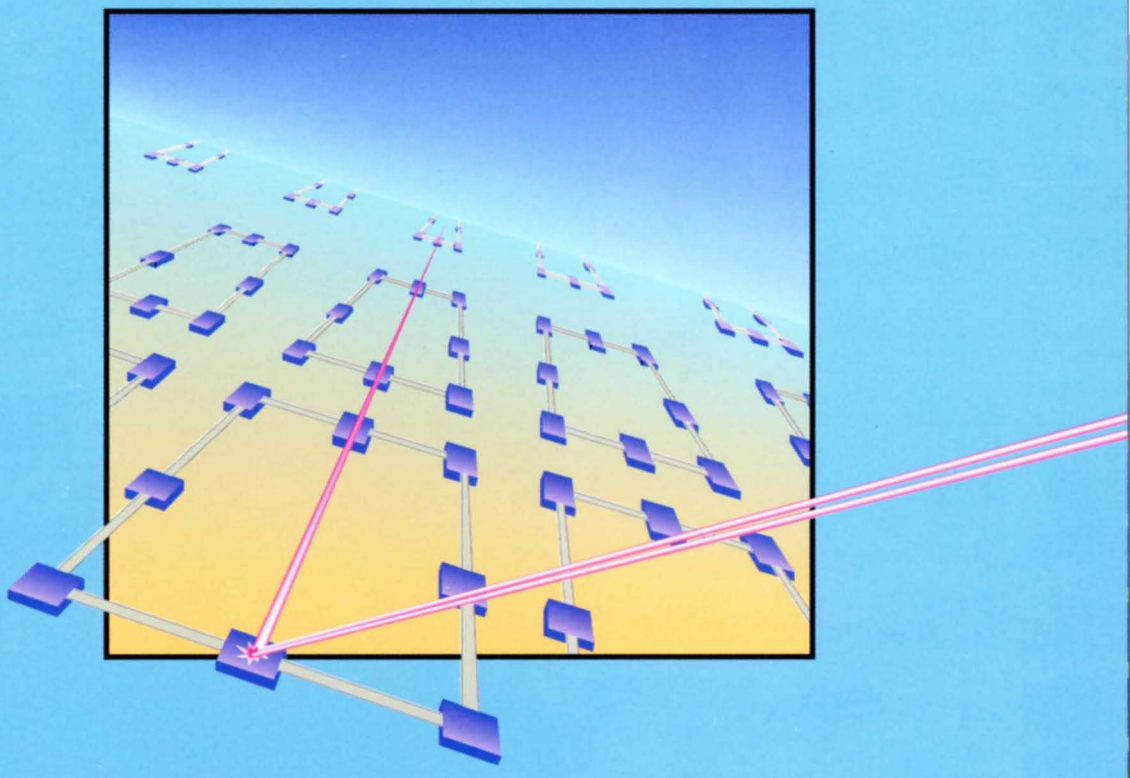
MC68840 UM/AD
REV. 1

FDDI



Fiber Distributed Data Interface
User's Manual

MC68840
IFDDI

MOTOROLA

MOTOROLA

MC68840 IFDDI

User's Manual

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Section 6
Port Operation

TABLE OF CONTENTS (Continued)

Section 7
Register Description

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Concluded)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

LIST OF ILLUSTRATIONS (Continued)

LIST OF ILLUSTRATIONS (Continued)

LIST OF ILLUSTRATIONS (Concluded)

LIST OF TABLES

LIST OF TABLES (Continued)

MC68840 ACRONYMS

A_FLAG—Indicates Destination Address Match
ALS—Active Line State
A_MAX —Maximum PHY Acquisition Time
ANSI—American National Standard Institute
ATE—Automatic Test Equipment
BADR—Burst Address
BIST—Built-In Self-Test
CAM—Content Addressable Memory
CDA—CAMEL Direct Access
C_FLAG—Indicates Frame Successfully Copied
C_MIN—Minimum Connect Time
CMOS—Complementary Metal-Oxide Semiconductor
CMT—Connection Management
CRC—Cyclic Redundancy Check
CT—Count, Actions for FSM
DA—Destination Address
DAS—Dual Attach Station
DMA—Direct Memory Access
EB—Elasticity Buffer
ED—Ending Delimiter
E_FLAG—Indicates Error Detected in Last Received Frame
ELM—Elasticity Buffer and Link Management Device
FC—Frame Control Field of FDDI Frame
FCG—FDDI Clock Generator
FCS—Frame Check Sequence
FDDI—Fiber Distributed Data Interface
FDX—Full Duplex
FIFO—First-In First-Out

LIST OF ACRONYMS (Continued)

FS—Frame Status
FSI—FDDI System Interface
FSM—Finite State Machine
H_FLAG—Indicates Higher Source Address Received
HLS—Halt Line State
IFDDI—Integrated Fiber Distributed Data Interface
ILS—Idle Line State
INFO—Information Field
IRI—Internal Control Register ID
IRT—Internal Control Register Type
JTAG—Joint Test Action Group
LAN—Local Area Network
LC_SHORT—Short Link Confidence Test
LCT—Link Confidence Test
LEM—Link Error Monitor
L_FLAG—Indicates Lower Source Address Received
LFSR—Linear Feedback Shift Register
LLC—Logical Link Control
LM—Link Management
LS—Line State
LSM—Line State Machine
LS_MAX—Maximum Line State Change Time
MAC—Media Access Control
MACIF—MAC Interface
M_FLAG—My Source Frame Received
MLA—My Long Address
MLS—Master Line State
MSA—My Short Address
MUX—Multiplexer
N_FLAG—Indicates No Copy Acknowledgment
NLS—Noise Line State
NOP—No Operation

LIST OF ACRONYMS (Continued)

NP—Node Processor
NPA—Node Processor Address Bus
NPD—Node Processor Data Bus
NPI—Node Processor Interface
NRZ—Non-Return to Zero Encoding
NRZI—Non-Return to Zero, Invert on One Encoding
NSA—Next Station Address
NS_MAX—Maximum Noise Time
PCI—Physical Connection Insertion
PCM—Physical Connection Management
PDU—Protocol Data Unit
PHY—Physical Layer of FDDI Standard
PMD—Physical Media Dependent
PNOP—Pipeline No Operation
PRH—Packet Request Header
PSF—Port Synchronization Function
QLS—Quiet Line State
RDA—Ring Data Assignment
REGWR—Register Read/Write Code
R_FLAG—Indicates Last Valid Token Was Restricted Token
RML—Ring Maximum Length
RPA—Ring Port Assignment
SA—Source Address
SAS—Single Attach Station
SD—Signal Detect
SMT—Station Management
TAP—Test Access Port
T_BID—Value This Station Bids During Claim Process
TB_MIN—Minimum Break Time
TCF—Indicates Transmit Logic Has Started Execution
T_FLAG—Indicates Last Token Captured Was Early
THT—Length of Transmit Time upon Receipt of Token Timer

LIST OF ACRONYMS (Concluded)

TL_MIN—Transmit Time Before Advancing to Next PCM State
T_MAX—Maximum TTRT
T_MIN—Minimum TTRT
TNE—Timer, Noise Events
T_NEG—Negotiated TTRT
T_OUT—Signaling TIme Out
TPC—Timer, Physical Connection
T_REQ—Requested TTRT
TRT—Token Rotation Time
T_SCRUB—Scrub Time
TTL—Transistor-Transistor Logic
TTRT—Target Token Rotation Time
TVX—Valid TransmissionTimer
ULS—Unknown Line State

SECTION 1
INTRODUCTION

Fiber distributed data interface (FDDI) is a 100-Mbps, fiber-optic-based, token-ring local area network (LAN) standard developed to accommodate rings of up to 1000 stations and a total ring length of 200 km. FDDI is an American National Standards Institute (ANSI) standard. This standard specifies the media access control (MAC) layer, the physical (PHY) layer, the physical medium dependent (PMD) function, and the station management (SMT) function.

1.1 OVERVIEW

The MC68840 integrated fiber distributed data interface (IFDDI) implements the MAC and PHY layers as defined in the ANSI X3T9 standard and also includes a non-standardized system interface to easily connect any host system to the FDDI network. Combined with an MC68836 FDDI clock generator (FCG), a single attach station is fully implemented according to the X3T9 standard (see Figure 1-1).

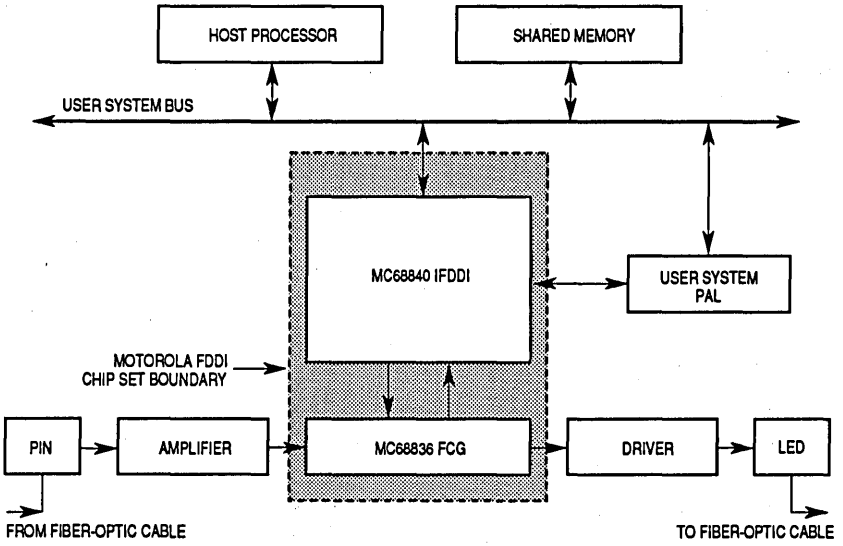


Figure 1-1. FDDI Chip Set

1.2 REVISION B ADDED FUNCTIONALITY

The following section describes new features provided with the IFDDI revision B that were not available on previous revisions of the IFDDI.

- Added Streaming Cipher capability for use in twisted pair systems as described in **Section 5.4 Twisted Pair Functional Operation.**
- Double Buffer Mode for streamlining the port to port DMA functions as described in **Section 5.1.9 Double Buffer Mode.**

1.3 FEATURES

The IFDDI consists of the FSI block, CAMEL (MAC and ELM) block, and a 64-entry CAM. A block diagram of the IFDDI is shown in Figure 1-2.

The IFDDI provides the following global features:

- On-Chip SMT Timer
  — Stopwatch Capability
  — Expiration Range from 40.9 µs to 2.68s at 12.5 MHz
  — Interrupt Set When Zero Value is reached
- Logically Separate, Programmable, 64-Entry, 48-Bit Internal CAM Handles Destination or Source, Individual or Multicast Addresses

The FDDI system interface (FSI) block provides the following features:

- Supports up to a 50-MHz Two-Clock Processor Access Cycle Time or 25-MHz System Bus Access Cycle Time (40-ns Cycle Time) in Normal Mode, with Higher Performance in Additional Modes
- Easily Connects to Any Bus Including Burst, High-Speed Processors, Little- and Big-Endian Buses, and Multiplexed/Nonmultiplexed Address/Data Buses
- Port Synchronization Function (PSF) Block Allows the FSI Block To Run at Clock Speeds other than SYMCLK while the CAMEL Block Remains at BYTCLK and SYMCLK Speeds (Split Clocks)
- Support of Pipeline Bus Access Allows the User More Control and Flexibility in Designing IFDDI/System Interface Control Logic
- Support of Burst Bus Access Allows the User To Connect the IFDDI to Buses such as the S-Bus
- Programmable Partitioned 8-Kbyte Internal RAM for Temporary Data Storage (Up to 1K of this RAM Is Used for Internal FSI Management)
- Four Fixed-Priority Transmit Rings and Two Receive Rings Support Synchronous/Asynchronous Frames
- Receive Filtering by Frame Control Field into Two Different Rings
- Ring Memory Structure with Multiple Buffers per Frame

- Data Path Parity Generation and Checking
- Two Identical 32-Bit Ports Allow Multiple Configurations Including 64-Bit Operations
- Port to Port DMA Capability

The CAMEL block provides the following features:

- Completely Implements ANSI FDDI PHY and MAC Standards
- 4B/5B Encoding and Decoding, Elasticity Buffer, and Smoother Functions
- Data Framing and Alignment to Byte Boundaries
- Performs Scrubbing and Provides for Nonconcatenation of Frames
- Hardware Assist for PCM State Machine Reduces Load on SMT Processing
- Line State Detector and Repeat Filter
- On-Chip LEM Detection and Counting
- Full-Duplex Operation
- Streaming Cipher Scrambling option; includes additional loopback features independant of scrambler activity.
- Independent Receive and Transmit Data Paths and State Machines Can Simultaneously Generate and Check CRC
- Supports Both One 16-Bit (My_Short_Address) and One 48-Bit (My_Long_Address) Individual Station Address on Chip
- Supports Several Bridging Facilities:
  — Can Reverse Bit Ordering on DA and SA
  — Supports Frame-Stripping Algorithm Based on Frame Counting and Void Frames
  — Allows Generating Frame CRC on a per-Frame Basis
  — Supports A and C Bit Handling for Transparent Bridging Mode
  — Supports Extended Address Recognition Timing for Address Recognition
- Supports Optional FDDI Standard Capabilities Such As:
  — Transmission and Reception of Additional Frame Status Indicators
  — Restricted Tokens
  — Synchronous Frames
- On-Chip Counters Support Station and Network Management Functions
  — Token Counter, Frame Counter, Lost Frame Counter, Error Frame Counter, and a Void Timer for Use in Latency Calculation
- Extensive Self-Test Capabilities and Data Generation and Checking

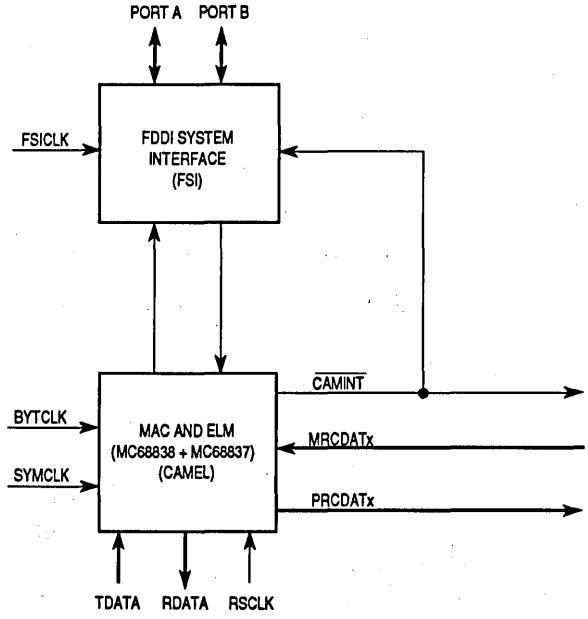


Figure 1-2. IFDDI Block Diagram

SECTION 2
BASIC OPERATION

To make a review of this chapter as productive as possible, it is recommended that the reader first become familiar with the FDDI protocol as outlined by the ANSI X3T9.5 standard.

This section is intended to give the reader the following information: guidelines for tailoring the IFDDI to many various system buses, detailed instructions on IFDDI initialization, an explanation of the IFDDI data structures and their handshaking mechanism, and the basics concerning transmission and reception of frames.

The subsections detail basic system configuration, data structures, software control of the IFDDI, and fundamental IFDDI timing.

2.1 SYSTEM CONFIGURATION

A basic system configuration is shown in Figure 2-1, and the following subsection gives a block by block description.

2.1.1 Block Description

A basic block description is discussed in the following paragraphs.

Host System

The host system can be any type of higher level system.

System Bus

The system bus communicates between the FDDI-specific protocol and the host system. Any system bus can be interfaced to the IFDDI. Examples of buses include VME, EISA, ISA, S-bus, and MacIntosh.

PALs

The PAL announces to the IFDDI via the ACNTL8–ACNTL0 lines what action the system bus wishes to take. It informs the system bus of what control information the IFDDI is requesting via the $\overline{\text{AREQ3}}$–$\overline{\text{AREQ0}}$ lines.

IFDDI

The IFDDI performs the interface between the system bus and the FDDI network. It also implements the ANSI MAC standard and the ANSI PHY standard. The IFDDI moves

data from the system bus to be transmitted to the FDDI network or moves data received from the FDDI network to the system bus.

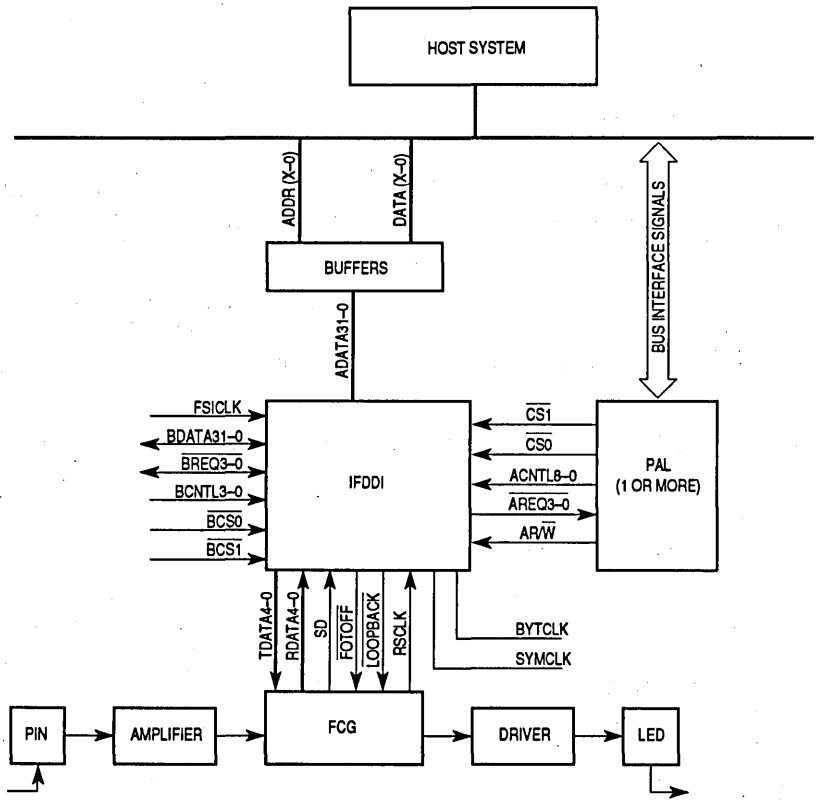


Figure 2-1. Basic System Configuration

Buffers

The buffers are used in the case when the IFDDI is unable to directly drive the system bus. The buffers may latch the address for the entire cycle. Please note that the IFDDI does not need the buffers in all cycles.

FCG

The FCG performs clock recovery and generation. It recovers the 125-MHz clock and performs serial to parallel conversion on the incoming serial bit stream from the FDDI receiver. The FCG performs parallel to serial conversion on the data to be transmitted from this station, which is sent in a serial bit stream to the FDDI transmitter.

2.1.2 Interface Pins

ADATAx, $\overline{\text{AREQx}}$, ACNTLx, ACSx, AR/$\overline{\text{W}}$, BYTCLK, SYMCLK, RSCLK, $\overline{\text{FOTOFF}}$, SD, $\overline{\text{LOOPBACK}}$, RDATAx, and TDATAx are critical interface pins. For more information, see **Section 8 Signal Description**. Note that port B is not used in this configuration.

2.2 DATA STRUCTURE

To facilitate communication between the FDDI media and the host, the IFDDI uses a hierarchy of data structures. The basic protocol unit of the FDDI network is the FDDI frame. When transmitting, the frame is written to system memory; when receiving, the frame is read from memory. A frame may be in a contiguous block of memory or may be segmented into different locations in system memory. Each section of memory used to hold either a frame segment or complete frame is referred to as a buffer. The IFDDI uses descriptors to point to each buffer in memory. For quick operation, these descriptors can be placed in a ring.

2.2.1 FDDI Frame Structure

The basic data structure in FDDI is the frame. The FDDI frame structure used by the network is shown in Figure 2-2.

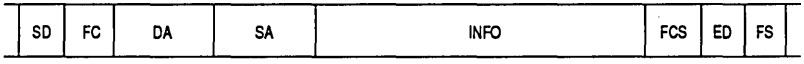


Figure 2-2. FDDI Frame Structure

The length of the various fields is as follows: starting delimiter (SD) field is 1 byte, frame control (FC) field is 1 byte, destination address (DA) field is 2 or 6 bytes, source address (SA) field is 2 or 6 bytes, INFO or data field is of variable length, and the frame check sequence (FCS) field, which contains the cyclic redundancy check (CRC), is 4 bytes in length. The end delimiter (ED) and frame status (FS) fields together are a minimum of two bytes. Interpacket idle symbols are not depicted. ANSI x3T9.5 defines the maximum FDDI frame size to be 4500 bytes.

