CIPA CPU ACLO to assert on the CIPA bus.

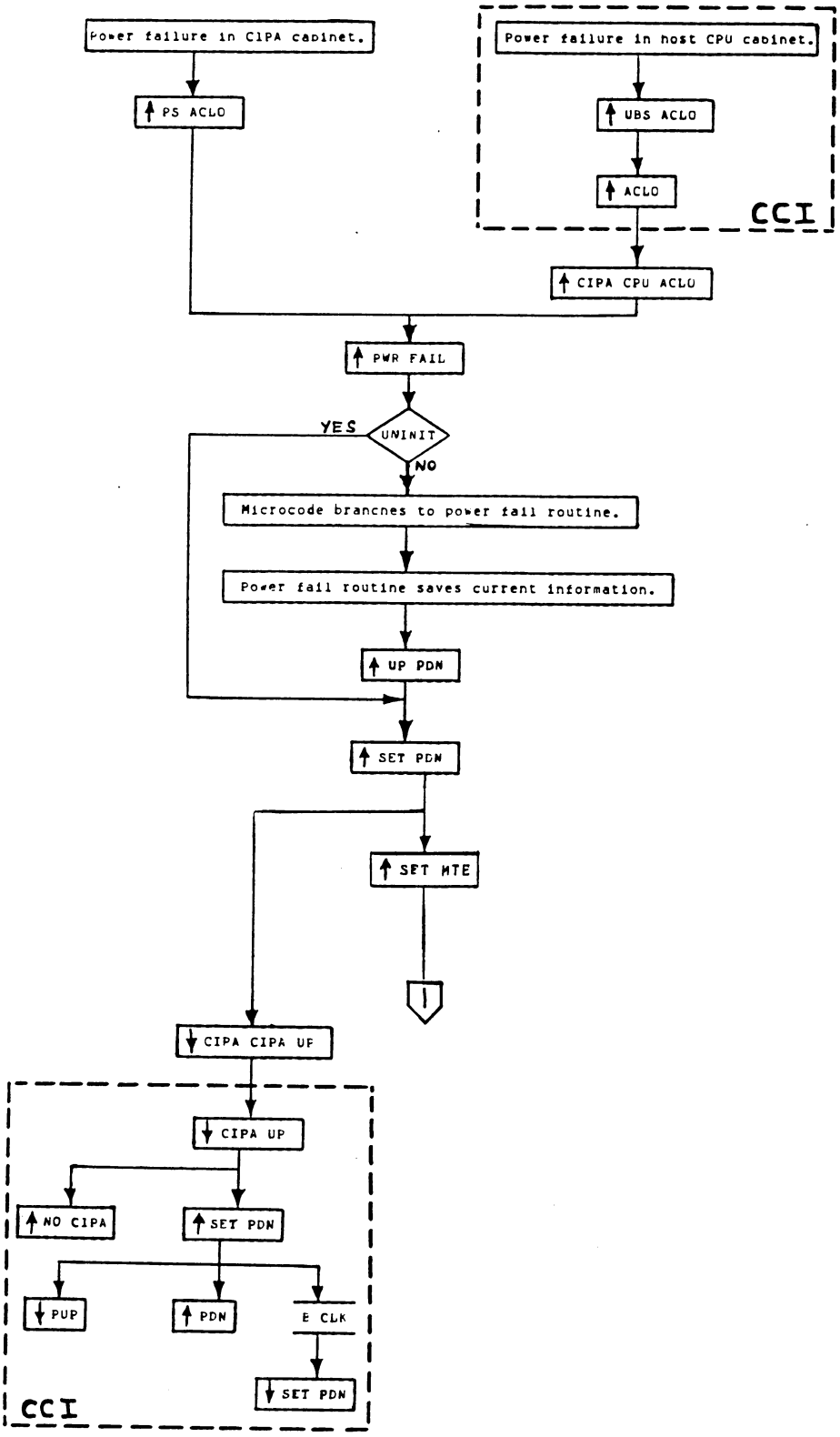


Figure 5-29 Power-Fail Sequence (Sheet 1 of 2)

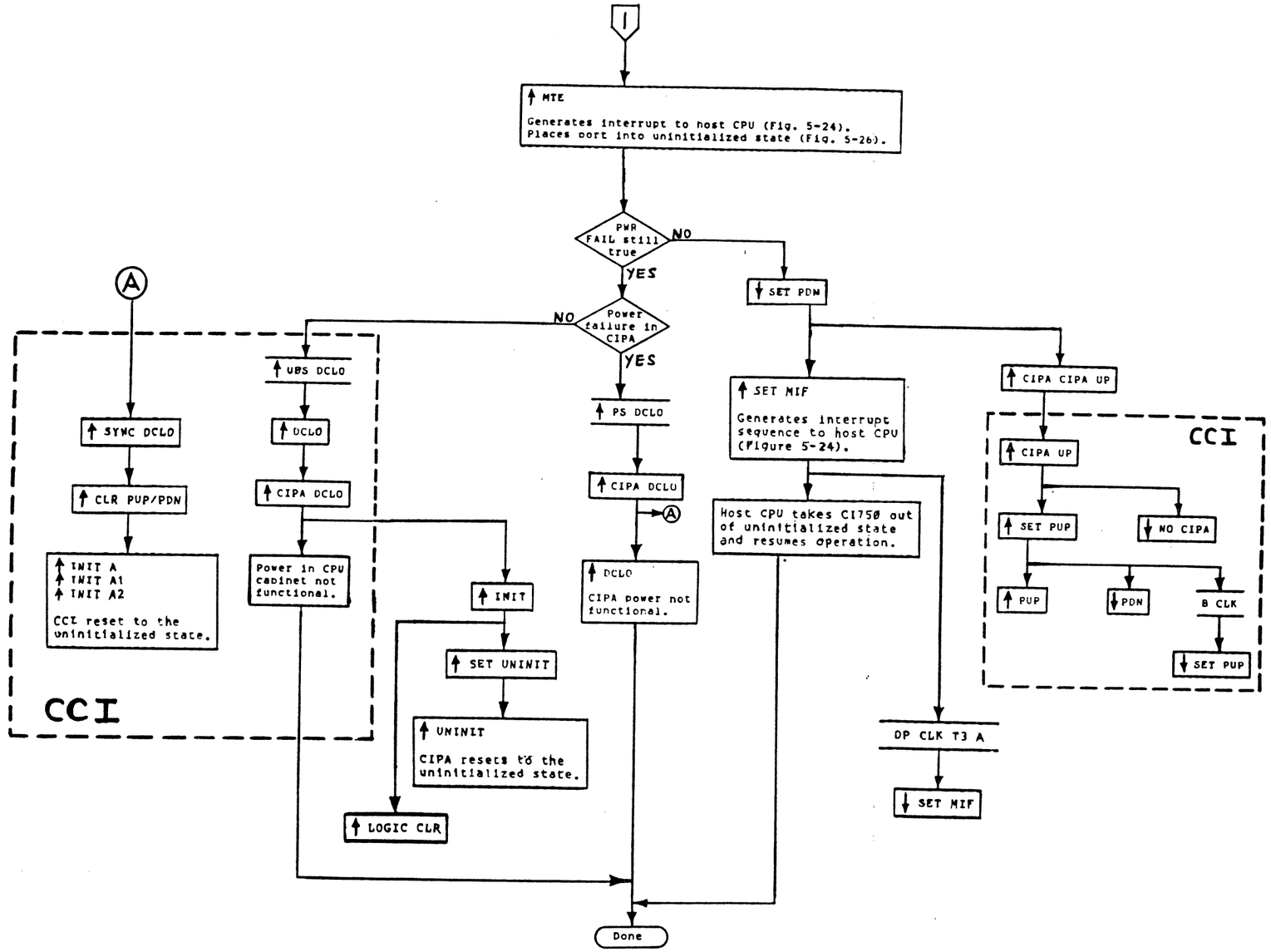


Figure 5-29 Power-Fail Sequence (Sheet 2 of 2)

PWR FAIL is applied to the microcode branching logic. If the port is not in the uninitialized state (UNINIT false) the microcode is running. PWR FAIL causes the microcode to branch to a power fail routine. The power fail routine functions to save current information so that operation may be resumed when power returns. The power fail routine returns UP PDN to the power fail logic resulting in the assertion of SET PDN.

If the port is in the uninitialized state (UNINIT true) when PWR FAIL asserts, SET PDN is asserted by the hardware - not the microcode. Referring to Figure 5-27, with UNINIT true, the assertion of PWR FAIL directly asserts SET PDN.

SET PDN is inverted and placed onto the CIPA bus as a negated CIPA CIPA UP signal. The false state of CIPA CIPA UP causes CIPA UP in the CCI to negate. The negation of CIPA UP is transferred through two flip-flops causing R2 to negate and NO CIPA and SET PDN to assert. SET PDN sets the PDN flip-flop asserting PDN, and resets the PUP flip-flop negating PUP.

SET PDN is asserted by ANDing NO CIPA and the R3 output of flip-flop E162. The next B CLK resets E162 thereby negating SET PDN. Thus SET PDN is formed into a 200 ns pulse.

PDN, PUP, and NO CIPA are bits in the CCI configuration register.

Back in the DP, SET PDN is applied to the interrupt logic where it asserts SET MTE and then MTE to generate an interrupt to the host CPU (Paragraph 5.12.1).

MTE is coupled from the interrupt logic to the initialization logic (Paragraph 5.12.2.2) where it asserts UNINIT to stop the microcode and place the port into the uninitialized state.

If a genuine power interruption is occurring, PWR FAIL will still be true and, if this is a CIPA power failure, PS DCLO will assert. When PS DCLO comes true it asserts CIPA DCLO. CIPA DCLO asserts DCLO indicating that power in the CIPA cabinet is no longer functional.

CIPA DCLO is also transferred to the CCI where it asserts SYNC DCLO, CLR PUP/PDN, and the "A" set of initialization signals (INIT A, INIT A1, INIT A2). CLR PUP/PDN and the "A" set of initialization signals clears the error bits in the CNFGR register and resets the CCI logic (except the FPLA logic array circuitry).

If the power failure occurred in the CPU cabinet, UBS DCLO will be asserted. UBS DCLO in turn asserts DCLO and CIPA DCLO. CIPA DCLO indicates that power in the CPU cabinet is no longer functional.

CIPA DCLO is also transferred to the DP where it assert INIT, SET UNINIT, UNINIT, and LOGIC CLR to reset the CIPA to the uninitialized state. Thus, due to the DCLO signal being interactive between the CPU cabinet and the CIPA, a power loss in one cabinet will reset the port module(s) in the other cabinet.

If only a transient AC power dip occurred, PWR FAIL may negate before PS DCLO asserts. In this case, the negation of PWR FAIL causes SET PDN to negate.

The negation of SET PDN asserts CIPA CIPA UP on the CIPA bus causing CIPA UP to assert in the CCI. The assertion of CIPA UP is transferred through two flip-flops causing R2 to assert. The assertion of R2 negates NO CIPA and asserts SEP PUP. SET PUP sets the PUP flip-flop asserting PUP, and resets the PDN flip-flop negating PDN.

SET PUP is asserted by ANDing the R2 input and the inverse of the R3 output of flip-flop E162. The next B CLK sets E162 thereby negating SET PUP. Thus SET PUP is formed into a 200 ns pulse.

The negation of SET PDN also causes SET MIF to assert until the next DP CLK T3 A clock pulse (the next DP CLK T3 A pulse resets the SET MIF flip-flop negating SET MIF). SET MIF is applied to the interrupt logic where it generates an interrupt to the host CPU just as during a normal power-up sequence. The CPU responds to the interrupt by taking the CI750 out of the uninitialized state (via an unsolicited CMI write sequence) and resuming normal operation.

Figure 5-30 illustrates when interrupt signals SET MTE and SET MIF occur with respect to SET PDN. When SET PDN asserts (due to a power failure), SET MTE asserts to generate an interrupt and place the port into the uninitialized state. When SET PDN negates (as a power-up sequence completes), SET MIF asserts to generate an interrupt.

5.12.3.3 Remote Reset Function -- In the maintenance mode the CI750 port can be reset from another node by means of a reset packet. The remote node sends the reset packet to the CI750 causing the port microcode to assert ASSERT FAIL and PF VLD (power fail valid) (Figure 5-27). ASSERT FAIL conditions a fail flip-flop to set while PF VLD gates the T1 clock to set the flip-flop. Setting the flip-flop asserts E62-5 which in turn asserts CIPA T ACLO on the CIPA bus. CIPA T ACLO is coupled to the CCI where it is synced by DELAYED B CLK to produce SYNC T ACLO. SYNC T ACLO then asserts T ACLO via flip-flop E164. SYNC T ACLO and T ACLO are ANDed with the inverse of PFD (power fail disable) and the inverse of PDN. When PFD and PDN are false (discussed later), the AND gate is enabled and UBUS ACLO is asserted on terminal C45 and then out to the CPU cabinet. UBUS ACLO functions to initiate a simulated power-down sequence within the host system.

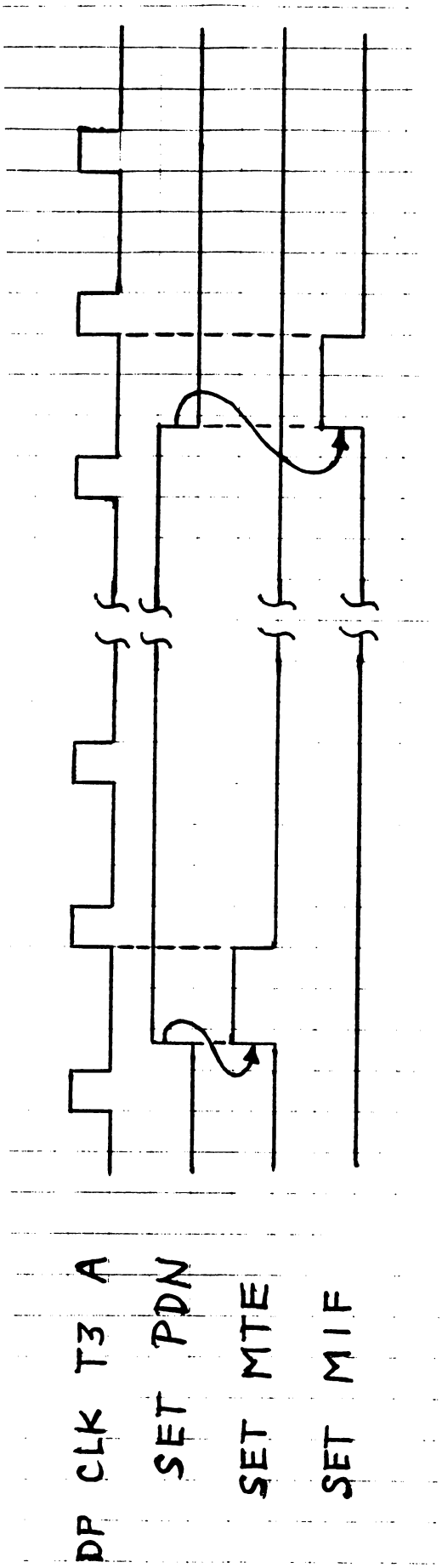


Figure 5-30 Power-Fail And Power-UP Interrupt Signals

The microcode then asserts ASSERT DEAD and PF VLD which similarly results in the assertion of SYNC T DCLO and T DCLO. The assertion of SYNC T DCLO and T DCLO causes UBUS DCLO to assert on terminal C93. The assertion of UBUS DCLO completes the simulated power-down sequence within the host system.

Note that the true state of T ACLO and T DCLO inhibit ACLO and DCLO from being asserted to the CCI and DP power down logic thereby preventing the logic from shutting down the port. Thus the port is still able to function while the host system is in the simulated powered-down state.

The microcode then asserts PF VLD with ASSERT DEAD negated resulting in the resetting of the dead flip-flop and the negation of SYNC T DCLO, T DCLO and UBUS DCLO in the CCI. With UBUS DCLO false, the host system attains a partial powered-up state.

When SYNC T DCLO negates, T DCLO is held true until the next B CLK pulse resets flip-flop E164. This insures that UBUS DCLO has negated before T DCLO goes false thereby preventing UBUS DCLO from possibly causing a spurious assertion of DCLO to the CCI and the DP.

The host system remains in the partial powered-up state (UBS ACLO still true) 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 SYNC T ACLO, T ACLO and UBUS ACLO in the CCI.

A delay of one B CLK between the negation of SYNC T ACLO and T ACLO prevents a possible spurious assertion of ACLO to the CCI and the DP.

With UBS ACLO false, the host system is now powered-up and the simulated power- fail sequence is completed.

T ACLO and T DCLO are bits in the CCI configuration register.

When the PFD bit in the configuration register is true, the T ACLO/T DCLO logic is inhibited from asserting UBUS ACLO and UBUS DCLO. This allows maintenance diagnostics to test the microword ASSERT FAIL and ASSERT DEAD bits without activating a power-fail sequence in the host CPU.

The true state of PDN also inhibits the T ACLO and T DCLO logic from asserting UBS ACLO and UBS DCLO. Thus, when the port is powered down (PDN true), inputs from the T ACLO and T DCLO logic do not disrupt the port power-up sequence.

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 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 in this chapter are keyed to the engineering circuit schematics (CS prints) by letter designation in parentheses. The letters specify the CS sheet that contains the 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 Overview

The CMI CIPA interface (CCI) module serves as an interface between the CMI (CPU memory interconnect) bus and the port logic of the CI750. The module follows CMI protocol and timing while interfacing with the CMI bus, and responds to port timing and signal formats when interfacing with the DP module in the CIPA cabinet.

The overview begins with a description of the CMI bus, CMI bus signals, and CMI bus timing. The overview then provides a block diagram of the CCI with a brief description of the CCI major components and their functions. Lastly, simplified flow diagrams are used to illustrate functioning of the major components during port initiated data transfers and unsolicited CMI transfers.*

* An unsolicited CMI transfer is a transfer wherein the CI750 is a slave (transfer not initiated by CI750).

In addition to serving as an overview of CCI operation, the simplified flow diagrams are related to flow diagrams found in other sections of Chapter 6. These sections expand on each area of the overview flows with more detailed diagrams and text as to how each of the areas performs its function. Consequently the overview flows are used, along with the CCI block diagram, throughout the rest of the chapter.

6.1.1 CMI Protocol

The CMI is a 45 line synchronous, interlocked communication bus. The bus is interlocked in that when a master and slave are communicating, the bus is locked out to other nexus.

The CI750 port is a master for all port initiated bus transfers (including interrupts). It is a slave for all unsolicited transfers.

6.1.1.1 Bus Signals -- The 45 bus lines are illustrated in Figure 6-1 and described in Table 6-1.

Table 6-1 CMI Bus Signals

No. of Lines	Name	Mnemonic	Function
32	Data/Address	CMI DATA <31:00>	Thirty two multiplexed lines that carry four bytes of data or a 32-bit command/address longword. The format of the data longword and the command/address longword is illustrated in Figure 6-2.
1	Busy	CMI DBBZ	Indicates that the CMI is busy doing a master/slave transfer.
1	Hold	CMI HOLD	The CPU asserts CMI HOLD to perform high priority transaction with a nexus. HOLD inhibits all other nexus from obtaining the CMI bus.

Table 6-1 CMI Bus Signals (Cont)

No. of Lines	Name	Mnemonic	Function
1	Wait	CMI WAIT	CMI WAIT is asserted by a nexus that is issuing an interrupt to the CPU. WAIT informs the CPU that an interrupt transaction is about to occur.
7	Arbitration	CMI ARB <7:1>	Arbitration for the CMI bus follows a distributive priority scheme. Each nexus (except the CPU) is assigned a priority ARB line (ARB 7 = highest priority). Among arbitrating nexus, the one with the highest arbitration priority will obtain the bus. Each nexus monitors all higher priority ARB lines. If a higher priority nexus is not arbitrating for the CMI bus, and the bus is free (CMI DBBZ and CMI HOLD false), a nexus can obtain the bus. The CPU is not assigned a CMI ARB line and thus becomes the lowest priority nexus on the CMI (ARB 0).

Table 6-1 CMI Bus Signals (Cont)

No. of Lines	Name	Mnemonic	Function
2	Status	CMI STATUS <1:Ø>	Each slave that is addressed by a master, return a two-bit status code to the master at the end of the transaction. The code indicates the success or failure of the transaction as shown below.

CMI STATUS 1 Ø	Meaning
Ø Ø	NXM (non-existent memory) (This is equivalent to no response.)
Ø 1	UCE (uncorrectable error)
1 Ø	CRD (corrected read data)
1 1	No error

1	Bus clock	CMI B CLK	B CLK is the bus clock that synchronizes all CMI bus activity. B CLK is a 6.25 MHz clock with a period (bus cycle) of 160 ns.

Referring to the command/address format in Figure 6-2B, note that bit 24 is always zero.

Also, that address bits <Ø1:ØØ> are not used because all CMI addresses are longword aligned. If only a portion of the data longword is of interest, the byte mask specifies which bytes are valid.

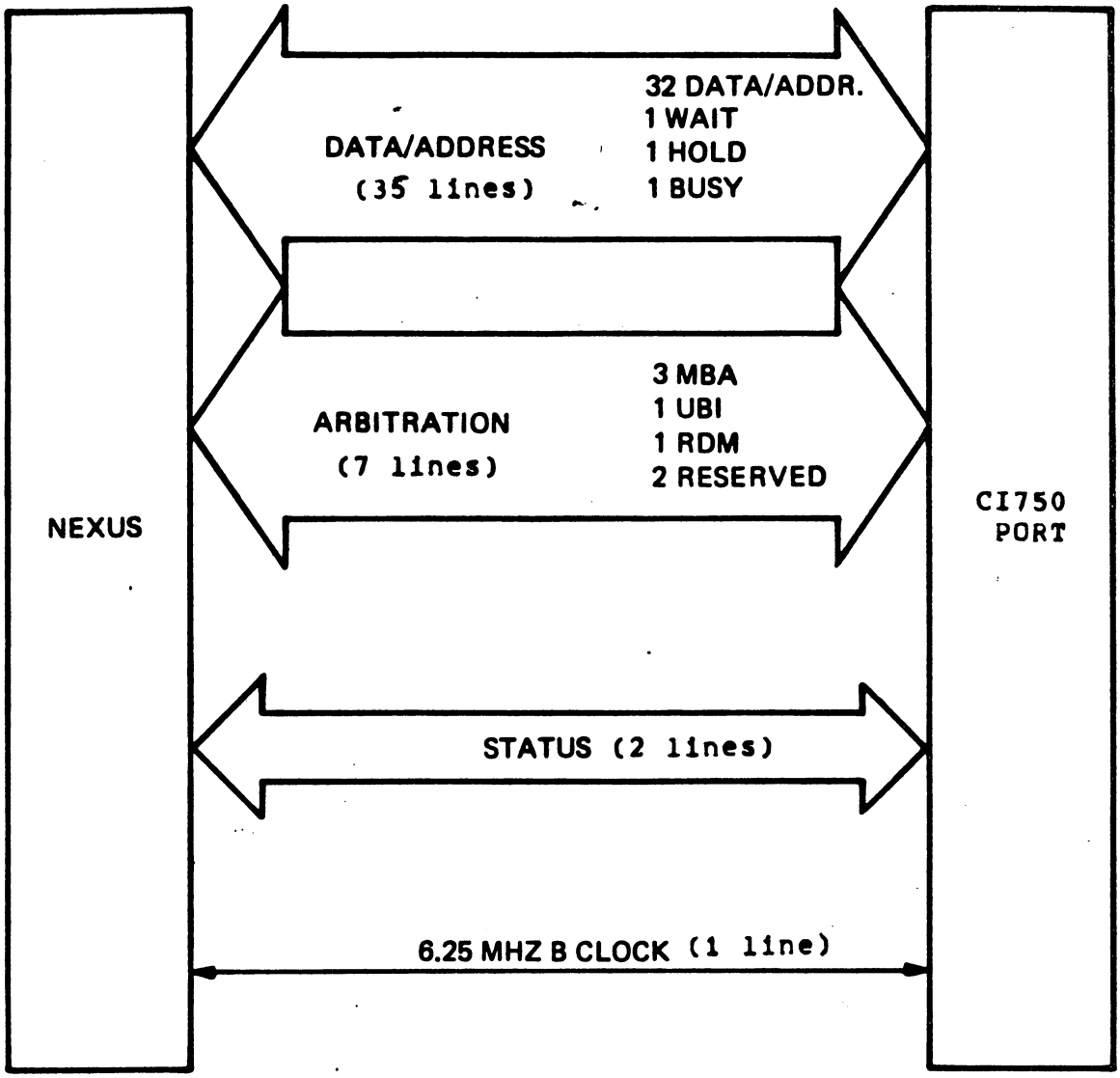
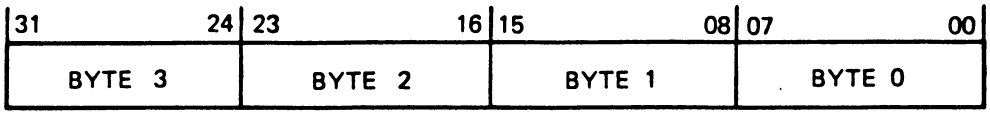
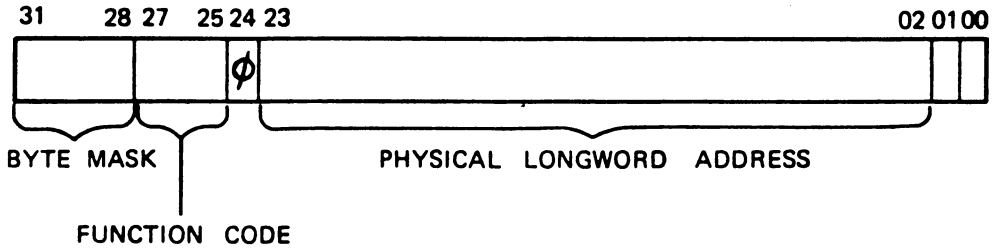


Figure 6-1 CMI Bus Signals



A. Data Format



B. Command/Address Format

Figure 6-2 CMI Data and Command/Address Formats

The three-bit function code shown in Figure 6-2B is defined in Table 6-2.

Table 6-2 Function Code

Function Bits 27	26	25	Function	Port Initiated Operation (CI750 is master)	Unsolicited Operation (CI750 is slave)
0	0	0	Read	A CMI device is read by CI750.	CI750 is read by the CPU.
0	0	1	Read Lock	Same as "read" plus all other CMI devices are locked off the CMI until the CI750 executes a "write unlock" function.	Treated as a "read" by the CI750.
0	1	0	Read With Modify Intent	Cannot be initiated by the CI750.	Treated as a "read" by the CI750.
0	1	1	Undefined	----	----
1	0	0	Write	A CMI device is written by the CI750.	The CI750 is written by the CPU.
1	0	1	Write Unlock	This function follows a "read lock" function. Same as a "write" plus all other CMI devices are unlocked so they can access the CMI.	Treated as a "write" by the CI750.
1	1	0	Write Vector	The CI750 writes an interrupt vector to the CPU.	Not applicable.
1	1	1	Undefined	----	----

6.1.1.2 Write Timing -- Timing for a CMI write transfer is illustrated in Figure 6-3.

The command/address cycle is the first bus cycle that occurs after the CMI master has won control of the bus. In the command/address cycle, the master nexus asserts DBBZ and places the command/address onto the CMI data/address lines.

In the next bus cycle, the master negates DBBZ, removes the command/address from the CMI, and places the write data onto the data/address lines. The slave nexus asserts DBBZ to hold the bus until it is ready to take the data. As seen in Figure 6-3, this could take more than one bus cycle.

When the slave is ready to take the write data, it negates DBBZ, takes the write data off the bus, and places status on the bus for the CMI master. The bus cycle following the negation of DBBZ is the last cycle of the write transfer and is designated as the status cycle. Note that the write data is still on the CMI during the status cycle.

If the slave is immediately ready to take the write data from the CMI, the transfer completes in only two bus cycles (command/address and status). In this case the slave takes the data and returns status, but does not assert DBBZ.

In a write transfer, only two of the four status states are used. These are "no error" and NXM (either the slave responded or it didn't). The two data error states are not applicable.

6.1.1.3 Read Timing -- Timing for a CMI read transfer is illustrated in Figure 6-4.

The command/address cycle is the first bus cycle that occurs after the CMI master has won control of the bus. In the command/address cycle, the master nexus asserts DBBZ and places the command/address onto the CMI data/address lines.

In the next bus cycle, the master negates DBBZ, removes the command/address from the CMI, and waits for the slave nexus to return the read data. The slave asserts DBBZ to hold the bus until it is ready to place the read data onto the bus. As seen in Figure 6-4, this could take more than one bus cycle.

When the slave is ready to place the read data onto the CMI, it negates DBBZ and places the read data and status onto the bus for the CMI master. The bus cycle following the negation of DBBZ is the last cycle of the read transfer and is designated as the status cycle.

If the slave is immediately ready to place the read data onto the CMI, the transfer completes in only two bus cycles (command/address and status). In this case the slave places the read data and status onto the CMI, but does not assert DBBZ.