2.2.2 Frame Structure within System Memory

The MAC block of the IFDDI automatically provides the SD, FCS, ED, and FS fields as well as the interpacket idle stream. Therefore, the user need not supply this information. The user must supply the FC, DA, SA, and INFO fields and also supply the packet request header through the host system.

Within system memory, the user data fields might look like those shown in Figure 2-3. Frame 1 shows an entire frame contained in one buffer. The PR1, PR2, PR3, FC, DA, and SA are all information supplied by the user. The INFO field is the data to be sent or received on the FDDI network. Frame 2 shows a frame segmented into three buffers. This technique is useful when the frame data is scattered through the system memory in different locations. This frees the implementor from being forced to use contiguous

memory blocks in a particular system implementation. The FDDI network consists of many of these frames being transmitted or received on the FDDI ring, with the maximum frame size being 4500 bytes.

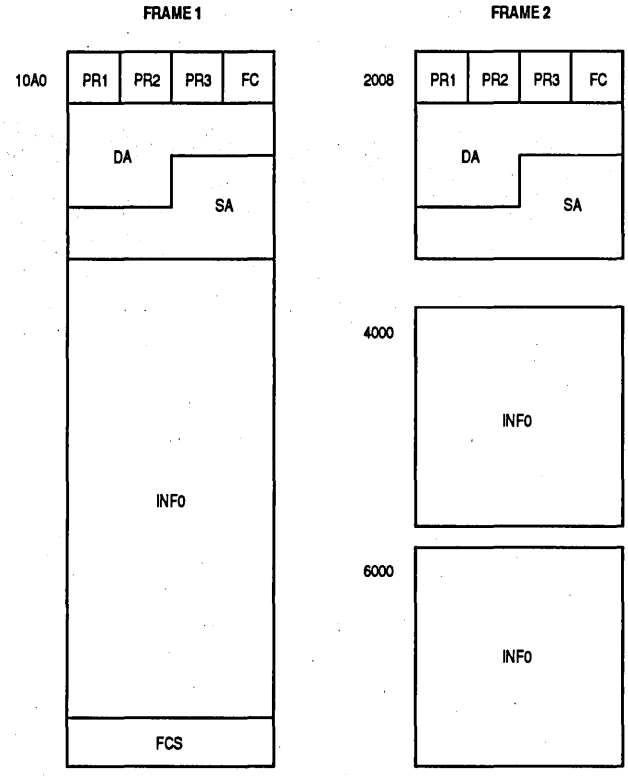


Figure 2-3. User Data Fields within System Memory

2.2.3 Descriptors

A single descriptor contains a pointer to where a buffer resides as well as the length of the buffer. It also contains control information indicating if this is the first, last, or only buffer in the frame. A descriptor points to a buffer, and one or more buffers contain an FDDI frame. Therefore, one or more descriptors describe an FDDI frame. A descriptor is eight bytes wide. The descriptors are defined on the IFDDI and are read into the IFDDI during transmit or receive operations.

2.2.4 Descriptor Rings

As previously mentioned, a frame is described by one or more descriptors. The descriptors are placed into a ring, which is a cyclical list structure. A descriptor ring is an easy way to organize many frames for quick and easy transmission to or reception from

the FDDI network. Rings are allowed to have sizes of as little as one descriptor and as many as 32K descriptors. The IFDDI supports four transmit and two receive rings.

2.2.5 Descriptor Ring Operation

After the ring is initialized with descriptors, the ring is defined by writing the DEFINE RING command to the command registers (CER/CMR) of the IFDDI. The higher layer software then tells the IFDDI that the ring is ready by writing to the ring ready register (RDY).

The handshaking mechanism by which the IFDDI knows that the descriptor ring is ready to be used is called the OWN bit, which is the most significant bit of the descriptor. When the system has finished preparing a descriptor, it then sets its OWN bit to one. After preparing a number of descriptors, the system indicates to the IFDDI that the ring is ready by writing to the ring ready (RDY) register. The IFDDI then starts using the descriptors for either transmission or reception. The IFDDI only reads in the descriptors that have their OWN bit set to one. When the IFDDI has completed operation on a descriptor, it generates an indication overwriting the descriptor it just processed. This indication has a reset OWN bit and contains status information about the descriptor it has overwritten. The reset OWN bit announces to the system that the descriptor is free to be redefined for new incoming or outgoing information.

In Figure 2-4, the host is able to write the next descriptor in the first indication with a reset OWN bit, which immediately follows the last descriptor the IFDDI owns. If the host continues to write new descriptors from this point and wraps back to where there are descriptors owned by the IFDDI, then the host will not be able to continue writing descriptors since the next descriptor is still owned by the IFDDI. The system has two ways of knowing that the first descriptor is no longer owned by the IFDDI—by polling this OWN bit or by waiting for the IFDDI to generate an interrupt. The interrupts may be set according to the RCC and RXC interrupts on status register 1.

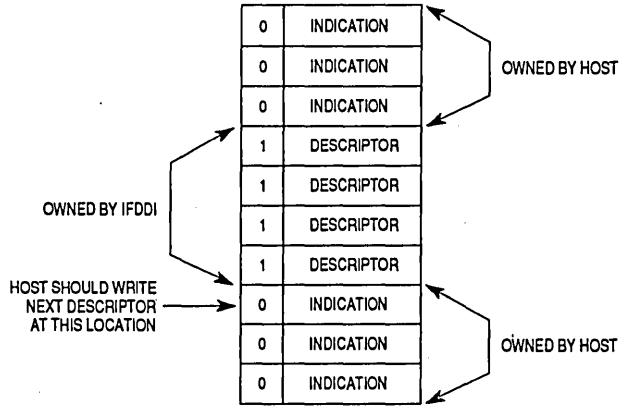


Figure 2-4. Descriptor Ring Operation

2.3 CONTROLLING THE IFDDI

2

Software control of the IFDDI is simplified through the IFDDI register set and through IFDDI commands. There are direct and indirect registers, which include configuration control registers, monitoring operations registers, and interrupt registers. Commands are issued either through the command register or through a ring. They are used by the IFDDI for ring-handling operations.

This subsection contains a brief discussion of the most critical registers and commands and can be used as a basic guide for how to use the IFDDI. There is a basic explanation for each of the following: initializing the FSI, using the ports, moving data, and defining rings. For details on commands, see **Section 9 Commands and Indications**, and for details on registers, see **Section 7 Register Description.**

2.3.1 Initializing the IFDDI

To configure the IFDDI for operation, the IFDDI configuration register (ICR) should be programmed for the appropriate functionality. In normal mode it should be 00H.

7	6	5	4	3	2	1	0
0	0	CTE	CDS	COM	CSM	CEO	DBM

IFDDI Configuration Register (ICR)

2.3.2 Using the Ports

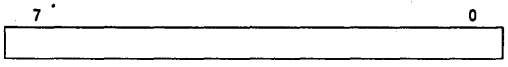
To set up the ports for operation, the port control register (PCR) should be programmed to enable port operation. The port memory page register (PMP) should also be programmed according to the page size of the memory that is connected to the IFDDI ports.

7	6	5	4	3	2	1	0
HPE	PC	SO	DO	0	DW	0	PE

Port Control Register (PCR)

To enable the port for operation, the PE bit should be set. The data width of the port (32 or 64 bits) is determined by the DW bit, and the data order (MSB first or last) of the port is determined by the DO bit. Synchronous or asynchronous port operation is determined by the SO bit.

The PMP register specifies the external memory page length size utilized on a port. Page sizes are encoded from 16 bytes (PMP = 00000000) to 16 Kbytes (PMP = 11111111).



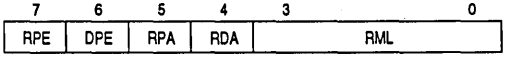
Port Memory Page Register (PMP)

2.3.3 Moving Data to/from the IFDDI

To perform data transfers to and from the host system, there are three relevant registers. First, the data register (DTR) is used to move data between the IFDDI port and the system bus. The address register (ADR) contains the address being used by the current port. The other port's address register (also ADR) represents the address being used by the other port. This is the most frequent operation that will be performed on the port side of any FDDI application.

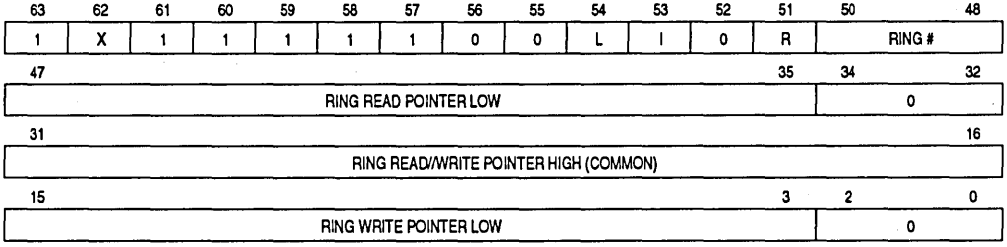
2.3.4 Defining a Ring

The first step in defining a ring is to set up the ring parameter register (RPR) for each ring that will be defined. The DEFINE RING command either defines a new receive or transmit ring, or it redefines an existing ring. To start the ring's operation, a write to the ring ready register (RDY) should be performed.



Ring Parameter Register (RPR)

To initialize which port the ring entries are accessed through, the RPA bit should be set appropriately. The RDA bit determines which port the data is transferred through. The maximum length of the ring is determined by the value of the RML field, with ring lengths available from 1 entry up to 32K entries.



DEFINE RING Command

The DEFINE RING command sets up the ring pointers, indicates whether or not this ring is associated with local memory, and indicates if the ring is ready upon definition or if ring execution should be held until a write is performed to the ring ready register (RDY).

2.3.5 Transmitting Frames

To initialize the system interface for proper transmission, the FIFO watermarks should be set appropriately for the bus latency of the host system by writing to the FIFO watermark register (FWR). Transmission is enabled by setting the transmit enable (TE) bit in the MACIF transmit control register (MTR). At the time of transmission, the frame can be transmitted either from a ring or by using the command registers (CMR/CER). To disable transmission for a specific ring, the TDx bit for that ring should be set.

7	6	5	4	3	2	1	0
TD3	TD2	TD1	TD0	TD7	TD6	0	TE

MACIF Transmit Control Register (MTR)

The FWR should be set according to the bus latency of the system for transmit operations so that an entire frame can be transmitted when this station has the token. For example, if the bus latency of the system bus is 50 µs, the number of latency bytes required is 50 µs * 12.5 Mbytes/s = 625 latency bytes. In other words, it takes 50 µs to get the system bus, and the throughput of the FDDI network is 12.5 Mbytes/s, so the IFDDI should store 625 bytes for a transmit operation to be continuous.

7	0

FIFO Watermark Register (FWR)

The TRANSMIT BUFFER DESCRIPTOR command can be issued either from a descriptor ring or directly from the CMR. This command includes pointers to the data buffer to be transmitted, and indicates whether this is the first, intermediate, or last buffer.

63	62	61	60	59	48
1	x	FIRST	LAST	0	

47	46	45	44	32
0	TOG	MODE	BUFFER LENGTH	

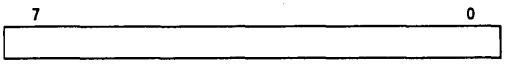
31	16
BUFFER ADDRESS HIGH	

15	0
BUFFER ADDRESS LOW	

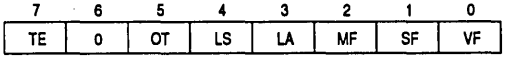
TRANSMIT BUFFER DESCRIPTOR Command

2.3.6 Receiving Frames

When receiving, the IFDDI fills empty buffer descriptors with incoming data until the watermark for reception is reached. The ring into which a frame is received depends on the sorting the user has requested based onn the frame's FC field. To prevent memory overflow, the user must determine the maximum receive space for reception operations. To initialize the system interface for proper reception, the FWR should be set. During reception, the FWR determines the amount of data that will be received before the system interface begins transferring the data to the system. The FWR should therefore be set up according to the amount of interaction with the system bus that the user desires the system interface to have. The incoming frames are sorted and directed according to their FC field, determined by the value in the receive frame type register (RFR). The maximum receive memory space register (RMR) determines the amount of internal memory that should be dedicated to reception operation. The next step is to set up the receive buffer descriptors for incoming frames. The final step is to enable reception operation using the MACIF receive control register (MRR).

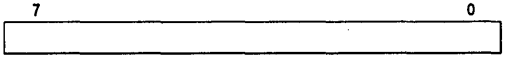


FIFO Watermark Register (FWR)



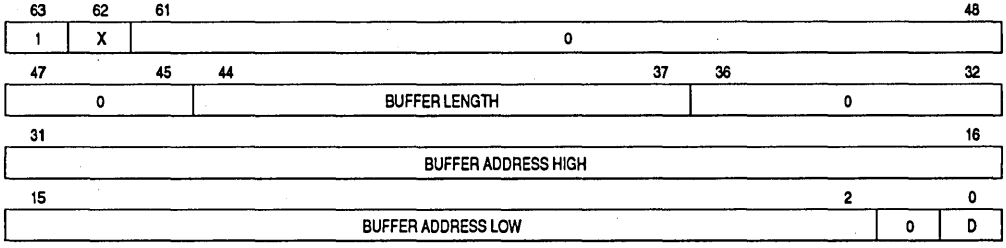
Receive Frame Type Register (RFR)

To receive station management (SMT) frames on a particular ring, the SF bit should be set. To receive logical link control (LLC) synchronous frames, the LS bit should be set; to receive LLC asynchronous frames, the LA bit should be set.



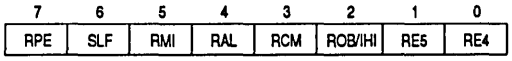
Maximum Receive Memory Space Register (RMR)

Included in the RMR value are both the receive internal ring FIFO and the receive internal data FIFO. The largest value allowed is 8K bytes (RMR = 255). The minimum value of RMR should be 10, which implies a receive buffer space of 352 bytes.



RECEIVE BUFFER DESCRIPTOR Command

The RECEIVE BUFFER DESCRIPTOR command contains the buffer length and pointers to the buffer address. The receive descriptors must be set up before reception operation is enabled.



MACIF Receive Control Register (MRR)

To enable reception, the RE5 or RE4 bit should be set. For normal mode and basic operation, all other bits should be reset.

2.3.7 Using Interrupt Capabilities

The IFDDI may generate interrupts for events such as ring events and error conditions, and for indicating command execution.

To enable interrupts on a given event, the bit corresponding to the interrupt event should be set in the interrupt mask register 1 (IMR1). When the corresponding bit in the status register 1 (SR1) is set, the IFDDI will generate an interrupt to the host system. For example, when bit 31 is set in IMR1, requesting that an interrupt be generated upon command execution, and bit 31 is set in SR1 by the IFDDI after command execution, an interrupt will be generated to the host system.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CDE	CFE	HEE	IEE	RVE5	RVE4	PEEB	PEEA	CIE	SIE	REE5	REE4	REE3	REE2	REE1	REE0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		RXE5	RXE4	RCCE3	RCCE2	RCCE1	RCCE0	0		RNE5	RNE4	RNE3	RNE2	RNE1	RNE0

Interrupt Mask Register 1 (IMR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CDN	CRF	HER	IOE	ROV5	ROV4	POEB	POEA	CIN	STE	RER5	RER4	RER3	RER2	RER1	RER0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		RXC5	RXC4	RCC3	RCC2	RCC1	RCC0	0		RNR5	RNR4	RNR3	RNR2	RNR1	RNR0

Status Register 1 (SR1)

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	0	0	0	CAMEL_INT	MAC_INT	ELM_INT

CAMEL Interrupt Location Register (CAMEL_INT_LOC)

The CAMEL interrupt location register is used to determine the source of the interrupt event that caused the assertion of the $\overline{\text{CAMINT}}$ pin. The CAMEL, MAC and ELM interrupt event registers are used to clear the core logic interrupts. The core logic interrupt signals are negated as a result of a node processor read operation to the interrupt event registers in the chip. This action also negates the $\overline{\text{CAMINT}}$ pin.

15	14	13	12	11	10	9	8
0	SD	LE_CTR	MINI_CTR	VSYM_CTR	PHYINV	EBUF_ERR	TNE_EXPIRED
7	6	5	4	3	2	1	0
TPC_EXPIRED	PCM_ENABLED	PCM_BREAK	SELF_TEST	TRACE_PROP	PCM_CODE	LS_MATCH	PARITY_ERR

ELM Interrupt Event Register (ELM_INT)

15	14	13	12	11	10	9	8
PH_INVALID	U_TOKEN_RCVD	R_TOKEN_RCVD	TKN_CAPTURE	BEACON_RCVD	CLAIM_RCVD	FRAME_ERR	FRAME_RCVD
7	6	5	4	3	2	1	0
DOUBLE_OVFL	RING_OP_CHNG	BAD_T_OPR	TVX_EXPIR	LATE_TKN	RCVRY_FAIL	DUPL_TKN	DUPL_ADDR

MAC Interrupt Event Register A (MAC_INT_A)

15	14	13	12	11	10	9	8
MY_BEACON	OTHER_BEACON	HIGHER_CLAIM	LOWER_CLAIM	MY_CLAIM	BAD_T_BID	PURGE_ERR	BRIDGE_STRIP_ERR
7	6	5	4	3	2	1	0
WON_CLAIM	0	SI_ERR	NOT_COPIED	FDX_CHANGE	BIT4_IS_S	BIT5_IS_S	BAD_CRC_SENT

MAC Interrupt Event Register B (MAC_INT_B)

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	0	0	RCDAT_PAR_ERR	VOID_TIME_REG_RDY	VOID_TIMER_OVF	TKN_CNT_OVF

MAC Interrupt Event Register C (MAC_INT_C)

2.3.8 Issuing Commands

Commands can be issued for a multitude of operations including ring initialization, single descriptor transmission, reading the current ring parameters, and local memory definition. Bits 31–0 are written to the command extension register (CER). Bits 63–32 are written to the CMR. Note that when issuing a command, the command will be executed as soon as the write to the CMR is complete; thus, the CER should be written first.

63	62	61	60	59	48
OWN	X	FIRST	LAST	X	
47					32
X					
31					16
X					
15					0
X					

Generic Descriptor Entry/Command

2.4 INITIALIZING THE PHY LEVEL (ELM BLOCK)

For fiber operation only, to initialize the ELM first configure ELM control register A (ELM_CNTRL_A). ELM_CNTRL_A performs various control operations, including timer

configuration and the ELM data path configuration. Next, configure ELM control register B (ELM_CNTRL_B). ELM_CNTRL_B controls whether the station is single or dual attach as well as how the ELM responds to different PCM events. Next, the maximum PHY acquisition time register (A_MAX), maximum line state change time register (LS_MAX), minimum break time register (TB_MIN), signaling time-out register (T_OUT), short link confidence test time register (LC_SHORT), scrub time register (T_SCRUB), and noise time register (NS_MAX) should be set (see Table 7-9). The ELM interrupt mask register (ELM_MASK) should be written to enable interrupts on the PCM_CODE, PCM_ENABLED, and PCM_BREAK states. The ELM interrupt event register (ELM_INT), violation symbol counter register (VIOL_SYM_CNT), and the link error counter register (LINK_ERR_CNT) should be read to reset them to zero. After completion of connection and link management operations, a PC_Start is written to the ELM_CNTRL_B register. For more information on the PCM process, see **11.1 ELM Initialization.**

15	14	13	12	11	10	9	8
0	NOISE_TIMER	TNE_16BIT	TPC_16BIT	REQ_SCRUB	ENA_PAR_CHK	VSYM_CTR_ INTRS	MINI_CTR_ INTRS
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCG_LOOP_ CTRL	FOT_OFF	EB_LOC_LOOP	LM_LOC_LOOP	SC_BYPASS	REM_LOOP	RF_DISABLE	RUN_BIST

ELM Control Register A (ELM_CNTRL_A)

15	14	11	10	8
CONFIG_CNTRL	MATCH_LS		MAINT_LS	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CLASS_S	PC_LOOP		PC_JOIN	LONG	PC_MAINT	PCM_CNTRL	

ELM Control Register B (ELM_CNTRL_B)

15	14	13	12	11	10	9	8
0	SD	LE_CTR	MINI_CTR	VSYM_CTR	PHYINV	EBUF_ERR	TNE_EXPIRED
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TPC_EXPIRED	PCM_ENABLED	PCM_BREAK	SELF_TEST	TRACE_PROP	PCM_CODE	LS_MATCH	PARITY_ERR

ELM Interrupt Event Register (ELM_INT)

When initializing the ELM core for twisted-pair operation, first initialize the ELM_CNTRL_A, ELM_CNTRL_B and CIPHER_CNTRL registers. The CIPHER_CNTRL register controls cipher enabling and disabling, loopback operation, signal detect operation, and the FOTOFF (quiet) timers. The following registers should be enabled for the desired quiet/scrambled quiet operation:

Maximum PHY acquisition time register (A_MAX),

Maximum line state change time register (LS_MAX),

Minimum break time register (TB_MIN),

Signaling time-out register (T_OUT),

Short link confidence test time register (LC_SHORT),

Scrub time register (T_SCRUB),

Noise time register (NS_MAX)

FOTOFF_ASSERT and FOTOFF_DEASSERT timers

See **Section 7.4 Twisted-Pair Operation Registers** and Table 7-12 for more information on twisted pair operation.

Finally, the ELM interrupt mask register (ELM_MASK) should be written to enable interrupts on the PCM_CODE, PCM_ENABLED, and PCM_BREAK states. The ELM interrupt event register (ELM_INT), violation symbol counter register (VIOL_SYM_CNT), and the link error counter register (LINK_ERR_CNT) should be read to reset them to zero. After completion of connection and link management operations, a PC_Start is written to the ELM_CNTRL_B register. For more information on the PCM process, see **11.1 ELM Initialization.**

15-14		13-9				8
PRO_TEST		Reserved				SDNRZEN
7	6	5	4	3-2	1	0
SDONEN	SDOFFEN	RXDATA_EN	FOTOFF_SRCE	FOTOFF_CNTRL	CIPHER_LPBCK	CIPHER_EN

Cipher Control Register

2.5 INITIALIZING THE MAC

A read of the MAC interrupt event A (MAC_INT_A), and MAC interrupt event B (MAC_INT_B) registers should be performed to reset these registers to zero. Next, the user should write to the my long address register A, B, and C (MLA_A, MLA_B, MLA_C) with the long address of this station. The my short address register (MSA) should be written with the short address of this station. The requested target request time register (T_REQ) and the TVX timer initial value register (TVX_VALUE) should be written. After initializing the timers, MAC operation is started by setting the MAC_ON bit in MAC control register A (MAC_CNTRL_A). The MAC interrupt mask register A (MAC_MASK_A) should be set to interrupt when the ring transitions out of the ring operational state. To check that the ring is operational, the MAC transmit status register (MTX_STATUS) should be read, checking bit 11 (RING_OP) if the RING_OP_CHANGE interrupt bit is set in the MAC interrupt event register B (MAC_INT_B).

15	14	13	12	11	10	9	8
MAC_ON	SET_BIT_5	SET_BIT_4	REVERSE_ADDR	FLUSH_SA47	COPY_ALL		COPY_OWN
7	6	5	4	3	2	1	0
COPY_EXTRA_SMT		COPY_IND_LLC	COPY_GRP_LLC	DSABL_BRDCST	RUN_BIST	RX_PARITY	NOTE_ALL_FRAMES

MAC Control Register A (MAC_CNTRL_A)

15	14	13	12	11	10	9	8
PH_INVALID	U_TOKEN_RCVD	R_TOKEN_RCVD	TKN_CAPTURE	BEACON_RCVD	CLAIM_RCVD	FRAME_ERR	FRAME_RCVD
7	6	5	4	3	2	1	0
DOUBLE_OVFL	RING_OP_CHNG	BAD_T_OPR	TVX_EXPIR	LATE_TKN	RCVRY_FAIL	DUPL_TKN	DUPL_ADDR

MAC Interrupt Event Register A (MAC_INT_A)

2.6 BASIC TIMING OF THE IFDDI PORTS

According to an internal algorithm, which involves FIFO states and watermarks, the IFDDI decides whether it wishes to be read or written by the external system bus. The IFDDI informs the system bus of what action it desires by setting the $\overline{\text{REQx}}$ lines accordingly. The system bus should access the DTR according to the request lines, but is not required to do so immediately. The IFDDI allows flexibility in when this operation is performed and even allows the system bus to access other registers without changing the request lines. If the IFDDI does not wish to be accessed, the IFDDI will set the $\overline{\text{REQx}}$ lines to the idle state.

Figure 2-5 shows an example of a basic bus cycle. The IFDDI first requests to be written by the host system. The host system reads the address to determine where to begin the read operation. The host system then issues two write cycles to the IFDDI, but the IFDDI stills requests a write cycle from the host. The host system instead chooses to write the CMR. This does not effect the state of the request lines. On the following data write, the IFDDI chooses not to be written and therefore puts the request lines into the idle state.

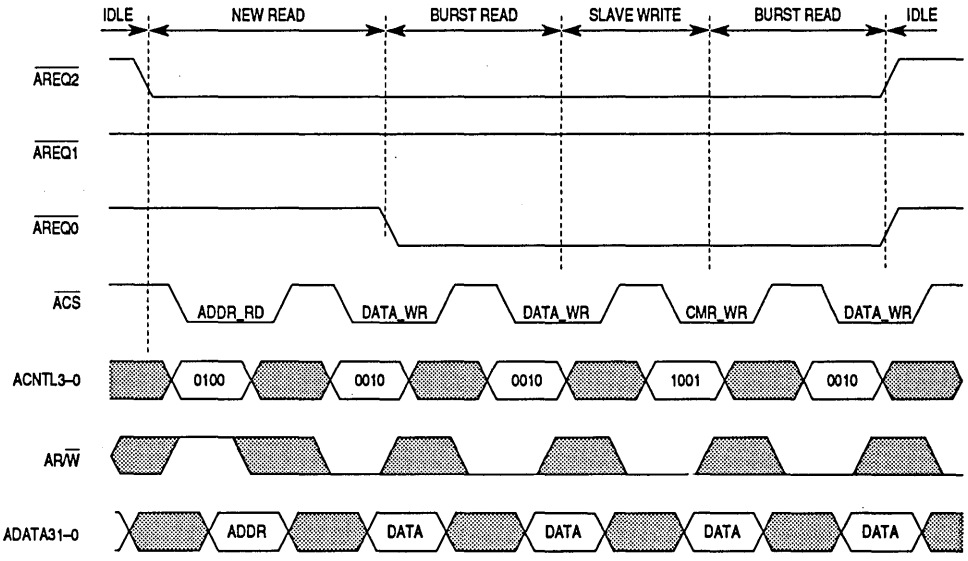


Figure 2-5. Basic Bus Cycle Example

SECTION 3
FEATURES SUMMARY

The IFDDI contains many features that are intended to enable simple solutions for many types of systems. The following paragraphs present a brief overview of these features.

3.1 TWO IDENTICAL PORTS ENABLE MULTIPLE CONFIGURATIONS

Having two identical ports enables multiple configurations as shown in Figures 3-1 to 3-5. Figure 3-1 describes two systems using the IFDDI as their media interface. Each system is unaware of the existence of the other system. Both systems are address/data multiplexed. System A can have a separate data order and page length from system B. The two systems do not share their transmit or receive rings, nor do they share their ring configurations.

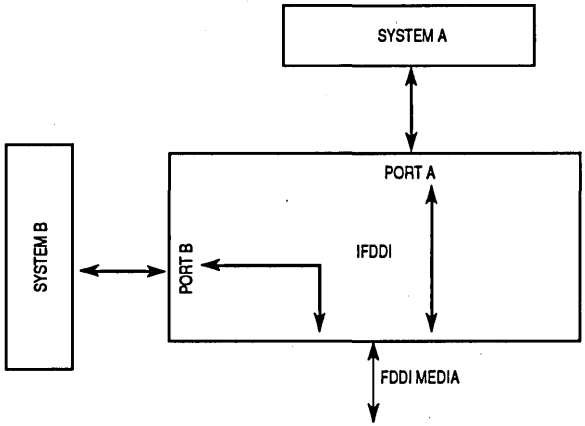


Figure 3-1. System Configuration Example 1

Figure 3-2 shows one system connected to the IFDDI. This system uses port A for addresses and port B for data. In this case, the IFDDI is operating in nonmultiplexed 32-bit mode.

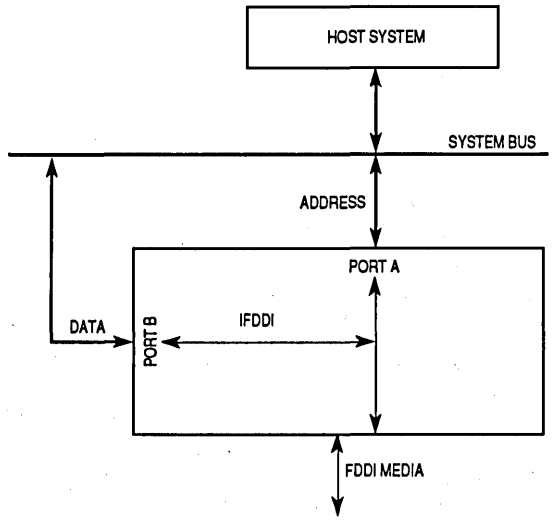


Figure 3-2. System Configuration Example 2

Figure 3-3 shows two systems using the IFDDI for both their media interface and intersystem communication. They perform this communication by sharing the direct memory access (DMA) channels (channels 6 and 7) provided by the IFDDI.

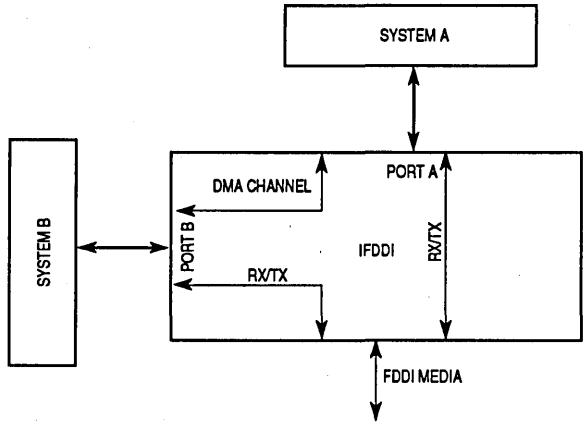


Figure 3-3. System Configuration Example 3

Figure 3-4 shows one system connected to the IFDDI. This system uses port A for the media interface, and port B is connected to a local memory device. Refer to **3.6 Extending Internal Memory Space (Local Memory)** for additional information.

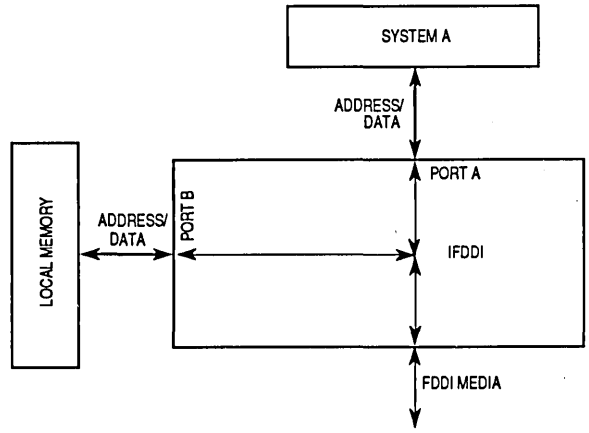


Figure 3-4. System Configuration Example 4

Figure 3-5 shows two systems connected to the IFDDI. The system connected to port B is a second processor that determines which frames are to be forwarded to the system via the IFDDI and which frames are to be discarded. This processor performs filtering on the incoming frames, forwarding only the system-relevant frames through the IFDDI DMA channels. The second processor is also able to process information that concerns only certain levels of the open systems interconnection (OSI) without forwarding them to the system.

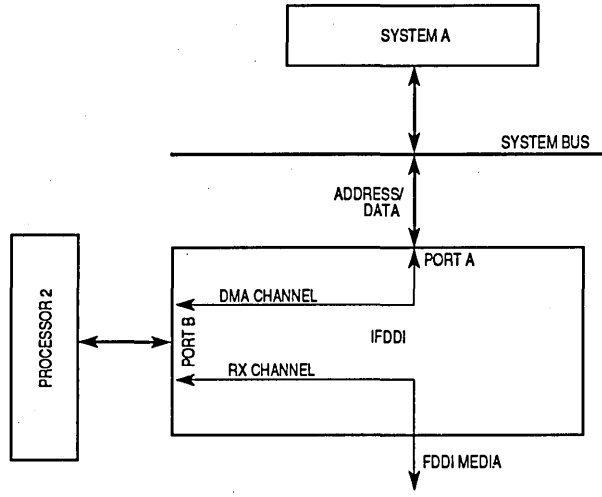


Figure 3-5. System Configuration Example 5

3.2 FOUR TRANSMIT, TWO RECEIVE, AND TWO COMMAND CHANNELS

The transmit channels can be dedicated to station management (SMT) asynchronous and synchronous frames as well as logical link control (LLC) asynchronous and synchronous frames. This configuration allows two processors to work simultaneously, with one processor dedicated to SMT traffic and one processor dedicated to LLC traffic. The separation between synchronous and asynchronous frames is useful due to the fact that synchronous frames have a higher priority in transmission than asynchronous frames. Note that the user chooses the desired configuration for these rings, and can even use these transmit rings for DMA operations between the ports.

The receive channels can also be dedicated to SMT traffic and LLC traffic, allowing two processors to work simultaneously. One processor may handle SMT traffic while the other processor handles LLC traffic. Incoming frames are filtered according to the configuration set up in their frame control (FC) field and the receive frame type register (RFR). The user is not forced to separate between SMT and LLC frames, but can use any configuration desired. For more information on the RFR, refer to **Section 7 Register Description.**

There are also two command channels provided. The only difference between a command channel and a transmit channel is that a transmit command using the command channel contains only one buffer. The same descriptors apply for both transmit and command channels.

3.3 DYNAMICALLY PARTITIONED 8-KBYTE INTERNAL RAM

Having a large internal memory allows flexibility in dealing with various bus latencies. For example, to support a 50-µs bus latency, there should be 50 µs x 12.5 Mbyte/s = 625 latency bytes. Even with this latency, the four transmit rings are able to work concurrently with remaining internal memory to perform reception operations. The dynamic allocation of memory allows the FIFOs to grow or shrink according to the memory that is needed for a specific channel at any given time, preventing any waste of internal memory.

3.4 FLEXIBLE BUS INTERFACE INCLUDING BURST BUS CYCLES

The IFDDI can easily be connected to almost any bus including high-speed processors, little- and big-endian buses, and multiplexed/nonmultiplexed data buses. The IFDDI can detect parity and parity errors and can support data widths of either 32 bits or 64 bits. For more information, see the port control register (PCR) description in **Section 7 Register Description.**

The IFDDI interfaces to external logic through its request lines. The request lines inform the external logic of what action the IFDDI wishes the external logic to perform. The external logic can respond to the IFDDI request lines by accessing the IFDDI through the control lines. It may do so immediately or may postpone the access. Only data register (DTR) accesses affect the request lines; thus, the external logic is free to access status registers, command registers, etc. The request lines indicate whether the access is on a page boundary by asserting $\overline{\text{AREQ0}}$ so that external logic is aware that may need to read

the address register. Moreover, the IFDDI can inform the external logic about what type of information is being accessed using $\overline{\text{AREQ3}}$.

Special attention was given to burst buses due to very tight timing requirements. In burst bus applications, the IFDDI should be able to output new data on each clock cycle, but not necessarily at all times. There are burst buses on which the next data should be output only if the previous data was acknowledged. For example, Figure 3-6 illustrates an S-bus interface that has a 40-ns clock cycle. This figure illustrates that only once the current data is acknowledged can the IFDDI start to output the next data. The next data must be valid 15 ns before the rising edge of the clock. This timing constraint does not give the IFDDI time to read the next register unless the pipeline mode is being used. In this mode, the IFDDI holds the current data in an internal output register while the next data is being read. Whenever the current data is acknowledged, the IFDDI is able to output the next data from the internal output register.

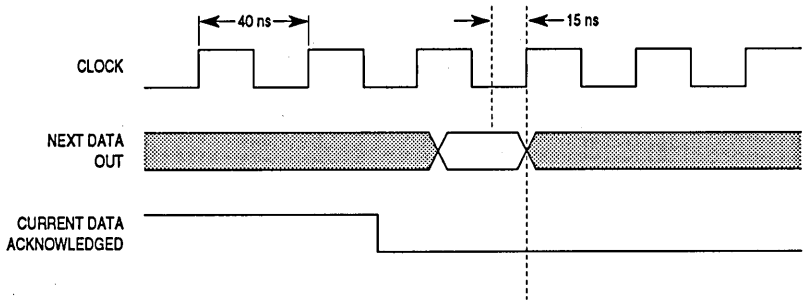


Figure 3-6. Simple Burst Bus Access

Another feature allowing the IFDDI to have easy access to burst buses is the burst limit counter. This counter counts the burst length, and when it expires, $\overline{\text{AREQ3}}$ is asserted, informing the external logic that it may end the current burst. The external logic is not required to use this hook. A second feature allowing easy IFDDI access to burst buses is the valid bit $\overline{\text{AREQ0}}$. The IFDDI asserts $\overline{\text{AREQ0}}$ to inform the external logic that the IFDDI does not wish to be accessed anymore. This hook can be used by buses that can terminate the burst in the middle of the cycle. For more information, refer to **Section 6 Port Operation.**

3.5 MULTIPLE REGISTER ACCESS METHODS

There are three groups of registers used by the IFDDI: FSI direct registers, CAMEL registers, and FSI indirect registers. There are three methods to access these registers: directly through the IFDDI control lines, indirectly through the IFDDI FSI control register (FCR), and indirectly through the IFDDI command register (CMR). To access the FSI direct registers and the CAMEL registers, one may use the direct access method. To access the FSI indirect registers and the CAMEL registers, one may use the indirect access through either the FCR or the CMR.

The multiple access methods provide flexibility in writing and reading to the registers. For example, the FCR write command (see **9.2.4 General Commands and Indications**) can place register accesses in transmit rings and then enable the ring. The IFDDI will then run through all the commands in that ring, reinitializing itself.

3.6 EXTENDING INTERNAL MEMORY SPACE (LOCAL MEMORY)

A low-performance system that is unable to keep up with the transmission speed of the FDDI ring (12.5 Mbps) may require local memory in addition to the internal 8 Kbyte of memory provided in order to perform a continuous transmission. The data is held in the local memory until the station gets the token, and the data is then transmitted from the local memory, allowing efficient use of the time the station holds the token. The local memory, therefore, actually smoothes the bursts on the FDDI ring. When recieving, the incoming data is held in the local memory until the system bus allows the IFDDI to transfer the data from local memory to the system memory.

To use local memory, a ring is defined with a parameter known as local memory mode (see **9.2.1.1 DEFINE RING Command** and **9.2.1.3 SET LOCAL MEMORY START ADDRESS Command**). Every transmit command on that ring has two stages: A DMA transfer from system memory to local memory and a transfer from local memory to the FDDI media (see Figure 3-7). Also refer to **Section 5 Functional Operation.**

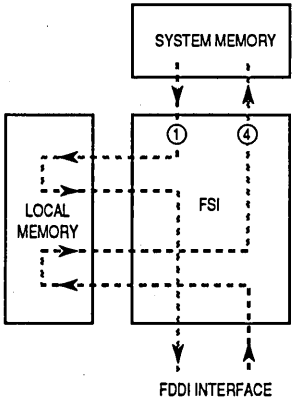


Figure 3-7. Local Memory Function

3.7 EASY-TO-USE DMA CHANNELS

Every transmit ring may contain commands to DMA data from system A to system B (see Figure 3-8). The command contains a pointer to the data on system A, and the length of the buffer to be transferred through DMA. System B contains destination descriptors that contain a pointer to the destination buffer and the length of the buffer. For more information, refer to **9.2.1.2 SET DESTINATION RING Command.**

This functionality can be easily used for both FDDI and non-FDDI applications. Both system A and system B can be residing on different or identical ports; thus, this feature can save the designer from being forced to use a DMA server in his system. A DMA channel that has its source and destination rings on the same port is useful for a system-level loopback test. For more information, refer to **Section 5 Functional Operation.**

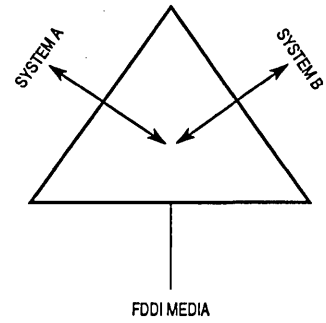


Figure 3-8. Simple DMA Functionality

3.8 MULTIPLE COMMANDS

The IFDDI uses commands and indications for its operation. According to user preference, these can be placed either in the command register or in a ring.