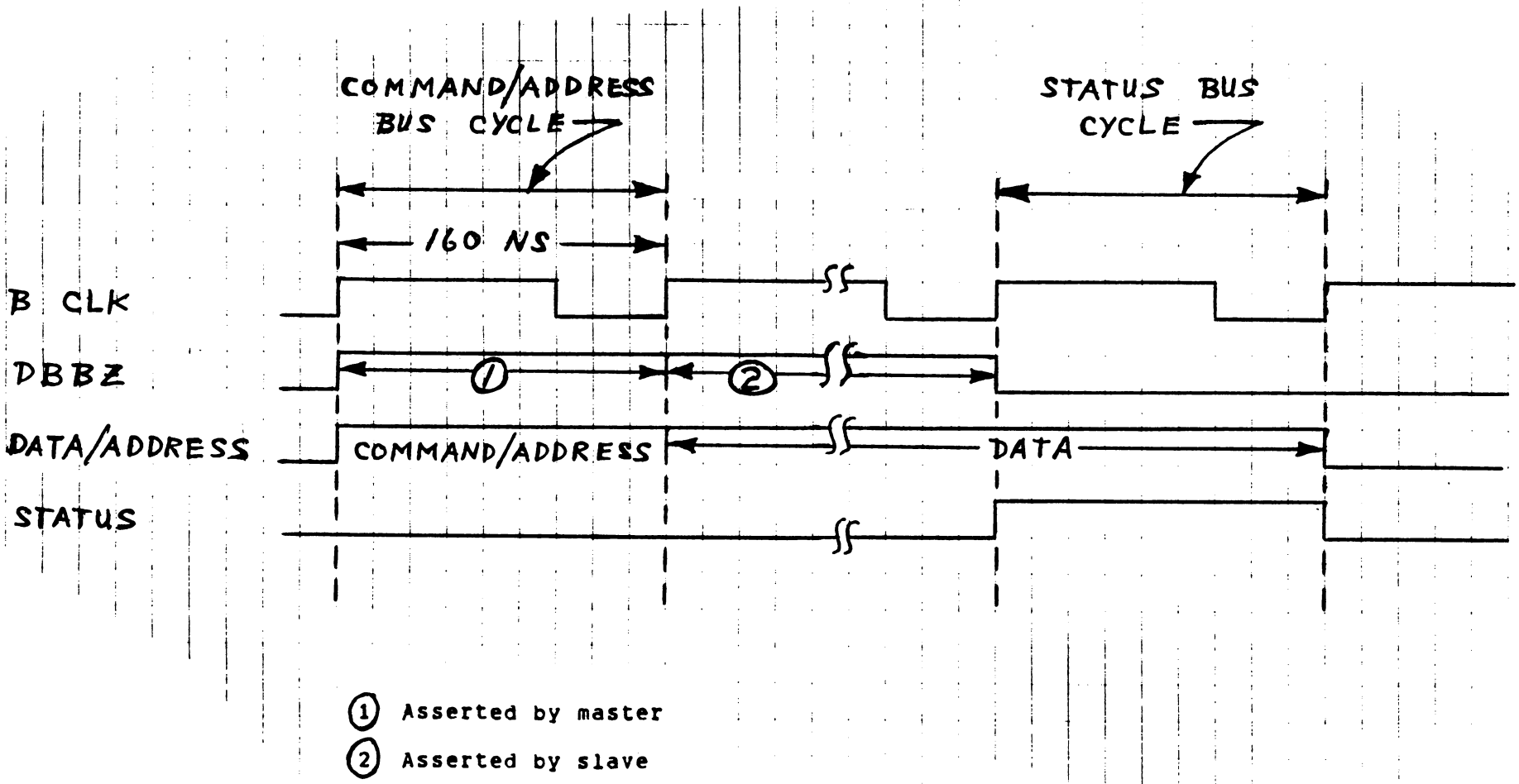


Figure 6-3 CMI Write Timing

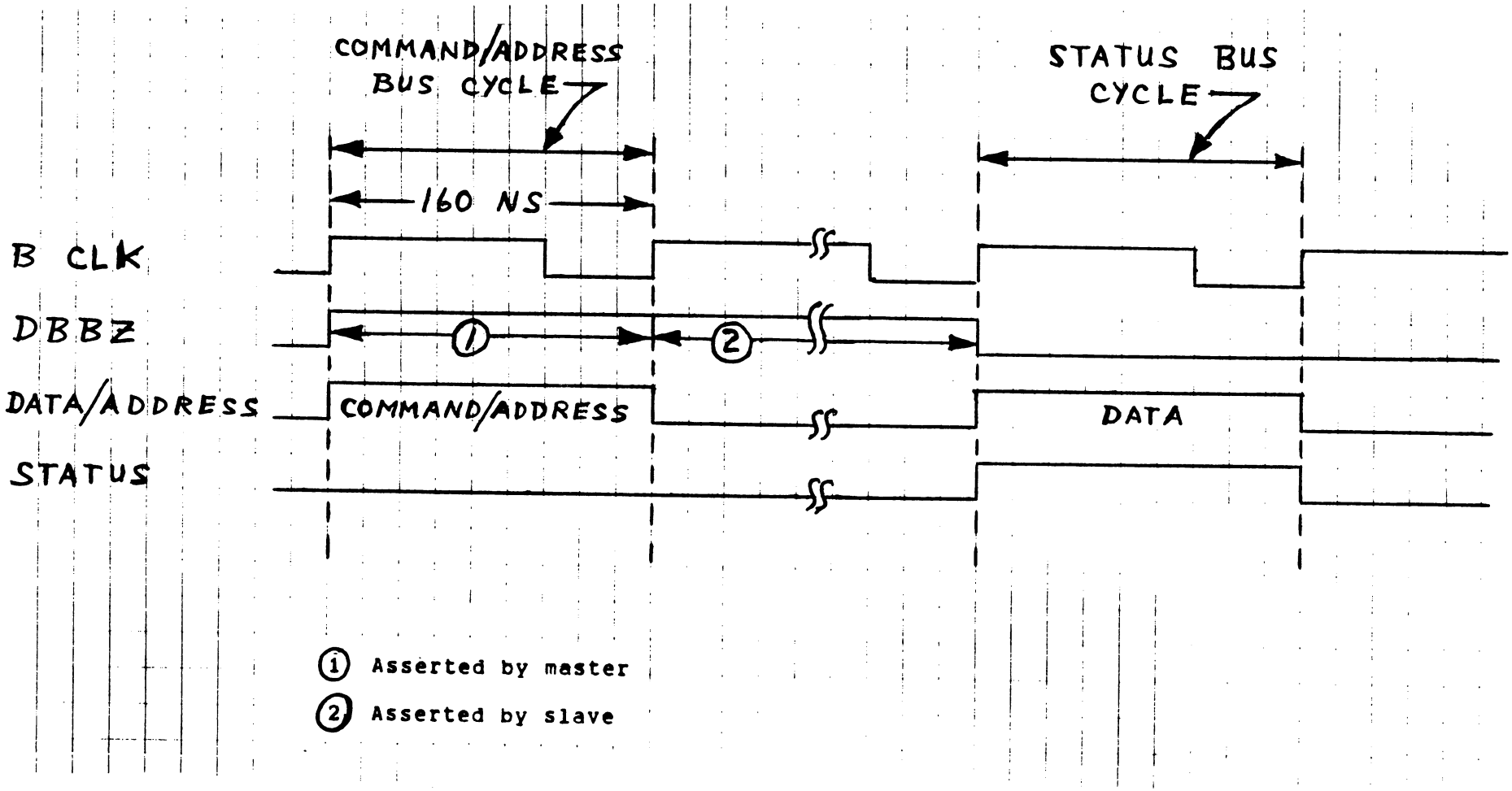


Figure 6-4 CMI Read Timing

In a read transfer, all four of the status states are used. The two data error states refer to the read data that is being transferred to the master.

6.1.1.4 Write Vector Timing -- Timing for a CMI write vector function is illustrated in Figure 6-5.

The command/address cycle is the first bus cycle that occurs after the interrupting nexus (master) has won control of the bus. In the command/address cycle the master (CI750) asserts DBBZ and places the interrupt vector onto the CMI data/address lines. CMI WAIT is already true, having been asserted by the interrupting nexus when the interrupt was initially generated.

During the next bus cycle (status cycle), the master negates DBBZ while the slave (CPU) returns status to the master. In an interrupt transaction, the CPU always returns a "no error" status. Note that the interrupt vector is on the CMI during both the command/address and status cycles.

6.1.2 Major Components

The major components of the CCI are shown in Figure 6-6 and listed below. This is followed by a description of each component.

1. Command/Address Hi Register
2. Address Lo Register
3. Byte Mask Register
4. XMIT File
5. Return Read Data Register
6. Interrupt Vector
7. CNFGR Register
8. CMI Mux
9. Address Decode Logic
10 Command/Address Hold Register
11. Address Offset Register
12. Function Register
13. Receive Write Data Register
14. RCV File

6.1.2.1 Command/Address Hi Register -- The command/address hi register receives the command and the high order address bits from the DP. The register contains two function bits and the six upper address bits. The register outputs onto the CMDADDR bus.

6.1.2.2 Address Lo Register -- The address lo register receives the 16 low order address bits from the DP. The lower eight bits of the register form a counter that is incremented to change the address for each transfer without reloading. The register outputs onto the CMDADDR bus.

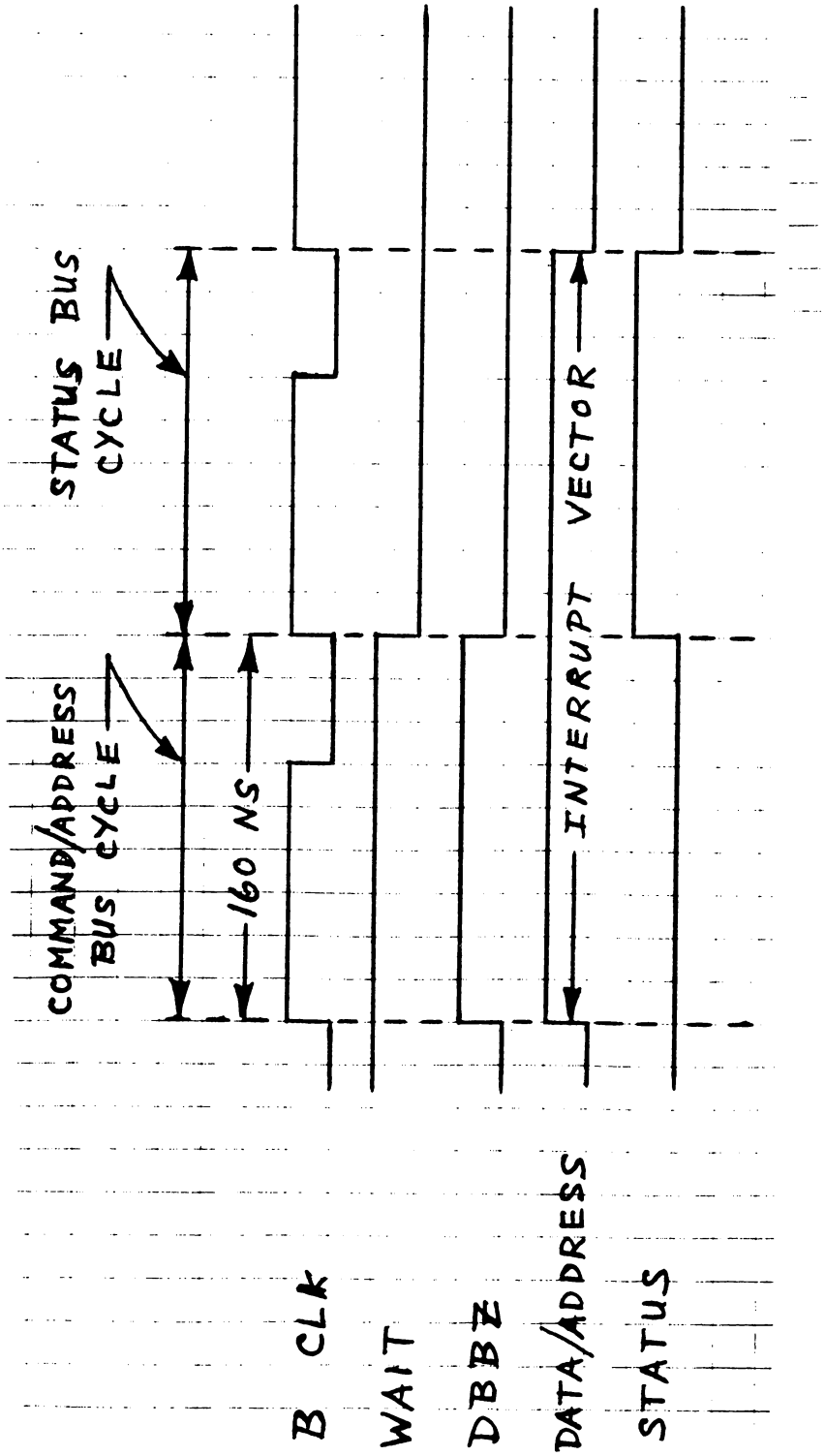
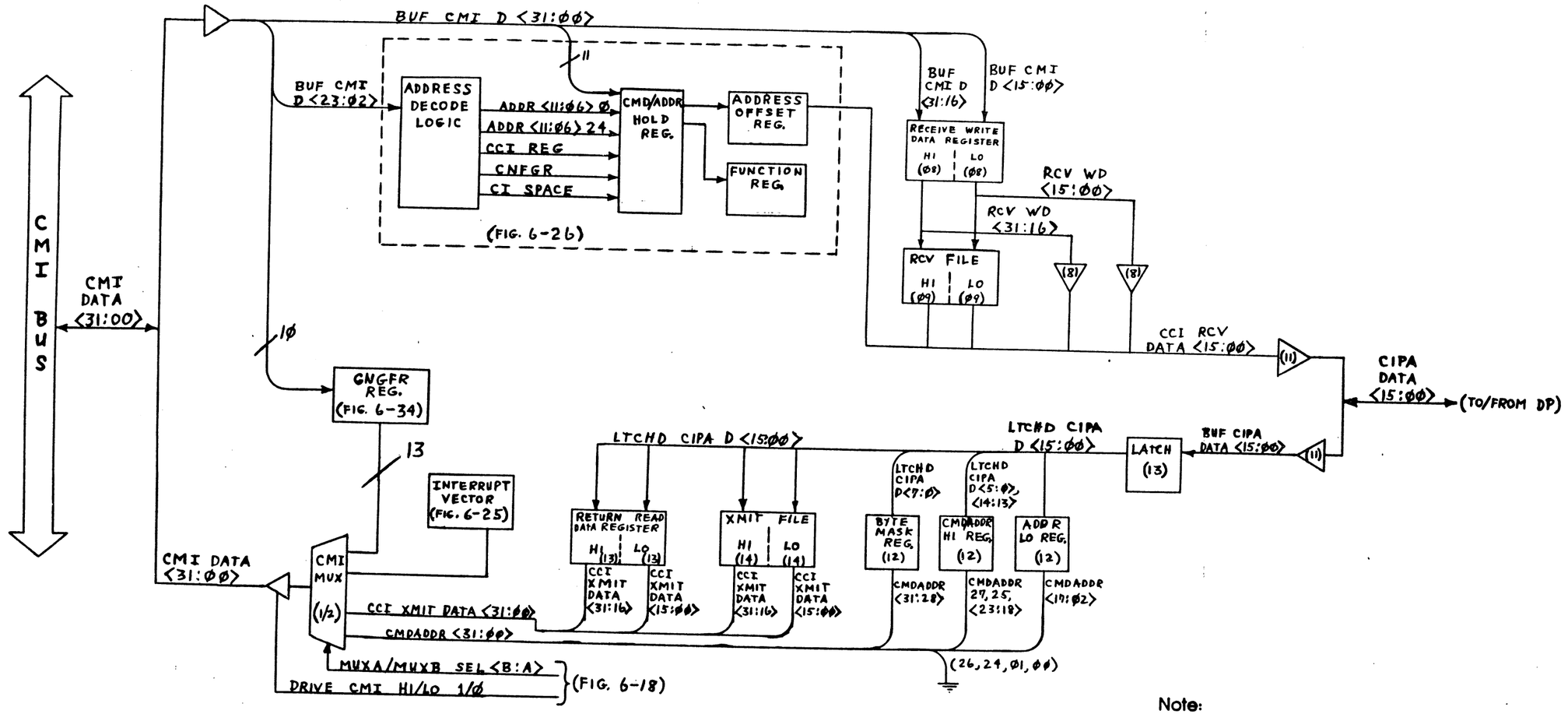


Figure 6-5 CMi Write Vector Timing

Figure 6-5 CMI Write Vector Timing



Note:
Letter designations in parentheses refer to engineering drawings containing corresponding logic.

Figure 6-6 CCI Block Diagram

6.1.2.3 Byte Mask Register -- The byte mask register receives the byte mask from the DP. The register is eight bits wide capable of holding two 4-bit byte masks. This capability is used for quadword data transfers. The register outputs onto the CMDADDR bus.

6.1.2.4 XMIT File -- The XMIT file is used as a buffer during port initiated write operations. The file stores the write data from the DP that is to be transferred to the CMI. The CCI unloads the XMIT file out to the CMI under CMI protocol.

The size of the XMIT file is 4 x 32; capable of holding four longwords. Buffering is implemented by dividing the file into halfs (A and B) with two longwords in each half. While one half is being loaded with data from the DP, the other half can be unloaded out to the CMI.

The file is loaded from a 16-bit LTCHD CIPA D bus hence, each longword location requires two load cycles to be filled. The file outputs onto the 32-bit CCI XMIT DATA bus.

6.1.2.5 Return Read Data Register -- The Return Read Data Register is used during CMI unsolicited read operations. The register receives data read from the DP that is to be transferred to the CMI. The CCI unloads the register out to the CMI under CMI protocol.

The Return Read Data Register is 32-bits wide. The register is loaded from the 16-bit LTCHD CIPA D bus hence, two load cycles are required. The register outputs onto the 32-bit CCI XMIT DATA bus.

6.1.2.6 Interrupt Vector -- The interrupt vector circuit supplies the interrupt vector to the CMI mux during a write vector function.

6.1.2.7 CNFGR Register -- The CNFGR register contains status, error, and control bits associated with operation of the CCI and the DP. The output of the CNFGR register is one of the selectable inputs to the CMI mux.

6.1.2.8 CMI Mux -- The CMI mux selects one of four data sources for the data/address lines on the CMI bus. The four data sources are:

1. Command/address bus (CMDADDR <31:00>)
2. Transmit data bus (CCI XMIT DATA <31:00>)
3. CNFGR register
4. Interrupt vector

6.1.2.9 Address Decode Logic -- The address decode logic decodes the command/address longword received during the command/address cycle of an unsolicited CMI operation. Outputs from the address decode logic indicate:

1. If the command/address longword is addressed to the CI750.

2. If the reference is to the CCI or the DP.

3. If the reference is a diagnostic maintenance function.

The outputs of the address decode logic are applied to the command/address hold register.

6.1.2.10 Command/Address Hold Register -- The command/address hold register is used to latch all the data received during the command/address cycle of an unsolicited CMI operation. The latched data is used during the execution of the unsolicited operation. The latched data includes all the outputs from the address decode logic as well as address and function data from the CMI. The data in the command/address hold register is transferred to the address offset register and the function register.

6.1.2.11 Address Offset Register -- The address offset register is used during an unsolicited CMI access to the DP. The register holds the offset address (offset from the CI750 base address) of the DP register to be accessed. The register receives the offset address from the command/address hold register.

6.1.2.12 Function Register -- The function register is used during an unsolicited CMI operation. The register holds function data received from the command/address hold register.

6.1.2.13 Receive Write Data Register -- The Receive Write Data Register latches the data off the buffered CMI data lines during the CMI status cycle. With the CMI data latched, the CMI bus can be released to allow another nexus to make a bus transfer.

The Receive Write Data Register is 32-bits wide and is longword loaded from the BUF CMI D bus. The full 32-bit register output is available to the RCV file. The register high word and low word outputs are coupled to the 16-bit CCI RCV DATA bus. Hence to output the register onto the CCI RCV DATA bus, two unload cycles are required.

6.1.2.14 RCV File -- The RCV file is used as a buffer during port initiated read operations. The file stores the read data obtained from the CMI (via the Receive Write Data Register) that is to be transferred to the DP. The DP unloads the RCV file when the CCI indicates that data is available in the file. Unloading of the RCV file is done under DP control.

The size of the RCV file is 4 x 32; capable of holding four longwords. Buffering is implemented by dividing the file into halfs (A and B) with two longwords in each half. While one half is being loaded with data from the CMI, the other half can be unloaded out to the DP.

The RCV file is longword loaded from the Receive Write Data Register. The RCV file high word and low word outputs are coupled to the 16-bit CCI RCV DATA bus. Hence, each longword location requires two unload cycles to be read out.

6.1.3 Simplified Flow Diagrams

Simplified flow diagrams are provided illustrating port initiated transfers, write vector functions, and unsolicited CMI transfers. The blocks of the flow diagram represent functions that occur in the execution of the transfer. Each block (or block area enclosed in dotted lines) includes a descriptive title and/or figure number. The title and/or figure number key the block into a section of Chapter 6 which describes in detail how the function represented by the block is carried out. Thus the simplified flows should be referenced throughout the rest of Chapter 6.

It can be seen that some of the flow diagram blocks appear more than once. This indicates that the function represented by the block repeats itself in the various types of transfers.

The CCI major components described in Paragraph 6.1.2 and illustrated in Figure 6-6 are also included in the flow diagram discussions. Therefore Figure 6-6 should also be referenced throughout the rest of Chapter 6.

6.1.3.1 Port Initiated Transfers -- A port initiated transfer is a CMI data transfer, initiated by the port in which the CI750 port is the bus master. Figure 6-7 is a simplified flow diagram of a port initiated transfer illustrating the major steps in the sequence. The illustration includes both a write and a read operation.

The port microcode initiates the transfer by loading the CMD/ADDR HI, the ADDR LO, and the byte mask registers. The three registers are loaded from the DP via the CIPA bus and the 16-bit LTCHD CIPA D bus. A separate write cycle is required to load each of the three registers.

At this point the flow sequence divides according to whether a write or read operation is being executed.

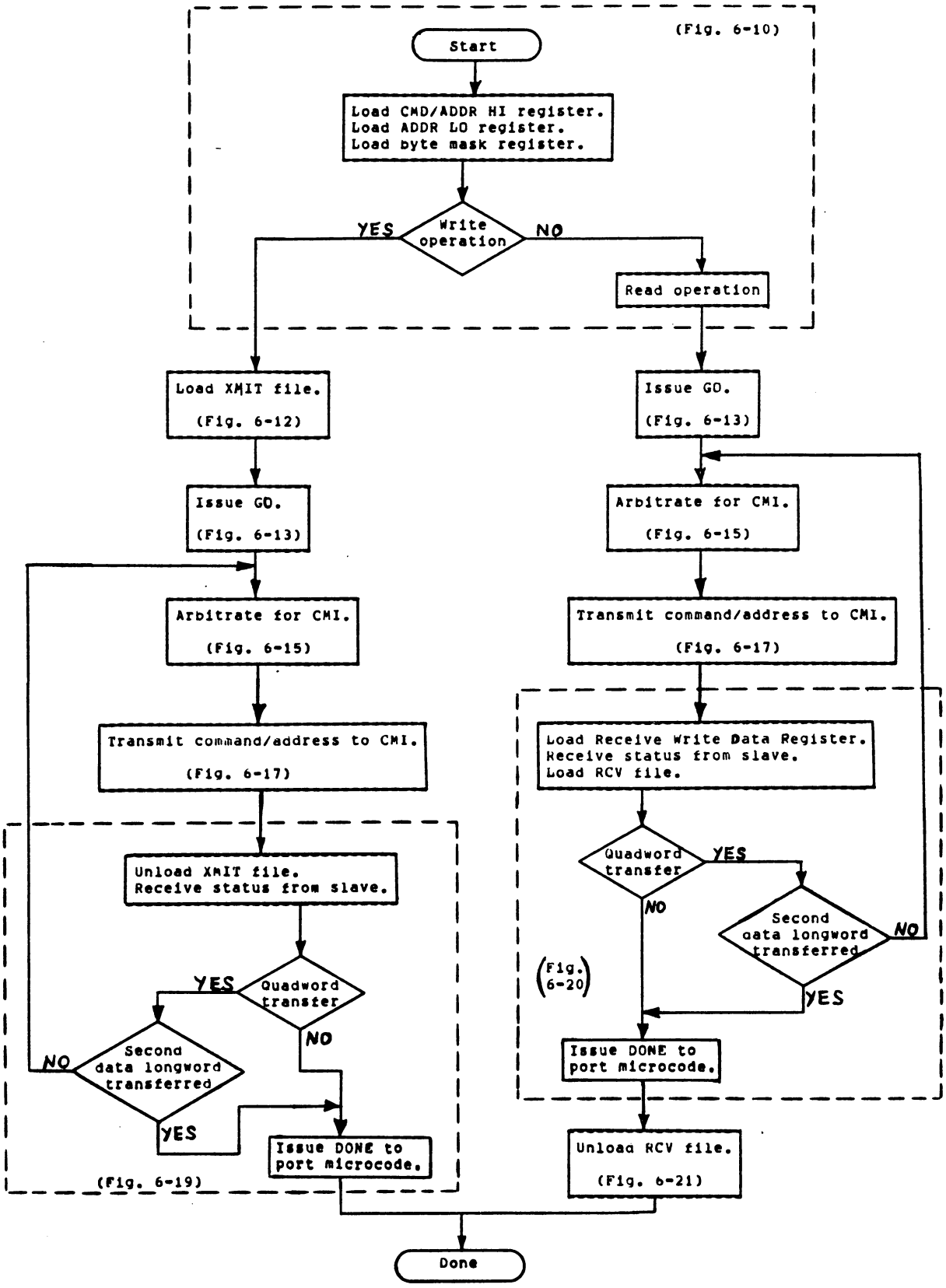


Figure 6-7 Flow Diagram of Port Initiated Transfers

A. Write Operation

If a write operation is executing, the XMIT file is loaded with write data from the DP. Two write cycles are required to load a longword into the file from the 16-bit LTCHD CIPA D bus. The XMIT file can hold up to four longwords of data.

After the XMIT file is loaded, the port microcode issues a GO command to initiate the transfer of write data over the CMI bus.

GO starts the arbitration process wherein the port attempts to gain control of the CMI.

When the arbitration is successful and the port has won the bus, a command/address bus cycle is executed. In the command/address bus cycle, the outputs of the CMD/ADDR HI, ADDR LO, and byte mask registers are combined to form a command/address longword on the CMDADDR bus (CMDADDR <31:00>). The command/address longword is selected by the CMI mux and placed onto the data/address lines of the CMI.

In the next bus cycle the data longword first loaded into the XMIT file is unloaded, selected by the CMI mux, and placed onto the CMI data/address lines. The transfer ends with the status bus cycle wherein the port receives status from the slave nexus.

If the transfer was to write a single longword of data, a DONE signal is issued to the port microcode informing it that the CMI transfer is completed. The flow diagram then terminates.

If a data quadword was to be transferred, the flow returns to the "arbitrate for CMI" block and repeats the sequence from that point. Note that only a single longword can be transferred on the CMI per write operation. If a quadword of data is to be transferred, the port must arbitrate for the CMI bus for each longword transferred.

B. Read Operation

If a read operation is executing, the port microcode issues a GO command to initiate the transfer of read data over the CMI bus.

GO starts the arbitration process wherein the port attempts to gain control of the CMI.

When the arbitration is sucessful and the port has won the bus, a command/address cycle is executed. In the command/address cycle, the outputs of the CMD/ADDR HI, ADDR LO, and byte mask registers are combined to form a command/address longword on the CMDADDR bus (CMDADDR <31:00>). The command/address longword is selected by the CMI mux and placed onto the data/address lines of the CMI.

When the slave nexus is ready with the requested read data, it initiates the status bus cycle wherein it places the data longword onto the CMI data/address lines and status onto the CMI status lines. During the status cycle the port takes the data longword off the CMI data/address lines and loads it into the RCV file (via the Receive Write Data Register). The port also takes status from the CMI during the status cycle.

If the transfer was to read a single longword of data, a DONE signal is issued to the port microcode informing it that the CMI transfer is completed. The sequence proceeds to the next block to unload the read data out of the RCV file.

If a data quadword was to be transferred, the flow returns to the "arbitrate for CMI" block and repeats the sequence from that point. Note that only a single longword can be transferred on the CMI per read operation. If a quadword of data is to be transferred, the port must arbitrate for the CMI bus for each longword transferred.

Unloading the RCV file out to the DP completes the read sequence.

6.1.3.2 Write Vector Function -- A write vector function is a special "port initiated" write transfer in which the port interrupts the CPU to send it an interrupt vector. Interrupt requests result from port error conditions such as parity errors, power-downs, etc. The interrupt vector transferred to the CPU contains the CPU interrupt starting address for the CI750. Figure 6-8 is a simplified flow diagram of a write vector function.

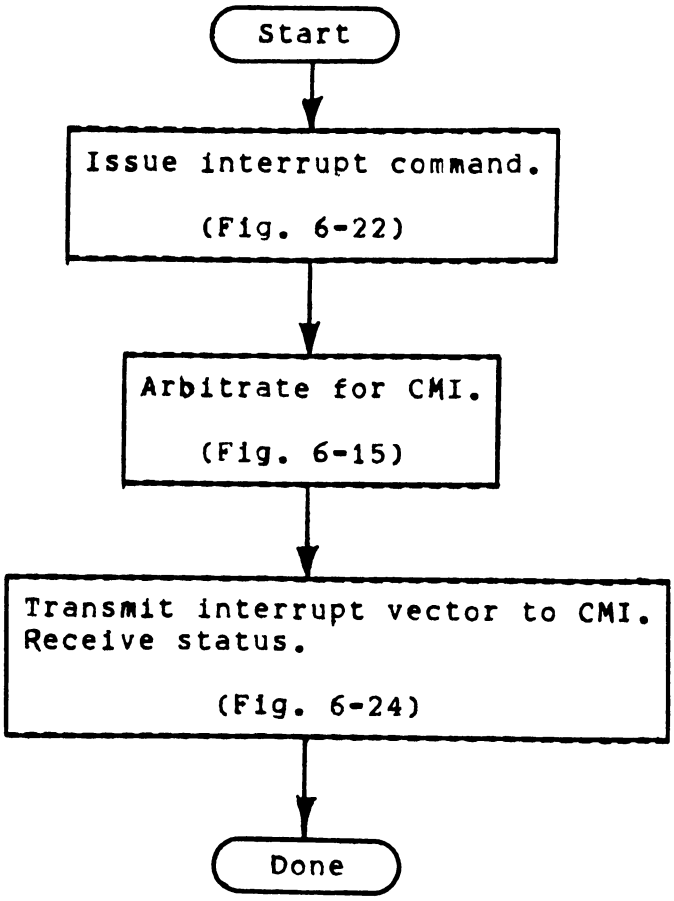


Figure 6-8 Write Vector Function Flow Diagram

Interrupt logic in the DP monitors those areas which could cause an interrupt. When the logic senses an interrupt condition it issues a command to the CCI which initiates the interrupt sequence.

The interrupt command starts the arbitration process wherein the port attempts to gain control of the CMI.

When the arbitration is sucessful and the port has won the bus, a command/address cycle is executed. In the command/address cycle, the CMI mux selects the interrupt vector and places it onto the data/address lines of the CMI. This is followed by the status cycle in which the the slave CPU sends status to the port. The port keeps the interrupt vector on the CMI data/address lines during the status cycle.

6.1.3.3 Unsolicited CMI Transfers -- An unsolicited CMI transfer is a CMI transaction in which the CI750 port is the slave. The port is addressed by the host CPU which commands either a write or a read of a port register. Figure 6-9 is a simplified flow diagram of unsolicited CMI transfers illustrating the major steps in the sequence. The illustration includes both a write and a read transfer.