3.8.1 Commands To Set Up or Define Rings

To transmit or receive data in the normal mode, only one ring is needed. DMA channels require two rings—one source ring and one destination/target ring. Transmitting or receiving through local memory requires one ring defined in the system and a special zone defined in the local memory.

Defining system rings for any of the above operations is performed using the DEFINE RING command. This command includes the ring number (0–7) and pointers to where the ring resides in memory. There is a bit in this command that indicates whether local memory is being used with this ring. For more information, see **9.2.1.1 DEFINE RING Command.**

Setting up destination rings is performed with the SET DESTINATION RING command. This command includes the ring number (0–7) and pointers to where the ring resides in memory.

Setting up local memory is performed with the SET LOCAL MEMORY START ADDRESS command. This command includes the ring number (0–7) and pointers to the local memory start address.

3.8.2 Commands To Perform a Transmit Operation

To perform a transmission, the frame is fragmented into buffers, and each buffer has a descriptor dedicated to it. The FIRST and LAST bits in the descriptor indicate which are first, last, and intermediate buffers in this frame. The length of the buffer and pointers to the buffer address are also included in the descriptor.

There is also a toggle bit in the descriptor that determines which port the data will be read from. This allows headers and other control information to be read from a different port than the port the actual data is read from. The frame is then constructed from the separate ports inside the IFDDI. Figure 3-9 illustrates a system in which each frame's payload needs to be inserted between a header and a trailer. The header and trailer are formed by a processor residing on port B. The actual descriptor ring resides on port A. The IFDDI is able to combine the separate buffers into one continuous frame.

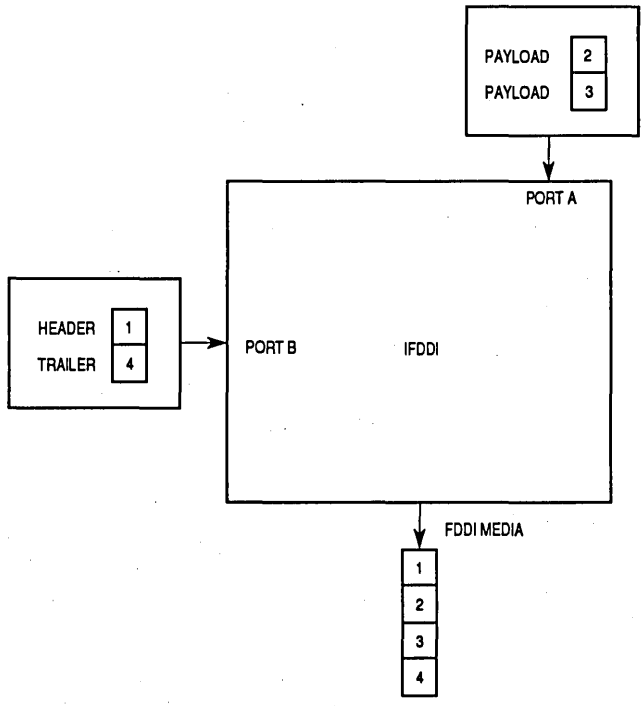


Figure 3-9. Continuous Frame from Fragmented Buffers

Also included in the transmit descriptor command is the ability to transmit a single frame at a time. This option is necessary for extremely low-performance systems in which the entire 4.5-Kbyte frame must reside within the IFDDI memory before the station receives the token. For more information, see **9.2.2.1 TRANSMIT BUFFER DESCRIPTOR Command.**

3.8.3 Commands To Receive

To receive data, the host system should prepare empty buffers. Each buffer is pointed to by a descriptor that also contains the length of the buffer. The receive descriptors must be set up before reception operation is enabled. For more information, refer to **9.2.2.10 RECEIVE BUFFER DESCRIPTOR Command.**

The indication returned after a buffer is received includes information such as whether a CRC error was found, the number of valid flags, the EAC bit values, the frame length, whether an overrun error occurred, whether there was a parity error, etc. For more information, refer to **9.2.2.11 RECEIVE FRAME NORMAL Indication, 9.2.2.12 RECEIVE ERROR Indication, 9.2.2.13 RECEIVE PORT ERROR Indication,** and **9.2.2.14 SPLIT MODE ERROR Indication.**

3.8.4 Commands To Perform DMA Operations

To perform DMA operations, two descriptors must be set up: a source of the DMA operation and a destination for the DMA operation. Both the DMA buffer descriptor and the destination ring buffer descriptor contain the pointer address and the buffer length. For more information, see **9.2.2.3 DMA BUFFER DESCRIPTOR Command, 9.2.2.4 DMA Indication (Source Side), 9.2.2.5 Destination Buffer Descriptor, 9.2.2.6 DMA Indication Without Error (Destination Side), 9.2.2.7 DMA ERROR Indication (Destination Side).**

A very short DMA of 24 bits can be achieved using the MAKE INDICATION command, in which the command itself contains the data. The destination descriptor will contain the same data as in the command. For more information refer to **9.2.2.8 MAKE INDICATION Command.**

3.8.5 Commands To Stop/Reset the Ring

To allow complete control of the IFDDI operation, two commands are provided: STOP RING command and RING RESET command. The STOP RING command allows the system to halt a specific ring operation. To resume the ring operation, a write to the ready register of the stopped ring must be performed through the FCR. For more information, see **9.2.1.5 STOP RING Command** and **7.2.2.1 Ring Ready Register.**

The RING RESET command allows the system to force a specific ring to the NOT DEFINED state. To redefine the ring, the DEFINE RING command should be used. For more information, see **9.2.1.5 RING RESET Command** and **9.2.1.1 DEFINE RING Command.**

3.8.6 Commands to Monitor the Network

The host can get receive indications on the following events:

1. A token has arrived—TOKEN CYCLE END indication (see **9.2.2.16 TOKEN CYCLE END Indication**).

2. The station has received its own frame—RECEIVE MY FRAME indication (see **9.2.2.15 RECEIVE MY FRAME Indication**).

To receive the TOKEN CYCLE END indication, the TE bit must be set in the receive ready frame type register (RFR). To receive the RECEIVE MY FRAME indication, the RMI bit must be set in the MACIF receive control register (MRR).

3.8.7 Commands To Control the CAM

The IFDDI contains a 64-entry internal CAM. An entry can be set up in the CAM using the SET UP CAM ENTRY command (see **9.2.3.2 SET UP CAM ENTRY Command**). A CAM entry can be read using the READ CAM ENTRY command (see **9.2.3.3 READ CAM ENTRY Command**). CAM entries can be compared by using the COMPARE CAM ENTRY command (see **9.2.3.6 COMPARE CAM ENTRY Command**). Also refer to **Appendix C CAM Operation in Non-FDDI Applications** for information on use of the CAM in non-FDDI applications.

3.8.8 Commands To Synchronize Between Rings

The IFDDI has the ability to prevent two separate ring processes from simultaneously using the same resource by using a common semphore. Each process should protect it's section(s) of critical code by prefixing these sections with the get-resource command and suffixing these sections with the release-resource command. After the IFDDI encounters the get-resource command, it does not proceed to the next command until the semaphore is reset. Before proceding to the next command, the IFDDI sets the semaphore, preventing any other process from accessing the initial process' critical code. The release-resource command resets the semaphore.

Figure 3-10 is an example of how to use the resource mechanism to transmit frame streams. The idea is to contiguously transmit all frames that belong to a specific data stream (or ring) without the frames being interleaved with a different frame stream coming from a separate ring. Note that in this figure, the transmit descriptors for the frame stream are prefixed by a get-resource command and suffixed by a release-resource command. From the perspective of the media, it is clear that all the frames belonging to Ring 0 are transmitted before those belonging to Ring 1.

3.8.9 Commands To Access Internal FSI memory

In most cases, only the IFDDI handles its internal memory; however, in rare cases, the user may need to access the internal memory. In these cases, the user may read or write this internal memory by using the MOVE command. This command moves 8 bytes of data between an external absolute memory address and an internal absolute memory address. The command contains pointers to both the external and internal memory as well as the direction of the transfer. For more information, see **9.2.4.4 MOVE Command**.

Note that writing to internal memory may interfere with the IFDDI work and, in the worst case, may overwrite important information. To avoid this situation, the GET BLOCK command should be used, which causes the IFDDI to extract one 32-byte block of internal memory from its queue of free blocks. The resulting indication of the GET BLOCK

command execution contains the address of the extracted block. Thus, the IFDDI will not use this block. For more information, see **9.2.4.6 GET BLOCK Command** and **9.2.4.7 GET BLOCK Indication.**

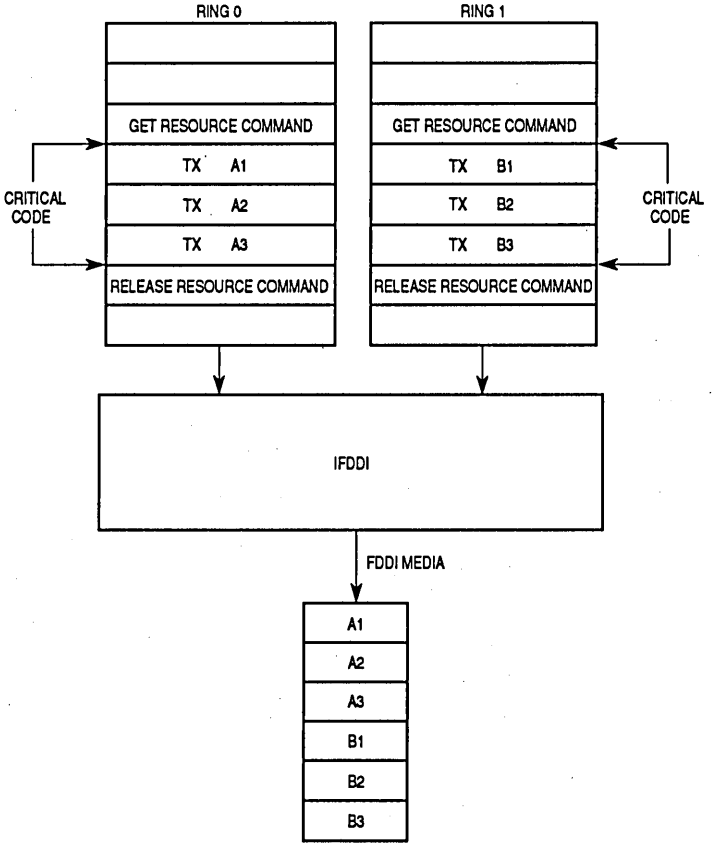


Figure 3-10. Example of Critical Code Protection

3.9 BUILT-IN RECEIVE FILTERING MECHANISM

The IFDDI can receive frames based on the frame control (FC) field. By using the receive frame type register (RFR), each ring can be programmed to receive specific types of frame. If the FC field does not match any of the receive types that are programmed, the reception operation for that frame is aborted. For example, a received LLC frame can be directed to ring 4, and SMT frames can be directed to ring 5. See **Section 7 Register Description** for additional information.

Another type of receive filtering supported by the IFDDI is known as split header mode. In this mode, the header for a frame is *always* directed to receive ring 4. The data portion of

the frame is directed according to the definition set in the RFR. The indication for the reception of the header portion will have the S-bit set to differentiate between this buffer and other buffers and will also have both the FIRST and LAST bits set.

3.10 EASILY CONFIGURABLE FOR SAS AND DAS

The IFDDI can easily be configured for use as a single attach station (SAS) or dual attach station (DAS) with an additional ELM. For a SAS, the user should connect the station as shown in Figure 3-11, set the CLASS_S bit to one in ELM control register B, and reset the BY_CNTRL bit to zero in the CAMEL control register.

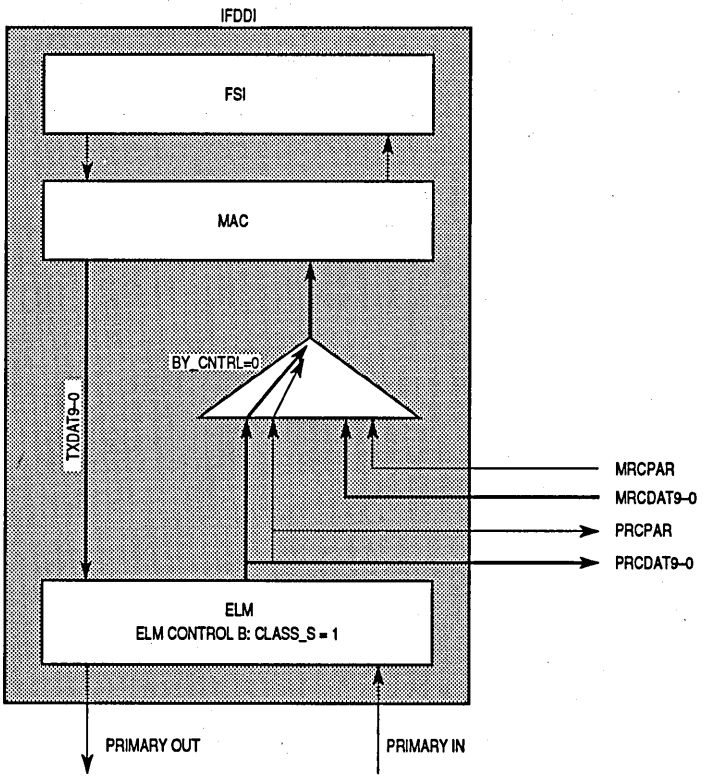


Figure 3-11. Single Attach Station

For a DAS, the user should connect the station as shown in Figure 3-12, reset the CLASS_S bit in ELM control register B, and set the BY_CNTRL bit to one in the CAMEL control register.

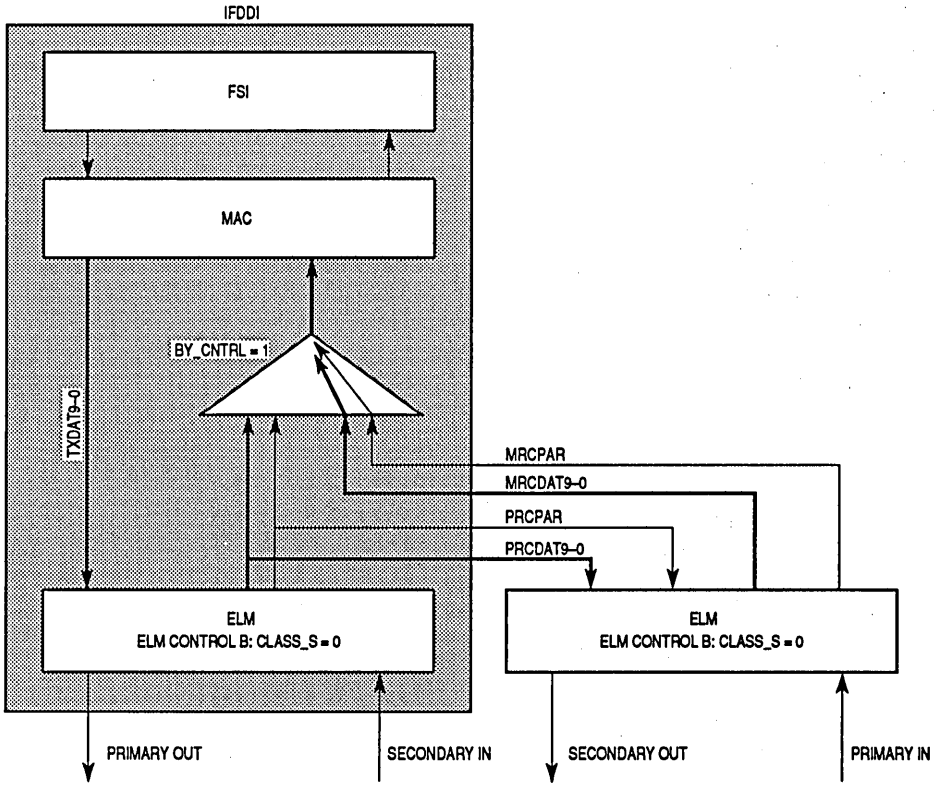


Figure 3-12. Dual Attach Station

3.11 IMPLEMENTS FULL MAC AND PHY ANSI STANDARDS

The IFDDI implements the full ANSI MAC and PHY standards, with some added functionality. Extra features include a PCM state machine within the ELM block, extra stripping and addressing modes for bridging in the MAC block, software access to each timer within both the MAC and ELM, and software access to the extra counters for both the MAC and ELM with the ability to monitor the counters through interrupts. There are also additional features for connecting an external CAM to the IFDDI. Lastly, there is the ability to interrupt or to mask the interrupts for important MAC and PHY events.

3.12 IMPLEMENTS STREAMING CIPHER PORTION OF TWISTED-PAIR STANDARD

The IFDDI contains streaming cipher capability, which in conjunction with a twisted-pair tranceiver will implement the twisted-pair standard for FDDI. Ciphering randomizes the transmitted data bits to reduce high frequency concentrations of the radiated spectrum at certain frequencies (that is, the spectrum is flattened and widened). When enabled, the

data is scrambled and unscrambled using the polynomial $X^{11} + X^9 + 1$. When the scrambler is disabled, the transmitted data is output directly from the PHY layer. Extra timers are also provided to allow flexiblility to the user in controling signal detect and line states during CONNECT, OFF and BREAK states.

3

3.12 INTERNAL 64-ENTRY CAM FOR ADDRESS RECOGNITION

The user can program the 64-entry CAM with any address for address matching. This feature is particularly useful when it is desirable to recognize more addresses (such as multicast addresses) than just the long and short addresses that are programmed into the MAC block. It is also useful for simple bridging applications that require up to 64 address entries. In addition to this internal CAM, there are two lines that enable the IFDDI to be connected to a large external CAM. The diagrams for this configuration are shown in Figures 3-13 and 3-14.

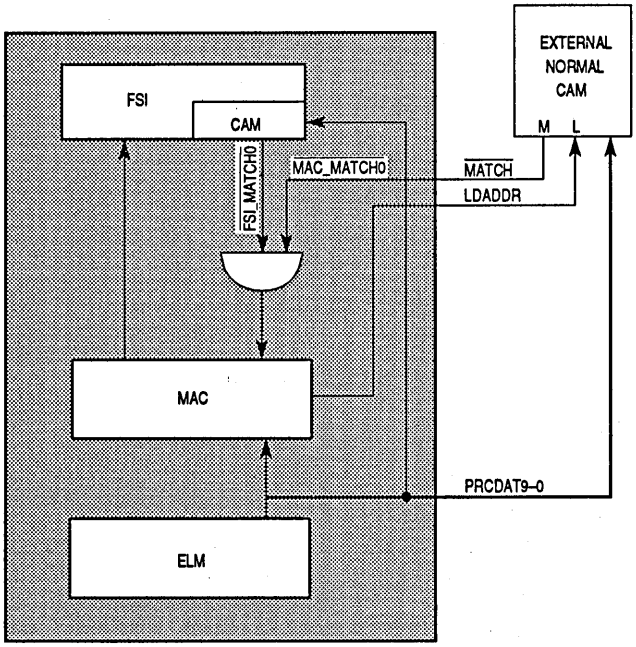


Figure 3-13. CAM Configuration In Normal Match Mode

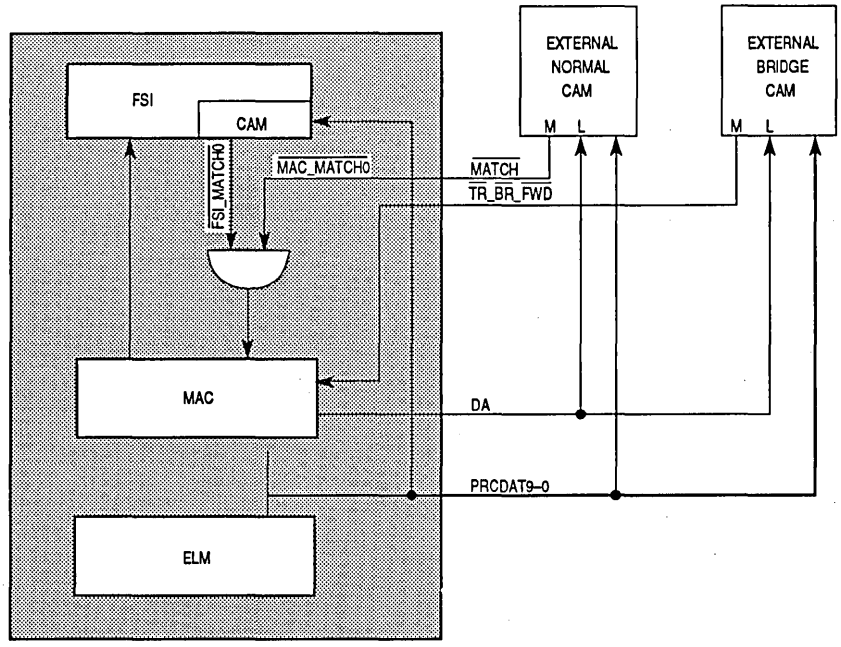


Figure 3-14. CAM Configuration in Extended Match Mode

3.13 BRIDGING FACILITIES

The IFDDI has extra functionality to make bridging applications simple to implement. A bridge is responsible for forwarding packets to stations that are on different network segments, which may or may not be of the same technology type, and for stripping packets that have been sent by the bridge.