The CCI contains address decode logic connected to the CMI data/address lines. When the decode logic detects an address that falls within the address range (I/O slot) of the CI750, it asserts CI SPACE. Other outputs from the decode logic indicate what area of the CI750 is being referenced.

The outputs from the decode logic, along with other command/address data from the CMI, is clocked into a command/address hold register. From the hold register, address data is clocked into an address offset register and function data is applied to a function register. If the command/address reference is to the CI750 (CI SPACE true), DBBZ is asserted on the CMI and the function data is clocked into the function register.

Note that most of the action described above occurs for every command/address cycle. All command/address longwords get decoded, checked for a CI750 reference (CI SPACE true), and latched into the command/address hold register. The command/address is on the bus for only one bus cycle during which it must be taken off the bus, decoded, and latched. If the reference was not to the CI750 port, the latched data is not used.

Actions that require that the reference be to the CI750, are the assertion of DBBZ and loading the function register. The assertion of DBBZ in response to a command/address cycle is done only by the slave that was referenced. Outputs from the function register are command signals that initiate and control CCI operations, hence these should only be asserted if the reference was to the CI750.

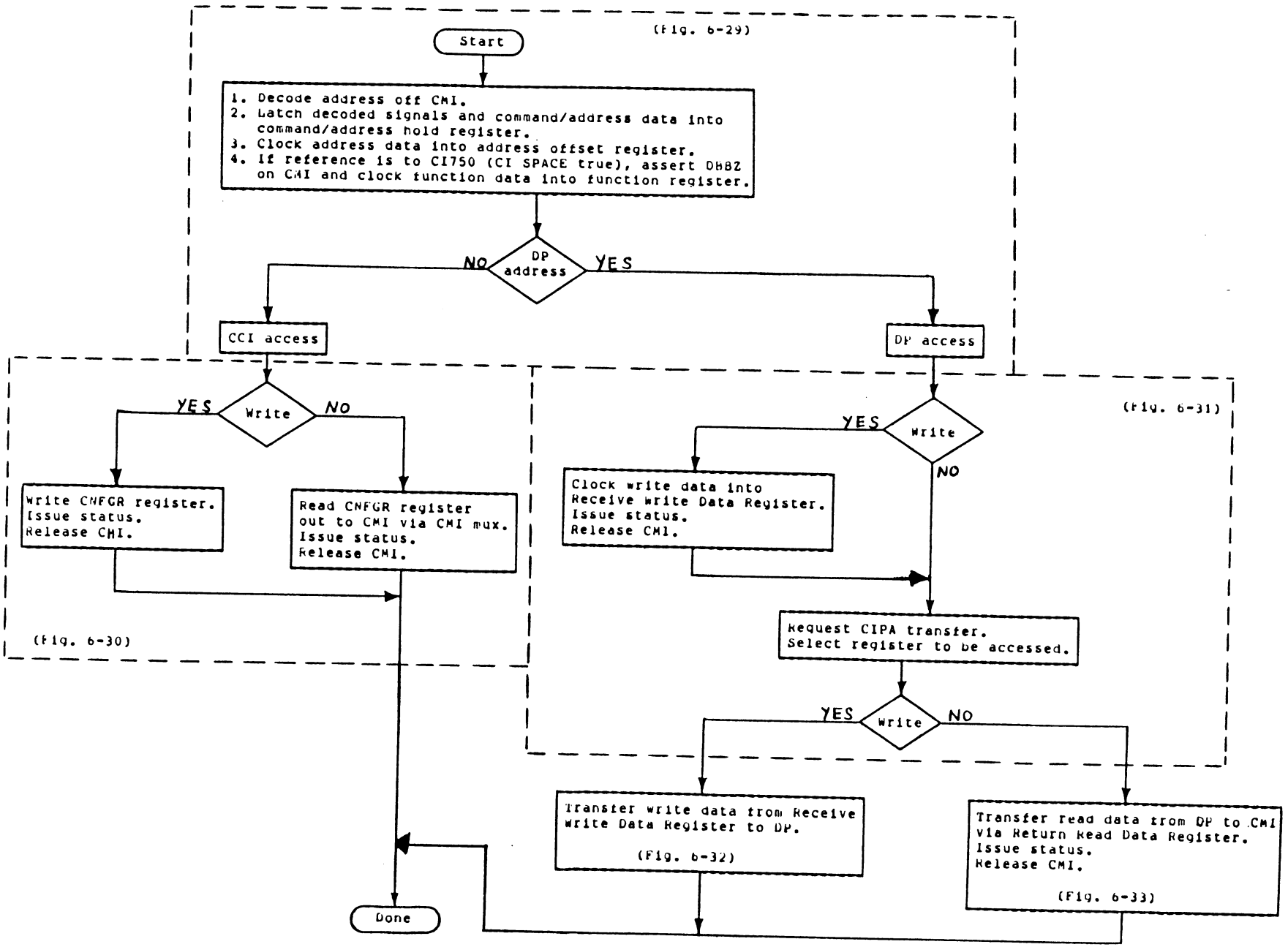


Figure 6-9 Flow Diagram Of Unsolicited CMI Transfer

At this point the flow sequence divides according to whether the reference address is to the CCI or the DP.

A. CCI Access

The only CCI register referenced (in normal mode) by an unsolicited operation is the CNFGR register. Hence, a reference to the CCI is a reference to the CNFGR register.

If the commanded function is a write, bits BUF CMI D 31, 23, 22, 20, 19, 17, 16, 14, 13, and 08 from the CMI, are written into the register during the status cycle. The CCI then returns status to the host CPU and releases the CMI.

If the commanded function is a read, the CMI mux selects the output from the CNFGR register for the CMI data/address lines. The CCI then returns status to the host CPU and releases the CMI.

B. DP Access

When the unsolicited reference is to the DP and the commanded operation is a write, the write data is taken off the CMI data/address lines and latched into the Receive Write Data Register. Status is then returned to the host CPU and the CMI is released. Thus the CMI bus is not tied up while the CCI interfaces with the DP.

In both a write and a read operation, a request is issued to the DP requesting access for a CIPA transfer. The DP responds by reading the CCI address offset register to determine which DP register is to be accessed.

If the commanded operation is a write, the DP transfers the write data from the Receive Write Data Register to the selected register in the DP via the CIPA bus. This completes the unsolicited write sequence to the DP.

If the commanded operation is a read, the DP reads the selected register and transfers the read data to the Return Read Data Register in the CCI via the CIPA bus. The read data is then unloaded from the Return Read Data Register onto the CCI XMIT DATA bus. The CCI XMIT DATA bus is selected by the CMI mux for output onto the data/address lines of the CMI bus. The CCI then returns status to the CPU and releases the CMI to complete the unsolicited read sequence.

6.2 Port Initiated Transfers

Figures 5-18 and 5-19 illustrated the sequencing of the CCI/DP Interface Control Logic (located in the DP) for port initiated data transfers between the DP and the CCI. The Interface Control Logic supplies a REG SEL <3:Ø> code that specifies the CCI location that is to be read or written. The associated data (CIPA DATA <15:ØØ>) is transferred over the CIPA bus.

The flow diagrams and descriptions given in section 6.2 are a detailed expansion of the general flow diagrams of port initiated CMI transfers given in Figures 6-7 and 6-8. Refer to Figures 6-7 and 6-8 and to the CCI block diagram (Figure 6-6) in the following discussion.

6.2.1 Load Command/Address And Byte Mask Registers

Figure 6-1Ø is a flow diagram of the "load command/address and byte mask registers" function. Figure 6-11 is a block diagram of the register control logic.

The command/address high word is taken from the data lines of the CIPA bus and applied to the CCI where it is buffered and becomes BUF CIPA DATA <15:ØØ>. BUF CIPA DATA <15:ØØ> is latched up in a latch which outputs LTCHD CIPA D <15:ØØ>. The latched data is applied to the CMD/ADDR HI register.

The CIPA REG SEL <3:Ø> code from the CIPA bus is applied to the CCI write decoder. CIPA REG SEL 3 is asserted when a CCI register is to be written (see Table 5-1Ø). The true state of CIPA REG SEL 3 enables the decoder (DIAGNOSE false) which then decodes the CIPA REG SEL <2:Ø> input to select the register to be written. In this case, the WRT CMDADDR HI output is asserted.

WRT CMDADDR HI loads the two function bits (LTCHD CIPA D <14:13>) and the six high address bits (LTCHD CIPA D <Ø5:ØØ>) into the CMD/ADDR HI register.

WRT CMDADDR HI also clears the read and write counters causing the read and write pointers to address location Ø in the XMIT and RCV files.

In addition, WRT CMDADDR HI clocks a quad flip-flop conditioned by bit 15 of the CMD/ADDR HI data (LTCHD D 15). The bit is Ø for a CMI longword transfer and a 1 for a CMI quadword transfer. When the bit is a 1 (quadword transfer), the flip-flop sets and asserts QUAD FLG indicating that a quadword transfer is in progress.

On the next DP to CCI transfer cycle, the address low word and its associated REG SEL code are taken from the CIPA bus. The address low word is latched and then applied to the ADDR LO register as LTCHD CIPA D <15:ØØ>. The REG SEL code is applied to the write decoder which outputs WRT ADDR LO. WRT ADDR LO loads the 16-bit low address into the ADDR LO register.

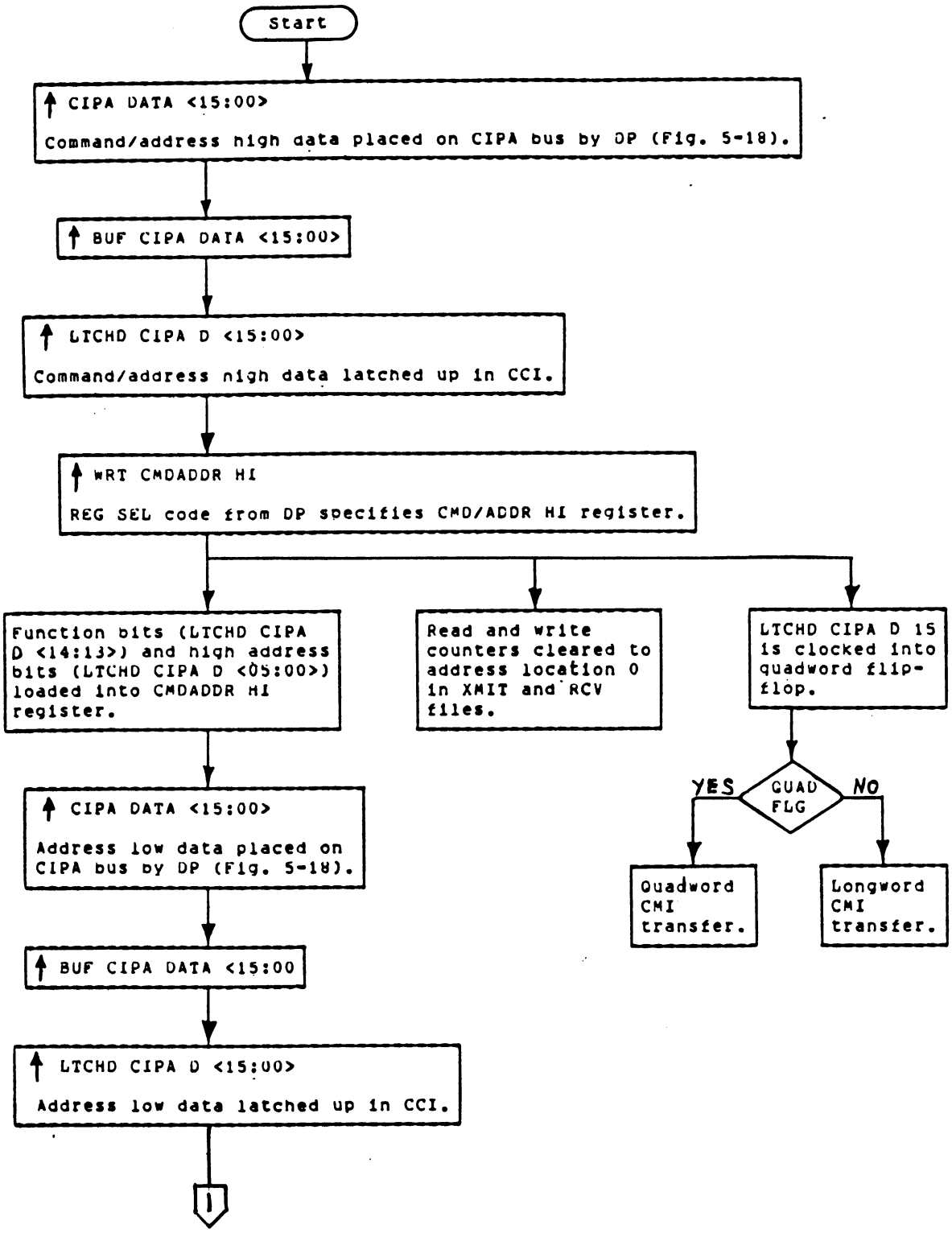


Figure 6-10 Load Flow Diagram for CMD/ADDR HI, ADDR LO, and Byte Mask Registers (Sheet 1 of 2)

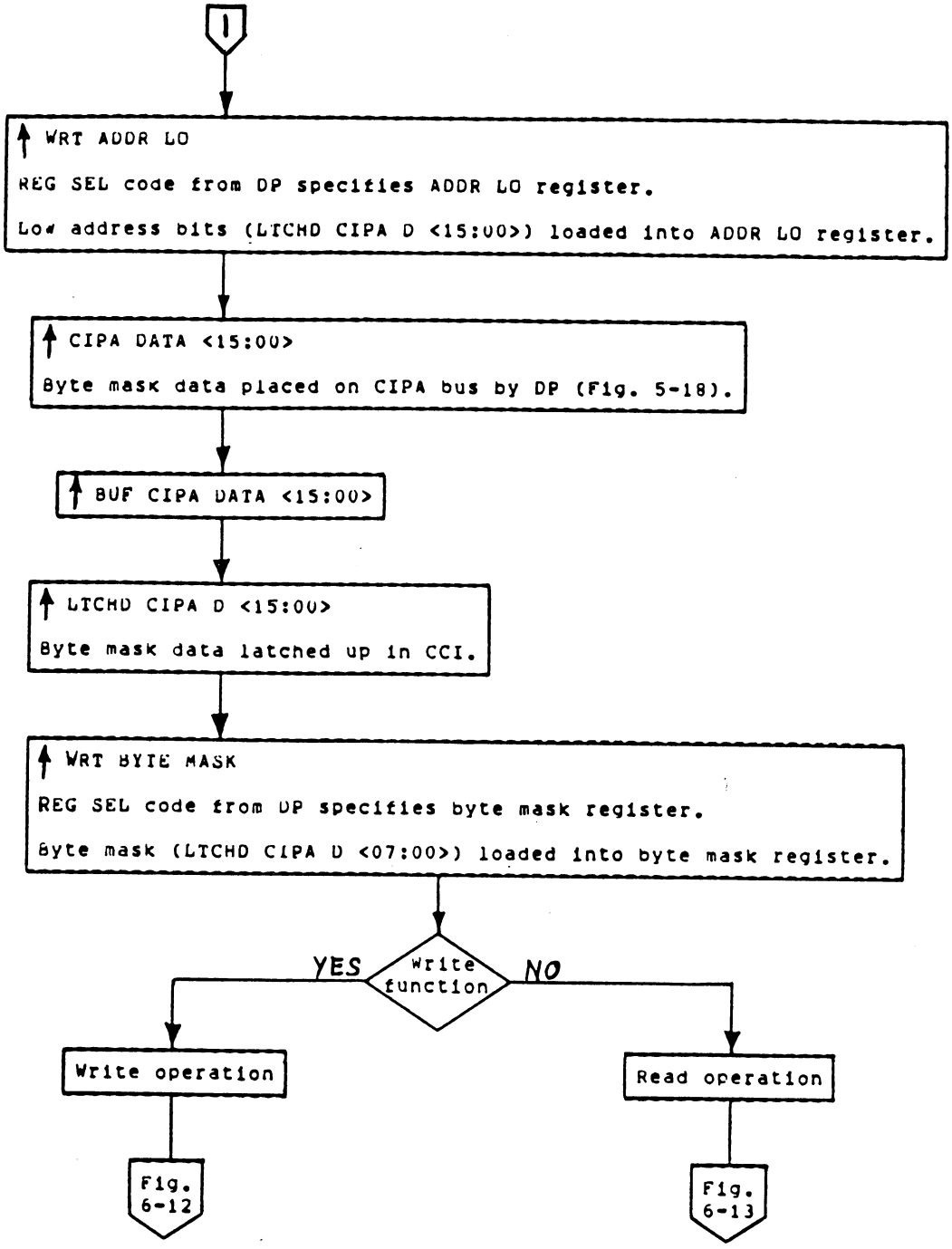
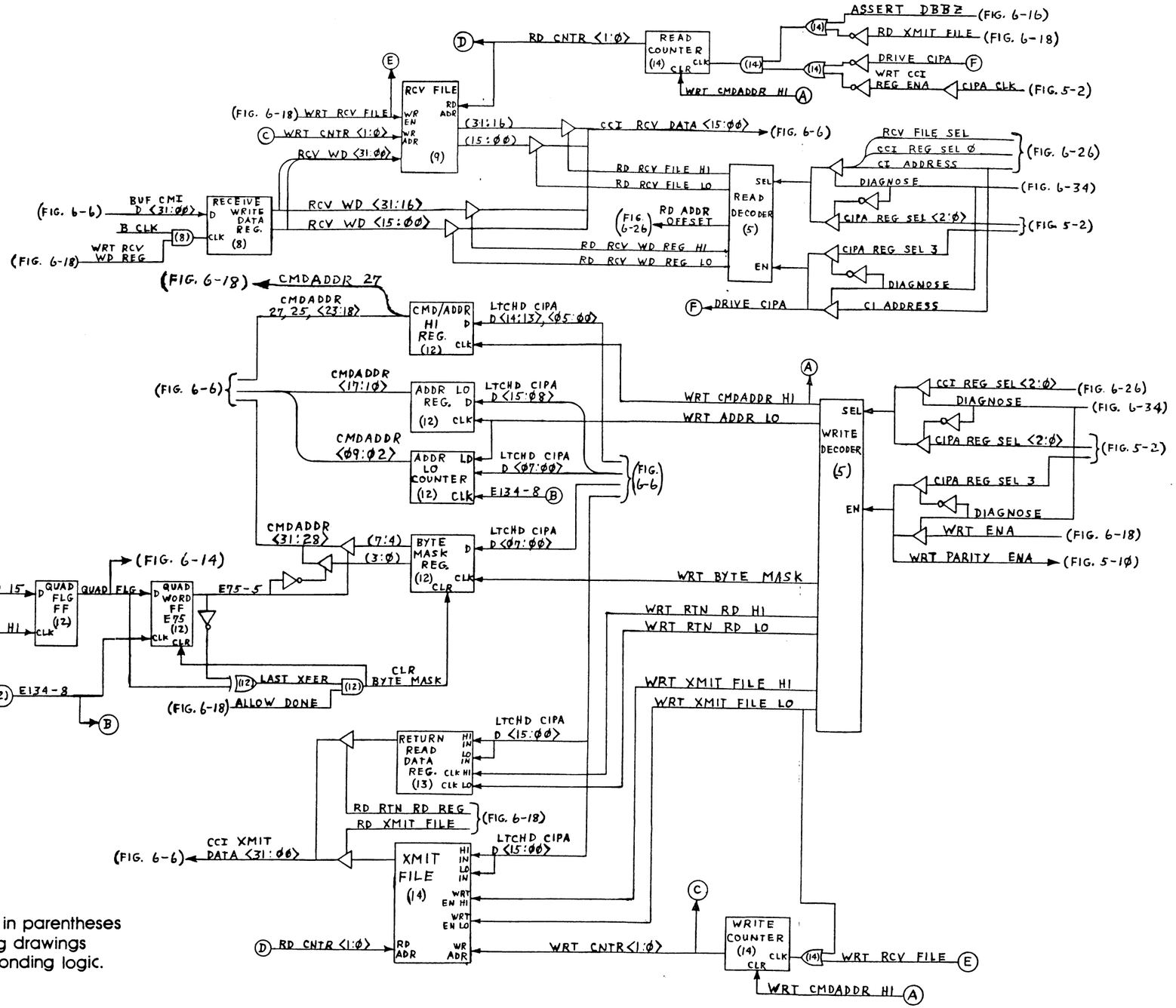


Figure 6-10 Load Flow Diagram for CMD/ADDR HI, ADDR LO, and Byte Mask Registers (Sheet 2 of 2)



Note:
Number designations in parentheses refer to engineering drawings containing corresponding logic.

Figure 6-11 CCI Register Control Logic

The lower eight bits of the ADDR LO register form an address counter. The counter is incremented after each CMI transfer by E134-8. E134-8 asserts for each command/address transfer cycle to the CMI (MUXA/MUXB SEL B and MUXA/MUXB SEL A both false; see Paragraph 6.2.2.4 and Table 6-3). Thus the CMD/ADDR and ADDR LO registers do not have to be reloaded for each transfer when transferring large data blocks. The microcode reloads the CMD/ADDR and ADDR LO registers with a new page address whenever a page boundary is crossed.

On the next DP to CCI transfer cycle, the byte mask and its associated REG SEL code are taken from the CIPA bus. The byte mask is latched and then applied to the byte mask register as LTCHD CIPA D <15:00>. The REG SEL code is applied to the write decoder which outputs WRT BYTE MASK. WRT BYTE MASK loads the eight- bit byte mask into the byte mask register.

Note that while an eight-bit byte is loaded into the byte mask register, the register output is only a four-bit nibble (CMDADDR <31:28>). The two halves of the register are muxed onto the output lines by E75-5 from flip-flop E75.

Flip-flop E75 is cleared by CLR BYTE MASK at the end of each CMI transfer. This negates the E75-5 output thereby enabling the lower four bits from the byte mask register onto the CMDADDR <31:28> output lines. The lower four bits of the register contain the byte mask for a longword transfer, and the byte mask of the first longword transferred in a quadword transfer. The upper four bits contain the byte mask of the second longword transferred in a quadword transfer.

If a quadword transfer is to occur, the quad flip-flop is clocked set by WRT CMDADDR HI asserting QUAD FLG. The true state of QUAD FLG conditions E75 to set. After the first CMI transfer has occurred, MUXA SEL A and MUXA SEL B both negate for the command/address cycle of the second transfer (Table 6-3) thereby asserting E134-8 and setting E75. The true state of E75-5 switches the upper four bits of the byte mask register onto the four output lines thereby placing the byte mask of the second longword onto the CMDADDR <31:28> output lines.

LAST XFER is asserted after each single longword transfer (QUAD FLG and E75-5 false) and after each quadword transfer (QUAD FLG and E75-5 true). LAST XFER is ANDed with ALLOW DONE from the CCI control logic to assert CLR BYTE MASK. CLR BYTE MASK resets the byte mask register and flip-flop E75. Resetting E75 negates E75-5 thereby selecting the lower four bits from the byte mask register as the next byte mask.

If the function commanded by the port microcode is a write, continue with Paragraph 6.2.2. If the commanded function is a read, continue with Paragraph 6.2.3.

6.2.2 Write Function

To execute a write function:

1. The XMIT file is loaded with the write data
2. The port arbitrates for the CMI bus
3. The command/address longword is placed on the CMI.
4. The XMIT file is unloaded out to the CMI

6.2.2.1 Load XMIT File -- Figure 6-12 is a flow diagram of the "load XMIT file" function.

The data high word and its associated REG SEL code are taken from the CIPA bus. The data high word is latched and then applied to the high word section of the XMIT file as LTCHD CIPA D <15:00>.

A two-bit pointer (WRT CNTR <1:0>) from a write counter, addresses one of the four longword locations in the XMIT file. The counter is cleared by WRT CMDADDR HI when the CMD/ADDR HI register is written, hence the pointer is initially addressing location 0 in the file.

The REG SEL code from the CIPA bus is applied to the write decoder which outputs WRT XMIT FILE HI. WRT XMIT FILE HI loads the latched data high word into the high word section of location 0 in the XMIT file.

Another DP to CCI cycle is required to write the data low word into the XMIT file. The data low word and its associated REG SEL code are taken from the CIPA bus. The data low word is latched and then applied to the low word section of the XMIT file as LTCHD CIPA D <15:00>.

The REG SEL code from the CIPA bus is applied to the write decoder which outputs WRT XMIT FILE LO. WRT XMIT FILE LO loads the latched data low word into the low word section of location 0 in the XMIT file.

WRT XMIT FILE LO also increments the write counter causing pointer WRT CNTR <1:0> to point to location 1 in the XMIT file.

If this is a quadword transfer, a second longword is transferred from the DP to the CCI where it is written into location 1 of the XMIT file. The REG SEL code will accompany both halves of the longword selecting the high word section of the XMIT file for the first half of the longword and the low word section of the file for the second half. The write counter is again incremented by WRT XMIT FILE LO to point to location 2 in the file.

Two more longwords could be written into locations 2 and 3 of the XMIT file before the file has to be read out to the CMI.

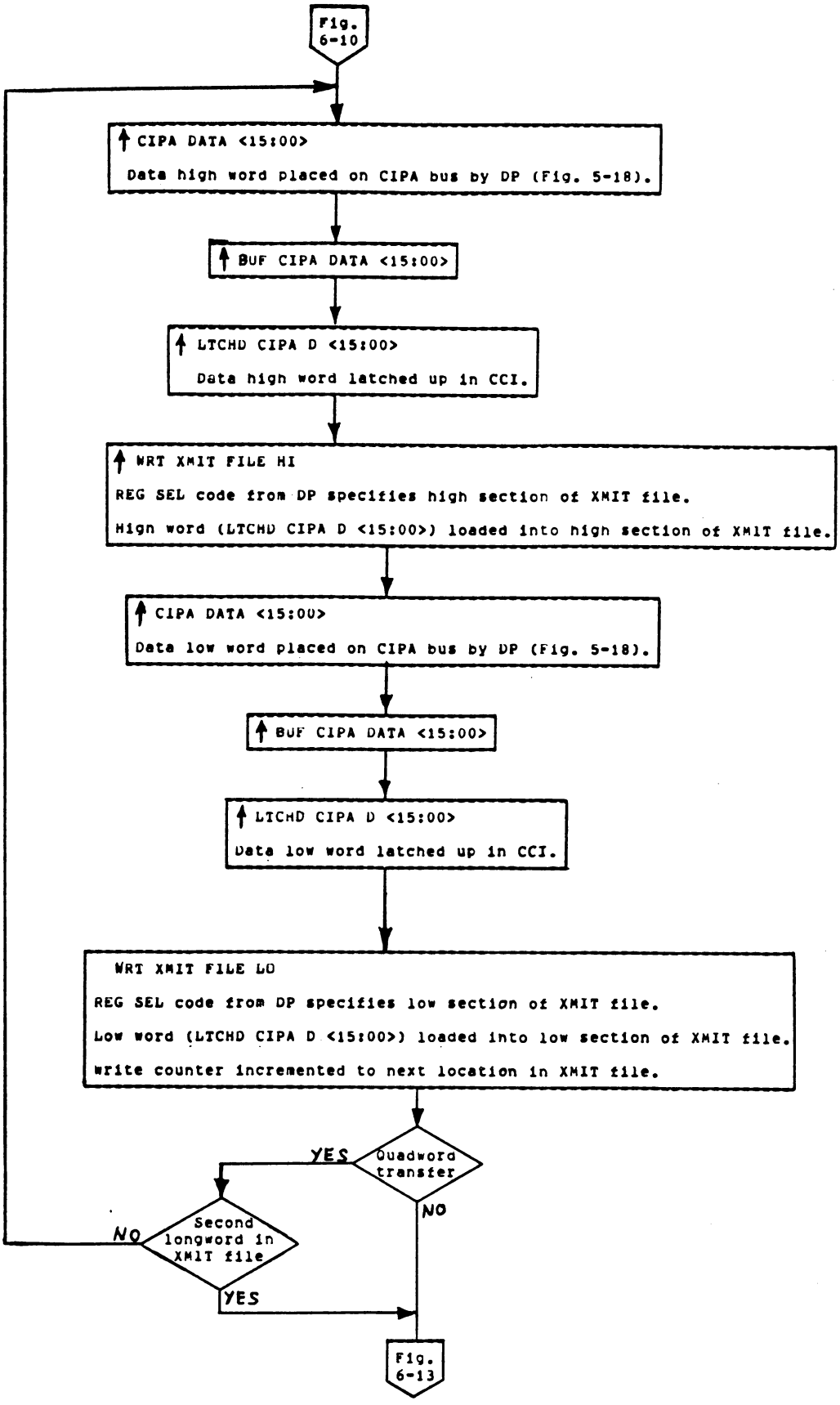


Figure 6-12 Load XMIT File Flow Diagram

6.2.2.2 Issue GO -- SET A GO is issued by the port microcode when the port is ready to arbitrate for control of the CMI bus. For a write operation, this is after the XMIT file has been loaded with the write data. For a read operation, this is after the byte mask register has been loaded.

Figure 6-13 is a flow diagram of the "issue GO" function. Figure 6-14 is a block diagram of the GO/DONE logic.

The port microcode asserts SET A GO to the DP. If the preceeding microinstruction has not been stretched out (UCODE STALL false) (Paragraph 5.11.2.3), GO flip-flop E1 is set and asserts E1-10. E1-10 in turn asserts CIPA A GO on the CIPA bus. CIPA A GO is received by the CCI where it becomes A GO and then SYNC A GO. SYNC A GO is applied to GO/DONE PAL E121.

Had the microcode asserted SET B GO (instead of SET A GO), a similar sequence would have occurred resulting in the assertion of SYNC B GO to PAL E121.

In response to SYNC A GO (or SYNC B GO), PAL E121 outputs POSSIBLE GO. If there is no CIPA transfer in progress to/from the DP (CIPA XFER false) and the CMI is not "read locked" (READ LOCK false), POSSIBLE GO will assert GO to the arbitration logic.

When a nexus executes a read lock function, it places all CMI nexus (except itself) into the "read lock" state. Thus when the CMI is "read locked", the nexus that executed the read lock function is the only one that can arbitrate for the CMI.

The GO/DONE logic senses a "read lock" function by detecting a BUF CMI D 27,25 function code of 0:1 (Table 6-2) during a command/address cycle. When function bits BUF CMI D 27,25 specify a "read lock" function, they condition a Read Lock flip-flop to be set by CLK RDLCK FF. CLK RDLCK FF asserts every command/address cycle (except port initiated command/address cycles) (Paragraph 6.3.1). If the function associated with the command/address cycle is a read lock function, the Read Lock flip-flop sets asserting READ LOCK.

If the GO/DONE logic attempts to issue a GO command while in the "read lock" state, POSSIBLE GO asserts but the assertion of GO is inhibited. The assertion of POSSIBLE GO enables a Read Lock counter (clocked by B CLK) to start counting.

When function bits BUF CMI D 27, 25 specify a "write unlock" function (1:1 code), the Read Lock flip-flop is conditioned to reset. During the command/address cycle of the "write unlock" function, CLK RDLCK FF asserts and resets the Read Lock flip-flop negating READ LOCK. The negation of READ LOCK clears the Read Lock counter and enables the GO AND gate.

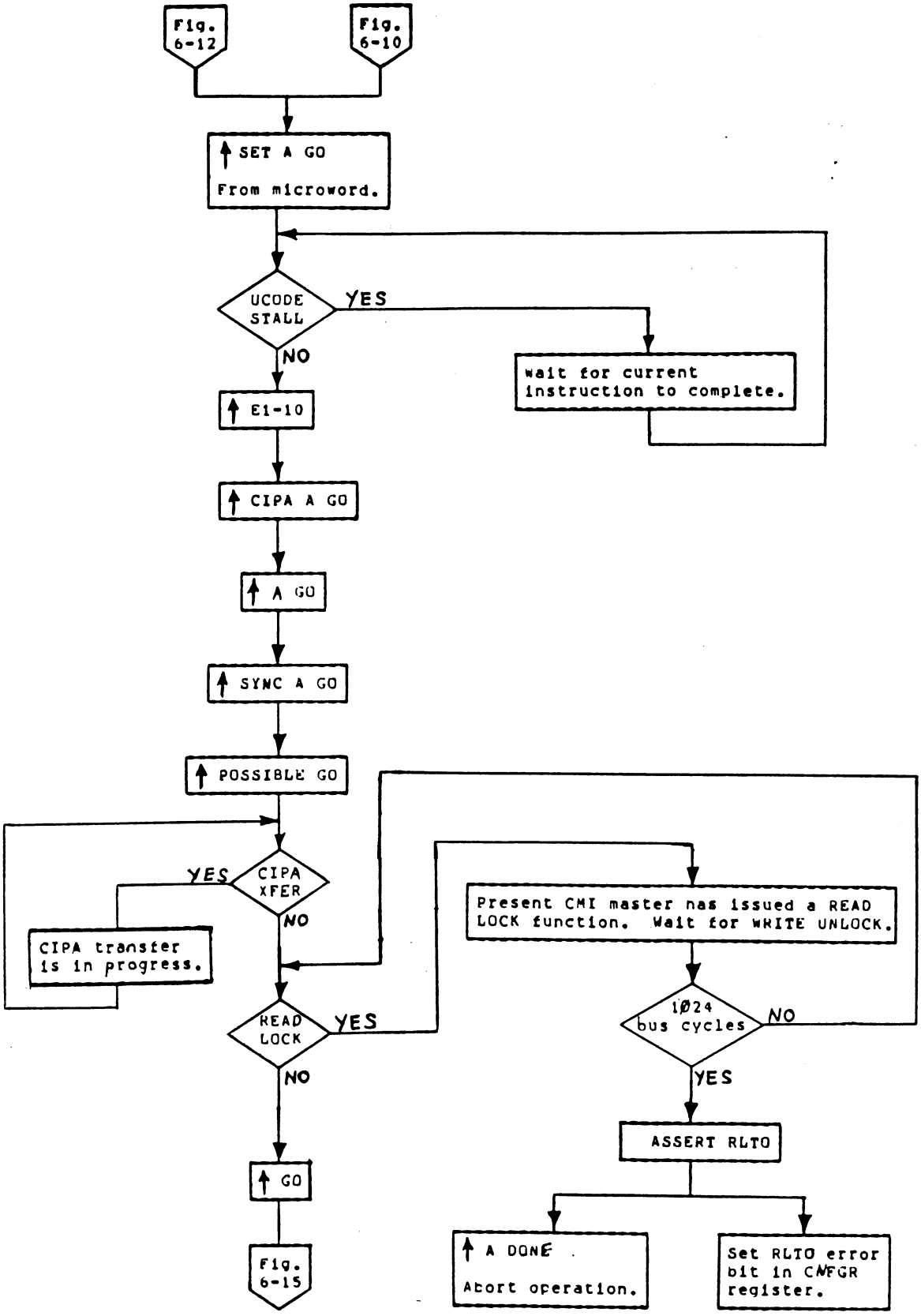


Figure 6-13 Issue GO Flow Diagram

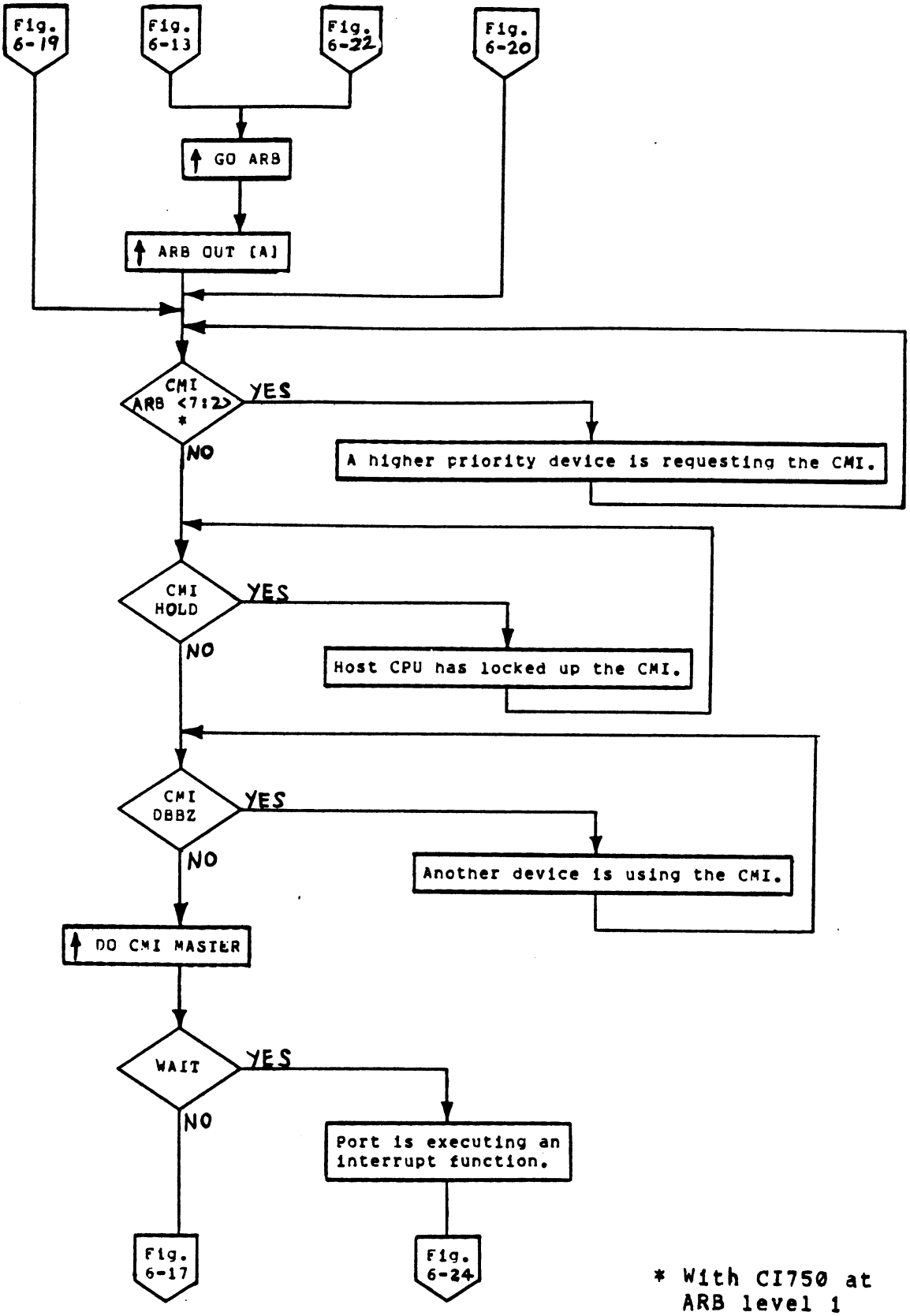
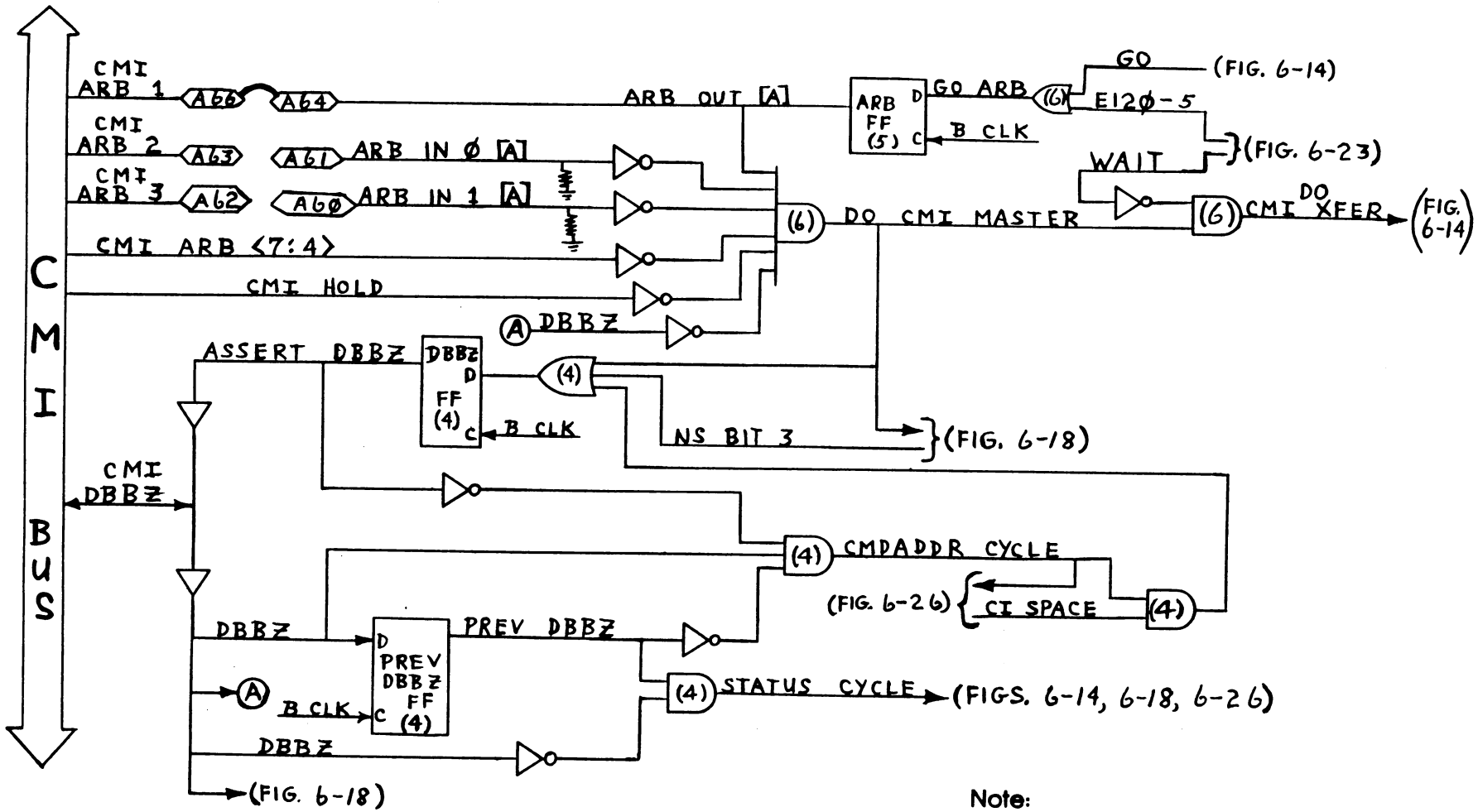


Figure 6-15 Arbitration Flow Diagram



Note:
Number designations in parentheses refer to engineering drawings containing corresponding logic.

Figure 6-16 Arbitration Logic

Similarly an arbitration level of 3 could be established by connecting terminal A64 to terminal A62. The assertion of ARB OUT [A] would assert CMI ARB 3 to inhibit arbitration by the CPU and by the CMI devices at ARB levels 2 and 1. Terminals A63 and A66 are left open so that the state of the CMI ARB 2 and 1 lines do not effect the CI750 arbitration process.

In addition to higher level arbitration lines, the ARB AND gate is also inhibited by the true state of CMI HOLD or DBBZ. CMI HOLD is asserted by the CPU to hold the CMI while an essential function is executing. DBBZ indicates that the CMI is being used by some other device.

If no higher arbitration levels are pending, and CMI HOLD and DBBZ are both false, the ARB AND gate will be enabled by ARB OUT [A] causing DO CMI MASTER to assert. The CI750 port now has control of the CMI bus.

A WAIT signal is received from the interrupt logic if the arbitration resulted from an interrupt command. If WAIT is false (arbitration due to "issue GO" function), the flow passes to the command/address function shown in Figure 6-17. If WAIT is true (arbitration due to interrupt command), the flow passes to the write vector function shown in Figure 6-24.

The arbitration flow diagram shows DO CMI MASTER being asserted from Figure 6-19 in the "write operation" sequence and Figure 6-20 in the "read operation" sequence. These inputs are used during quadword data transfers after the first longword has been transferred and the arbitration logic must regain control of the CMI to transfer the second longword. This is discussed further in the discussions associated with Figure 6-19 (Paragraph 6.2.2.5) and Figure 6-20 (Paragraph 6.2.3.4).

6.2.2.4 Command/Address Cycle -- During the command/address cycle, the port asserts DBBZ and places the command/address on the CMI bus. Figure 6-17 is a flow diagram of a port initiated command/address cycle.

The next B CLK after a successful arbitration (DO CMI MASTER asserted) that was initiated by an "issue GO" function (WAIT false), initiates the command/address cycle. B CLK sets a DBBZ flip-flop (conditioned to set by DO CMI MASTER) asserting ASSERT DBBZ and then CMI DBBZ on the CMI bus. CMI DBBZ indicates that the CMI is busy. The true state of CMI DBBZ inhibits the arbitration process in all other CMI devices.

CMI DBBZ asserts DBBZ which negates DO CMI MASTER thereby conditioning the DBBZ flip-flop to reset on the next B CLK. Thus CMI DBBZ is asserted by the port for only one bus cycle (the command/address cycle).

DBBZ is also applied to a PREV DBBZ flip-flop conditioning it to set.

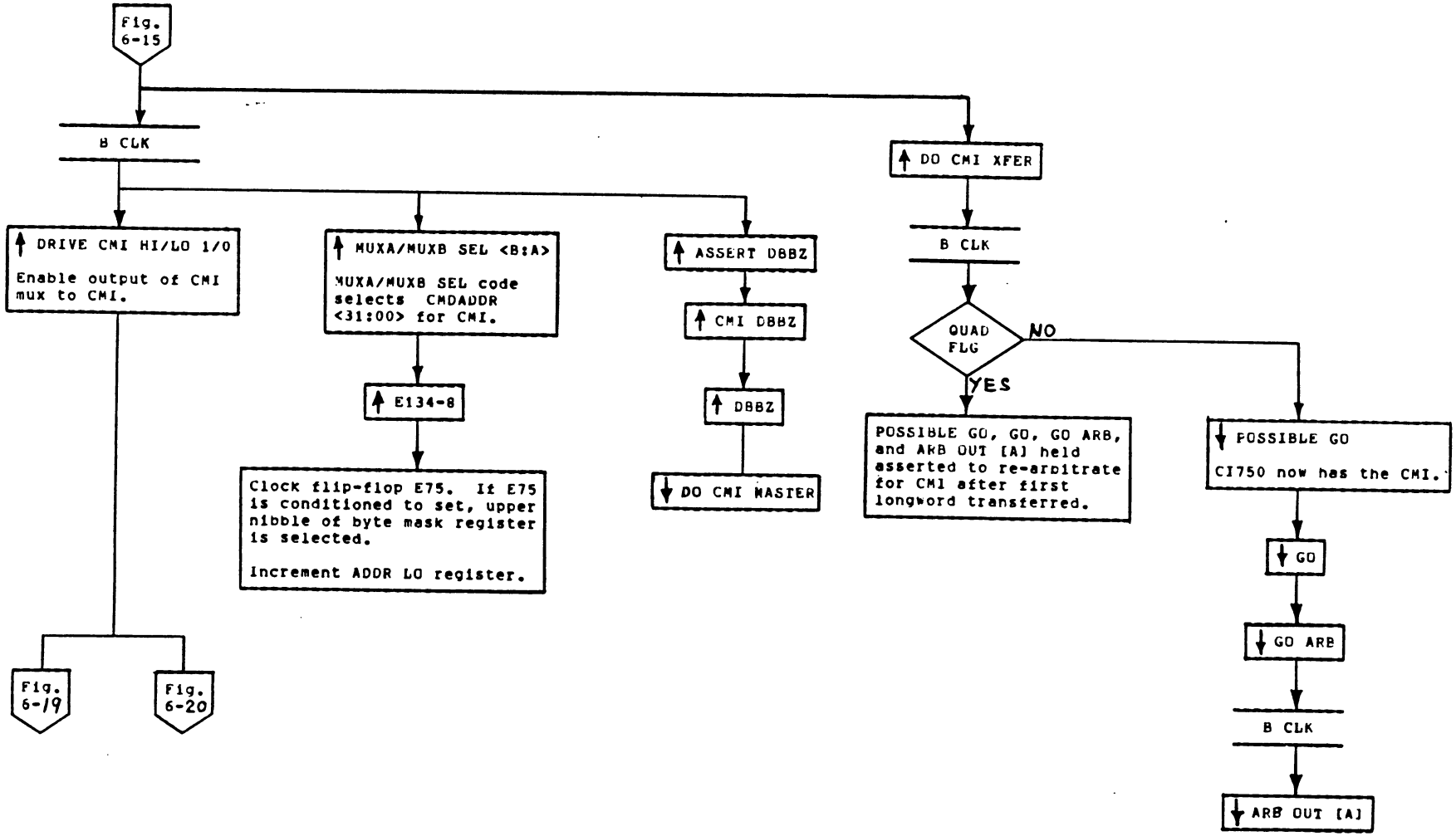


Figure 6-17 Flow Diagram Of Port Initiated Command/Address Cycle

6.2.2.5 **Unload XMIT File And Status Cycle** -- The next B CLK in the main flow initiates the period in which the port transfers the data in the XMIT file to the CMI and negates CMI DBBZ. This period is followed by the status cycle in which the port receives status from the slave device. Figure 6-19 is a flow diagram illustrating the unloading of the XMIT file and the status cycle.

NOTE

In the case of a two-cycle transfer, the port places the data onto the CMI and negates CMI DBBZ during the status cycle (Paragraph 6.1.1.2).

With DO CMI MASTER false (Figure 6-16), B CLK resets the DBBZ flip-flop negating ASSERT DBBZ and CMI DBBZ. If the slave is not ready to accept the write data, it will assert CMI DBBZ until it has taken the data from the bus. The flow diagram shows this to be one bus cycle although it may be longer.

B CLK also sets the PREV DBBZ flip-flop asserting PREV DBBZ.

Logic array E26 in the CCI control logic (Figure 6-18), samples function bit CMDADDR 27. The true state of CMDADDR 27 indicates this to be a write operation. The array responds by asserting ENA XMIT. ENA XMIT in turn asserts RD XMIT FILE which enables the data out of location Ø of the XMIT file (read counter reset to Ø when CMD/ADDR HI register was loaded; see Figures 6-1Ø and 6-11).

Logic array E26 also outputs the MUXA/MUXB SEL code to select CCI XMIT DATA <31:ØØ> from the XMIT file as the CMI mux input.

The data longword is placed on the CMI bus and transferred to the slave device. The slave holds CMI DBBZ asserted until it has taken the data.

The next B CLK initiates the status bus cycle. During the status cycle the slave takes the write data off the CMI, negates CMI DBBZ, and places the status bits (CMI STATUS <1:Ø>) (Table 6-1) on the CMI. The negation of CMI DBBZ negates DBBZ which in turn asserts STATUS CYCLE.

The next assertion of B CLK to the logic array causes the array to assert WRT ENABLE STATUS and negate DRIVE CMI HI/LO 1/Ø. WRT ENABLE STATUS enables the status decoder which decodes the status bits from the CMI and outputs any error condition to the configuration register. The negation of DRIVE CMI HI/LO 1/Ø inhibits the output of the CMI mux thereby removing the write data from the CMI bus.

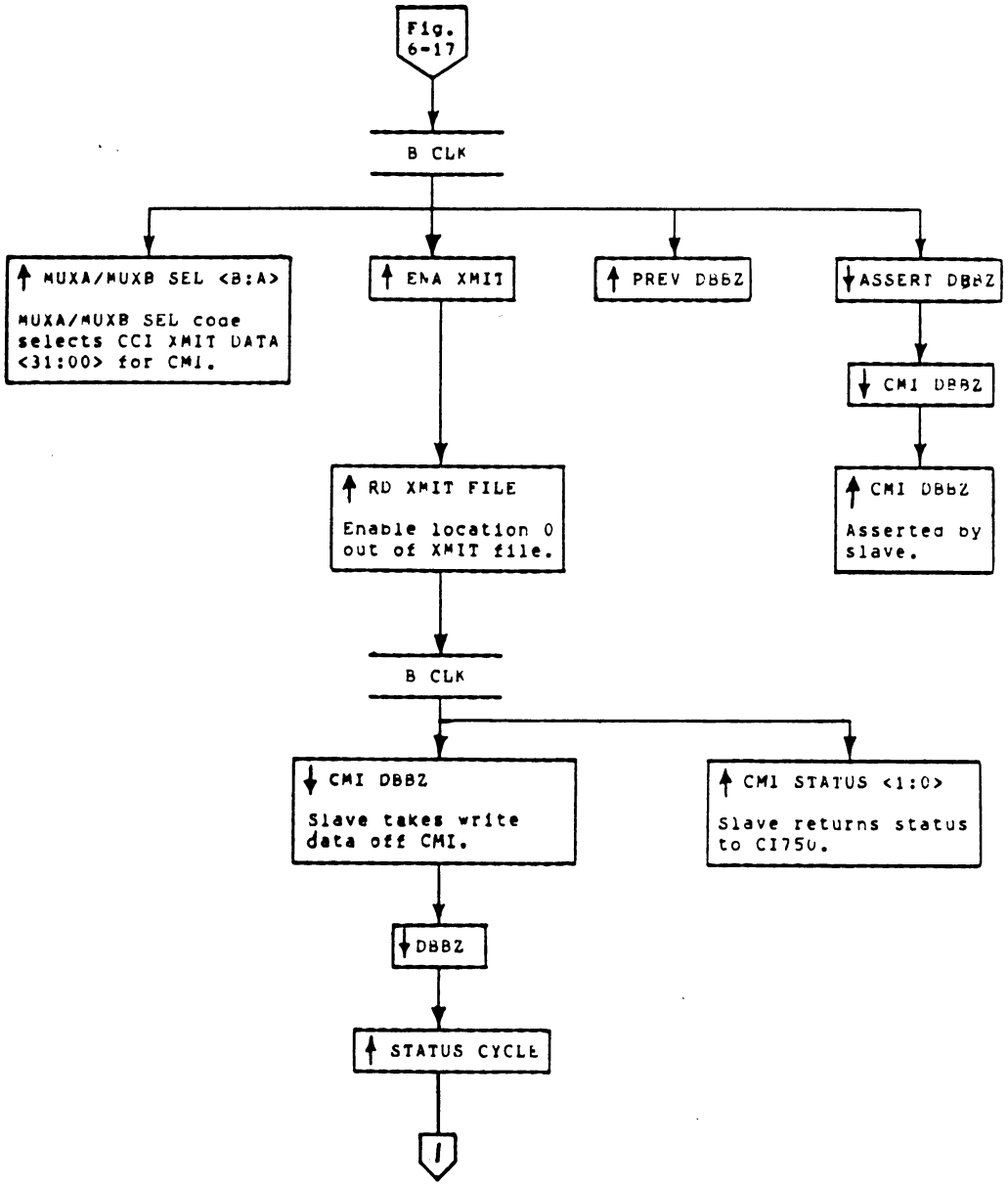


Figure 6-19 Unload XMIT File and Status Cycle Flow Diagram (Sheet 1 of 2)

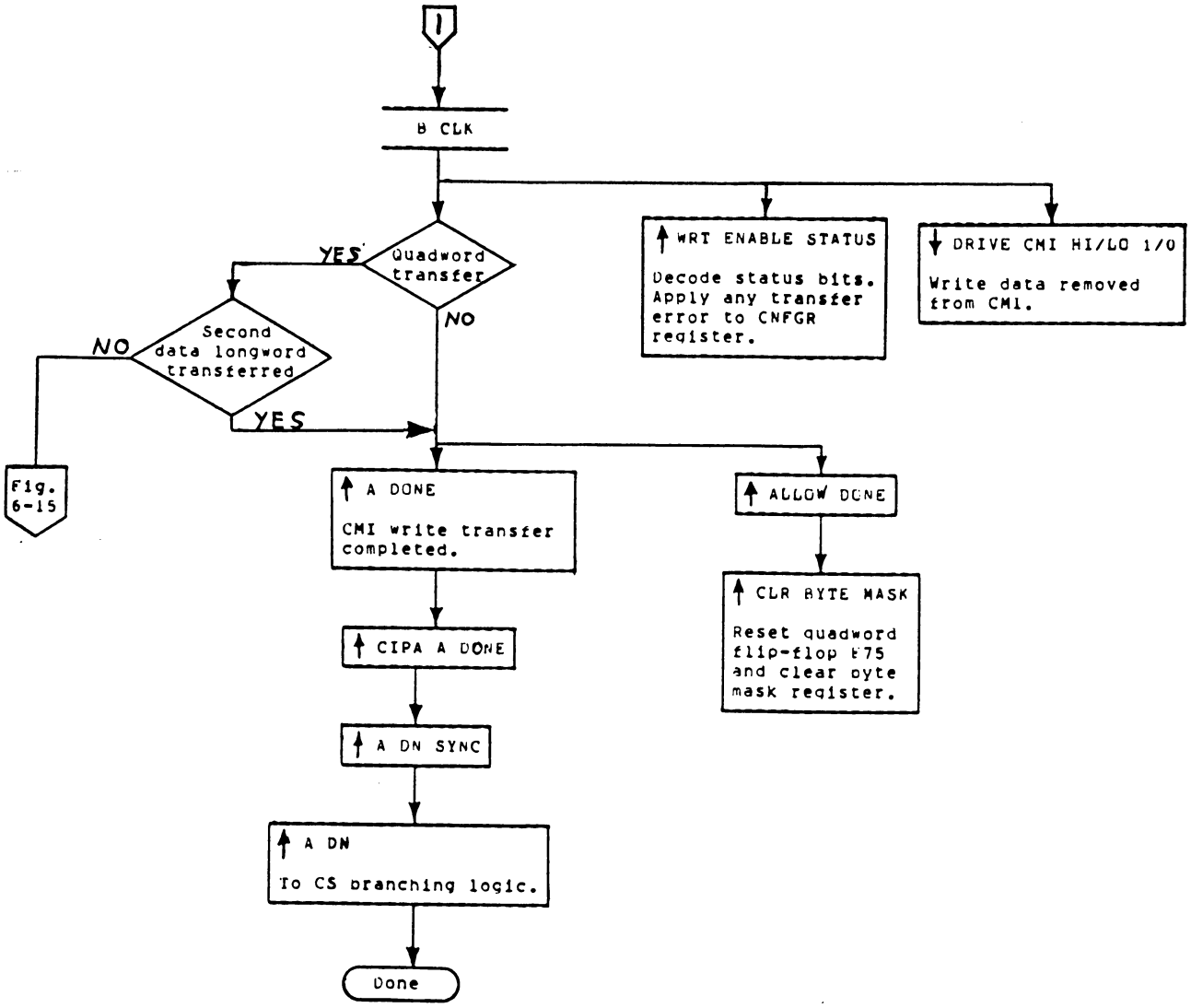


Figure 6-19 Unload XMIT File and Status Cycle Flow Diagram (Sheet 2 of 2)

If this is a quadword transfer and only the first longword has been transferred, the flow diagram returns to the arbitration function (Figure 6-15) to rearbitrate for the CMI and transfer the second longword. When a quadword transfer was sensed during the command/address cycle, the arbitration sequence was executed up to the assertion of ARB OUT [A] (Figure 6-17). Now with DBBZ negated, the arbitration sequence can proceed. If in the interim, the CPU has not asserted CMI HOLD or a higher priority nexus is not requesting the CMI, the CI750 will regain the CMI for the second half of the quadword transfer.

If this is the last transfer of the write function, the CCI control logic asserts ALLOW DONE which in turn asserts CLR BYTE MASK. CLR BYTE MASK clears the byte mask register and resets quadword flip-flop E75 in the CCI register control logic.

In addition, GO/DONE PAL E121 sensing the true state of STATUS CYCLE, outputs A DONE indicating that the write transfer function is completed. A DONE is placed on the CIPA bus as CIPA A DONE and then coupled to the DP as A DN SYNC. A DN SYNC then causes A DN to be asserted to the CS branching logic. The port microcode is thereby informed of the completion of the commanded write function.

6.2.3 Read Function

To execute a read function:

1. GO is issued from the port microcode
2. The port arbitrates for the CMI bus
3. The command/address is placed on the CMI
4. The RCV file is written from the CMI
5. The RCV file is read out to the DP

6.2.3.1 Issue GO -- SET A GO is issued by the port microcode resulting in the assertion of GO to the arbitration logic. The Issue GO sequence is shown in Figure 6-13 and described in Paragraph 6.2.2.2.

6.2.3.2 Arbitration -- GO is received by the arbitration logic which proceeds to arbitrate for control of the CMI bus. When the port gains control of the bus, it asserts DO CMI MASTER. The arbitration sequence is shown in Figure 6-15 and described in Paragraph 6.2.2.3.