The IFDDI provides many different options for bridge reception operations. First, the IFDDI may operate in promiscuous mode, where all LLC frames are copied. This option allows higher layer software to perform sorting on the incoming LLC traffic. Promiscuous mode is enabled by setting the COPY_IND_LLC and COPY_GRP_LLC bits in the MAC_CNTRL_A register.

A second frame reception option is to use an external CAM for holding addressing or routing information. This option makes extra address recognition logic necessary, which is provided by the $\overline{\text{MATCH}}$, LDADDR/$\overline{\text{TR\_BR\_FWD}}$, and $\overline{\text{REJECT}}$ pins of the IFDDI. The data bus used for external CAM address recognition is the PRCDATx data bus. If the extra address recognition logic requires extended timing, extended timing can be enabled by setting the EXT_DA_MATCH bit in the MAC_CNTRL_B register. This ability is particularly important in source routing bridges.

The IFDDI also provides functionality for transparent bridging. Transparent bridges require special handling of the A and C bits contained in a frame since they forward frames, giving an indication of copying the frame, but simply repeat the status of the address bit. Transparent bridging is enabled by setting the EXT_DA_MATCH bit in MAC_CNTRL_B, which not only provides extended timing, but also changes the functionality of the LDADDR output pin to be $\overline{TR\_BR\_FWD}$, an input pin.

For transmission operation, the IFDDI provides a wide range of options. The most essential operation of a bridge is the ability to transmit data frames with a source address that is not the station's MLA or MSA (long 48-bit address, short 16-bit address). In the case of the IFDDI, the upper layer software provides the source address as well as the destination address, frame control, and data fields of the frame.

Because a bridge transmits frames with source addresses that do not necessarily match the stations MLA or MSA, a bridge must use special stripping algorithms to remove frames from the ring that it has transmitted. Void frame/sent count stripping is provided by setting the BRIDGE_STRIP bit in the MAC_CNTRL_B register. When the BRIDGE_STRIP bit is set, the station (or bridge) will increment a counter for each frame that it has sent and send the special void frame at the end of the transmission. The station will then strip from the ring every frame that is received and decrement the counter until one of the following conditions is true: the counter reaches zero, a special void frame is received, a beacon frame is received, or a token is received.

The final option available to the user for transmission operation of a bridge is the ability to disable generation of the frame CRC on a per-frame basis. A bridge must forward a frame if received remotely without initiating any changes to it (i.e., recalculating the CRC), but if it is transmitting a frame generated locally, then the bridge does calculate the CRC before transmitting the frame. CRC generation is enabled or disabled by the APPEND_CRC bit in the packet request header of the frame. For more information, see **10.4.1.2 Transmit Data Interface.**

Another bridging option provided by the IFDDI concerns an Ethernet or token bus to FDDI bridge. This type of bridge must reverse the destination and source address fields upon transmission or reception of a frame since FDDI and Ethernet (or token bus) standards have different address representations. Reversing is performed automatically when the REVERSE_ADDR bit in the MAC_CNTRL_A register is set.

More detail on these features is provided in the following sections: **5.2.3 CAM Interface Functional Operation, 8.6 CAM Interface Signals,** and **7.3.3 MAC Core Registers.**

3.14 SPLIT CLOCKS

The IFDDI provides for the clock that drives the system interface (FSICLK) to be separated from the clock that drives the media interface (SYMCLK). If a separate FSICLK is used, the FSICLK must be greater than SYMCLK. This configuration is useful to be able to run the system interface block at the system speed for faster accessing while the media interface block remains at the FDDI standard speed of SYMCLK. When the two clocks operate at the same frequency, they should be tied together with as little skew as

possible. The port synchronization function (PSF) block is responsible for synchronization between the two blocks, but it may be bypassed in cases where the two clocks are at identical frequencies.

3.15 JTAG COMPATIBILITY AND BUILT-IN SELF-TEST (BIST)

There are two types of testing provided with the IFDDI: BIST and JTAG. BIST is useful for checking the MAC and ELM blocks when they are first powered up to ensure that the blocks are functioning properly. JTAG is an IEEE standard in which all I/Os are connected to a shift register for data to be shifted in and out of the I/Os. JTAG can confirm that board-level components are interconnected in the correct manner and can check that the IFDDI performs its required function.

SECTION 4
BLOCK DESCRIPTION

The IFDDI contains two major blocks consisting of the FSI and CAMEL cores. The IFDDI also contains the PSF, which performs clock sysnchronization, the SMT counter and the node processor interface.

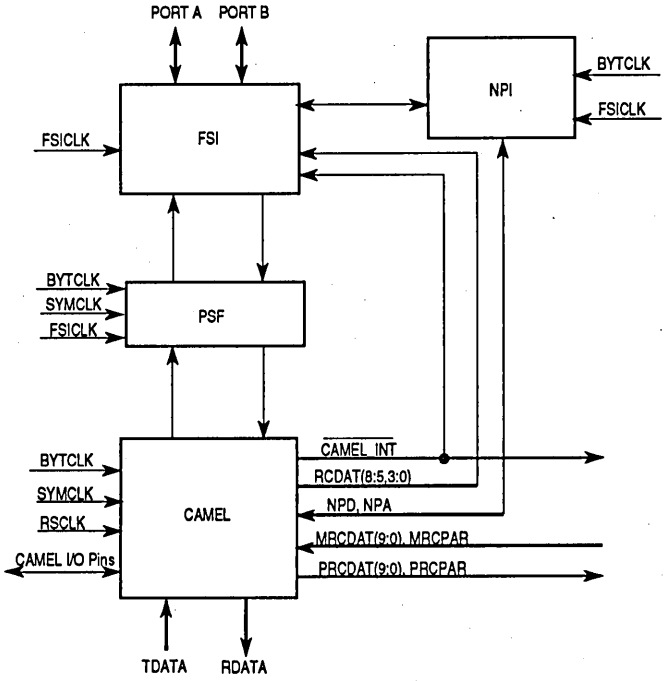


Figure 4-1 IFDDI Functional Block Diagram

4.1 FSI BLOCK DESCRIPTION

The FDDI system interface (FSI) has four main blocks: the MAC interface unit, the internal memory array, the port control unit, and the main control unit. Figure 4-2 illustrates the internal architecture of the FSI.

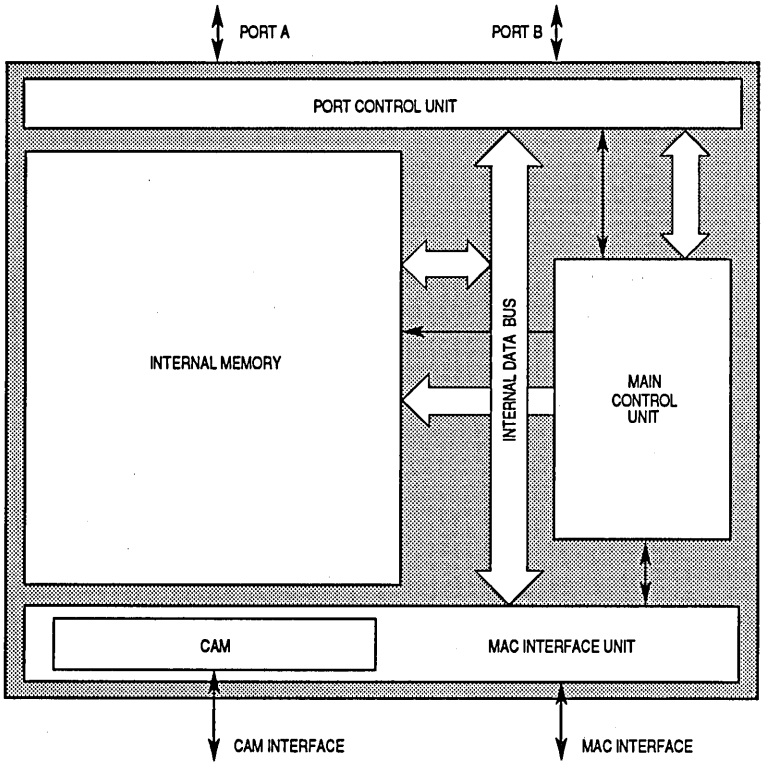


Figure 4-2. FSI Internal Architecture

4.1.1 Main Control Unit

The main control unit is responsible for controlling the other blocks and the timing between them. It performs processor command decoding and execution, internal virtual memory address generation, and handles priorities between various memory array accesses.

4.1.2 MAC Interface Unit

The MAC interface unit is responsible for interfacing the FSI to the CAMEL block and for transferring data between the internal memory and the CAMEL block. It's internal data path is synchronized to FSICLK. When SYMCLK and FSICLK are not identical in frequency the PSF block synchronizes between the MAC interface unit and the CAMEL block since the CAMEL block is synchronized to SYMCLK. The MAC interface contains separate receive and transmit buses, each of which can transmit either data or control information.

The MAC interface also contains small hardware FIFOs for transmission and reception. During normal FSI operation, frames that are not meant for this station are discarded during an early stage of processing while still inside the receive hardware FIFO in the

MAC interface. When receiving data, the MAC interface transfers the information from the hardware FIFO in blocks of 8 bytes into the internal memory array. When transmitting data, the MAC interface gets the control information and frames from the internal memory in blocks of 8 bytes to transfer to the hardware transmit FIFO in the MAC interface.

The MAC interface includes a selector mechanism that operates on the received frames as follows: when the MAC interface starts to receive data, it decodes the frame control (FC) field of the frame being received and, according to the value of the FC, determines into which receive ring to transfer the data. This receive ring selection is programmable by the user upon initialization.

The MAC interface also has one rejection signal, which is the input receive reject ($\overline{\text{REJECT}}$). The rejection signal may be used when the MAC is configured in promiscuous mode, since it enables external logic to decide whether to receive the frame based on any arbitrary pattern or algorithm implemented by that external logic.

4.1.3 Internal Memory

The internal memory is a memory array of 8K bytes arranged as 512 rows of 128 bits plus parity. It is used to temporarily store transmit and receive frames and descriptors. All data FIFOs supported by the FSI are mapped inside this memory array. The memory array is shared by port A, port B, and the MAC interface. Memory address generation is centralized inside the main controller.

4.1.3.1 Internal Transmit Data Structures and Command Issuance. For each descriptor ring in external memory, the FSI has two internal FIFOs in the internal memory. One is an internal data FIFO for temporary data storage; the other is an internal ring FIFO for temporary storage of commands, buffer descriptors, and indications.

Commands can also be issued directly to the FSI through the command register as opposed to through the descriptor rings as previously described. When transmit commands are issued directly to the command register, an internal command data FIFO is used for temporary storage of transmitted data.

The internal data and internal ring FIFOs exist only if their rings are defined. When a transmit ring is complete (there are no more frames to transmit), its internal data and ring FIFOs occupy 64 bytes (32 bytes each) of internal memory. During operation, the size of a transmit FIFO changes dynamically. If the transmit FIFO is not transmitting, the maximum allowed transmit data FIFO length is defined by its "watermark." The maximum transmit ring FIFO length is 160 bytes (five blocks of 32 bytes). When the transmit data FIFO is transmitting, the maximum size of that FIFO inside the internal memory is twice its watermark, and the maximum transmit ring FIFO length can be 288 bytes (nine blocks of 32 bytes). Note that only one transmit FIFO will ever be transmitting data at any time.

4.1.3.2 Internal Receive Data Structures. Much like the transmit side, the FSI has two internal FIFOs in the FSI internal memory for each receive ring in external memory. One is an internal data FIFO for temporary data storage of received data, and another is an internal ring FIFO for temporary storage of receive descriptors and indications. Once the

receive ring is defined and the receive operation is enabled for a ring, the data received from the network is saved inside the internal data FIFO. If the receive ring is ready, the FSI reads buffer descriptors and transfers the data into the location pointed to by the buffer descriptor. The internal memory space reserved for each receive ring is used for both the internal data FIFO and the internal ring FIFO.

In addition to received frames, other receiver event indications (e.g., TOKEN CYCLE END and RECEIVE MY FRAME indications) may occupy the internal ring FIFO using one entry (8 bytes) per indication.

4.1.4 Port Control Unit

The port control unit is responsible for data transfers between the internal memory and the host processor. Each port has its own external address generator and a block counter to support DMA operation. (Note that external bus timing is provided by the bus control logic outside the FSI.)

Each port of the port control unit may handle both input and output transfers at the same time. The FSI implements an internal priority between the ports to schedule the next required transfer request. When the FSI is ready to execute a transfer, it indicates the transfer type by asserting the request signals for that port. Once the request signals are asserted, they will not change until the access cycle is complete. A change of request signals from an active request to either some other request or to the idle state may be done only during the data read/write. At the end of the data cycle, the chip select signal is negated, and the next cycle request may be sampled by the bus control logic. Note that only accesses to the port data register have an effect on the port state and on address updating. All other registers may be accessed at any time regardless of the condition of the request signals.

Data is transferred through ports A and B as block transfers. On transmission, when a block of data is transferred into the internal memory, the port control logic assembles the data into 16-byte blocks with internal byte parity. It then performs byte-swapping, depending upon whether the bus connected to the port is big-endian or little-endian. The FSI fully supports byte-swapping, even for 64-bit data. Since the byte-swapping capability is selectable per port, it is possible to use processors with two different byte orders within a single system. After the 16 bytes of information are assembled, they are transferred to the appropriate location in the internal memory. When the frame to be transmitted includes many buffers, the port control logic is responsible for sequentially assembling all data of the frame so that the data is seen in internal memory as a contiguous block.

When the block transfer is from internal memory to the system (reception), the port control logic is responsible for synchronizing between the internal and external transfers. The data is transferred from internal memory to the port in blocks of 16 bytes.

The accesses to the ring are handled by the port control unit, which generates the address for each one of the rings to read commands and descriptors and to write indications.

The address counter of the external memory address is updated each time the data is read or written by either an external processor or the bus control logic. The address itself may be accessed by selecting the appropriate control pins.

The ports can be configured to operate separately as 32 bits each or can be combined as 64 bits. When configured separately, each port can be accessed to supply 32-bit data or 32-bit address or 32-bit address and data multiplexed. The ports can be configured to operate together in that the address generator of port A may be accessed from port A or port B and vice versa. This allows the ports to be configured as a single 64-bit data port with 32-bit address multiplexed or as a 32-bit data port with 32-bit address not multiplexed.

The port control unit also handles the byte order of the external bus (little endian/big endian). According to its initial definition, each port is able to transfer the data between the system and internal memory according to the type of bus the FSI is connected to. It must be noted that the command and indication definitions (bit and field assignment) and the addresses in the ring buffer descriptors are independent of the type of bus the FSI is connected to. Byte order does not matter since the bits and fields are defined explicitly for addresses and commands and indications.

The port control unit can determine whether the addresses of the current location to be accessed in external memory and the previous location accessed in external memory are on the same page. The limits of the page definition are supplied to the FSI upon initialization. The result of this comparison is supplied to the external bus controller on the request signals so that it can generate the appropriate timing for signals that access a dynamic memory using page or nibble modes.

4.2 CAMEL BLOCK DESCRIPTION

The CAMEL block consists of two principal sub-blocks: the elasticity buffer and link management (ELM) and the media access controller (MAC). A block diagram of the CAMEL is shown in Figure 4-3.

4

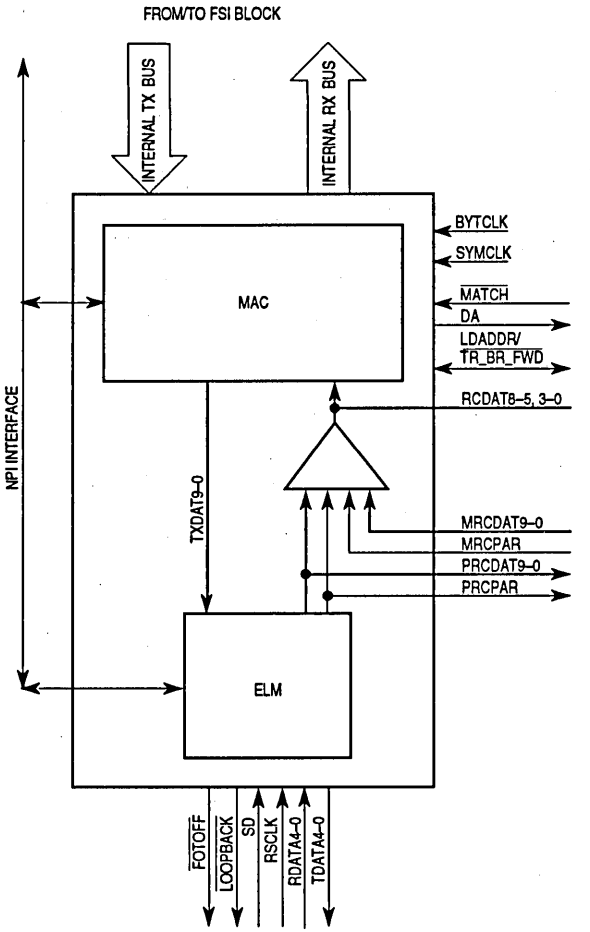


Figure 4-3. CAMEL Block Diagram

The CAMEL block was taken from the latest revision of the CAMEL chip. The CAMEL has seven interface groups: FSI transmit and receive, CAM interface, FCG transmit and receive interface, FCG control interface, dual attach station (DAS) interface, Node

Processor interface (NPI);, and test interface. The FSI transmit and receive interface is connected internally to the port synchronization function (PSF).

The CAMEL interrupt line is connected to an output pin ($\overline{\text{CAMINT}}$) and to a status bit in the FSI status register 1 (SR1). The FSI SR1 bit name is CAMEL interrupt (CIN). The $\overline{\text{CAMINT}}$ pin and the CIN bit in the FSI SR1 are used to notify an external processor of the occurrence of a CAMEL interruptible event. The processor must read the appropriate CAMEL interrupt register to determine the interrupt or interrupts that caused the event. The interrupt will remain set until the appropriate CAMEL interrupt register is read, clearing the interrupting event(s), or until the interrupt is masked by writing a zero to the appropriate CAMEL interrupt mask register bit. The port interrupt may be masked by writing zero to the CIE bit in the FSI interrupt mask register 1 (IMR1).

The CAMEL internal registers may be accessed by the interrupt handler using CAMEL direct access or the FSI control register (FCR) access. If FCR access is used, note that special precautions should be taken because the FCR is not *always* free to be used by the interrupt handler. This case occurs when a noninterrupt task uses the FCR and the access cycle is not complete.

The FCG (MC68836) transmit and receive interface and the FCG control interface are connected to external pins.

The DAS interface enables the connection of another ELM to the IFDDI. The ELM receive lines (PRCDAT9–PRCDAT0 and PRCPAR) are connected to output pins and to the CAMEL internal MUX. The other ELM receive lines (MRCDAT9–MRCDAT0 and MRCPAR) are connected from the input pins to the CAMEL internal MUX. The output of the MUX, which selects between MRCDATx and PRCDATx, is an internal bus, RCDAT9–RCDAT0. RCDATx also runs internally from the CAMEL core to the IFDDI internal CAM. An external CAM is connected to either MRCDATx or PRCDATx, depending on the system configuration.

The CAMEL counter segmentation test mode is controlled by the CSM bit in the IFDDI configuration register (ICR). After power-up reset, CSM = 0 and the counters operate normally. The CSM may be changed using the control register write access to enable/disable the counter segmentation test mode.

4.3 ELM FUNCTIONAL BLOCK DESCRIPTION

Figure 4-4 shows a diagram of the functional blocks and the data paths of the ELM. The following paragraphs describe the functions of the various blocks in the ELM.

The ELM implements the PHY functions of the FDDI standard, including data framing, elasticity buffer, encoding, decoding, smoothing, line state detection, and repeat filter. The ELM also contains the following station management (SMT) functions: physical connection management (PCM), physical connection insertion (PCI), and link error monitor (LEM).