6.2.3.3 Command/Address Cycle -- The assertion of DO CMI MASTER initiates the command/address bus cycle. During the command/address cycle the port asserts CMI DBBZ and places the command/address longword on the CMI bus for transmission to the slave device. The command/address sequence is shown in Figure 6-17 and described in Paragraph 6.2.2.4.

6.2.3.4 Status Cycle And Load RCV File -- The next B CLK after the command/address cycle initiates the period in which the port negates CMI DBBZ. This period is followed by the status cycle in which the port receives the read data and status from the slave device. The read data is loaded into the RCV file via the Receive Write Data Register. Figure 6-20 is a flow diagram of the status cycle and the loading of the RCV file.

NOTE

In the case of a two-cycle transfer, the port negates CMI DBBZ during the status cycle (Paragraph 6.1.1.3).

With DO CMI MASTER false (Figure 6-16), B CLK resets the DBBZ flip-flop negating ASSERT DBBZ and CMI DBBZ. The slave will assert CMI DBBZ while it obtains the requested data and places it on the CMI. The flow diagram shows this to be one bus cycle although it may be longer.

B CLK also sets the PREV DBBZ flip-flop asserting PREV DBBZ.

After the slave has obtained the requested data and placed it on the CMI, it negates CMI DBBZ causing the negation of DBBZ in the CCI. The port uses the negation of CMI DBBZ as an indication that the read data and the status bits are on the CMI and the next bus cycle is the status cycle.

The negation of DBBZ also causes STATUS CYCLE to assert (PREV DBBZ true).

On the next B CLK (start of status cycle), the CCI control logic responds to the false state of DBBZ and outputs WRT RCV WD REG. WRT RCV WD REG clocks the buffered write data (BUF CMI D <31:00>) from the CMI into the Receive Write Data Register.

The control logic also outputs WRT ENABLE STATUS to enable the status decoder. The status decoder decodes the CMI status bits from the slave and outputs any error condition to the CNFGR register.

A third output asserted by the CCI control logic is WRT ENA. On the next B CLK, WRT ENA asserts WRT RCV FILE. WRT RCV FILE loads the read data (RCV WD <31:00>) from the Receive Write Data Register into location 0 of the RCV file. The RCV file write pointer (WRT CNTR <1:0>) is pointing to location 0 due to the write counter being cleared by WRT CMDADDR HI during the command/address cycle (Figure 6-11).

WRT RCV FILE also increments the write counter causing the write pointer to address location 1 in the file. This would be the file location of the next longword if this were a multi-longword transfer.

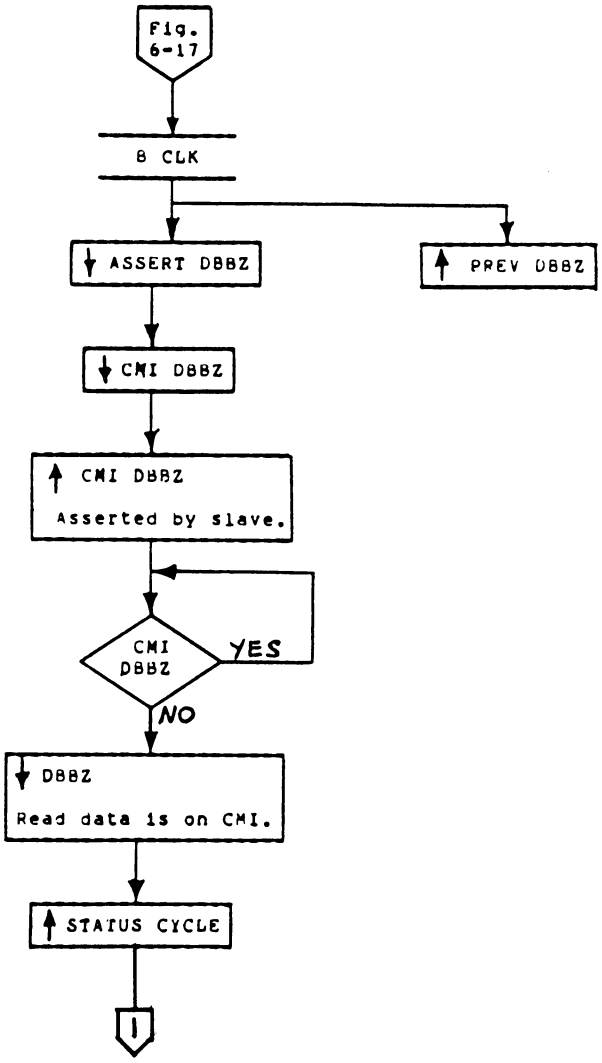


Figure 6-20 Status Cycle and Load RCV File Flow Diagram (Sheet 1 of 2)

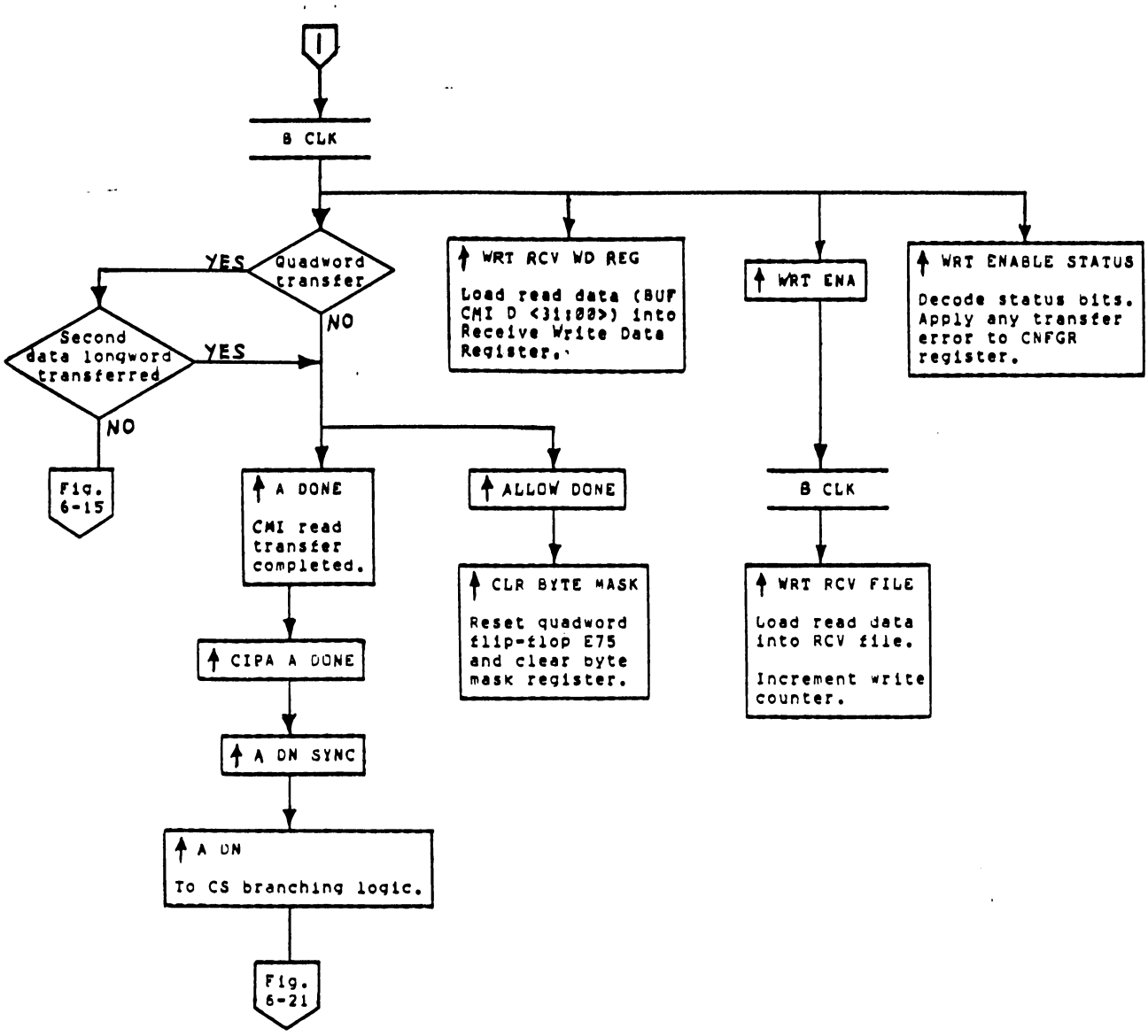


Figure 6-20 Status Cycle and Load RCV File Flow Diagram (Sheet 2 of 2)

If this is a quadword transfer and only the first longword has been transferred, the flow diagram returns to the arbitration function (Figure 6-15) to re-arbitrate for the CMI and transfer the second longword. When a quadword transfer was sensed during the command/address cycle, the arbitration sequence was executed up to the assertion of ARB OUT [A] (Figure 6-17). Now with DBBZ negated, the arbitration sequence can proceed. If in the interim, the CPU has not asserted CMI HOLD or a higher priority nexus is not requesting the CMI, the CI750 will regain the CMI for the second half of the quadword transfer.

If this is the last transfer of the read function, the CCI control logic asserts ALLOW DONE which in turn asserts CLR BYTE MASK. CLR BYTE MASK clears the byte mask register and resets quadword flip-flop E75 in the CCI register control logic.

In addition, GO/DONE PAL E121 sensing the true state of STATUS CYCLE, outputs A DONE indicating that the read transfer function is completed. A DONE is placed on the CIPA bus as CIPA A DONE and then coupled to the DP as A DN SYNC. A DN SYNC then causes A DN to be asserted to the CS branching logic. The port microcode is thereby informed of the completion of the commanded read function.

6.2.3.5 Unload RCV File -- Figure 6-21 is a flow diagram of the "unload RCV file" sequence. Figure 6-21 interfaces with Figure 5-19 in Chapter 5. Figure 5-19 illustrates the sequence that requests an unload of the read data in the RCV file and receives the data that is unloaded.

The port microcode requests that the read data in the RCV file be transferred to the DP by asserting the REG SEL code for a read of the RCV file. The REG SEL code is applied to the CCI read decoder via the CIPA bus. REG SEL 3 is false for a register read function (Table 5-10). A negated REG SEL 3 enables the read decoder which then decodes REG SEL bits <2:0> (Figure 6-11). The decoder outputs RD RCV FILE HI enabling the high section of the RCV file. The high word (CCI RCV DATA <15:00>) is transferred from location 0 of the RCV file to the CIPA bus as CIPA DATA <15:00> and then applied to the DP.

The microcode repeats the preceding sequence in order to retrieve the low half of the data longword. The DP places the REG SEL code for a read of the low word section of the RCV file, on the CIPA bus. The CCI read decoder responds to the code by asserting RD RCV FILE LO. RD RCV FILE LO enables the low section of the RCV file. The low word (CCI RCV DATA <15:00>) is transferred from location 0 of the RCV file to the CIPA bus as CIPA DATA <15:00> and then applied to the DP.

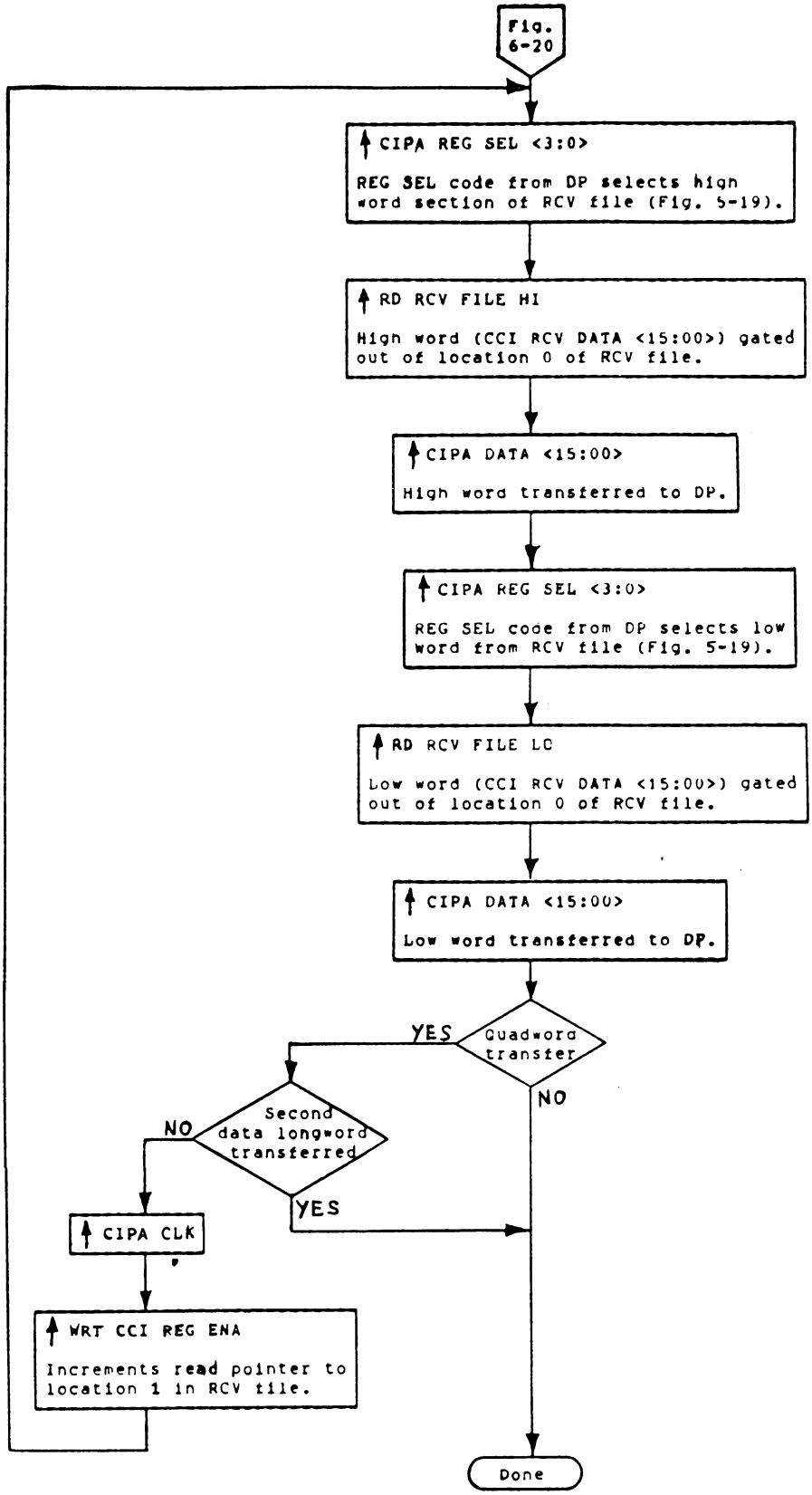


Figure 6-21 Unload RCV File Flow diagram

If this is a quadword read of the RCV file and only the first longword has been read out, the RCV file read pointer (RD CNTR <1:0>) is incremented to location 1 and the read sequence is repeated. The read pointer is incremented by CIPA CLK from the DP. CIPA CLK becomes WRT CCI REG ENA which then clocks the CCI read counter.

If this is the last transfer from the RCV file, the port read sequence of the CMI is completed.

6.2.4 Write Vector Function

The write vector function consists of two CMI cycles: a command/address cycle and a status cycle. The write vector function is described in three sequences. These are:

1. Issue Interrupt
2. Arbitration
3. Write Interrupt Vector

6.2.4.1 Issue Interrupt -- Figure 6-22 is a flow diagram of the "issue interrupt" function. Figure 6-23 is a block diagram of the "issue interrupt" logic.

When an interrupt condition occurs in the CI750, the DP places CIPA PORT INT on the CIPA bus which is then applied to the CCI "issue interrupt" logic as PORT INT. This action is illustrated in Figure 5-24 and described in Paragraph 5.12.1.

PORT INT is transferred through two flip-flops as D1 PORT INT and SYNC PORT INT. SYNC PORT INT is used to clock a BR flip-flop which then outputs an asserted BR.

BR is a bus request placed on the Unibus at the BR priority level established by the 16-pin socket E31. BR is applied to four AND gates with the second input of each gate tied to an E31 pin. The socket connects a + voltage from pin 12 to the pin at the specified BR level. For the CI750, this is normally pin 13 for a BR level of 4. This enables the BR4 AND gate resulting in the assertion of UBS BR4 on the Unibus.

The CPU arbitrates the BR requests from all the CMI devices. When it is ready to respond to the CI750's interrupt request, it asserts a bus grant to the CI750 at the same priority level as the BR request (level 4). Bus grants are daisy-chained on the Unibus from one device to another. Connector socket E31 connects all the bus grant inputs ([A]) to their corresponding outputs ([A+1]) for all priority levels except level 4. The bus grant input at level 4 (UBS BG4 [A]) is applied to the "issue interrupt" logic as BG IN. There is no output at the corresponding BG OUT terminal.

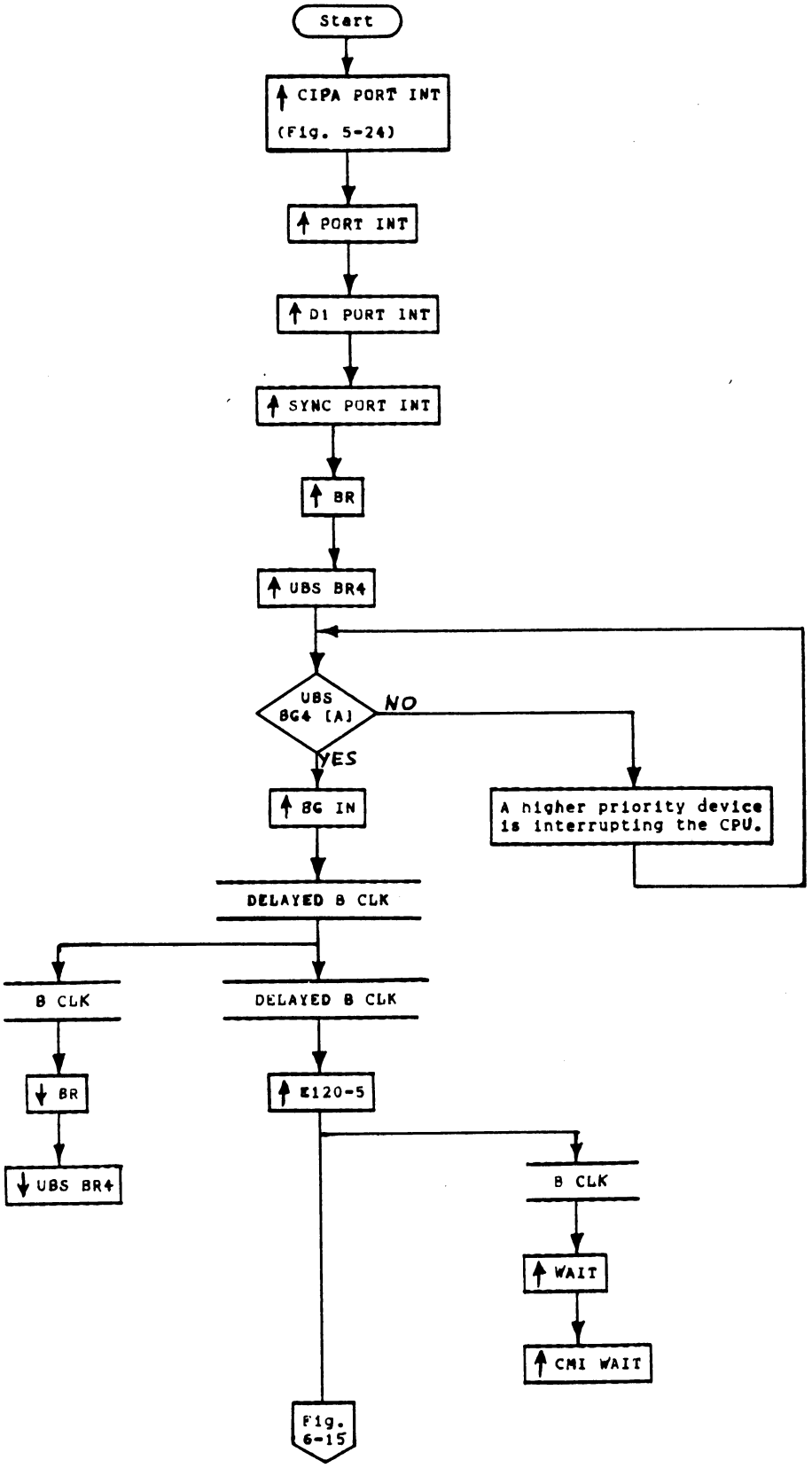
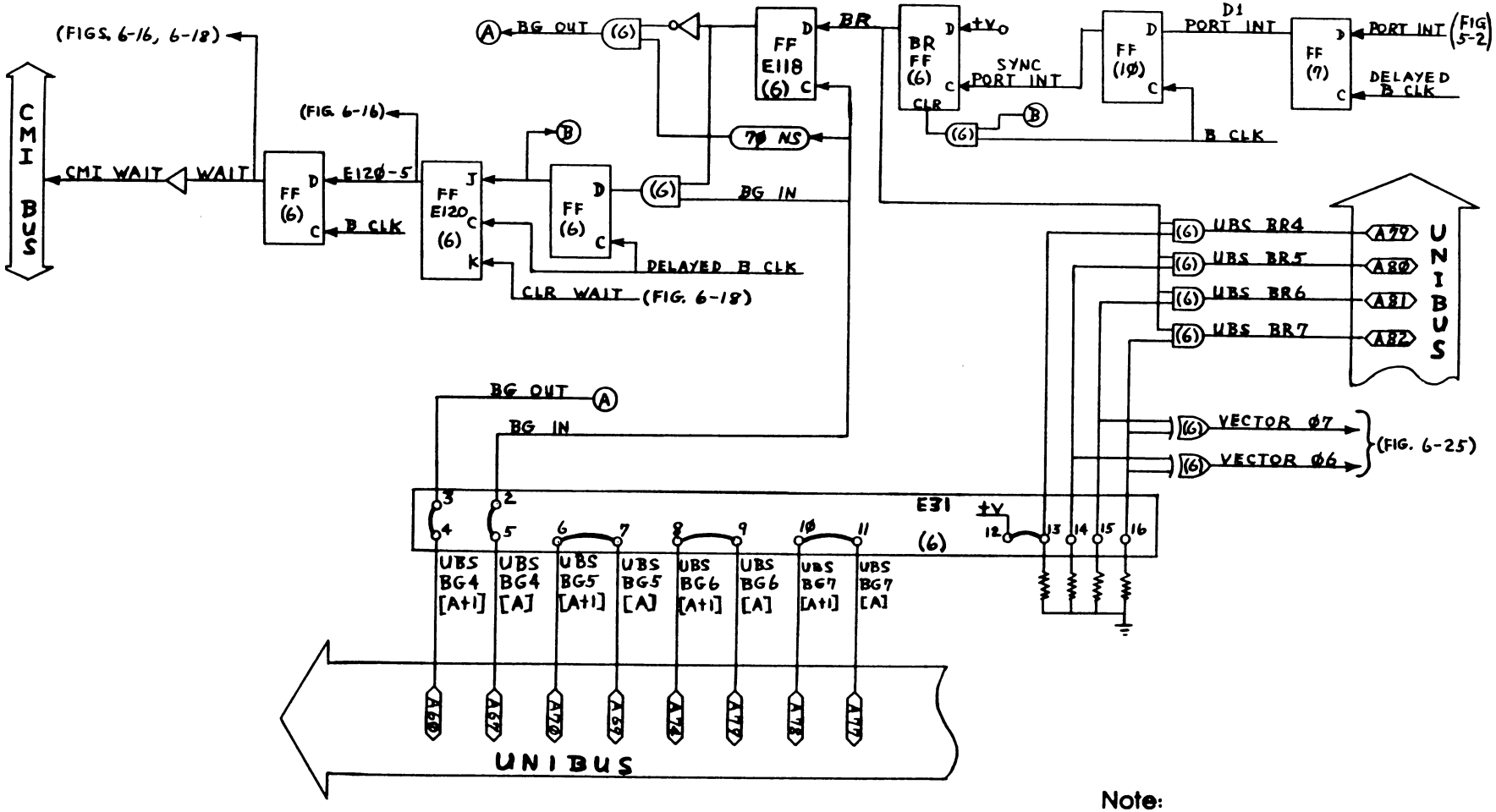


Figure 6-22 Issue Interrupt Flow diagram



Note:
Number designations in parentheses refer to engineering drawings containing corresponding logic.

Figure 6-23 Issue Interrupt Logic

If some other device also at BR level 4 had issued the bus request, the interrupt logic would output the bus grant at the BG OUT terminal and then to the Unibus as UBS BG4 [A+1] thereby passing the bus grant along to the device that had issued the bus request.

BG IN clocks a flip-flop (E118) conditioned to set by BR from the BR flip-flop. E118 sets and in so doing inhibits the assertion of BG OUT. Had BR been false (no bus request from the CI750), E118 would not have set and BG OUT would have been asserted (after a 70 ns delay*). The assertion of BG OUT would have returned the bus grant to the Unibus via the UBS BG4 [A+1] terminal.

* Time allowed for flip-flop E118 to set in the event the CI750 is requesting an interrupt.

BG IN is also applied to a chain of three flip-flops. The output of the first flip-flop resets the BR flip-flop thereby removing the bus request (UBS BR4) from the Unibus. The output of the second flip-flop (E120-5) is applied to the arbitration logic where it asserts GO ARB in the arbitration sequence.

The output of the third flip-flop (WAIT) is applied to the arbitration logic, the CCI control logic, and the CMI.

In the arbitration logic, WAIT inhibits the assertion of DO CMI XFER to the GO/DONE logic. Thus the GO/DONE logic does not re-assert GO if a quadword operation happened to be in progress when the interrupt occurred.

In the CCI control logic, WAIT indicates to logic array E26 that an interrupt transfer (not a data transfer) is executing. The array can then output the proper MUXA/MUXB SEL code for the "write vector" function (Paragraph 6.2.4.3).

On the CMI, CMI WAIT informs the host CPU of the pending "write vector" function.

6.2.4.2 Arbitration -- E120-5 is received by the arbitration logic which proceeds to arbitrate for control of the CMI bus. When the port gains control of the bus, it asserts DO CMI MASTER. The arbitration sequence is shown in Figure 6-15 and described in Paragraph 6.2.2.3.

6.2.4.3 Write Interrupt Vector -- Upon completion of a successful arbitration, the arbitration logic asserts DO CMI MASTER. The next B CLK initiates the command/address cycle during which the port asserts DBBZ and places the interrupt vector on the CMI bus. This is followed by the status cycle in which the CPU returns status to the port. Figure 6-24 is a flow diagram of the "write interrupt vector" sequence.

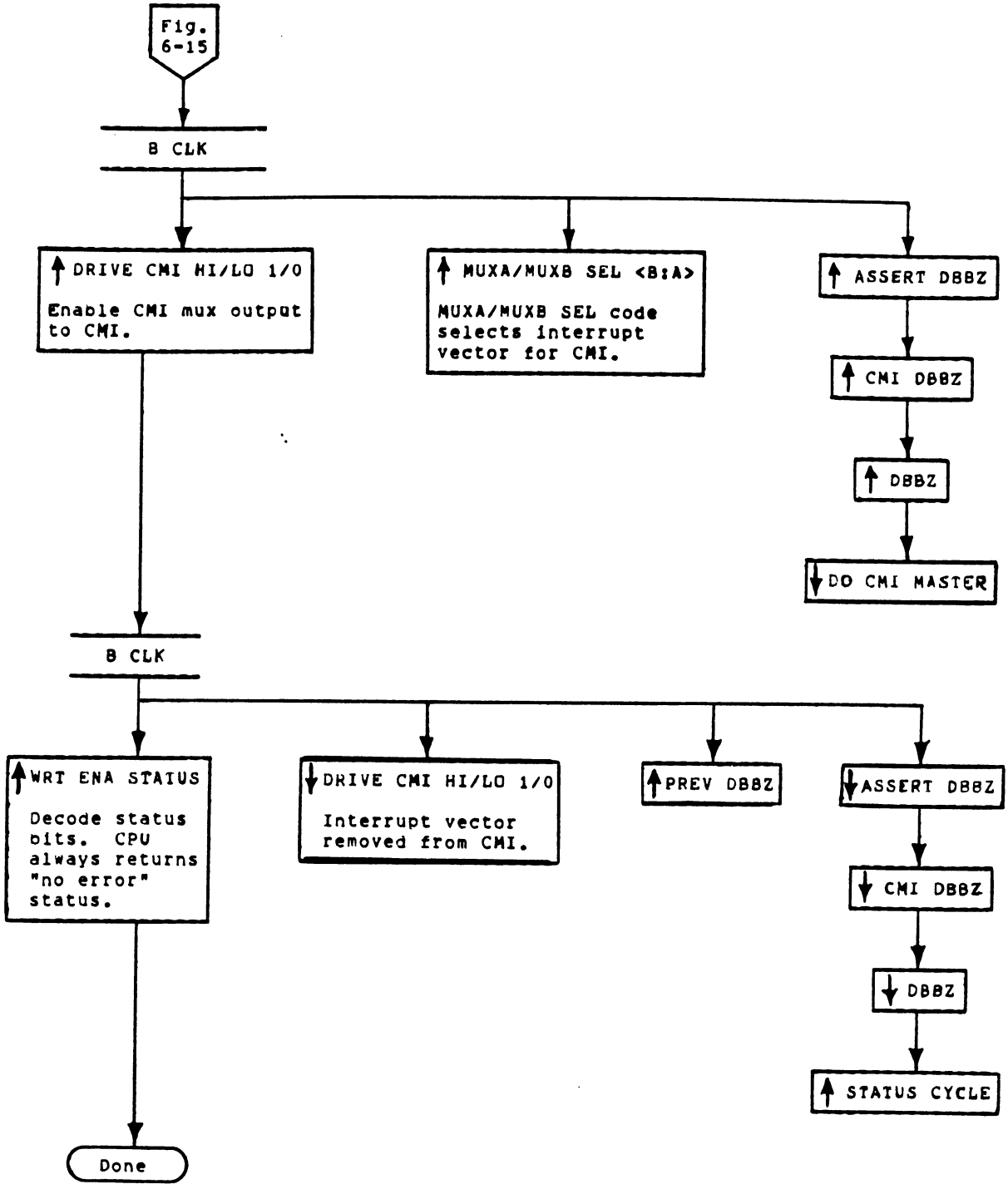


Figure 6-24 Write Interrupt Vector Flow Diagram

B CLK sets a DBBZ flip-flop (conditioned to set by DO CMI MASTER) asserting ASSERT DBBZ and then CMI DBBZ on the CMI bus (Figure 6-16). The true state of CMI DBBZ inhibits the arbitration process in all other CMI devices.