4.3.1 Framer

The framer accepts 5-bit-wide parallel data as well as the RSCLK from the clock recovery device. Generally, data coming into the framer is not framed into proper FDDI symbols. The framer is used to align the incoming data to form proper symbols before the data is passed to the elasticity buffer. A starting delimiter that is used at the beginning of each frame is detected by the framer and used to determine proper symbol boundaries for the data. The framer has been designed such that the starting delimiter (the JK symbol pair) can be detected independently of previous framing.

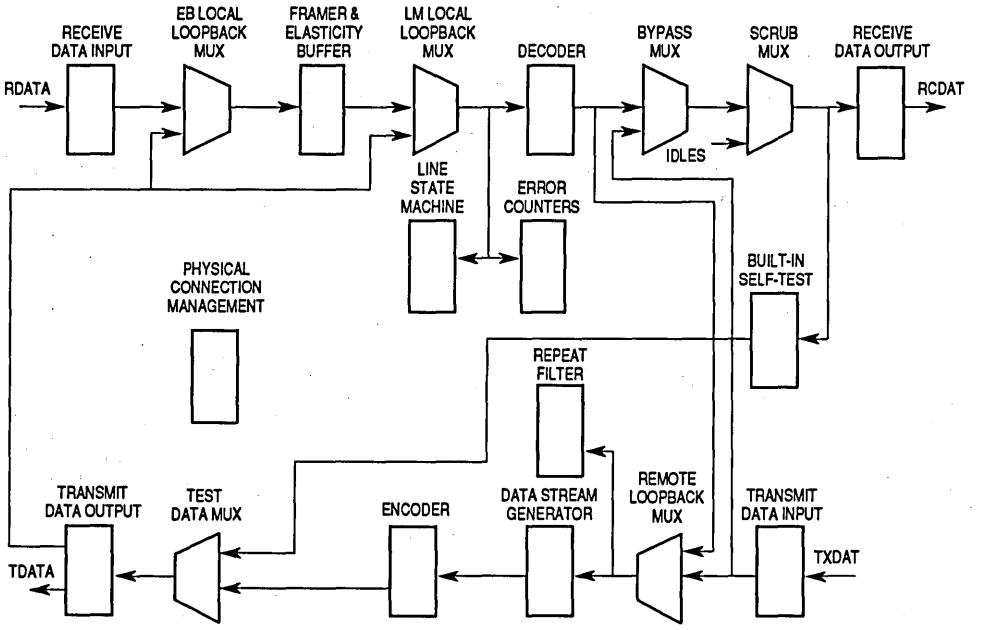


Figure 4-4. ELM High-Level Functional Block Diagram

4.3.2 Elasticity Buffer

The elasticity buffer performs the necessary buffering to allow data to pass between different FDDI stations with independent station clocks. The elasticity buffer consists of an 80-bit buffer and some control circuitry. The buffer is used to compensate for the differences in the transmit and receive clock frequencies in the station. Data is clocked into the buffer by the RSCLK and clocked out of the buffer by the BYTCLK. RSCLK is also used to drive all the input circuitry, including the input controller and input pointer. BYTCLK is also used to drive the output circuitry, including the output pointer, the output controller, the overflow/underflow detection circuitry, and the output buffer.

4.3.3 Data Path Multiplexers

The receive data path and transmit data path of the ELM include six multiplexers (MUXs) for the purpose of altering the normal flow of data through the core (see Figure 4-4). Altering the data paths may be necessary for physical connection insertion and removal and for testing and diagnostics. All receive and transmit paths internal to the ELM are 10 bits (two symbols) wide.

4.3.3.1 ELASTICITY BUFFER LOCAL LOOPBACK MUX. In normal operating mode, the elasticity buffer local loopback MUX puts data held in the RDATAx latch onto the receive data path of the ELM.

When the EB_LOC_LOOP bit in ELM control register A (ELM_CNTRL_A) is set or when built-in self-test (BIST) is running, the MUX loops back the data in the TDATAx latch onto the receive data path. This creates a path whereby data from the MAC can traverse the entire transmit and receive data paths of the ELM and be returned to the MAC. BIST uses this loopback along with the remote loopback MUX to create a loop that covers the entire transmit and receive data paths.

NOTE

When this loopback mode is in effect, data is internally looped back using BYTCLK, and RSCLK is not used in elasticity buffer operation.

4.3.3.2 LINK MANAGEMENT LOCAL LOOPBACK MUX. In normal operating mode, the LM local loopback MUX receives data from the elasticity buffer and passes it through the receive data path at a point before the decoder. When the LM_LOC_LOOP bit in ELM control register A is set, this MUX loops back the data in the TDATAx latch onto the receive data path. The LM local loopback MUX provides a method of testing most ELM circuitry without the influence of the framer or elasticity buffer.

4.3.3.3 BYPASS MUX. In normal mode, the Bypass MUX sends the data output provided by the decoder to the Scrub MUX. When the PCI is in the REMOVED, INSERT_SCRUB, or REMOVE_SCRUB state, or when the SC_BYPASS bit in ELM control register A is set while the PCM is in the MAINT state or the CONFIG_CNTRL bit in ELM control register B (ELM_CNTRL_B) is also set, the data in the TXDATx latch is put on the receive path to the scrub MUX. On power-up, this bypass path will be in effect. The delay through a bypassed ELM is one BYTCLK.

When the CLASS_S bit in the ELM control register B is set (as in SAS), whenever the PCI would normally be in the REMOVED state, it will be in the INSERTED state. Thus, before entering INSERT_SCRUB or after leaving REMOVE_SCRUB, rather than putting the ELM in the bypass mode, PHY_INVALID is output on PRCDATx.

4.3.3.4 SCRUB MUX. The scrub MUX selects its input from either constant idle (I) symbol pairs or the output of the bypass MUX. When the REQ_SCRUB bit in ELM control register A is set while the PCM is in the MAINT state or the CONFIG_CNTRL bit in ELM control register B is set, or when the PCI is in the INSERT_SCRUB or REMOVE_SCRUB

state, the output of the scrub MUX is I-symbols. Otherwise, the output of the bypass MUX is placed onto the PRCDATx bus.

This MUX is used during physical connection insertion and removal to output I-symbols on PRCDATx when scrubbing the ring.

4.3.3.5 REMOTE LOOPBACK MUX. In normal operating mode, the remote loopback MUX puts the data held in the TXDATx latch onto the transmit data path of the ELM. When the REM_LOOP bit in ELM control register A is set (and the EB_LOC_LOOP and LM_LOC_LOOP bits are not set) or when BIST is running, this MUX loops back the data from the decoder onto the transmit data path. This loopback creates a path on which received data from the FCG can traverse the entire receive and transmit data paths of the ELM and be transmitted back to the FCG. BIST uses this loopback along with the elasticity buffer loopback to create a loop that covers the entire receive and transmit data paths.

4.3.3.6 TEST DATA MUX. In normal operating mode, the test data MUX sends the data output by the encoder to the TDATAx latch. When BIST is running, the test data MUX selects the input from the BIST block. Test data is returned to the BIST block at the output of the scrub MUX.

4.3.4 Decoder

The decoder performs the 4B/5B decoding of received data symbols (see Table 4-1). The five bits of aligned data are decoded into four bits of data and one control bit, with the high-order bit being the control bit. The decoded symbol pairs are then sent to the MAC. Although the decoder operates on symbol pairs, each symbol is decoded independently of the other. Until the PCM has completed establishing a connection for the physical link, the PHY invalid (PI) symbol is returned to the MAC on both PRCDAT9–PRCDAT5 and PRCDAT4–PRCDAT0 if the CLASS_S bit in ELM_CNTRL_B is set.. If the CLASS_S bit is not set, TXDATx is looped back from the MAC transmit output onto PRCDATx. In addition, a violation (V) symbol is generated when an input error condition has been detected, such as an elasticity buffer error (buffer overflow or underflow). PI takes precedence over V; therefore, if an elasticity buffer error occurs while the current line state is Quiet, Halt, Master, or Noise, then PI is given to the MAC.

Table 4-1. 4B/5B Decoding of Data

Symbol	RDATA(9–5)(4–0)	PRCDAT(9–5)(4–0)
Q	00000	10000
I	11111	10111
H1	00100	10100
J	11000	11100
K	10001	10011
T	01101	11101
R	00111	10001
S	11001	11001
H5	00001	10100
H4	00010	10100
V1	00011	11000
V2	00101	11000
V3	00110	11000
H3	01000	10100
V4	01100	11000
H2	10000	10100
PI	XXXXX	11111
0	11110	00000
1	01001	00001
2	10100	00010
3	10101	00011
4	01010	00100
5	01011	00101
6	01110	00110
7	01111	00111
8	10010	01000
9	10011	01001
A	10110	01010
B	10111	01011
C	11010	01100
D	11011	01101
E	11100	01110
F	11101	01111

X = Don't care.

4.3.5 Encoder

The encoder performs the 4B/5B encoding of data symbols to be transmitted over the physical medium (see Table 4-2). The four bits of data and one control bit from the MAC are encoded into a unique 5-bit symbol that is sent to the FCG circuitry. Although the encoder operates on symbol pairs, each symbol is encoded independently of the other. When the GOBBLE_BYTE signal is asserted by the repeat filter (see **Section 4.3.6 Repeat Filter**), the input data symbols are ignored, and the encoder outputs an I-symbol pair.

A parity error on the internal TXDATx bus sets the ELM's internal bus to a pair of INVALID symbols (1100011000). If the RF_DISABLE bit in ELM control register A is clear, the repeat filter changes these INVALID symbols to I-symbols. Parity detection can be enabled or disabled with the ENA_PARITY_CHK bit in ELM control register A.

The encoder can be disabled via the ENCOFF pin.

4.3.6 Repeat Filter

The repeat filter operates on the symbol stream at the output of the remote loopback MUX. Only Idle and Active line states are allowed to propagate through the station. Invalid line states will be turned into an I-symbol stream. Also, if the repeat filter detects a corrupted frame, it truncates the frame by transmitting four Halt (H) symbols and then an I-symbol. The H-symbols will cause the next MAC in the logical ring to count the frame as a lost frame.

Another function of the repeat filter is called the GOBBLE_BYTE. When the repeat filter detects a fragment (i.e., a frame in which an I-symbol appears before the ending delimiter), it instructs the encoder to change the previous symbol pair to Idles. After passing through repeat filters in other stations, the fragment will eventually be completely converted to Idles.

The repeat filter is defined in the ANSI FDDI PHY document. The ELM is a byte-wide implementation.

Table 4-2. 4B/5B Encoding of Data

Symbol	TXDAT(9–5)(4–0)	TDATA(9–5)(4–0)
Q	10000	00000
I	10111	11111
H	10100	00100
J	11100	11000
K	10011	10001
T	11101	01101
R	10001	00111
S	11001	11001
INVALID	11110	11111
INVALID	10010	11111
INVALID	10101	11111
INVALID	10110	11111
INVALID	11111	11111
INVALID	11000	11111
INVALID	11010	11111
INVALID	11011	11111
0	00000	11110
1	00001	01001
2	00010	10100
3	00011	10101
4	00100	01010
5	00101	01011
6	00110	01110
7	00111	01111
8	01000	10010
9	01001	10011
A	01010	10110
B	01011	10111
C	01100	11010
D	01101	11011
E	01110	11100
F	01111	11101

4.3.7 DATA STREAM GENERATOR

The data stream generator uses a multiplexer to generate a symbol pair at the request of the PCM or the repeat filter, or by transmitting the symbol pair from TXDATx. When the PCM is in the MAINT state, the symbol sourced is defined by the state of the MAINT_LS field in the ELM control register B. The repeat filter and the PCM state machine can be

turned off, but while operating, they generate symbol pairs according to their internal algorithms.

4.3.8 Data I/O Ports

The ELM contains four ports for the input and output of network data: RDATAx, PRCDATx, TXDATx, and TDATAx. The signal timing for these ports is shown in **Section 13 Electrical Characteristics.**

4.3.8.1 RECEIVE DATA INPUT. RDATAx is a 5-bit (symbol-wide) data bus coming from the FCG to the ELM. Data is clocked in synchronously with RSCLK. RSCLK is also used to clock the data through the framer and into the elasticity buffer.

4.3.8.2 RECEIVE DATA OUTPUT. PRCDATx is a 10-bit (symbol-pair-wide) data bus going from the ELM to the MAC block (or, in a DAS, to another ELM). Data is latched inside the ELM on each rising edge of BYTCLK and is available to the MAC.

4.3.8.3 TRANSMIT DATA INPUT. TXDATx is a 10-bit (symbol pair wide) data bus coming from the MAC to the ELM. Data is latched on each rising edge of BYTCLK.

For the operation of this bus when connected to an external ELM, as in the case of a Dual Attach Station, refer to **3.10 Easily Configurable for SAS and DAS.**

4.3.8.4 TRANSMIT DATA OUTPUT. TDATAx is a 5-bit (symbol-wide) data bus going from the ELM to the FCG. The 10-bit internal data bus is latched initially by the ELM on each rising edge of BYTCLK. Bits 9–5 are then latched by the rising edge of SYMCLK following the *rising* edge of BYTCLK. Bits 4–0 are then latched by the next rising edge of SYMCLK, which follows the *falling* edge of BYTCLK. Data is available to the FCG after each rising edge of SYMCLK.

4.3.9 Connection Management

The following three logic blocks implement facilities to provide for PCM and link monitoring. These blocks implement helpful time-critical state machines and monitoring logic for use by SMT and CMT pseudocode.

4.3.9.1 LINE STATE MACHINE. In FDDI networks, a special group of symbols called line state symbols (Q—Quiet, H—Halt, I—Idle, and the JK symbol pair) are transmitted to establish the physical connection between neighboring stations. These line state symbols are unique in that they can be recognized independently of symbol boundaries. Line states are comprised of consecutive line state symbols as defined in **5.2.1.1 Line State Machine Operation.**

4.3.9.2 LINK ERROR MONITOR. The LEM provides an indication of the inbound link quality to the PCM. The PCM uses this information to determine if the Link Confidence Test passes to establish a new connection. Once a link is active, the PCM continually runs an LEM test to detect and isolate links having an inadequate bit error rate.

4.3.9.3 PHYSICAL CONNECTION MANAGEMENT. CMT defines the operation of PHY layer insertion and removal and the connection of PHY entities to the MAC entities. PCM, a subset of CMT, is the management of a physical connection between the PHY being managed and another PHY.

PCM consists of two entities: the state machine and the pseudocode. The ELM implements the PCM state machine; whereas, the pseudocode is implemented by driver software.

The PCI state machine works in conjunction with the PCM state machine. The PCI controls ring scrubbing and the insertion and removal of a station on the ring.

4.3.10 Test Logic

The ELM has built-in self-test (BIST) capability. See **Section 11 Test Operation** for more details

4.4 MAC FUNCTIONAL BLOCK DESCRIPTION

The MAC performs all the functions of the ANSI FDDI MAC standard. A block diagram of the MAC is illustrated in Figure 4-5.

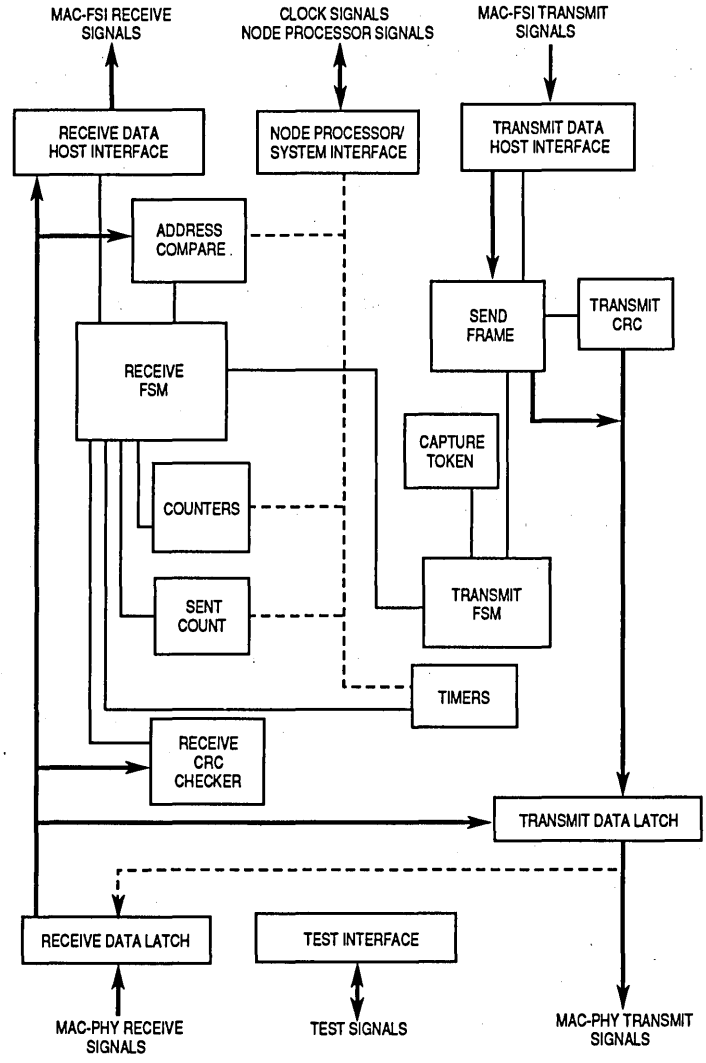


Figure 4-5. MAC Block Diagram

4.4.1 Receive Data Path

The receive data path is the internal data bus associated with receiving packets from the ring. It connects to the RCDATx bus (data bus from the ELM) through a pipeline latch in the receive latch logic and to the internal RPATHx bus (data bus to the FSI block). Only the receive data latch containing the current received symbol pair drives this internal

data bus. Usually, several different logic blocks are concurrently reading and processing this symbol pair. These logic blocks perform the following functions:

- Decode the input symbol pair, recognize the beginning of frames, and use this information to run the receiver finite state machine.
- Compare the DA field of a received packet to this station's individual 16- or 48-bit address.
- Compare the SA field of a received packet to this station's individual 16- or 48-bit address.
- Run the external CAM address matching logic.
- Compare the INFO field of claim frames to this station's requested token rotation time.
- Perform CRC checking on the received packet.
- Store the frame status indicators that have been received.
- Keep a count of the number of good and bad frames received.

4.4.2 Receive Latch

The receive latch clocks data from the ELM. The data signals are latched by the falling edge of an internal clock, SAMPLE_CLK. SAMPLE_CLK is derived by using the falling edge of SYMCLK to sample BYTCLK. The latched data signals are then relatched by BYTCLK to provide stable symbols for MAC processing.

4.4.3 Receive CRC Checker

This logic block checks the FCS field of received frames and operates a byte at a time, using the data symbol pair currently on RCDATx. The CRC checker receives a signal from the receive finite state machine telling it when to initialize for another CRC check and receives a strobe from the receive finite state machine for each symbol pair to be included in the CRC check. This block provides a valid CRC signal to the receive finite state machine. This signal should only be examined after the CRC checker has processed the last byte of the incoming frame's FCS field.

4.4.4 Sent Count

Sent count is used in bridgestrip mode to determine if a frame was sent by this station. All data frames, as well as special void frames, are counted.

4.4.5 Counters

This block holds the following counters:

- The frame count register is a 16-bit unsigned count of the number of frames (good or bad) that have been received since the last time this counter was read and reset or the chip was reset.

- The error count register is a 6-bit unsigned count of the number of error frames (i.e., frames containing a bad CRC or an invalid data length) that were detected by this station, but by no previous station, since the last time this counter was read and reset or the chip was reset.
- The lost count register is a 6-bit unsigned count of the number of frame format errors that have occurred since the last time this counter was read and reset or the chip was reset.
- The token count register is a 16-bit unsigned count of the number of tokens (restricted or unrestricted) that have been received since the last time this counter was read and reset or the chip was reset.

These counters do not stick at their largest value, but always wrap around to zero. These registers receive their increment signals from the receive finite state machine.

4.4.6 Receive Finite State Machine

This finite state machine implements the receiver process as described in the FDDI MAC standard—i.e., it controls all aspects of parsing and validating frames and tokens, determines whether a frame should be received or stripped, detects ring errors, and notifies the transmit finite state machine of any relevant events. This block also decodes and forwards the received FC field, decodes the current symbol pair on the receive data path, and parses and forwards the received frame status field. This block contains all the status flags described in the MAC standard: E_FLAG, A_FLAG, C_FLAG, N_FLAG, and R_FLAG.

4.4.7 Address Comparator

This block performs part of the DA actions, SA actions, and CT actions as specified in the FDDI MAC standard. Specifically, this block compares the DA and SA fields of received frames to this station's individual addresses (my short address register and my long address register). It also compares the INFO field of received claim frames (the token rotation time requested by another station) to this station's desired token rotation time.