CMI DBBZ asserts DBBZ which negates DO CMI MASTER thereby conditioning the DBBZ flip-flop to reset on the next B CLK. Thus CMI DBBZ is asserted by the port for only one bus cycle (the command/address cycle).

DBBZ is also applied to a PREV DBBZ flip-flop conditioning it to set.

Also occurring during the command/address cycle is the assertion of DRIVE CMI HI/LO 1/0 by logic array E26 in the CCI control logic (Figure 6-18). When the array senses that the CMI bus has a master (DBBZ true) and that the CI750 is that master (DO CMI MASTER true), it asserts DRIVE CMI HI/LO 1/0 to gate the output from the CMI mux to the CMI bus.

Logic array E26 also senses the true state of WAIT from the interrupt logic. WAIT indicates to the memory array that an interrupt sequence is executing. Accordingly the memory array outputs the MUXA/MUXB SEL code that selects the interrupt vector for the CMI mux input (Table 6-3).

The next B CLK initiates the status cycle. With DO CMI MASTER false, B CLK resets the DBBZ flip-flop negating ASSERT DBBZ, CMI DBBZ, and DBBZ.

B CLK also sets the PREV DBBZ flip-flop asserting PREV DBBZ. With PREV DBBZ true and DBBZ false, STATUS CYCLE asserts.

In addition, B CLK causes logic array E26 to assert WRT ENA STATUS and negate DRIVE CMI HI/LO 1/0. WRT ENA STATUS enables the status decoder which decodes the status bits from the CMI. The CPU always returns a "no error" status in response to a write vector function. Hence there is no output from the status decoder.

The negation of DRIVE CMI HI/LO 1/0 inhibits the output of the CMI mux thereby removing the interrupt vector from the CMI bus.

The CPU now takes the appropriate action in response to the CI750 interrupt.

The interrupt vector (Figure 6-25) is a 32-bit command/address longword. The byte mask field (bits <31:28>) is all zeros and the function code (bits <27:25>) is 6. The address field (bits <23:00>) is determined by the CMI I/O "frequency slot" and the BR level assigned to the CI750. This is discussed below.

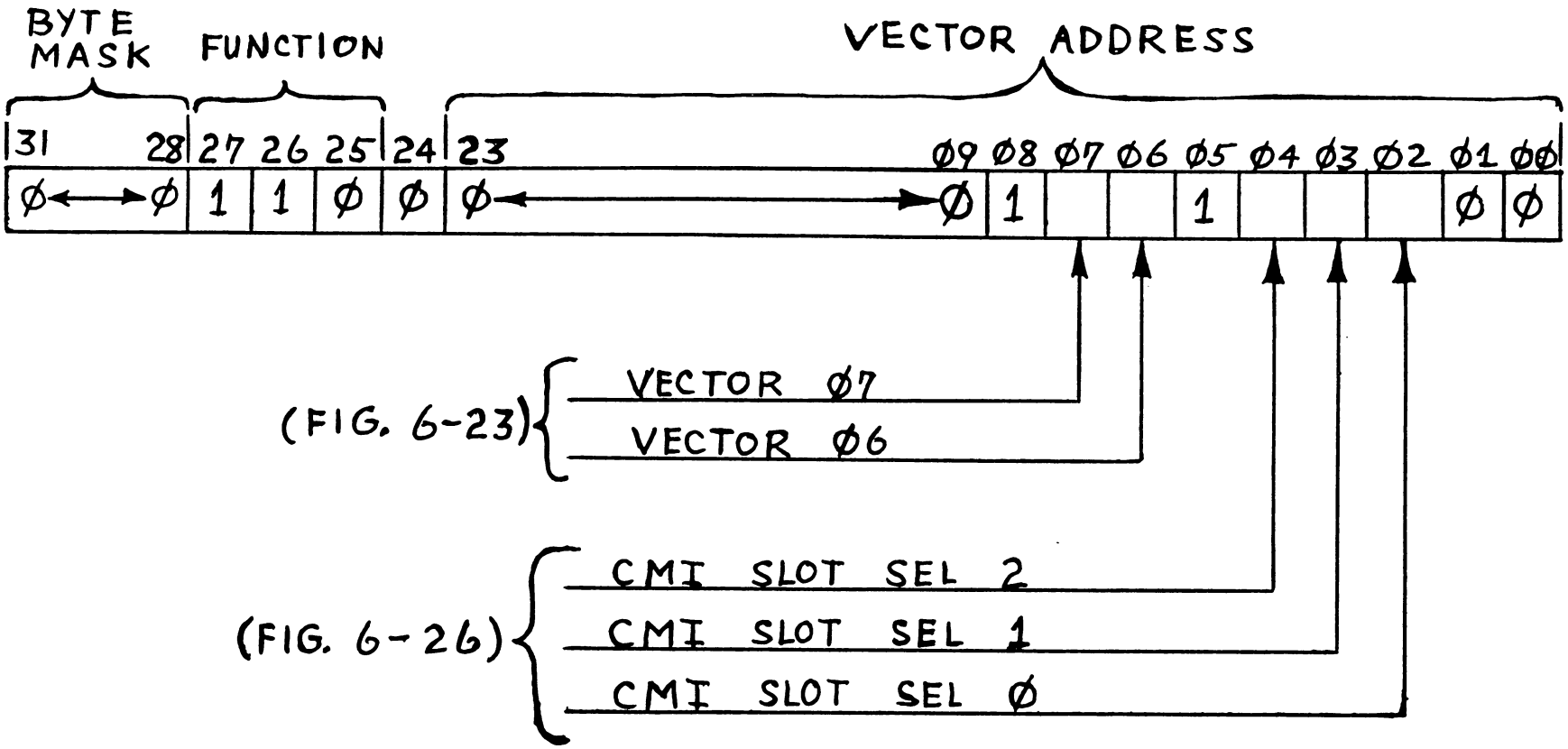


Figure 6-25 Interrupt Vector

Most of the vector bits are supplied by the CMI mux by use of + voltage pull-up and ground connections. These include the byte mask bits, the function bits and 19 of the 24 address bits. Five of the address bits are selectable. Table 6-4 lists all possible values of the interrupt vector.

Table 6-4 Interrupt Vector Values

I/O Slot No.	BR Level	Interrupt Vector Bits								Interrupt Vector <31:00> (hex)
		07	06	05	04	03	02	01	00	
10	4	0	0	1	0	1	0	0	0	0C00 0128
	5	0	1		0	1	0			68
	6	1	0		0	1	0			A8
	7	1	1		0	1	0			E8
11	4	0	0		0	1	1			2C
	5	0	1		0	1	1			6C
	6	1	0		0	1	1			AC
	7	1	1		0	1	1			EC
12	4	0	0		1	0	0			30
	5	0	1		1	0	0			70
	6	1	0		1	0	0			B0
	7	1	1		1	0	0			F0
13	4	0	0		1	0	1			34
	5	0	1		1	0	1			74
	6	1	0		1	0	1			B4
	7	1	1		1	0	1			F4
14	4	0	0		1	1	0			38
	5	0	1		1	1	0			78
	6	1	0		1	1	0			B8
	7	1	1		1	1	0			F8
15	4	0	0		1	1	1			3C
	5	0	1		1	1	1			7C
	6	1	0		1	1	1			BC
	7	1	1		1	1	1			FC

The five selectable bits are 07, 06, 04, 03, and 02. Three of the selectable bits (04, 03, 02) are established by the I/O "slot" in which the CI750 is located. There are six I/O "slots" (numbered 10 through 15 inclusive) which could be assigned to the CI750. Each slot has its own base address. Bits 04, 03, and 02 are connected to terminals designated as CMI SLOT SEL <2:0> respectively. By use of jumpers, the three SLOT SEL bits are made 1's or 0's according to the I/O slot assigned to the CI750.

The three SLOT SEL bits are used in the address decode logic to establish the I/O slot that the logic will recognize as being CI750 addresses. The address decode logic and the use of the SLOT SEL bits is described in Paragraph 6.3.1 (Command/Address Cycle) and Figure 6-26. It is sufficient here to say that the value of the SLOT SEL bits have already been established by the selection of the CI750 I/O slot.

The remaining two selectable address bits (07 and 06) are designated as VECTOR <07:06> respectively. The value of these bits is established by the BR level selected by jumper socket E31 (Figure 6-23). As seen in Table 6-4 and Figure 6-23, the binary value of bits VECTOR <07:06> increases from 00 to 11 as the BR level changes from BR4 to BR7.

Table 6-4 lists the binary values of vector bits <07:00>, and the hex values of the entire vector longword, for all four BR levels in each I/O slot. The normal selections for the CI750 is BR4 and I/O slot 15, resulting in a normal interrupt vector value of 0C00 013C.

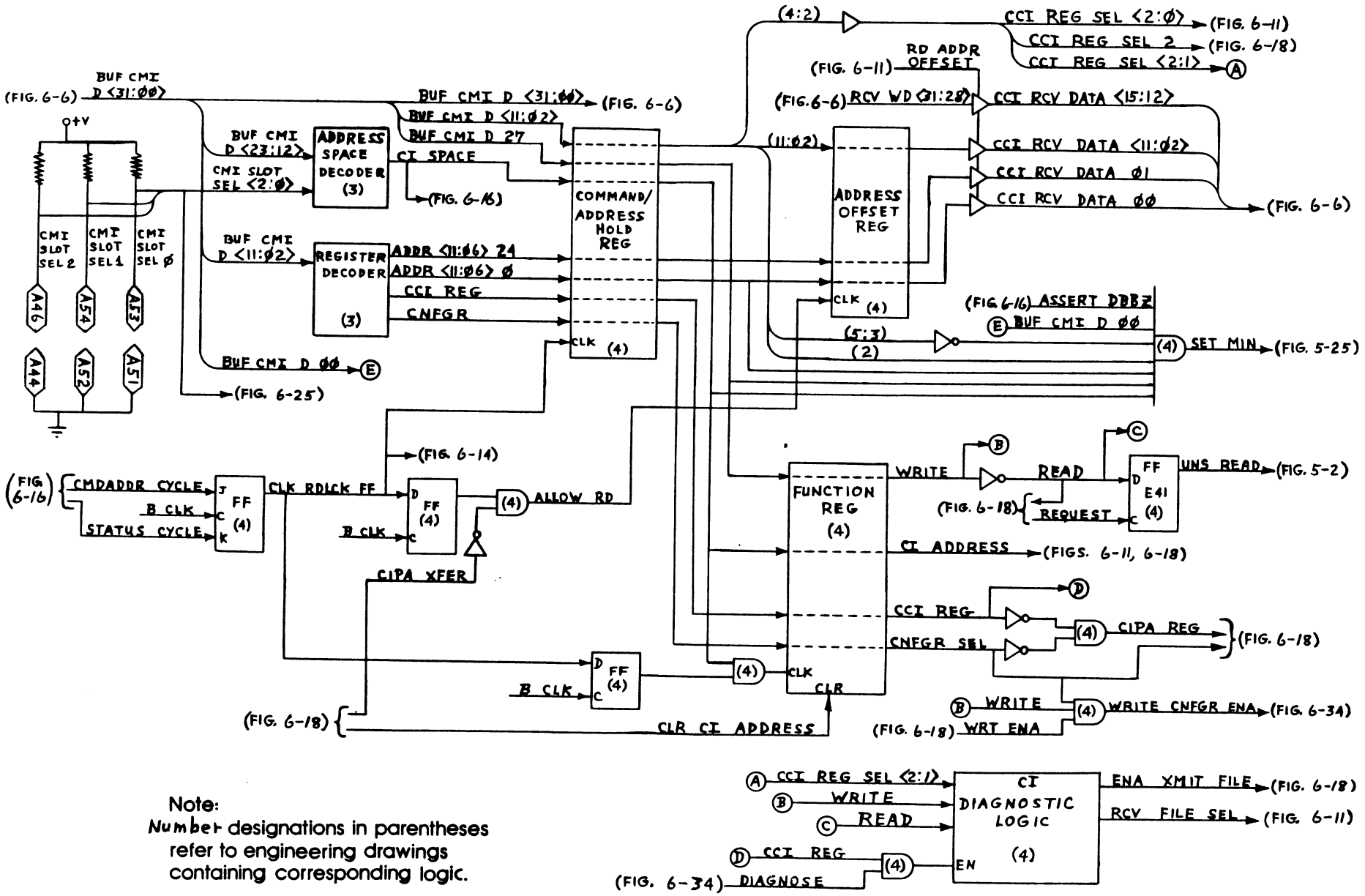
6.3 UNSOLICITED CMI OPERATIONS

The flow diagrams and descriptions given in section 6.3 are a detailed expansion of the general flow diagram of unsolicited CMI transfers given in Figure 6-9. Refer to Figure 6-9 and the CCI block diagram (Figure 6-6) in the following discussion.

6.3.1 Command/Address Cycle

In the command/address cycle, the CPU addresses the CI750 port and the register that is to be accessed. It places the address on the CMI along with DBBZ and the requested function (read or write). The CI750 takes the command/address longword off the CMI and decodes the address and function fields. If the CI750 determines that the command/address reference is for it, it asserts DBBZ on the CMI while the operation executes.

Figure 6-26 illustrates the logic involved in decoding the address field. The logic includes an address space decoder to determine if the CI750 is being addressed, and a register decoder to determine the register or register area that is being addressed. The decoders are discussed in Paragraphs 6.3.1.1 and 6.3.1.2.



Note:
Number designations in parentheses refer to engineering drawings containing corresponding logic.

Figure 6-26 Unsolicited Decode And Register Logic

The byte mask bits (BUF CMI D <31:28>) are ignored by the CI750. Function bits BUF CMI D <26:25> are also ignored. Function bit BUF CMI D 27 is used to specify either a read or a write operation (these are the only two unsolicited functions; see Table 6-2).

6.3.1.1 Address Space Decoder

Address bits BUF CMI D <23:12> are applied to an address space decoder where they are compared with the base address of the CI750 (Figure 6-27). If a match is obtained, the logic asserts CI SPACE indicating that the port is being addressed by the CPU and the command/address data is for the CI750.

Bits 15, 14 and 13 in the address space decoder are designated as SLOT SEL <2:0> respectively. The bits are connected to backplane terminals and by use of jumpers, can take on values of one or zero. The jumpers select the port base address from six possible values thereby placing the 8K of CI750 address space in one of six I/O "frequency slots" as shown in Table 6-5. The CI750 is normally in I/O slot no. 15.

Table 6-5 CI750 I/O Slots

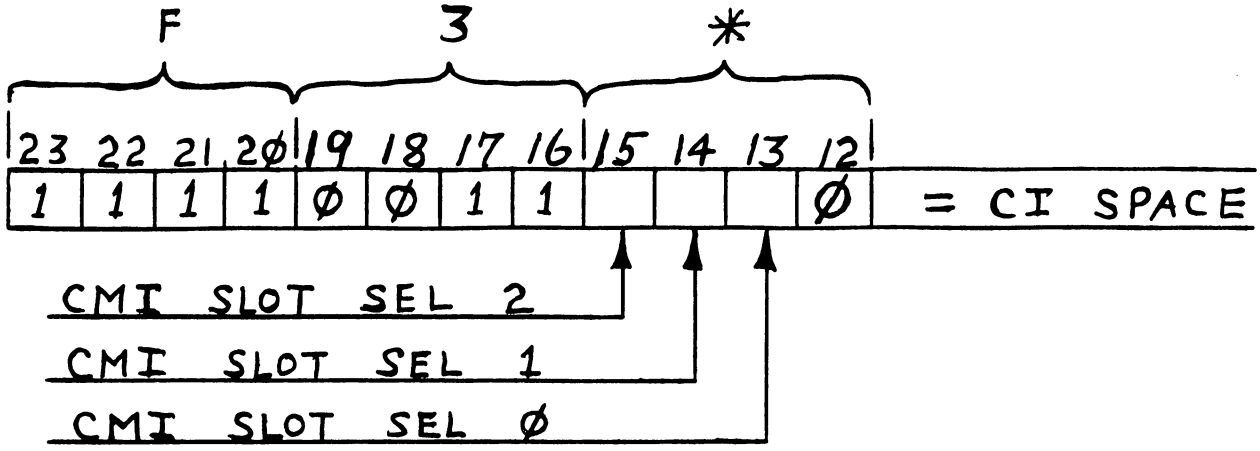
I/O Slot No.	Base Address	Bit 15	Bit 14	Bit 13
		SLOT SEL 2	SLOT SEL 1	SLOT SEL 0
10	F34000	0	1	0
11	F36000	0	1	1
12	F38000	1	0	0
13	F3A000	1	0	1
14	F3C000	1	1	0
15	F3E000	1	1	1

6.3.1.2 Register Decoder -- Offset Address bits BUF CMI D <11:02>* are applied to a register decoder where they are decoded to specify the port register or register area that is to be accessed. Figure 6-27 illustrates the response of the register decoder to the address bits. Figure 6-28 illustrates CI750 address space relative to the signals asserted by the register decoder. The total CI750 address space is 8K. As seen in Figure 6-28, all the CI750 registers (including the LS and VCDT) are located within the first 4K of address space.

* The two lowest address bits (BUF CMI D <01:00>) are ignored as all unsolicited CMI transfers are longword aligned.

In the range from 000 to 03C (hex), ADDR <11:06> 0 is asserted. In this range are all the port hardware registers.

At address 000 the reference is to the CNFGR register and the decoder asserts CNFGR.



* Normally E

A. Address Space Decoder

11	10	09	08	07	06	05	04	03	02	
1	0	0	1	0	0	X	X	X	X	= ADDR <11:06> 24
0	0	0	0	0	0	X	X	X	X	= ADDR <11:06> 0
0	0	0	0	0	0	1	X	X	X	= CCI REG
0	0	0	0	0	0	0	0	0	0	= CNFGR

X = Don't care

B. Register Decoder

Figure 6-27 CI750 Address Responses

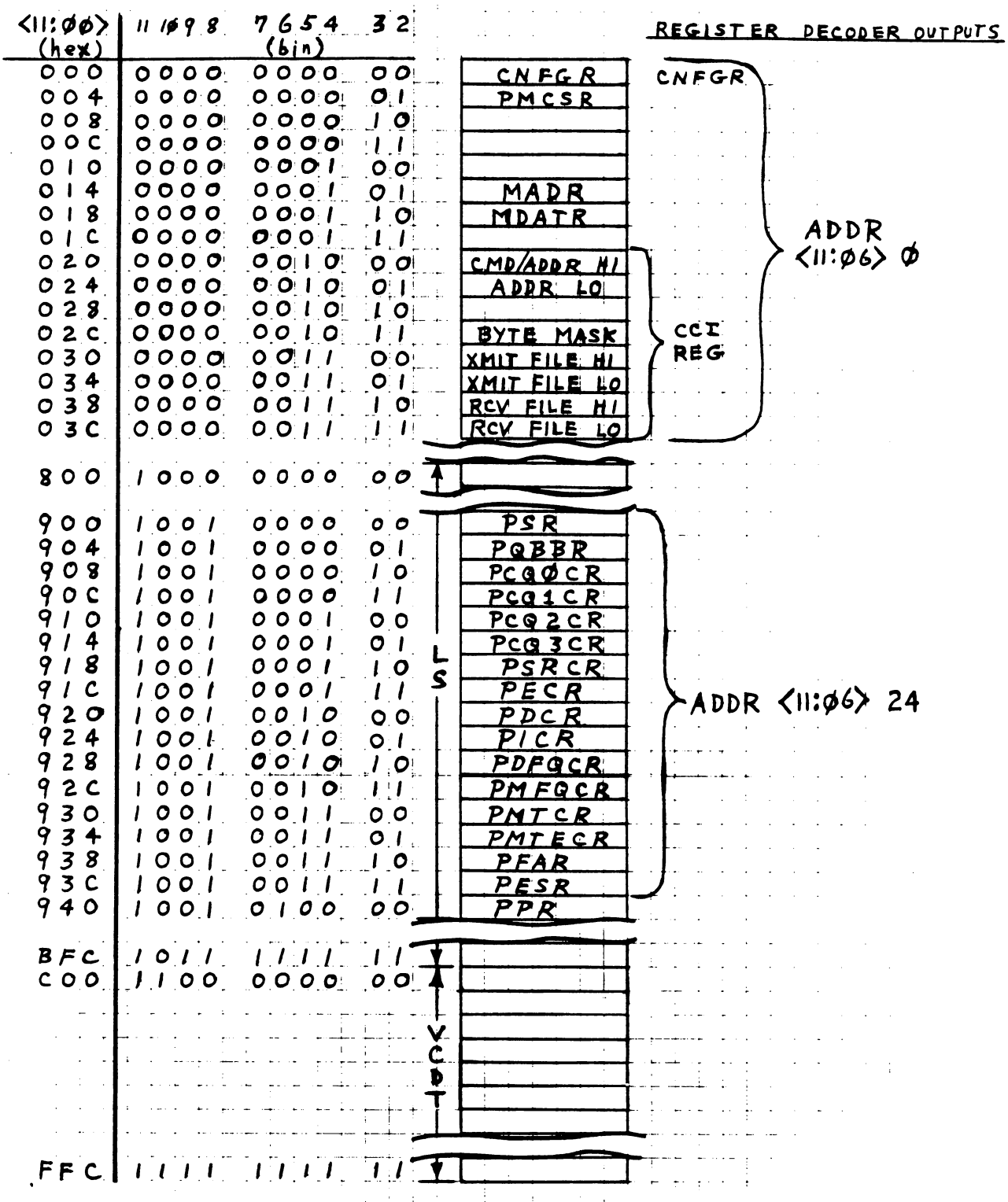


Figure 6-28 CI750 CMI Address Space vs Register Decoder Outputs

From address 020 to 03C, CCI REG is asserted indicating a maintenance function. The registers in the 020 to 03C range (CMD/ADDR HI, ADDR LO, byte mask, XMIT file, and RCV file) are not accessed by the host CPU for normal unsolicited operations. They are accessed only for diagnostic testing of the CCI hardware. The CNFGR register is the only CCI register accessed by a normal unsolicited operation. (Hence, the false state of CNFGR SEL indicates access to the DP.)

The LS (local store) area in the DP extends from 800 to BFC. In the range from 900 to 93C, ADDR <11:06> 24 is asserted. In this range are most of the software registers associated with the port architecture.

The range from C00 to FFC is for the VCDT (virtual circuit descriptor table) in the DP.

6.3.1.3 Command/Address Sequence -- Figure 6-29 is a flow diagram of the unsolicited command/address sequence.

The CPU asserts CMI DBBZ on the CMI and places the command/address on the CMI data/address lines (CMI DATA <31:00>). As shown in Figure 6-16 (arbitration logic), CMI DBBZ asserts DBBZ which in turn asserts CMDADDR CYCLE. The true state of CMDADDR CYCLE indicates this to be a command/address cycle from some other nexus on the CMI, not a CI750 initiated command/address cycle (ASSERT DBBZ false).

The next B CLK (designated as B CLK [1]) ends the command/address cycle. B CLK [1] asserts CLK RDLCK FF which loads the outputs from the address space decoder and register decoder into a command/address hold register. Address bits BUF CMI D <11:02> and function bit BUF CMI D 27 are also loaded into the hold register.

In addition, B CLK [1] sets the PREV DBBZ flip-flop asserting PREV DBBZ.

If CI SPACE is true, B CLK [1] sets the DBBZ flip-flop (CMDADDR CYCLE true) asserting ASSERT DBBZ. ASSERT DBBZ asserts CMI DBBZ on the CMI bus thereby holding the bus until the unsolicited function is completed. If CI SPACE were false, the CMI command/address was not for the CI750 and the port would not assert CMI DBBZ.

ASSERT DBBZ also negates CMDADDR CYCLE indicating the end of the command/address cycle. Note that the false state of CMDADDR CYCLE no longer conditions the DBBZ flip-flop to set. However NS BIT 3 is now asserted by logic array E26 to hold the flip-flop set (thereby keeping CMI DBBZ asserted) until the unsolicited operation is completed.

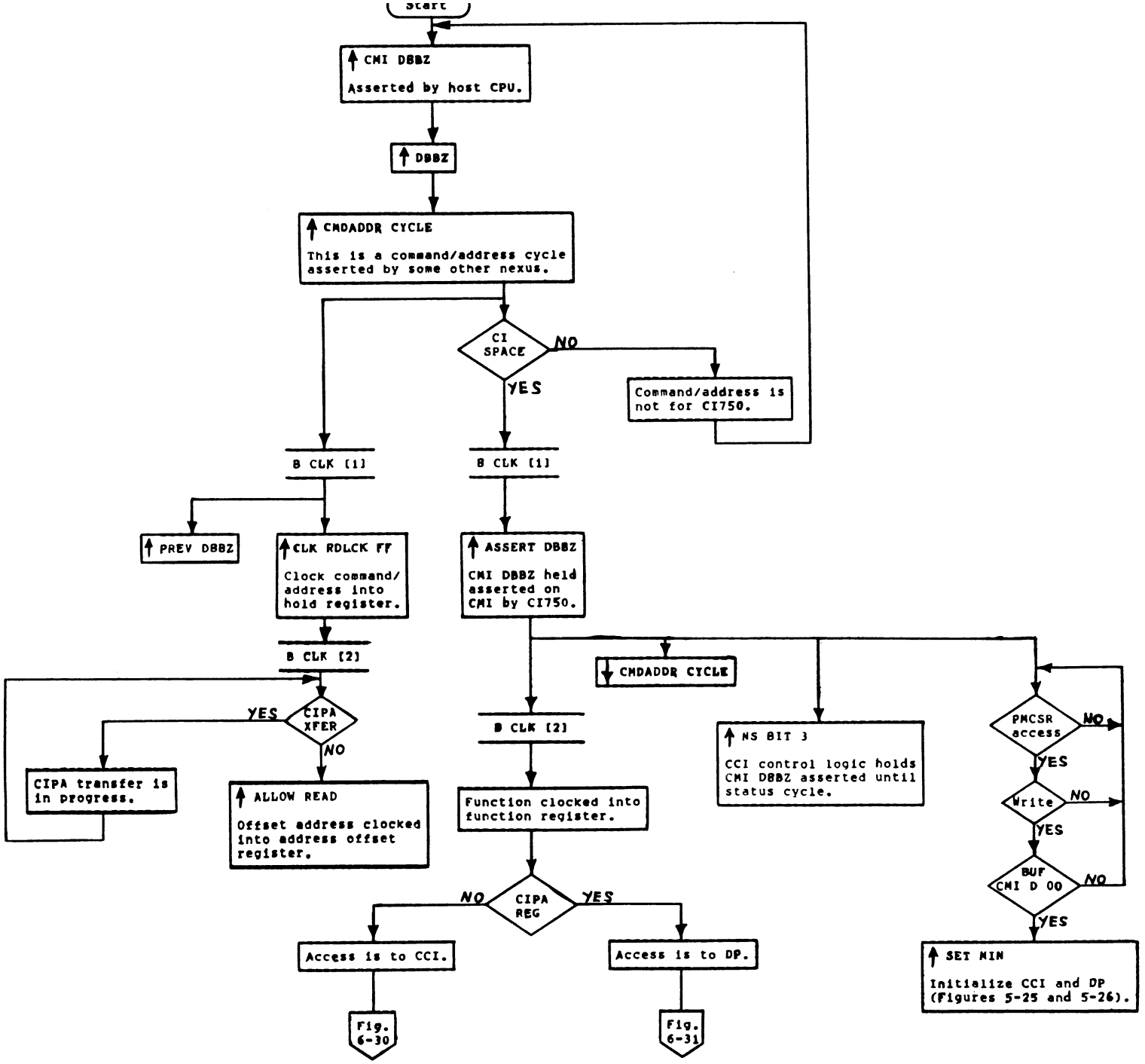


Figure 6-29 Flow Diagram Of Unsolicited Command/Address Sequence

Still another ASSERT DBBZ branch exists possibly resulting in the assertion of SET MIN. This is a special case of a "write DP" function and is discussed in Paragraph 6.3.3.3.

If there is no CIPA transfer in progress to/from the DP (CIPA XFER false), the next B CLK (B CLK [2]) will assert ALLOW RD which clocks the offset address from the hold register into the address offset register. The base offset address is address bits <11:02>, ADDR <11:06> 24, and ADDR <11:06> 0. ADDR <11:06> 24 and ADDR <11:06> 0 output from the address offset register as CCI RCV DATA <01:00> respectively.

B CLK [2] also clocks function data from the hold register into the function register. If the CCI REG and CNFGR SEL outputs from the function register are false, CIPA REG asserts indicating that access is to a register in the DP. CIPA REG is applied to the CCI control logic to indicate a CIPA transfer.

If CIPA REG is false, continue with Paragraph 6.3.2 for a CCI access. If CIPA REG is true, continue with Paragraph 6.3.3 for a DP access.

6.3.2 Read/Write CCI

Figure 6-30 is a flow diagram of all unsolicited read/write operations of the CCI. An unsolicited access to the CCI is either a maintenance function or a reference to the configuration register.

6.3.2.1 Maintenance Function -- CCI maintenance functions are capable of writing and reading all the CCI registers.

If a maintenance function is executing, CCI REG is asserted from the function register to enable the CCI diagnostic logic (Figure 6-26). The CCI diagnostic logic samples the READ and WRITE commands from the function register, and the CCI REG SEL code from the command/address hold register, to generate enabling signals for the RCV file and the XMIT file. The CCI REG SEL code is also applied to the read and write decoders in the CCI register control logic (Figure 6-11) for additional register and function selection.

After the register and the function are selected, the diagnostic maintenance routines are run.

6.3.2.2 Writing the CNFGR Register -- If a maintenance function is not executing, CNFGR SEL is asserted from the hold register indicating that the unsolicited reference is to the CNFGR register.

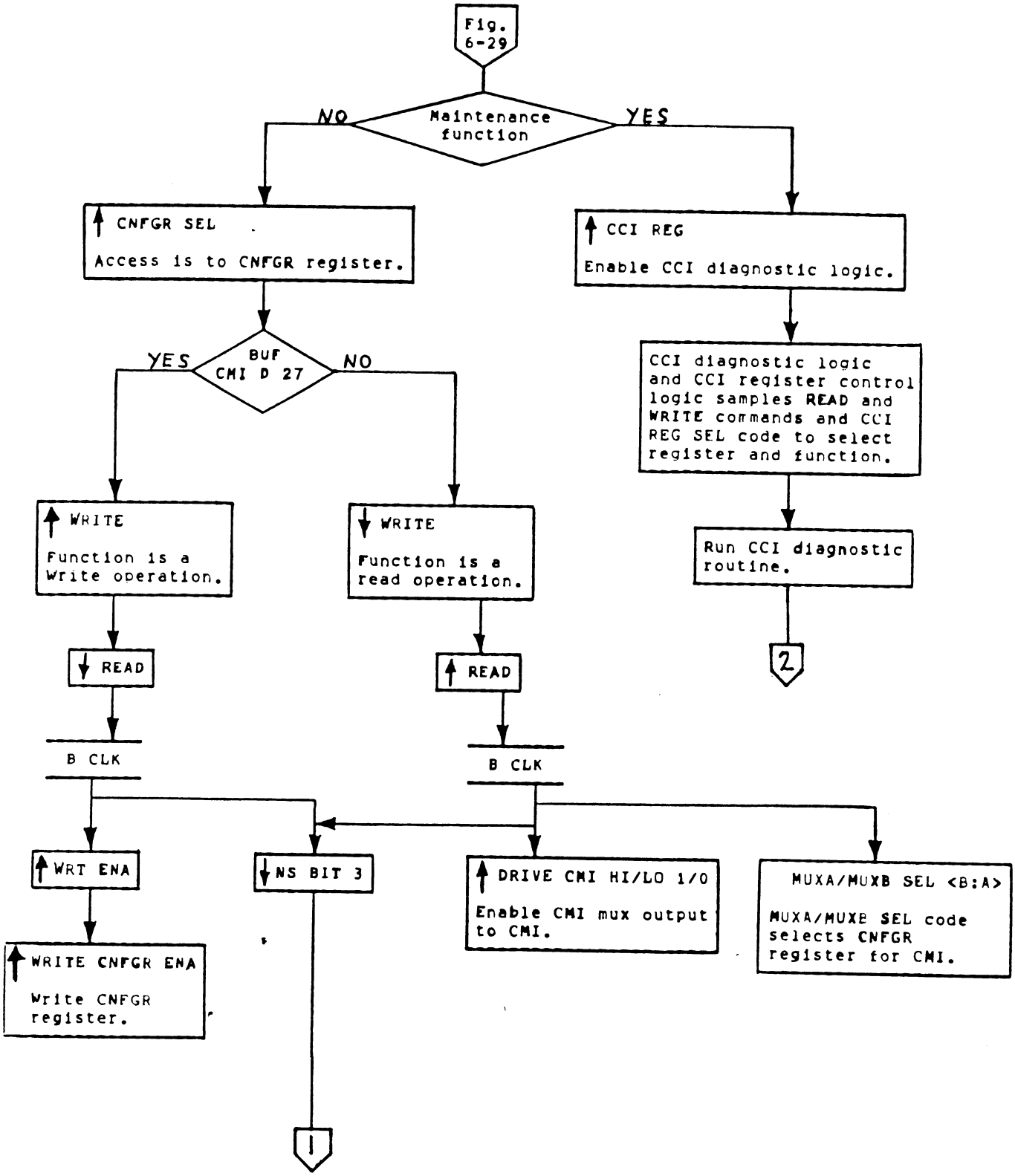


Figure 6-30 Read/Write CCI Flow Diagram
(Sheet 1 of 2)

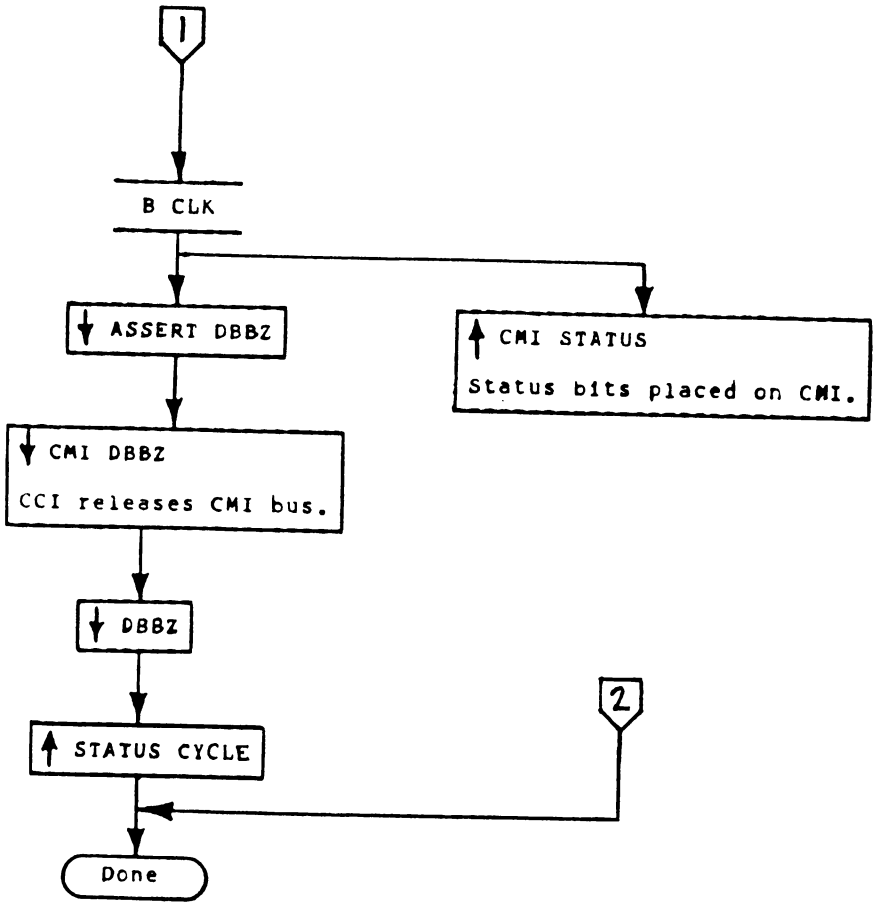


Figure 6-30 Read/Write CCI Flow Diagram
(Sheet 2 of 2)

When function bit 27 is true, a write operation is specified. In this case the function register outputs an asserted WRITE resulting in a negated READ. The false state of READ is sensed by the CCI control logic (Figure 6-18) which responds by asserting WRT ENA on the next B CLK. CNFGR SEL, WRITE, and WRT ENA are ANDed to generate WRT CNFGR ENA. WRT CNFGR ENA clocks write data bits BUF CMI D 31, 23, 22, 20, 19, 17, 16, 14, 13 and 08 into the CNFGR register. Due to the relatively quick access to the CNFGR register (as compared to having to access a register in the DP), the ten write data bits are taken directly from the CMI. Latching them up in a holding register is not necessary.

The B CLK that asserted WRT ENA also caused NS BIT 3 from the CCI control logic to negate thereby conditioning the DBBZ flip-flop to reset.

The following B CLK resets the DBBZ flip-flop negating ASSERT DBBZ. The negation of ASSERT DBBZ negates CMI DBBZ and then DBBZ. The negation of DBBZ causes STATUS CYCLE to assert (PREV DBBZ true) indicating that this is the status cycle and valid status data is on the CMI.

The B CLK that reset the DBBZ flip-flop also caused the CCI control logic to assert the CMI status bits on the CMI.

6.3.2.3 Reading the CNFGR Register -- If a maintenance function is not executing (CNFGR SEL true) and function bit 27 is false, a read operation is specified. In this case the function register outputs a negated WRITE resulting in an asserted READ. The true state of READ is sensed by the CCI control logic which responds by asserting DRIVE CMI HI/LO 1/0 and MUXA/MUXB SEL <B:A> on the next B CLK. DRIVE CMI HI/LO 1/0 enables the CMI mux output to the CMI. The asserted MUXA/MUXB SEL code selects the CNFGR register output for the CMI mux input. The register bits are thus transferred to the data/address lines on the CMI.

The B CLK that asserted the DRIVE CMI HI/LO 1/0 enabling signal and the MUXA/MUXB SEL code, also caused NS BIT 3 from the CCI control logic to negate. The negation of NS BIT 3 conditions the DBBZ flip-flop to reset.

The following B CLK completes the read sequence by negating CMI DBBZ and asserting the status bits on the CMI bus just as for the "write CNFGR register" operation.

6.3.3 Read/Write DP

Read/write access to the DP involves three operations. These are:

1. CIPA Transfer Request - The CCI requests a CCI/DP transfer and provides the DP with the address of the register to be accessed. If this is a write function, the operation includes taking the write data off the CMI, loading it into the Receive Write Data Register, and releasing the CMI bus.

2. Write DP - Write data is transferred from the CCI to the selected register in the DP.

3. Read DP - Read data is taken from the selected register in the DP and transferred to the CMI via the CCI. The CMI bus is then released.

Paragraph 5.10.3 (Unsolicited Request Operations) describes corresponding actions occurring in the DP while the three operations are executing.

6.3.3.1 CIPA Transfer Request -- Figure 6-31 is a flow diagram of the "CIPA transfer request" operation.

When function bit 27 is true, a write function is specified. In this case the function register (Figure 6-26) outputs an asserted WRITE indicating that a DP register is to be written. The assertion of WRITE results in the negation of READ. The false state of READ is sensed by the CCI control logic (Figure 6-18) along with the true state of CIPA REG. In response to the asserted CIPA REG and the negated READ, the CCI control logic asserts WRT RCV WD REG on the next B CLK. WRT RCV WD REG is applied to the Receive Write Data Register where it loads the write data from the CMI bus into the register.

The CCI control logic also negates NS BIT 3. The negation of NS BIT 3 conditions the DBBZ flip-flop to reset (Figure 6-16). On the next B CLK the DBBZ flip-flop resets negating ASSERT DBBZ. The negation of ASSERT DBBZ negates CMI DBBZ and then DBBZ. The negation of CMI DBBZ releases the CMI bus.

The negation of DBBZ causes STATUS CYCLE to assert (PREV DBBZ true) indicating that this is the status cycle and valid status data from the CCI control logic is now on the CMI.

In addition, the CCI control logic asserts SET REQUEST. SET REQUEST asserts for both write and read functions. The assertion of SET REQUEST asserts REQUEST which is placed onto the CIPA bus as CIPA REQUEST. The DP responds to CIPA REQUEST by returning CIPA GRANT and the REG SEL code that specifies a read of the offset register (Paragraph 5.10.3). CIPA GRANT indicates to the CCI that the DP is executing the requested function. CIPA GRANT asserts GRANT and then SYNC GRANT which in turn negates REQUEST.

The REG SEL code is applied to the CCI read decoder which asserts RD ADDR OFFSET. RD ADDR OFFSET enables the output of the offset register (CCI RCV DATA <11:00>) onto the CIPA. The 12-bit offset address specifies the DP register to be accessed for the unsolicited function.

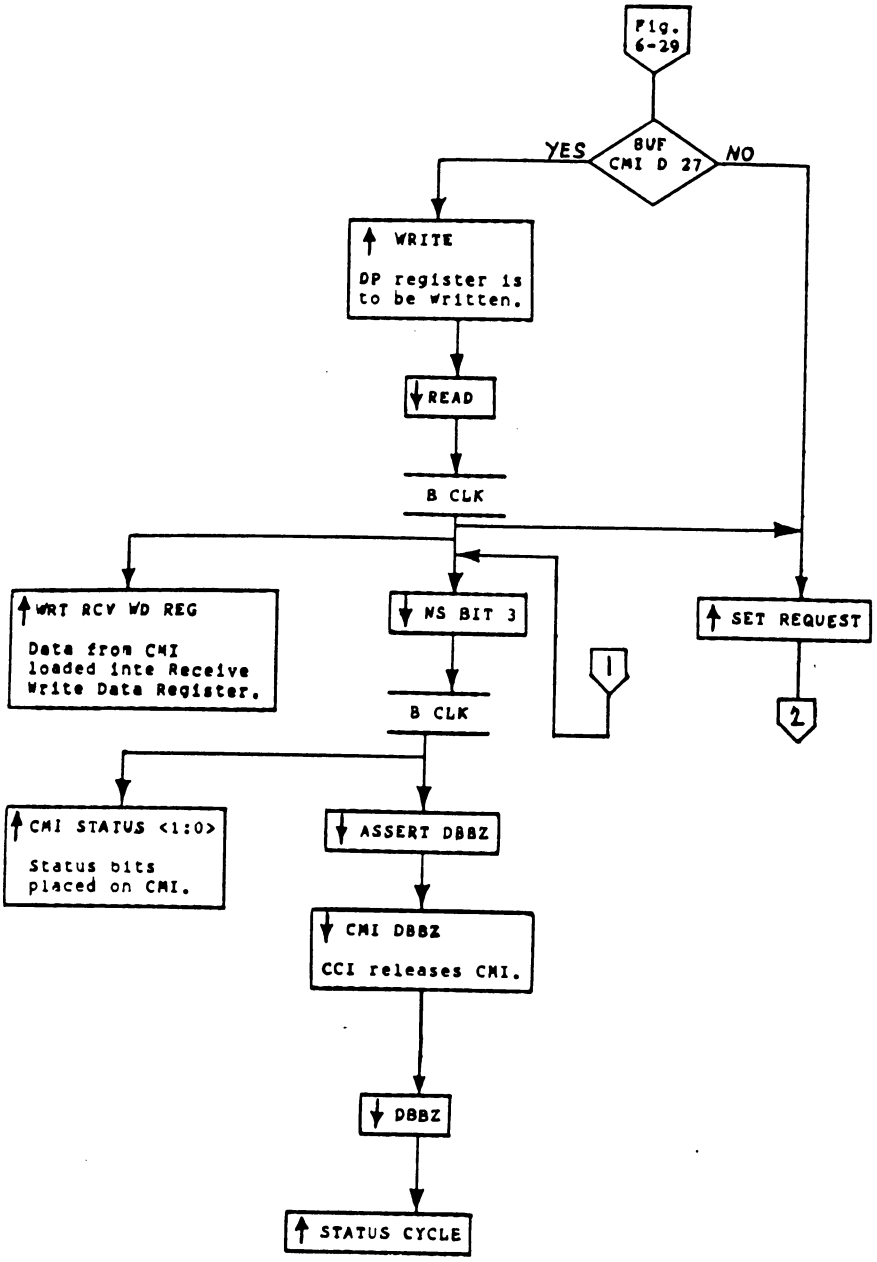


Figure 6-31 Flow Diagram of CIPA Transfer Request (Sheet 1 of 2)

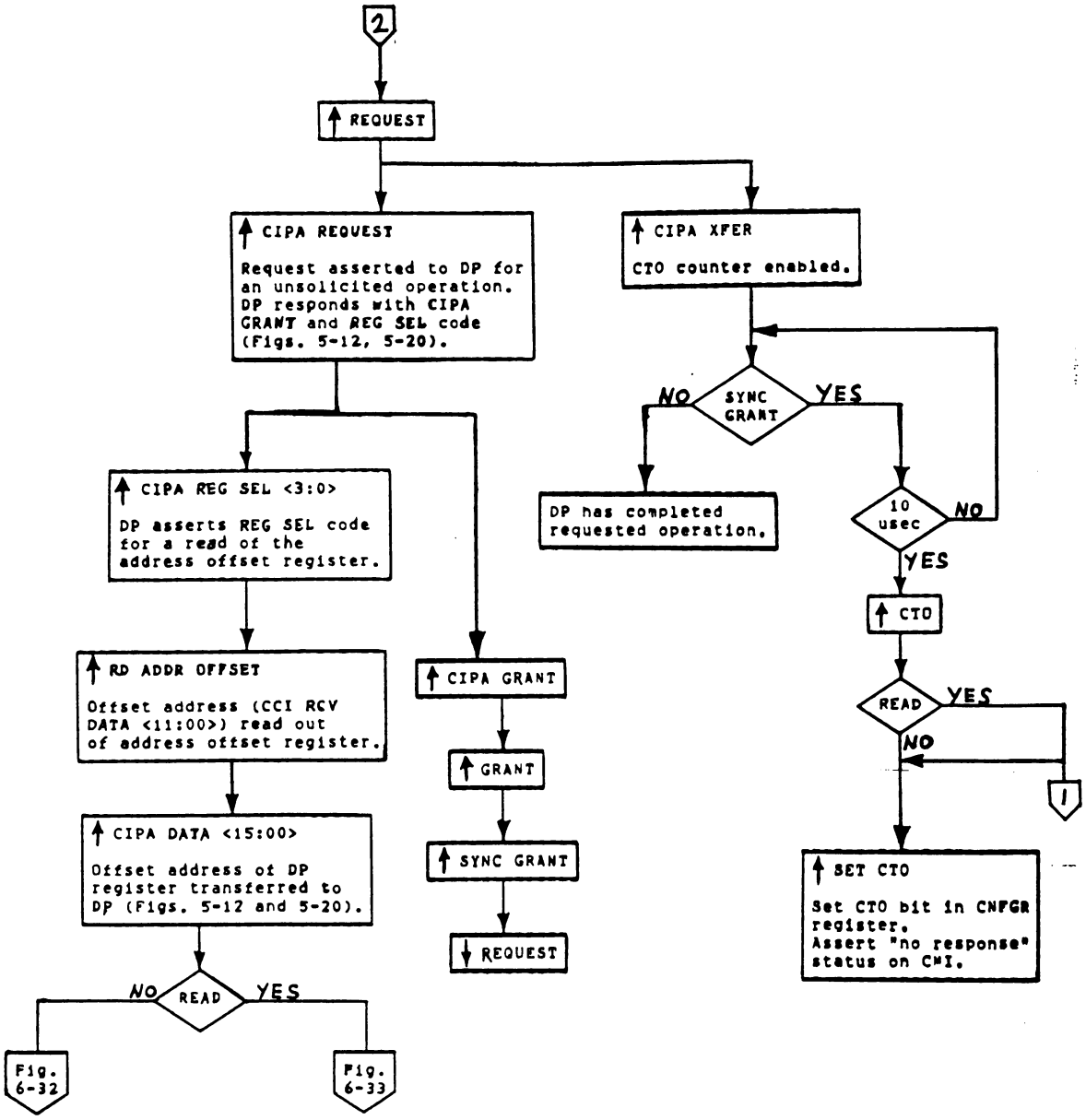


Figure 6-31 Flow Diagram of CIPA Transfer Request (Sheet 2 of 2)

The address offset on the CIPA (CIPA DATA <15:00>)* is applied to the DP to enable the register that is to be accessed.

* Bits CIPA DATA <15:12> are added to the 12-bit offset address to insure that these lines are not at a tri-state level. They serve no function in the DP.

If the requested function is a write operation (negated READ from function register), continue with Paragraph 6.3.3.2. If the requested function is a read operation (asserted READ from function register), continue with Paragraph 6.3.3.4.

It is seen in Figure 6-31 that when the unsolicited request was sent to the DP (REQUEST asserted), CIPA XFER was asserted which enabled a CTO (CIPA time-out) counter in the CCI control logic. The return of CIPA GRANT from the DP asserted GRANT and then SYNC GRANT which in turn negated REQUEST. With REQUEST false, SYNC GRANT holds CIPA XFER asserted thereby keeping the CTO counter enabled. The counter is incremented by B CLK and continues to run until SYNC GRANT is negated by the DP. The DP negates SYNC GRANT after it has completed the unsolicited operation. If the DP has not completed its read or write of the selected register and negated SYNC GRANT within 10 microseconds, the CTO counter asserts CTO. CTO in turn asserts SET CTO which sets the CTO bit in the CNFGR register.

In addition, if this is a read operation, the logic array (sensing the true state of CTO) negates NS BIT 3. Negating NS BIT 3 releases the CMI bus on the next B CLK and places status on the CMI as shown in Figure 6-31. In this case, the status bits placed on the CMI bus will indicate a "no response".

If this is a write operation, the CMI bus would have been released right after the write data was taken off the bus.

6.3.3.2 Write DP

The write DP operation consists of transferring the write data from the Receive Write Data Register to the DP. The CMI bus has already been released.

Figure 6-32 is a flow diagram of the "write DP" operation.

READ from the function register is applied to the "D" input of flip-flop E41 in the unsolicited decode and register logic (Figure 6-26). The flip-flop is clocked by REQUEST from the CCI control logic. With READ false, the flip-flop output (UNS READ) remains false resulting in a negated CIPA READ sent to the DP. The DP interprets the negated CIPA READ as a write command. The DP then returns the REG SEL code for a high word read of the Receive Write Data Register.

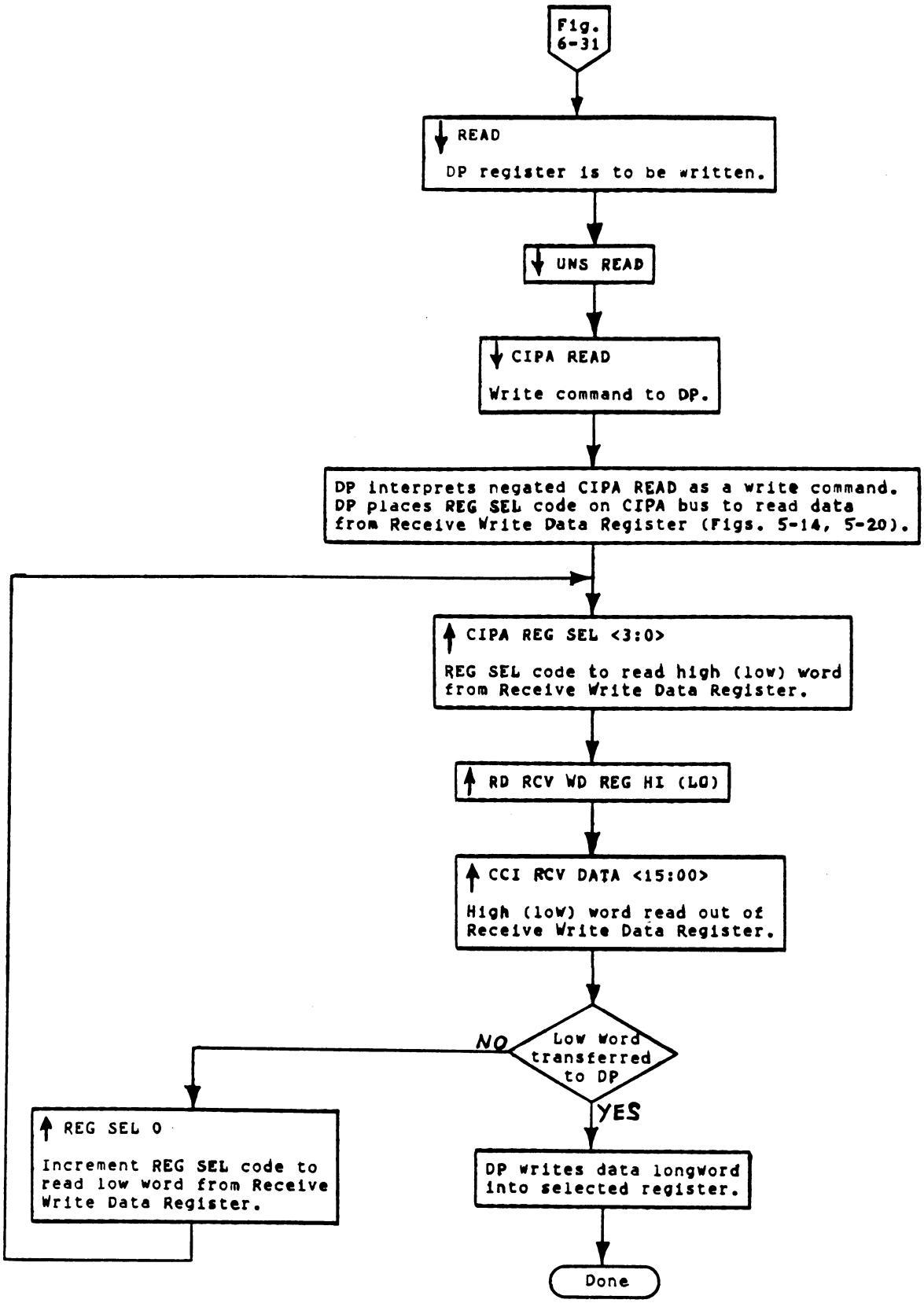


Figure 6-32 Write DP Flow Diagram

The REG SEL code (CIPA REG SEL <3:0>) is applied to the read decoder in the CCI. The decoder outputs RD RCV WD REG HI to the Receive Write Data Register. RD RCV WD REG HI enables bits <31:16> from the register onto the CCI RCV DATA bus. From the CCI RCV DATA bus, the high word is transferred to the DP via the CIPA bus as CIPA DATA <15:00>.

The DP accepts the data high word and increments the REG SEL code to read the low word from the Receive Write Data Register. The DP increments the REG SEL code by asserting REG SEL 0 changing it from a 0 to a 1 (see Table 5-10).

The incremented REG SEL code is returned to the read decoder in the CCI which outputs RD RCV WD REG LO. RD RCV WD REG LO is applied to the Receive Write Data Register where it enables bits <15:00> from the register onto the CCI RCV DATA bus. From the CCI RCV DATA bus the low word is transferred to the DP via the CIPA bus as CIPA DATA <15:00>.

The DP then proceeds to write the data longword into the selected register to complete the unsolicited write operation.

6.3.3.3 Maintenance Initialize (MIN) -- A special case occurs when the MIN bit in the PMCSR in the DP is to be written.

When ASSERT DBBZ asserts during the bus cycle following the command/address bus cycle (Figure 6-29), the command/address in the hold register is examined by a SET MIN AND gate (Figure 6-26). Address bit <11:06> 0 and address bits <05:02> are examined. If the address bits show the access is to the PMCSR (Figure 6-28), and the commanded function is a write (CMI bit 27 true), and if the PMCSR MIN bit (bit 00) is being referenced (BUF CMI D 00 true), then SET MIN asserts.

SET MIN initiates an initialization sequence within the CCI and the DP. The initialization sequence and the associated logic is described in Paragraph 5.12.2 and illustrated in Figures 5-25 and 5-26).

6.3.3.4 Read DP -- The read DP operation consists of receiving the read data from the DP, loading the data into the Return Read Data Register, transferring the data from the Return Read Data Register to the CMI, and releasing the CMI.

Figure 6-33 is a flow diagram of the "read DP" operation.

READ from the function register is applied to the "D" input of flip-flop E41 in the unsolicited decode and register logic (Figure 6-26). The flip-flop is clocked by REQUEST from the CCI control logic. With READ true, the flip-flop sets asserting UNS READ and sending an asserted CIPA READ to the DP. The DP then reads the selected register and places the high word of the read data on the CIPA bus along with the REG SEL code for a high word write of the Return Read Data Register.

6.4 CNFGR REGISTER

The CNFGR register contains 13 information bits and an 8-bit adaptor code. The remaining 11 bits of the CNFGR register are zeros supplied by the CMI mux.

The CNFGR register is written by the assertion of WRITE CNFGR ENA from the function register logic (Paragraph 6.3.2.2). Only 10 of the 13 information bits are writeable. The other three (NO CIPA, T ACLO, T DCLO) are read only.

The CNFGR register is read by selecting it as the input to the CMI mux and enabling the mux output to the CMI (Paragraph 6.3.2.3).

The CNFGR register bit fields are shown in Figure C-4 in Appendix C (Hardware Registers). The CNFGR register logic is illustrated in Figure 6-34. The information bits and the 8-bit adapter code are described below.

6.4.1 Adapter Code

The adapter code is an ID that identifies the CI750 on the CMI bus. The code number is 38 (hex) and is supplied by the CMI mux via +V pull-up and ground connections. The adapter code is read only.

6.4.2 PDN, PUP, NO CIPA

PDN (power-down) is a flag indicating that the port is powering down. PDN is set by the assertion of SET PDN which comes true whenever ACLO asserts in the CIPA cabinet or in the host CPU cabinet. PDN can be cleared by writing the CNFGR register with BUF CMI D 23 asserted. PDN is also cleared by SET PUP or by CLR PUP/PDN from the CCI initialize logic.

PUP (power-up) is a flag indicating that the system is powered up. PUP is set by the assertion of SET PUP which comes true when ACLO negates in both the CIPA cabinet and the host CPU cabinet. PUP can be cleared by writing the CNFGR register with BUF CMI D 22 asserted. PUP is also cleared by SET PDN or by CLR PUP/PDN from the CCI initialize logic.

NO CIPA is a flag indicating the ready state of the CIPA cabinet. NO CIPA is false if the CIPA is present, powered-up, and initialized. Otherwise NO CIPA is true. NO CIPA is read only.

A complete description of the PDN, PUP, and NO CIPA functions is given in Paragraph 5.12.3 (Power Control Function) and in Figures 5-27, 5-28, and 5-29.