This logic block has two parts:

1. A register array contains this station's individual 16- and 48-bit address (my short address register and my long address register) and this station's desired token rotation time. This register block contains a byte-wide read-only port that feeds into the comparator and into the transmit data path when generating claim, beacon, and void frames.
2. The comparison logic part contains a byte-wide comparator that gets its inputs from the register array and from the receive data path.

The receive finite state machine controls this logic, and the results of the comparison are passed to the receive finite state machine.

4.4.8 Receive Host Interface

The FSI receive interface logic controls the RPATHx and RCCTLx buses that pass received frames to the FSI. It strips off the delimiters before passing the frame to the FSI and handles all the extra control and handshake lines required for the RPATHx bus. The receiver FSM controls this logic and receives abort/flush signals from it. The FSI receive interface logic is completely separate from, and does not communicate with, the FSI transmit interface logic.

4.4.9 Transmit Data Path Operation

The transmit data path is the internal data path associated with the transmission of data onto the ring. The send frame logic assembles a packet consisting of preamble (idles), JK, FC, DA, SA, INFO field, CRC, T, and the FS indicators from various sources. Frame_Data is multiplexed with either idles or repeat data from the receive data path in the transmit latch logic. The transmit latch logic contains the TXDATx pipeline latch that drives the TXDATx internal bus. The TXDATx bus passes a symbol pair to the ELM chip on the following BYTCLK cycle after the MAC chip has received it.

Frame data at the transmit latch logic can have any one of the following sources:

- The transmit data path latch—contains the symbol pair passed from the FSI via the TPATHx bus. This latch is part of the FSI TX interface logic.
- The delimiter generator—transmits frame delimiters like idles, JK, TT, etc., in response to requests from the transmit FSM.
- The transmit CRC generator—appends the frame check sequence to the end of the data field.
- The address registers—hold this station's individual addresses and its value of desired token rotation time for void, claim, and beacon frames. These registers are properly a part of the receive data logic, but they can be accessed by the transmit data logic as needed.

4.4.10 Transmit Data Host Interface.

The FSI transmit interface logic controls the TPATHx bus over which the FSI passes packets to be transmitted to the MAC. This logic handles all the extra control and handshake lines required for the TPATHx bus. For example, it controls the reception of packet request information from the FSI, the notification that the MAC is ready for the next packet or packet request header, etc. This logic does not communicate with the FSI receive logic.

4.4.11 Send Frame Logic

The send frame block is responsible for the actual transmission of a frame including the sequencing and sending of (i.e., multiplexing of) the preamble, the appropriate delimiters (e.g., JK, TR, RR, etc.) the FC field (for token, claim, beacon, and void frames), the DA and SA fields (for claim, beacon, and void frames where they could be my long address register, broadcast, or null address), information fields (claim and beacon frames), general

data (FC, SA, DA, and INFO fields for FSI frames), the CRC field, and the requested frame status. The transmit finite state machine tells this logic what kind of frame to send and when to start sending it.

4.4.12 Capture Token Logic

This logic block holds the packet request header passed to it by the FSI. The packet request header contains all the information that the MAC transmitter needs to determine when it can send this frame (i.e., asynchronous/synchronous, token type, immediate mode or not, etc.), what type of token to issue later, whether a CRC is to be generated, etc. In addition to parsing this information, this block contains the logic to determine when a token can be captured and issued and when a frame can be transmitted. It feeds these signals into the transmit finite state machine that takes the appropriate action, depending upon its current state.

4.4.13 Transmit CRC Generator

This logic block appends the FCS field (a 32-bit CRC) onto transmitted frames. Some packets may already contain the FCS field, in which case this logic block may be used as a CRC checker instead of a CRC generator. When used as a checker, a bit is set in the interrupt event register when the transmitted CRC field is incorrect. This causes an interrupt if this event is not masked in the interrupt mask register. This logic block computes the CRC on data bytes it reads from the FRAME_DATA bus that comes from the send frame block. It also receives control signals from the transmit finite state machine.

4.4.14 Transmit Finite State Machine

The transmit finite state machine implements the transmit process as described in the FDDI MAC standard. Specifically, the transmit finite state machine repeats packets from other stations on the ring, determines when it can capture the token, how long it can hold the token, what type of token to issue, what kind of frame to send (but does not actually send it), and participates in the ring recovery procedures, etc.

4.4.15 Timers

This logic block contains all the MAC protocol timers (TRT, THT, and TVX timers), the associated register fields used to load them (T_MAX, T_NEG, T_BID_RC/T_INFO, and TVX_VALUE), and the LATE_CT. The MAC standard specifies an additional register, T_OPR, which can contain redundant information and therefore is not implemented.

The TVX timer is used to ensure that a good frame (i.e., correct CRC and valid data length) or a nonrestricted token is seen by this station on a regular basis. It can be used to detect events such as a babbling station, an infinitely circulating restricted token, a lost token, etc.

The TRT timer is used to determine the time taken for each rotation of the token. The transmit finite state machine uses this information to decide whether the token has been

lost and whether this station can then transmit when it receives the token. If the token takes too long to get to this station, it implies that the ring is busy and this station may have to defer its lower priority transmission. This timer also times the claim and beacon recovery procedures to determine if they will complete.

The THT timer controls the length of time that a station can transmit when it has captured the token. This timer is loaded with the value of the TRT timer when a token is captured. A station can then transmit asynchronous frames until this timer expires. This procedure ensures that the sum of this stations' transmissions and that of the previous station's transmissions during this rotation of the token will be (approximately) less than the mutually agreed upon token rotation time.

The Void timer is used to monitor the length of time that a void frame will travel over the ring (ring latency). The void time register contains the latency of the last properly transmitted and received Void frame. This timer can be used in conjunction with the Token counter to measure the ring utilization.

This logic block is connected to the receive data path to be able to load a bid time register from claim frames that are being received. In addition, it receives signals to enable/disable the TVX timer and to reset the TVX timer from the receiver FSM. This block's only outputs (other than to the node processor/system interface) are TRT expired, TVX expired, THT expired, and LATE_CT equals zero interrupts.

4.4.16 Transmit Data Latch (and Repeat Function)

This logic block contains the repeat function and multiplexes the repeat path with FRAME_DATA from the send frame logic block. The repeat function reads symbol pairs from the receive data path and repeats them to the ELM. Usually this logic transmits the symbol pair just received, but when receiving the FS indicators, the logic may need to modify the received R and S symbols according to the values in the frame status logic and the values in the register associated with repeating additional FS symbols. Also, this logic selectively replaces the last symbol pair with an I-symbol pair when a frame is stripped or detected as a fragment.

4.4.17 Test Logic

The MAC has BIST capability. See **Section 11 Test Operation** for more details.

4.5 TWISTED PAIR SUPPORT BLOCK DESCRIPTION

Twisted pair support can be broken into three distinct categories: streaming cipher, FOTOFF control and signal detect filtering.

4.5.1 Streaming Cipher

When ciphering is enabled, the streaming cipher scrambler uses the $X^{11} + X^9 + 1$ polynomial to flatten and widen the spectrum created by the transmission of data bits on the twisted-pair media. The descrambler seaches for any of the repeating line states (ILS,

HLS, QLS or MLS) to aquire synchronization with the remote scrambler. Once synchronized, the descrambler will properly de-scramble the received data using the $X^{11} + X^9 + 1$ polynomial.

When ciphering is not enabled in the CIPHER_CNTRL register, the ELM operates as described in **Section 4.3.**

4.5.2 FOTOFF Control

When using fiber media, whenever the physical connection machine (PCM) in the PHY block enters either the BREAK or OFF states, no light energy is to be transmitted onto the fiber. The IFDDI insures this through use of the FOTOFF output pin by turning off the transmit data output of the FCG (clock generation and recovery). This bypasses any information the FCG's NRZI encoder is attempting to output to the fiber.

When using twisted-pair media, a different mechanism must be used. The propsed standard requires that a scrambled-quiet be transmitted for a minimum time (currently 50uS). At any point following this minimum time, true-quiet can be sent. The FOTOFF control provided by the IFDDI is flexible. A user can choose to send scrambled-quiet during the BREAK state and true-quiet during the OFF state. A user is also allowed to transmit only scrambled-quiet at all times, never transmitting true-quiet.

4.5.3 Signal Detect Operation

The Signal Detect block filters the Signal Detect (SD) input from the clock recovery device for presentation to the PHY/ELM device. In fiber-optic systems, this block's three filters should be left disabled. In twisted-pair systems, the SD input is optionally filtered by data path transistion sensors. In addition, SD can be delayed to allow the descrambler to initialize.

The Signal Detect block performs three types of filtering: a turn-off filter; a turn-on filter; and an NRZ filter. The turn-off filter enables the SD input to the PHY block only when the RDATA(4:0) bus is active. The turn-on filter delays the assertion of SD for a fixed amount of time to allow the descrambler to aquire lock. The NRZ filter enables SD only when the de-scrambled NRZDATA(4:0) bus is active. The Signal Detect ouput to the PHY layer is called Signal Detect Filtered (SDF).

Because of the increased functionality of signal detect, the SD input is now synchronous to the RSCLK input and not the BYTCLK input.

4.6 MISCELLANEOUS FEATURES BLOCK DESCRIPTION

The CAM interface, port synchronization function, SMT timer, and port interface blocks are discussed in the following paragraphs.

4.6.1 CAM Operation

The CAM is a 64-entry by 48-bit CAM that is used for address matching. The CAM includes a valid bit with each CAM entry. The CAM control signals include a load address signal to mark the beginning of a received address field and an output indication signal if an address is matched.

The CAM lines are connected internally to the RCDATx bus. The addressing match mode selection is performed according to the CAMEL MAC configuration. If the CAMEL MAC is in extended match mode, the internal CAM will execute comparisons only on the destination address. If the CAMEL MAC is in normal match mode, the internal CAM will execute comparisons on the destination address and the source address.

The CAM is manipulated through the CAM commands (see **Section 9 Commands and Indications**). As a result of these commands, the system or a node processor can write, read, or clear an entry in the CAM or compare data with the CAM contents. A write operation of a CAM command updates the contents of the entry pointed to by the command with the data supplied in the command and either sets the corresponding valid bit to indicate a valid address or resets the valid bit to clear or invalidate the entry. A read operation of a CAM command supplies the host processor with the contents of the entry pointed to by the command. Upon reset, each CAM entry's valid bit is reset.

4.6.2 Port Synchronization Function (PSF)

To increase the IFDDI clock rate, the IFDDI internal clock is separated from SYMCLK. The PSF block synchronizes between the FSI MAC interface unit (which operates with FSICLK) and the CAMEL MAC sub-block (which operates with SYMCLK and BYTCLK). The synchronization causes up to a three BYTCLK delay in the receive path between the CAMEL MAC and the FSI.

NOTE

The PSF will operate properly only if FSICLK frequency ≥ SYMCLK frequency. In cases where FSICLK = SYMCLK, both pins should be tied together.

4.6.3 SMT Timer

The SMT timer was designed to be used by SMT software. The timer counts in 512-BYTCLK increments. The expiration time ranges from 512 BYTCLKs (40.96 µs at 12.5 MHz) to 2^{25} BYTCLKs (2.684 sec at 12.5 MHz). The expiration time is set by writing to the SMT timer load value registers, STL0 (least significant bits) and STL1 (most significant bits). Writing to STL1 causes the timer to load the value contained in the load registers and to begin counting down. When the timer reaches zero, the STE bit of status register 1 (SR1) is set, causing an interrupt if enabled. The timer is then reloaded with the timer load value and continues to count down.

The current value of the timer can be read at any time by reading the SMT timer registers, STR0 and STR1. The fact that the timer stops counting when either of these registers is

read serves a dual purpose. First, it guarantees consistent readings since the timer value could change between reading the most significant and least significant parts. It also provides a method of turning off the timer when it is not needed to prevent unnecessary interrupts. The timer will then remain disabled until the next write to STL1.

4.6.4 Node Processor/Port Interface

The CAMEL internal registers are accessed through the node processor interface block, which is part of the port interface. This interface provides access to the CAMEL internal registers from the following sources:

1. CAMEL direct access *only* through port A—see **Section 7 Register Description**
2. FSI control register (FCR)—see **Section 7 Register Description**
3. CONTROL REGISTER WRITE command—see **Section 9 Commands and Indications**

SECTION 5
FUNCTIONAL OPERATION

This section details the functional operation of the FSI and CAMEL blocks.

5.1 FSI CORE FUNCTIONAL OPERATION

The function of the FDDI system interface (FSI) device is the transfer of frame data between the system bus/memory and the FDDI media access controller (MAC) core. The FSI core may also serve as a DMA engine and transfer data from one memory area to another.

The following sections serve as an overview for the data DMA functionality and as an introduction to the data types used with the Motorola IFDDI. **Section 11 Initialization and Programming** has more detailed information on realizing the functionality introduced here.

5.1.1 Data Flow Functional Overview

This subsection illustrates some of the possible configurations for the transfer of data between the system interface and the FDDI or for data transfers from one memory area to another over the system port interface(s).

5.1.1.1 DIRECT TRANSMIT AND RECEIVE OPERATION. Figure 5-1 shows the simplest and most immediate method for the transmission and reception of frame data to and from the FDDI.

Figure 5-1 shows the use of both ports A and B. However, the implementor may desire the use of only one port, in which case, the idle port could be used to expand the interface to 64-bit multiplexed address/data operation, to provide a nonmultiplexed address source, or one of the ports may simply be left idle.

In the direct transmit and receive implementation, the user may use up to four transmit channels or rings (numbers 0–3) and two receive channels (numbers 4 and 5). The ring numbers are fixed; however, they can be individually assigned to either port via the ring parameter registers. In Figure 5-1, Tx ring 1 and Rx ring 4 are assigned to the port on the system interface, and Tx ring 3 and Rx ring 5 are assigned to the port on the local memory system.

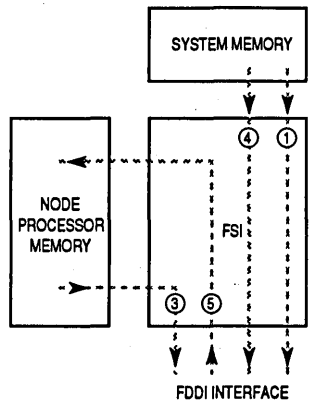


Figure 5-1. Direct Transmission and Reception

5.1.1.2 TRANSMIT AND RECEIVE OPERATION WITH LOCAL MEMORY. The FDDI protocol allows a station to transmit frames only when it holds a token. Systems in which the bandwidth allocated for the IFDDI at any moment is lower than the FDDI bandwidth of 12.5 Mbytes might not be able to sustain a transmission operation throughout the token-holding period and could be forced to release the token. The solution to this problem is to divide the frame transmission flow into two steps. The first step is to transfer the frame to local memory private to the IFDDI. The second step is to transmit the frame from the local memory to the FDDI media. The second step can occur only when a station holds the token, or an average of 1/N of the time where N is the number of stations on the network. The first activity can be done constantly allowing a system to have a bandwidth of only (12.5 Mbytes)/N. An example of this implementation is illustrated in Figure 5-2.

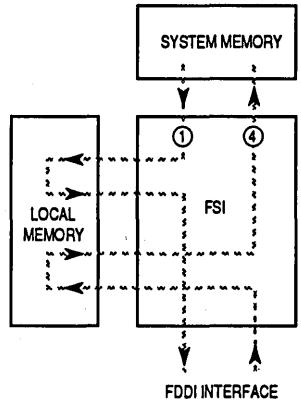


Figure 5-2. Extended Local Memory Option

In this configuration, transmit data from the system is first transferred to the local or extended memory. When an entire frame has been transferred, the frame data is available to be transferred to the FDDI. Receive data from the FDDI is first put into local memory and will subsequently be transferred to system memory. Note that the local memory is assumed to be readily available to the FSI with low system bus contention. This ensures that frame data is transferred between the local memory and the FDDI in a timely fashion. The transfer of data between the local memory and system memory may then occur at a slower pace, determined by the amount of contention on the system bus port.

Figure 5-2 portrays two channels, one receive and one transmit, that uses the local memory. All transmit and receive channels may make use of local memory, and even mixed use is possible. Figure 5-3 shows channels 2 and 5 using local memory and channels 1 and 4 using the direct transfer method.

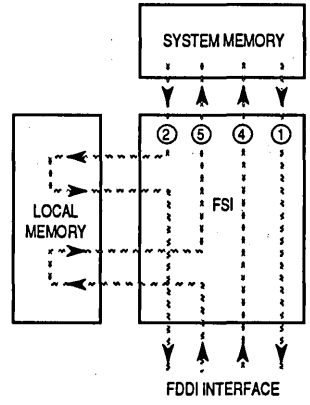


Figure 5-3. Direct and Expanded Local Memory Options

The local memory may also be shared by a local processor and have specific channels assigned to it. However, the implementor must ensure that the local memory interface provides sufficient bandwidth to handle the data flow. Such an application is shown in Figure 5-4.

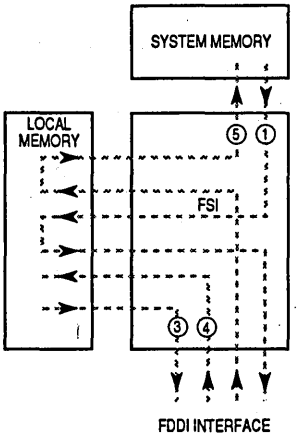


Figure 5-4. Dual Ports with Local Memory Option

Figure 5-4 shows the assignment of Tx ring 1 and Rx ring 5 on the system memory port, implemented perhaps due to a low bus latency. Tx ring 3 and Rx ring 4 are assigned to the local bus using direct transfer.

5.1.1.3. MEMORY TO MEMORY DMA OPTIONS. The FSI also provides facilities to use the DMA channels, or rings, for the purpose of transferring data from one memory location to another, even across ports. The FSI is capable of a variety of data DMA transfers as shown in Figure 5-5.

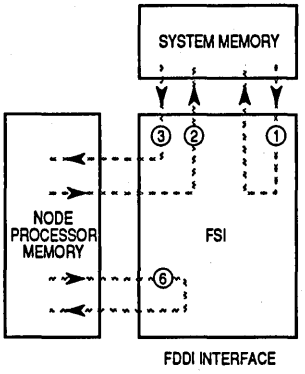


Figure 5-5. Memory to Memory DMA

Figure 5-5 only shows memory to memory DMA operations. The memory DMA operations may occur simultaneously with any of the FSI data transfer from either port's memory and the FDDI. Memory to memory DMA operations may be used to transfer any block of data.

For example, host processors on one port may use this facility to download data or code to another processor's memory space on the other port.

Any transmit ring (0–3) not using local memory or any command channel (6–7) can be used for DMA operations and frame data transmission.

5.1.2 FSI Data Structures

The primary purpose of the FSI is to facilitate the transfer of frame data packets between the system memory and the FDDI network, primarily the FDDI MAC core. This subsection defines the nature of the frame data, what the user needs to know about the memory format of the data, and the data structures that the FSI uses to effect the data transfer. When the FSI is used to perform memory to memory DMA data moves, the user may disregard the nature of the data since that data may not necessarily be destined for the FDDI network.

The external memory frame data can exist in one contiguous buffer or can be broken up into several buffers. Each buffer in memory is accessed by the FSI through the use of a descriptor. Descriptors may be thought of as commands, commands to receive or transmit frame data, or in the case of performing a memory to memory move, a DMA data command.

Descriptor rings are conveniently organized ring structures that are collections of descriptors and commands. A descriptor ring may be conveniently thought of as defining a DMA channel. Commands that concern setting up and manipulating descriptor rings are discussed in detail in **Section 9 Commands and Indications**. There are two receive descriptor rings (4 and 5) that contain only receive buffer descriptors and four transmit descriptor rings (0–3) that can contain general commands and transmit buffer descriptors.

In addition, two command channels (6 and 7) are provided for the use of general commands, transmit frame commands, and DMA commands for memory to memory data moves, but not receive descriptor commands.

5.1.2.1 FDDI FRAME STRUCTURE. The basic data structure in FDDI is the frame. The FDDI frame structure that the user is most involved with is depicted in Figure 5-6.

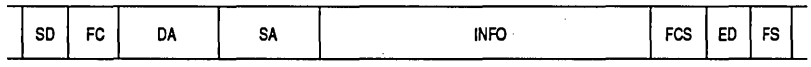


Figure 5-6. FDDI Frame Structure

The length of the various fields is as follows: frame control (FC) field is 1 byte, destination address (DA) field is 2 or 6 bytes, source address (SA) field is 2 or 6 bytes, INFO or data field is of variable length, and the frame check sequence (FCS) field, which contains the cyclic redundancy check (CRC), is 4 bytes in length. The end delimiter (ED) and frame status (FS) fields are together a minimum of two bytes. Interpacket idle symbols are not depicted. The MAC core automatically provides the FCS, ED, and FS fields as well as the

interpacket idle stream. The user must supply the packet request header (PR1, PR2, PR3), FC, DA, SA, and INFO fields. Bridge applications have the option of supplying the CRC for the FCS field and suppressing the CRC generation by the MAC for data integrity.