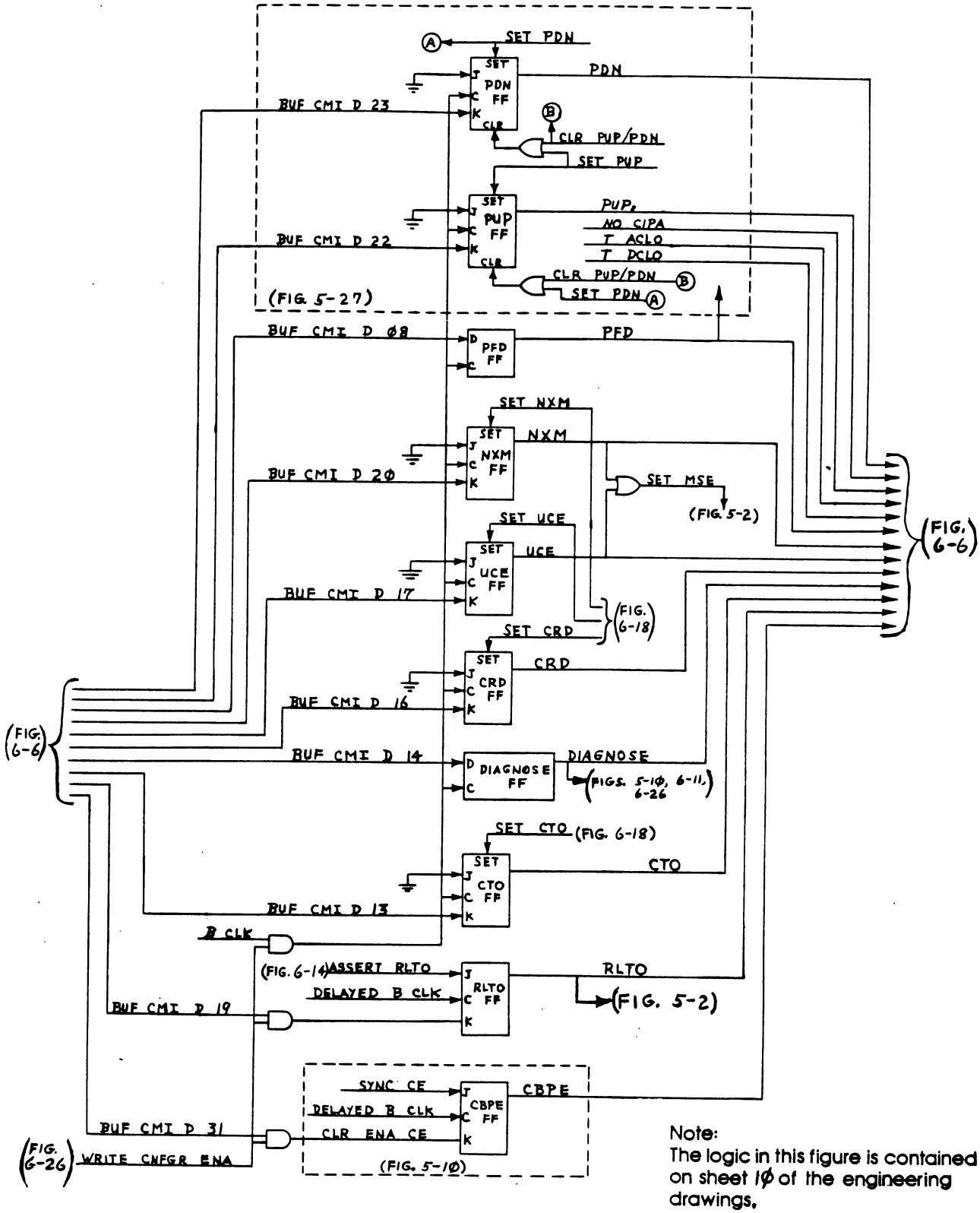


Figure 6-34 CNFGR Register Logic

6.4.3 T ACLO, T DCLO, PFD

T ACLO (transmitted ACLO) and T DCLO (transmitted DCLO) result from a reset command issued by a remote node on the CI cluster. T ACLO and T DCLO function to power down and power-up the host system while leaving the CI750 port operational. T ACLO and T DCLO are set and cleared by the port microcode. Both bits are read only.

PFD (power fail disable) asserts to inhibit T ACLO and T DCLO from powering down the host system during maintenance testing. Thus diagnostics can check the T ACLO and T DCLO function without affecting the host system. PFD is set by writing the CNFGR register with BUF CMI D 08 asserted. PFD is cleared by writing the CNFGR register with BUF CMI D 08 negated.

A complete description of the T ACLO, T DCLO, and PFD functions is given in Paragraph 5.12.3 (Power Control Function) and in Figure 5-27.

6.4.4 NXM, UCE, CRD

NXM (non-existent memory), UCE (uncorrectable error), and CRD (corrected read data) indicate status of a port initiated CMI transfer. The CMI status bits are returned by the addressed nexus and applied to a status decoder in the CCI control logic (Figure 6-18). The decoder outputs SET NXM, SET UCE, or SET CRD if any of these transfer errors occurred. No output is asserted by the decoder for an error free transfer.

NXM indicates a "no response" by a nexus that was addressed by the port. NXM is set by the assertion of SET NXM from the status decoder. NXM can be cleared by writing the CNFGR register with BUF CMI D 20 asserted. NXM asserts SET MSE (memory system error) to the CS branching logic (Figure 4-10) via the CIPA bus (Figure 5-2).

UCE indicates an uncorrectable error is contained in the data returned by a nexus that was addressed by the port for a read operation. UCE is set by the assertion of SET UCE from the status decoder. UCE can be cleared by writing the CNFGR register with BUF CMI D 17 asserted. UCE asserts SET MSE to the CS branching logic (Figure 4-10) via the CIPA bus (Figure 5-2).

CRD indicates a correctable error occurred in the data returned by a nexus that was addressed by the port for a read operation. CRD is set by the assertion of SET CRD from the status decoder. CRD can be cleared by writing the CNFGR register with BUF CMI D 16 asserted.

6.4.5 DIAGNOSE

DIAGNOSE asserts to place the CCI into the diagnostic maintenance mode of operation. DIAGNOSE enables test logic for reading and writing the CCI registers so that diagnostic routines can check out the CCI hardware.

DIAGNOSE is set by writing the CNFGR register with BUF CMI D 14 asserted. DIAGNOSE is cleared by writing the CNFGR register with BUF CMI D 14 negated.

6.4.6 CTO

CTO (CIPA time-out) indicates an unsolicited read or write of the DP did not complete within 10 microseconds. CTO is set by the assertion of SET CTO from the CCI control logic.

During an unsolicited read of the DP, the port holds CMI DBBZ asserted on the CMI bus until the read operation is completed. If the read operation is not completed within 10 microseconds, the CCI control logic places a NXM status code on the CMI bus, releases the CMI bus, and asserts SET CTO to the CNFGR register.

During an unsolicited write of the DP, CTO indicates that the write data taken off the CMI was not written into the DP.

CTO can be cleared by writing the CNFGR register with BUF CMI D 13 asserted.

A complete description of the CTO function is given in Paragraph 6.3.3.1 (CIPA Transfer Request).

6.4.7 RLTO

RLTO (read lock time-out) indicates that more than 1024 bus cycles have occurred since a CMI nexus executed a read lock function without executing a write unlock function. RLTO is set by the assertion of ASSERT RLTO from the GO/DONE logic (Figure 6-14).

When a CMI nexus executes a read lock function, the CI750 GO/DONE logic is inhibited from asserting GO until the nexus executes a write unlock function. If a write unlock function has not occurred after 1024 bus cycles (163.8 microseconds), ASSERT RLTO is asserted to the CNFGR register.

RLTO can be cleared by writing the CNFGR register with BUF CMI D 19 asserted.

RLTO asserts SET MSE SYNC in the CS branching logic (Figure 4-10) via the CIPA bus (Figure 5-2).

A complete description of the read lock function is given in Paragraph 6.2.2.2 (Issue GO).

6.4.8 CBPE

CBPE (CIPA bus parity error) indicates a parity error has occurred during a data transfer over the CIPA bus in either direction (DP to CCI or CCI to DP). CBPE is set by the assertion of SYNC CE from the CCI parity logic. CBPE can be cleared by a write to the CNFGR register when BUF CMI D 31 is asserted.

A complete description of the CCI parity logic is given in Paragraph 5.7.4 (CIPA ERROR) and Figure 5-10.

6.5 PARITY GENERATION AND CHECKING

Parity generation and checking within the CCI is described in Chapter 5 along with the DP parity and checking function (Paragraph 5.7 and Figure 5-10).

6.6 INITIALIZE AND POWER CONTROL FUNCTIONS

The power control functions of power-up, power-down, ACLO, DCLO, and initialize are described in Chapter 5 along with the DP initialize and power control functions (Paragraphs 5.12.2 and 5.12.3, and Figures 5-25 through 5-30 inclusive).

APPENDIX A
CI750 MNEMONIC GLOSSARY

ACK	Acknowledge
ACLO	AC low
ADD	Address
ADDR	Address
ADR	Address
ALT	Alternate
ALU	Arithmetic logic unit
AR	ACK receive (state)
ARB	Arbitration
ARBC	Arbitration counter
ASRT	Assert
ATTN	Attention
AX	ACK transmit (state)
B	Bus
BG	Bus grant
BR	Branch
BR	Bus request
BTO	Boot timeout
BUF	Buffer
BUF	Buffered
C	Carry
CBPE	CIPA bus parity error
CCI	CMI to CIPA interface
CDEST	Complement destination
CDET	Carrier detect
CE	CIPA error
CHAR	Character
CHK	Check
CI	Computer interconnect (formerly ICCS and IPA)
CIPA	Computer interconnect port adapter
CLK	Clock
CLR	Clear
CMD	Command
CMD/ADDR	Command/address
CMDADDR	Command/address
CMI	CPU memory interconnect
CMP	Compare
CNFGR	Configuration register
CNODE	Complement node
CNT	Counter
CNTL	Control
CNTR	Counter
COMP	Complementary
CRC	Cyclic redundancy check
CRD	Corrected read data
CRY	Carry
CS	Control store

CSA	Control store address
CSPE	Control store parity error
CTO	CIPA time-out
D	Data
DBBZ	Data bus busy
DCLO	DC low
DEL	Delay
DET	Detector
DFE	Decoded file enable
DLY	Delay
DLYD	Delayed
DN	Done
DP	Data path (module)
DPUP	Decoded push/pop
DST	Destination
DST CMP	Destination compare
ECL	Emitter coupled logic
EN	Enable
ENA	Enable
ENB	Enable bit
ERR	Error
EXT	External
FCN	Function
FE	File enable
FLG	Flag
FPLA	Field programmable logic array
GEN	Generate
HDR	Header
HI	High
HTO	Header time-out
IB	Internal bus
IB DST	IB destination
IB SRC	IB source
ICCS	Intercomputer communications switch (see CI)
IMUX	Input mux
INH	Inhibit
INIT	Initialize
INT	Internal
INT	Interrupt
INTR	Interrupt
IPA	Interprocessor adapter (see CI)
IPE	Input parity error
JMPR	Jumper
JSR	Jump to subroutine
LD	Load
LO	Low

LS	Local store
LSA	Local store address
LSB	Least significant bit
LSPE	Local store parity error
LT	Less than
LTCHD	Latched
MADR	Maintenance address register
MAINT	Maintenance
MCLR	Maintenance clear
MD	Miscellaneous data
MDATR	Maintenance data register
MDECODER	Manchester decoder
ME	Manchester encoded
MIE	Maintenance interrupt enable
MIF	Maintenance interrupt flag
MIN	Maintenance initialize
MISC	Miscellaneous
MISC CNTL	Miscellaneous control
MLD	Maintenance load
MLOAD	Maintenance load
MLOOP	Maintenance loop
MR	Message receive (state)
MSB	Most significant bit
MSE	Memory system error
MSG	Message
MTD	Maintenance timer disable
MTE	Maintenance error
MX	Message transmit (state)
NACK	Negative acknowledge
NS	Next state
NXM	Non-existent memory
OP	Operation
OPE	Output parity error
OVFL	Overflow
PAL	Programmable array logic
PAR	Parity
PB	Packet buffer (module)
PBIR	PB in register
PC	Program counter
PDN	Power-down
PE	Parity error
PE	Phase encoded
PF VLD	Power fail valid
PFD	Power fail disable
PICR	Port initialize control register
PMCSR	Port maintenance control/status register
PMTCR	Port maintenance timer control register
PMUX	Packet buffer mux
PREV	Previous
PROM	Programmable read-only memory

PROP	Propagate
PSA	Programmable starting address
PSR	Port status register
PSRCR	Port status release control register
PUP	Power-up
PUP	Push/pop
PW	Pulse width
PWR	Power
QUAD	Quadword
RAM	Random access memory
RB	Receive buffer
RBPE	Receive buffer parity error
RBUF	Receive buffer
RCAR	Receiver carrier
RCV	Receive
RCVD	Received
RCVR	Receiver
RD	Read
RD	Read data
RDAT	Receive data
RDLCK	Read lock
REC	Receiver
REG	Register
RINIT	Receive (state) initialize
RLTO	Read lock time-out
RSEL	Register select
RSVD	Reserved
RTN	Return
RTS	Return from subroutine
RXMIT	Receive (state) transmit
S/D	Source/destination
SEL	Select
SEL CC	Select condition code
SEQ	Sequencer
SRC	Source
T	Time
T ACLO	Transmitted AC low
T DCLO	Transmitted DC low
TABORT	Transmit (state) abort
TACK	Transmit ACK
TBUF	Transmit buffer
TDATA	Transmit data
TDEST	True destination
TINIT	Transmit (state) initialize
TO	Time-out
TPATH	Transmit path
TR	Trailer
TTL	Transistor-transistor logic
TXMIT	Transmit (state) transmit

UBS	UNIBUS
UCE	Uncorrectable error
UCODE	Microcode
UNINIT	Uninitialized
UNS	Unsolicited
UNSOL	Unsolicited
UWORD	Microword
VALDAT	Valid data
VCDT	Virtual circuit descriptor table
VRD	Valid receive data
WACK	Wait for ACK
WD	Word
WD	Write data
WE	Write enable
WP	Wrong parity
WR	Write
WRT	Write
XBIR	External bus input register
XBOR	External bus output register
XBUF	Transmit buffer
XBUS	External bus
XFER	Transfer
XLATE	Translate
XMIT	Transmit
XTAL	Crystal

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.

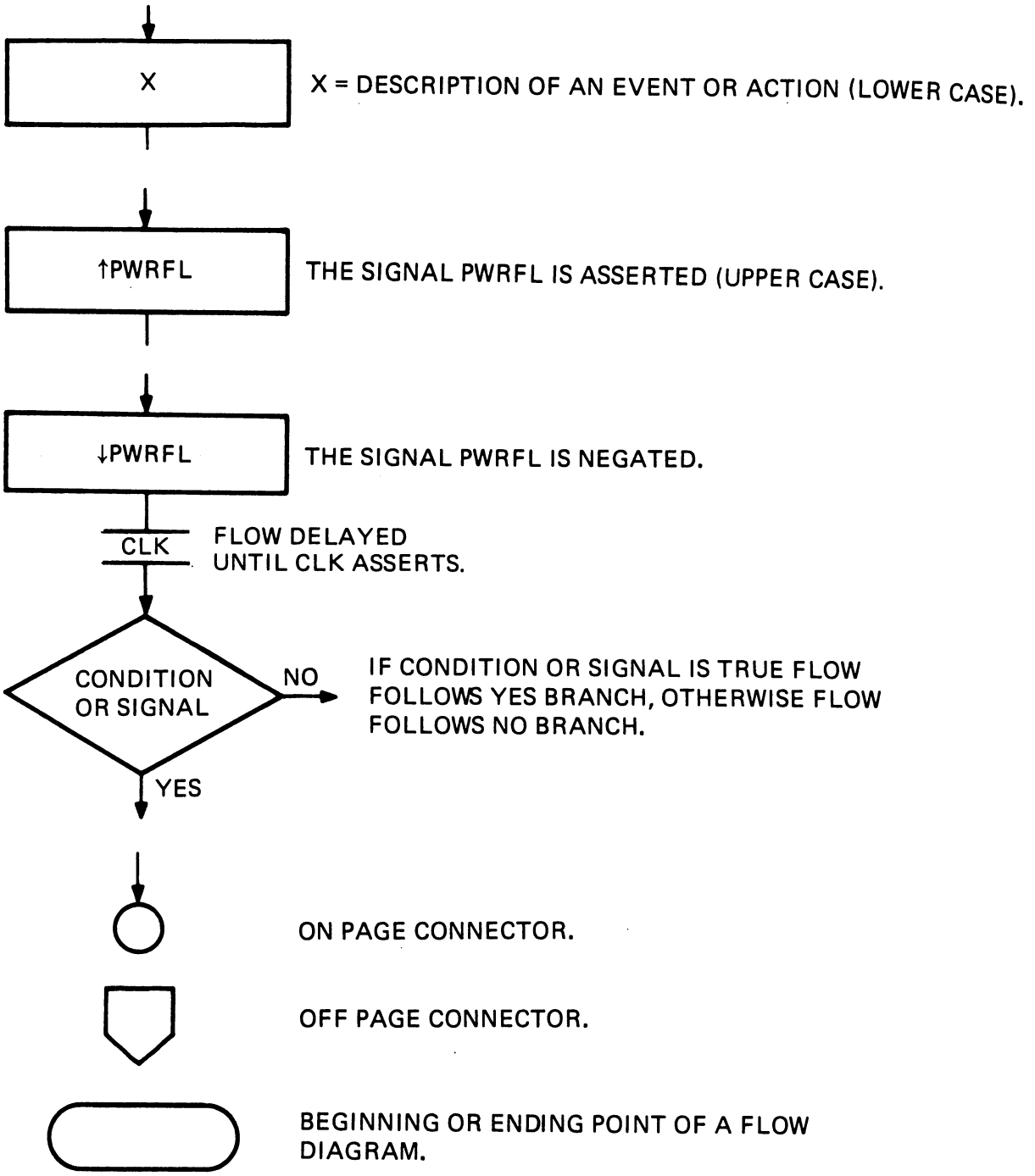


Figure B-1 Flow Diagram Symbols TK-6071

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 Data 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.

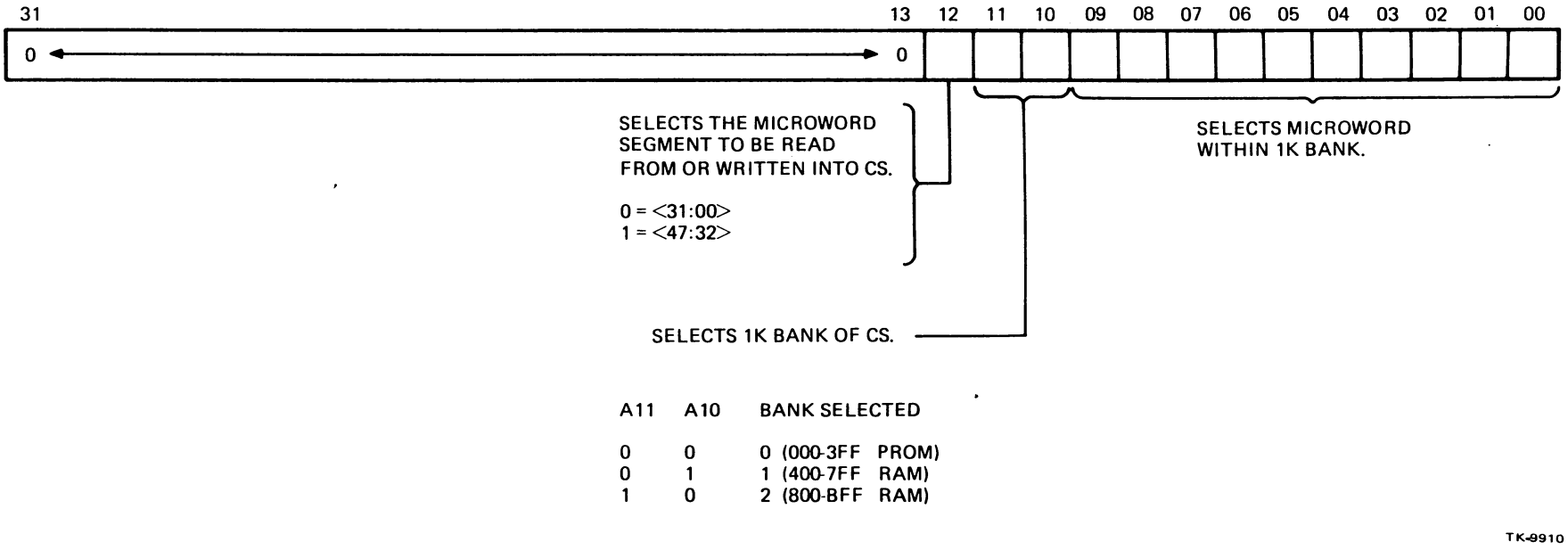


Figure C-1 Maintenance Address Register (MADR) Bit Fields

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.

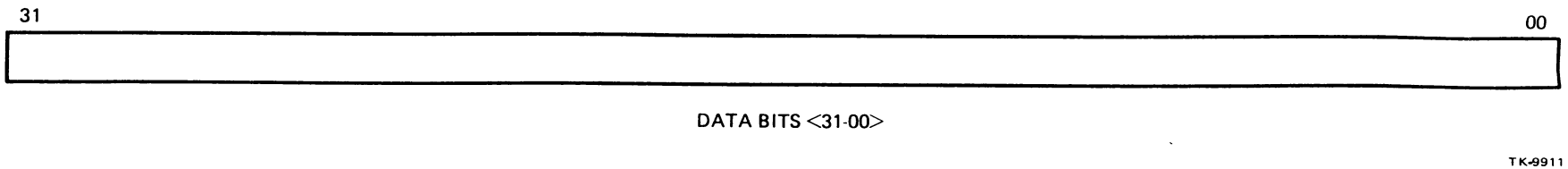


Figure C-2 Maintenance Data Register (MDATR)

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 5.5.3 and Table 5-4.

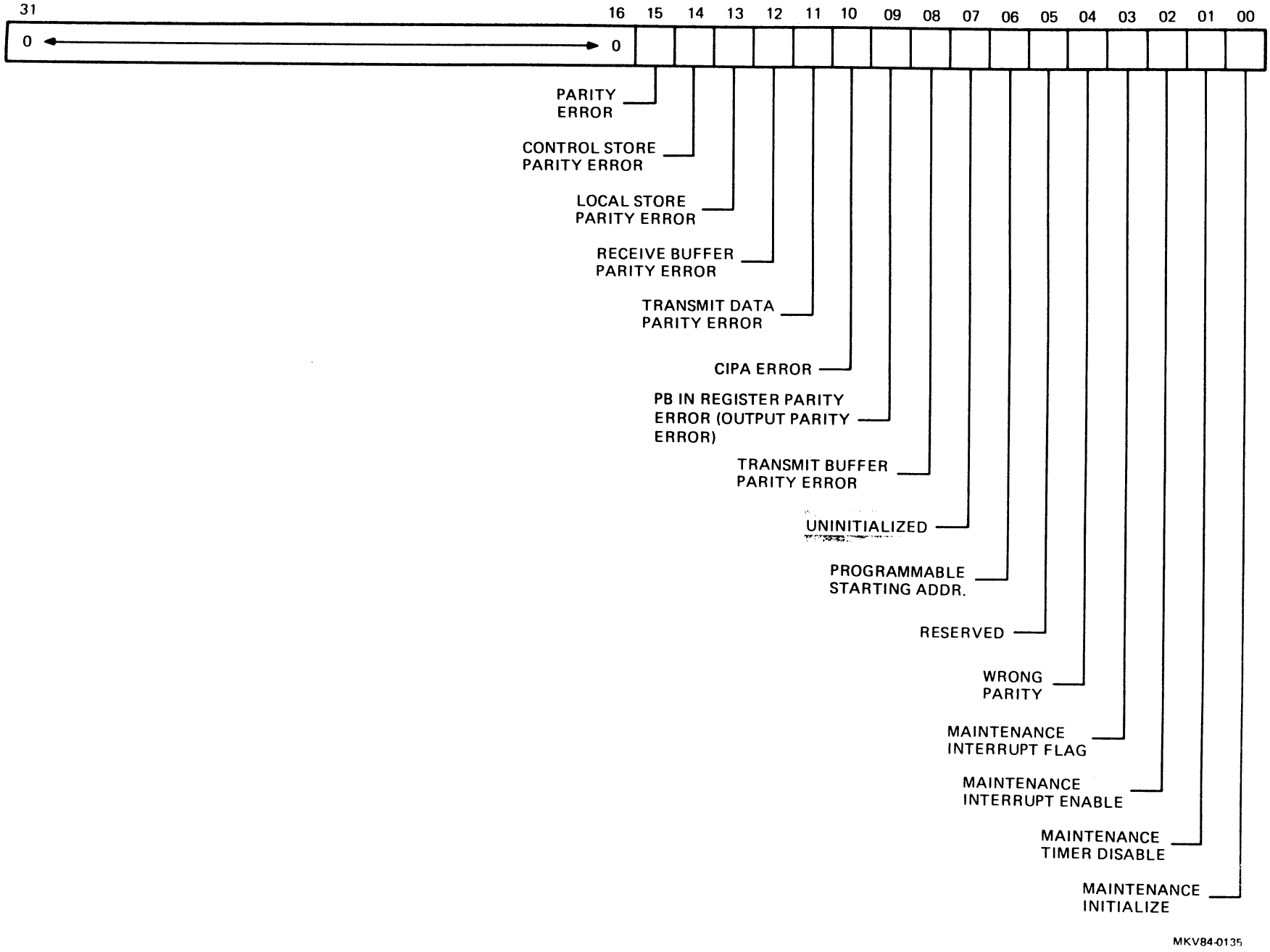


Figure C-3 Port Maintenance Control/Status Register (PMCSR) Bit Fields

C.4 CNFGR -- Configuration Register

Figure C-4 illustrates the function of the CNFGR bits.

The register address = XXXXX000

The CNFGR register contains CI750 status and control bits and the CI750 CMI adapter code.

Table C-1 describes the CNFGR bit functions. Refer to Paragraph 6.4 for a more detailed discussion of the CNFGR register.

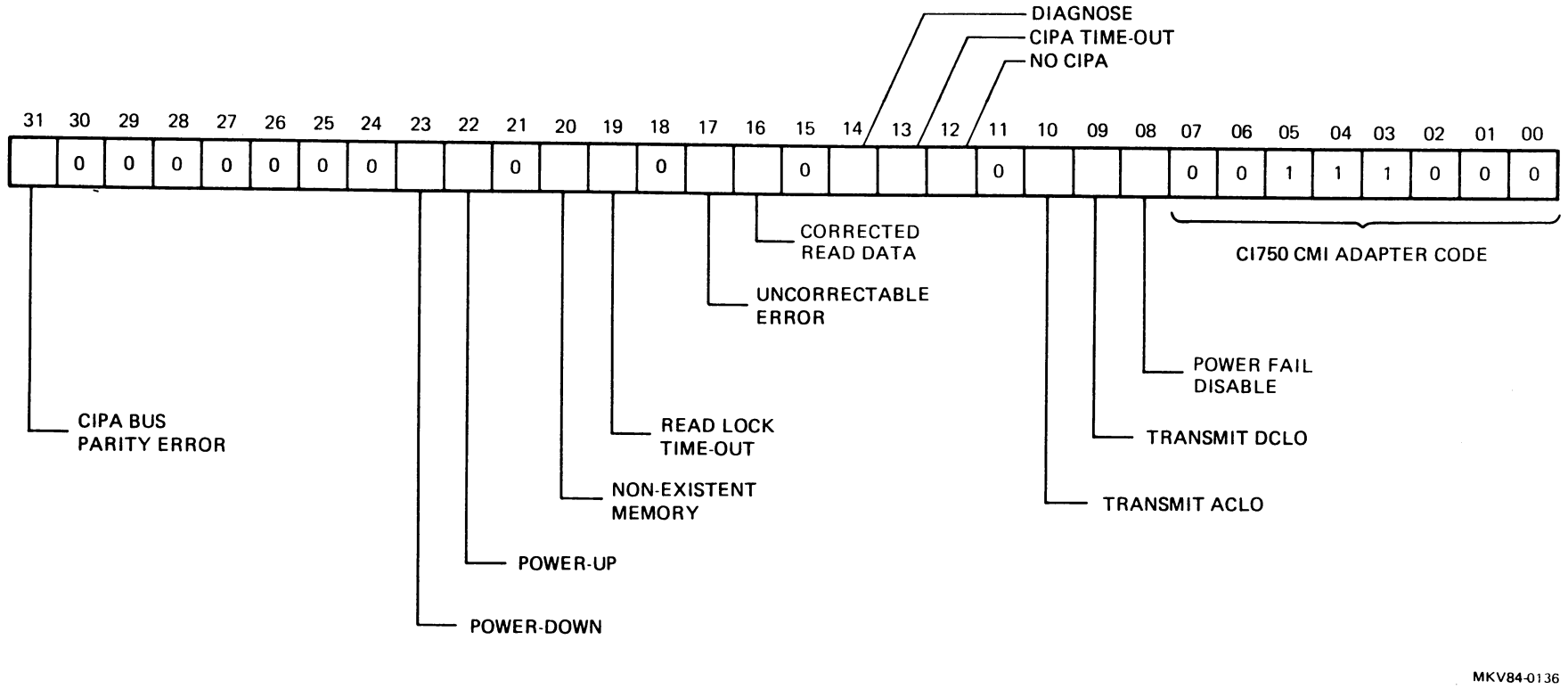


Figure C-4 Configuration Register (CNFGR) Bit Fields

Table C-1 CNFGR Bits

Bit	Mnemonic	Description
31	CBPE	CIPA Bus Parity Error: Set when a parity error is detected on a DP to CCI or CCI to DP data transfer.
30:24	0	Reserved. Read as 0s.
23	PDN	Power-Down: Set if the port is powering down. PDN is set by the assertion of ACLO in the CIPA cabinet or host CPU cabinet. The PDN bit is cleared by writing a 1 to it or by setting the PUP bit.
22	PUP	Power-Up: Set if the port is powered up. PUP is set when ACLO negates in both the CIPA cabinet and host CPU cabinet. The PUP bit is cleared by writing a 1 to it or by setting the PDN bit.
21	0	Reserved. Read as 0.
20	NXM	Non-Existent Memory: Set when the port initiates a CMI transfer and does not receive any response from the slave nexus. The NXM bit is cleared by writing a 1 to it.
19	RLTO	Read Lock Time-Out: Set when 1024 CMI bus cycles have occurred (163.8 microseconds) since a CMI nexus (other than the CI750) executed a read lock function without executing a write unlock function. The RLTO bit is cleared by writing a 1 to it.
18	0	Reserved. Read as 0.
17	UCE	Uncorrectable Error: Set when the CI750 receives a UCE status from a slave nexus during a read operation. UCE indicates an uncorrectable error is contained in the read data returned by the slave. The UCE bit is cleared by writing a 1 to it.

Table C-1 CNFGR Bits (Cont)

Bit	Mnemonic	Description
16	CRD	Corrected Read Data: Set when the CI750 receives a CRD status from a slave nexus during a read operation. CRD indicates a correctable error occurred in the read data returned by the slave. The CRD bit is cleared by writing a 1 to it.
15	0	Reserved. Read as 0.
14	DIAGNOSE	Diagnose: DIAGNOSE is a control bit that is set to place the CI750 into the diagnostic maintenance mode of operation.
13	CTO	CIPA Time-Out: Set when an unsolicited read or write of the DP did not complete within 10 microseconds.
12	NO CIPA	NO CIPA: Cleared if the CIPA cabinet is present, powered-up, and initialized. Otherwise this bit is set.
11	0	Reserved. Read as 0.
10	T ACLO	Transmitted ACLO: Set and cleared by the port microcode to effect a power-down and power-up of the host system while keeping the CI750 powered up.
09	T DCLO	Transmitted DCLO: Set and cleared by the port microcode. Used in conjunction with T ACLO to effect a host system power-down and power-up while keeping the CI750 powered up.
08	PFD	Power Fail Disable: PFD is a control bit that is set to inhibit T ACLO and T DCLO from powering down the host system during maintenance testing.
07:00	----	Adapter code: These bits contain the CI750 CMI adapter code.

Digital Equipment Corporation • Bedford, MA 01730