Within system memory, the user data fields might look like those shown in Figure 5-7. Two examples are shown—frame 1, in which the entire frame is contained in one buffer, and frame 2, in which the frame is segmented into three buffers.

5.1.2.2 EXAMPLE MEMORY ORGANIZATION. The three bytes of packet header are shown in Figure 5-7 as they contain control information for the MAC. These must be supplied by the user and are fully described in **Section 10 IFDDI as an FSI.** The FCS field is shown in one of the data buffers. This field is usually generated by the MAC core, and the user does not have to account for it in the data buffer. However, in the case of a bridge implementation, the user will normally receive the FCS field and retransmit the data frame without recomputing the FCS. Only 48-bit addresses are shown here.

NOTE

The FSI core does not interpret any of the data fields.

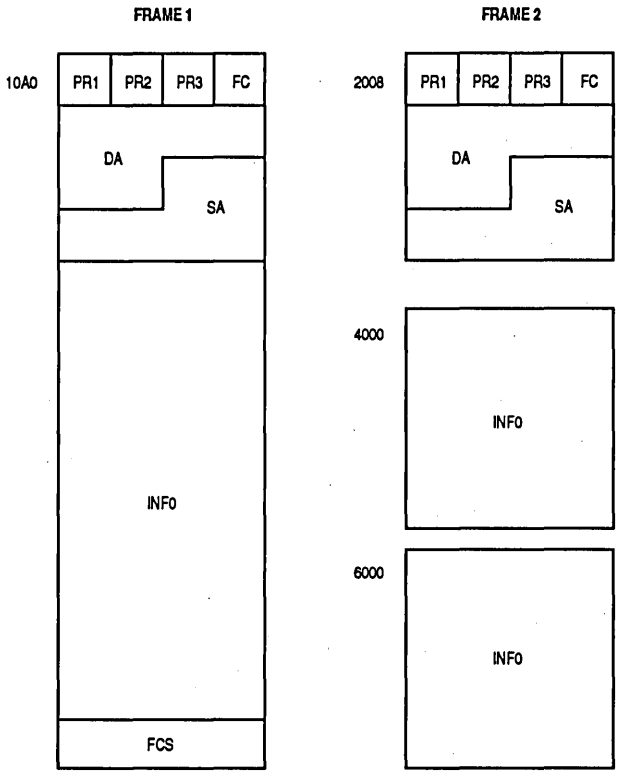


Figure 5-7. Example Memory Data Organization

In the data transmit case, the data buffers may be aligned to any byte boundary; furthermore, each data buffer may contain an arbitrary number of data bytes.

In the data receive case, the data buffers must be aligned to a 32- or 64-bit address, depending on whether the user has implemented the 32- or 64-bit bus interface. The length of the receive buffer is specified by the receive buffer length register (see **Section 7 Register Description**) or by the buffer length field of the receive buffer descriptor.

Some applications may find the receive data buffer structure insufficiently granular. For example, systems set up to use single receive buffers for maximum length FDDI frames will need to set the receive buffer length to 8 Kbytes. Note that it is possible to set the receive buffer descriptor pointers to buffers at 4.5-Kbyte increments, but host software should verify that actual frame length did not exceed pointer spacing since the FSI core will not check for the user in this case.

5.1.3 Descriptors

The data structure used to convey information about the memory data buffers to the FSI core is called descriptors. Descriptors are a type of 8-byte wide command to the FSI core

to transmit data, receive data, transfer data, or to issue commands to the FSI core. Descriptors describe each data buffer. There are three types of descriptors:

1. TRANSMIT BUFFER DESCRIPTOR Command (see **9.2.2.1 TRANSMIT BUFFER DESCRIPTOR Command**)
2. DMA BUFFER DESCRIPTOR Command (see **9.2.2.3 DMA BUFFER DESCRIPTOR Command**)
3. RECEIVE OR DESTINATION BUFFER DESCRIPTOR Command (see **9.2.2.10 RECEIVE BUFFER DESCRIPTOR** and **9.2.2.5 DESTINATION BUFFER DESCRIPTOR**, respectively).

The descriptors contain the address of the data buffer and the size of the data buffer.

Each descriptor contains an OWN bit to indicate whether the FSI core or the host "owns"—that is, can use the descriptor and its associated data buffer. The FIRST and LAST bits contain additional control information. When a frame exists in one contiguous data block, both the FIRST (F) and LAST (L) bits are set. Otherwise, the FIRST bit indicates the first data block of a frame, and the LAST bit indicates the last data block. Intermediate data blocks have both the FIRST and LAST bits reset.

For data transmission, the frame data may be described by up to 16 descriptors. A frame that is received or a data structure that is the destination of a memory to memory transfer may require the use of an indefinite number of descriptors and buffers. Thus, a very large receive frame may be characterized by a large number of small buffers.

Descriptors can also be used for ring definition or for executing commands through a transmit ring.

5.1.4 Descriptor Rings

The descriptor/commands may be provided to the IFDDI individually via the FSI command register (CMR). While this approach may be useful for diagnostics, it is inefficient for practical use. To facilitate the IFDDI's access to the memory data, the descriptors may be grouped in rings called descriptor rings, which are circular arrays in user memory. Defining a descriptor ring is logically equivalent to defining a data DMA channel.

The user can program four transmit descriptor rings and two receive descriptor rings. The transmit descriptor rings are used for queuing frames for transmission, for scheduling commands to be executed by the FSI core, and for memory to memory DMA data transfers. The receive descriptor rings contain only buffer descriptors that describe the data buffers for received data. Figure 5-8 shows a small descriptor ring that might well describe the two frames shown in Figure 5-7.

In Figure 5-8 there are two frames queued. The first frame is contained in one data buffer, indicated by having both the F and L bits set in the descriptor. The second frame is in three buffers and requires three descriptors. The first descriptor has the F-bit set, the last descriptor has the L-bit set. Not shown is that all the valid descriptors for these frames will also have the OWN bit set to indicate to the FSI core which descriptors are valid.

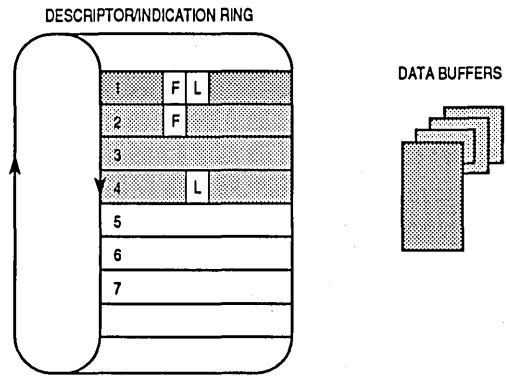


Figure 5-8. Example Descriptor Ring

The FSI core has a ring buffer descriptor FIFO and a data FIFO allocated in FSI core internal memory for each descriptor ring. Valid entries in the external ring buffer descriptors are first transferred to the internal ring buffer descriptor FIFO until either the internal ring buffer descriptor FIFO is full (20 entries) or until no other valid external ring buffer descriptors exist. The internal copy of the descriptor ring reduces bus accesses, especially during critical data transfers as illustrated in Figure 5-9.

The user must be aware that, due to the nature of the ring structure, an invalid entry (one with the OWN bit reset) must be provided following the valid entries to prevent the FSI core from wrapping around and re-reading already copied descriptors. To illustrate further, a transmit ring designed with an RML of four descriptors in which all four descriptors are valid will be repeatedly read by the FSI, resulting in multiple copies of the ring. Since the FSI operates on its internal snapshot, more frames will be sent than desired.

The IFDDI executes commands sequentially from the internal transmit ring buffer descriptor FIFO. An internal state machine for each descriptor ring handles the pointers to the current command and keeps the state of the descriptor ring.

The four transmit descriptor rings have a fixed priority between them. Transmit descriptor ring 0 always has the highest priority; transmit descriptor ring 1 has the next highest priority, etc. Placement of transmit frames into the proper ring is the responsibility of the host processor and depends on the frame's type and priority. For example, if synchronous frames are to be transmitted properly, they must be placed into transmit descriptor ring 0. Note that, if frames are placed into a higher priority transmit descriptor ring while a lower priority transmit descriptor ring is being serviced, the frames in the higher priority transmit descriptor ring are serviced as soon as they meet the FSI core requirements for transmission.

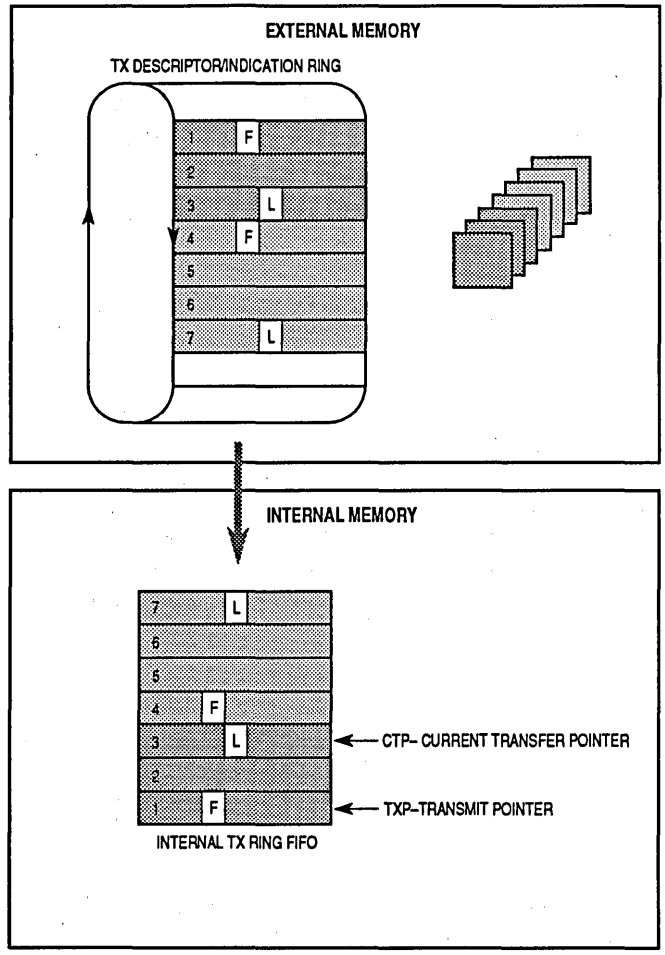


Figure 5-9. Internal Descriptor FIFO

When observing accesses to a ring, it is possible for more data and descriptor accesses to occur than might be expected. For example, a transmit descriptor ring containing TRANSMIT BUFFER commands, followed by a STOP command for the same ring, followed by additional TRANSMIT BUFFER commands will likely have some TRANSMIT BUFFER commands following the STOP command, along with some of their associated data, transferred to the FSI core prior to the STOP command being executed. However, the STOP command is properly executed in order, and none of the TRANSMIT BUFFER commands following the STOP command are executed until the ring is restarted.

After the required operation is complete, the FSI core writes an indication to the address pointed to by the ring write pointer. This write pointer may point to the same location as

the original buffer descriptor (writing over the original buffer descriptor) or to another location. The indication is written with the OWN bit reset.

5.1.5 Destination Rings

Each of the four transmit rings and the two command channels may have associated destination rings. The destination rings define the destination buffers for memory to memory DMA data transfers.

To perform memory to memory DMA data transfers, a destination ring must be defined by using a SET DESTINATION RING command for each destination ring in addition to the definition of the normal (source) ring. The DMA data transfer may occur between the memory systems on port A and B or may occur from memory to memory on either port. Figure 5-10(a) shows a data transfer between ports, and Figure 5-10(b) shows the transfer in the memory on one port.

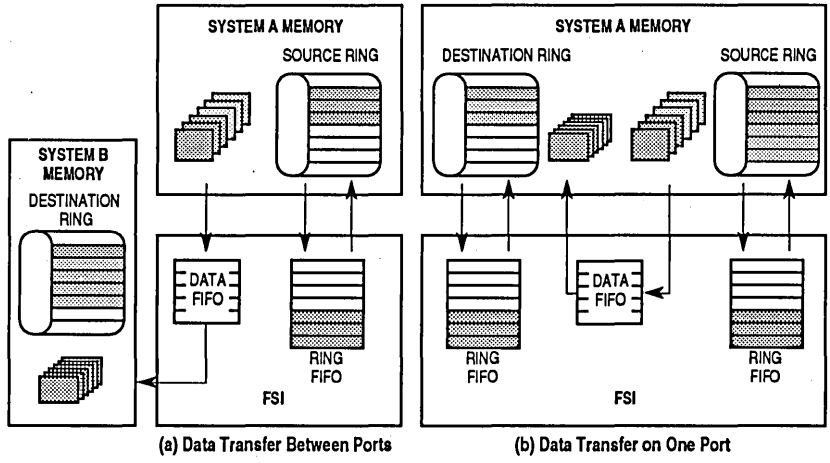


Figure 5-10. Source and Destination Descriptor Rings

Although Figure 5-10(a) shows the destination descriptor ring as existing in the same memory as the data buffers, it could have been defined and could exist in the system A memory, and only the data accesses will occur over the port to system B memory. The data buffers at the destination do not have to mirror the source. The DMA data transfer may be used to compact or to scatter data.

5.1.6 Buffer Descriptor Ring States

Since data buffer transmission and reception are dependent upon the buffer descriptor ring states, a short description of the buffer descriptor ring state machine operation is given in the following subsections.

5.1.6.1 STATE DESCRIPTIONS. The state descriptions are as follows:

Not Defined Internal	The ring is not defined. No action is taken by the FSI core, and no memory space is reserved for this ring.
Ready	There is at least one entry in this ring that has not yet been read by the FSI core or when the FSI core thinks that it can read an entry as when a DEFINE RING command is given even though no entries may yet be valid.
Empty	The ring is empty when the FSI core decides that all the ring entries have been read (i.e., the first entry with the OWN bit reset has been read by the FSI). Note, however, that there is at least one entry that has not yet been executed, (either data is still being transmitted or a command is still being executed).
Stopped	No other action is taken by the FSI core on the entries in the ring or the data that goes with the entries, even if this data is inside the FSI core. In the stopped state, there are no remaining indications inside the FSI core unless the ring stopped itself.
Complete	The ring is empty. All the entries have been executed by the FSI core, and all the indications have been written back to the ring.

NOTE

Only when the ring is in the complete or stopped state can the ring be redefined.

5.1.6.2 STATE TRANSITIONS. The following transitions are used in the ring state machine (see Figure 5-11):

A. The ring is defined and enabled using the DEFINE RING command.
B. The ring is defined by the DEFINE RING command but is not enabled.
C. All valid entries have been read by the FSI core.
D. A ring ready control access (access to the FCR) has been done by the host.
E. All indications have been written back to the ring.
F. The RING STOP command has been executed.
G. A parity or operational error has been detected during a descriptor read.
H. The RING RESET command has been executed.

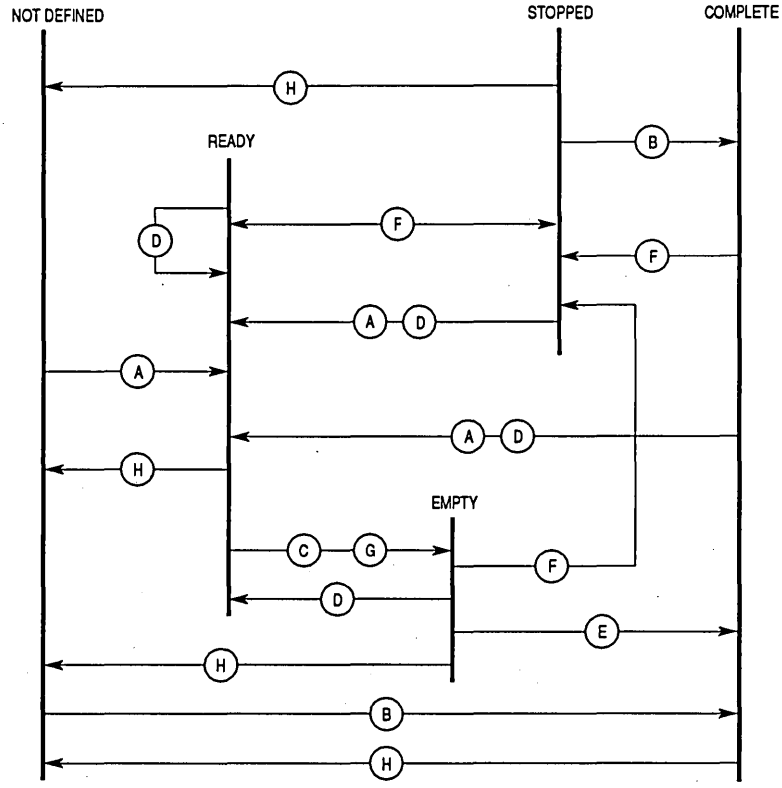


Figure 5-11. Descriptor Ring States

5.1.6.3 TRANSMISSION OPERATION. During the transmission process, buffer descriptors and then the data identified by the buffer descriptors are moved into the FSI core internal memory. When the appropriate conditions occur—for example, the watermark limit or an end of frame is detected—the frame is then moved into the FSI core hardware transmit FIFO and finally presented to the MAC core. This progressive movement is controlled by hardware signals from the MAC and the FSI core transmission logic, the IFDDI main controller, and the FSI port control working with the system bus control logic.

5.1.6.3.1 Normal Transmission Process. The transmit process starts when the host processor has accomplished the preparation of a frame or a number of frames in system memory and has informed the FSI that the transmit ring is ready (see Figure 5-12). The FSI core will then read a number of descriptors from the ring into its internal memory (1) and will start to transfer the data buffers specified by these descriptors from the system memory into its internal memory (2). When either enough data has been transferred in to reach the watermark of the FIFO or an entire frame is in the internal data FIFO, frame transmission will begin (3). When the frame has been transmitted, the indication is

generated and placed on top of the last buffer descriptor of the frame. The last step of the transmit process is to write the indications of the frame to the system memory (4). At this point, all frame descriptors will be cleared from the internal memory.

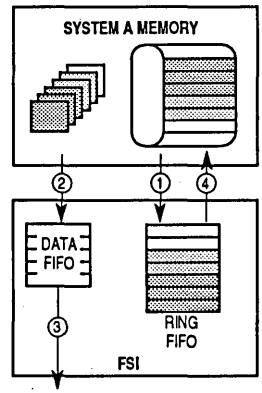


Figure 5-12. Normal Transmission Operation

5.1.6.3.2 Endless Transmission. Figure 5-13 illustrates two examples that provide for the endless transmission of frame data by the FSI core. In the first example, a number of different frames (four shown here) can be endlessly transmitted. In the second example, a single frame is endlessly transmitted. The latter example may be of use when the IFDDI is to transmit directed beacon frames until informed by SMT to stop such transmissions.

The ring read pointer used for the transmit commands must be different from the ring write pointer for the indications so that the indications will not invalidate the transmit commands.

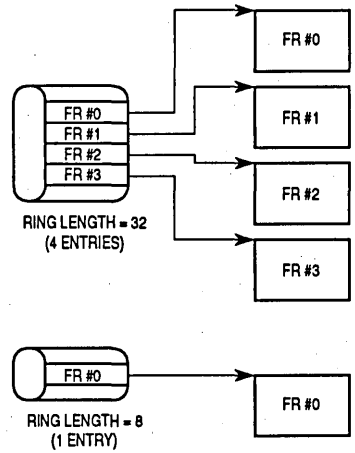


Figure 5-13. Endless Repeat of Data Frame Transmission

5.1.6.4 RECEPTION OPERATION. Once the receiver is enabled, the MAC interface transfers the received data into one of the internal receive data FIFOs according to the FC of the received data frame. When receive buffer descriptors exist (OWN bit is set), the FSI core reads them from the receive ring. If the receive ring is defined, the FSI core tries to always maintain a number of buffer descriptors inside the FSI core internal ring FIFO to reduce the time between reception of a frame into the FSI internal memory and when the FSI begins to transfer it to external memory. The FSI core begins to transfer a frame to external memory when either the full frame exists in the FSI core internal memory or the receive FIFO watermark has been reached.

5.1.6.4.1 Normal Reception Process. The receive process shown in Figure 5-14 starts with the reception of data from the MAC into one of the receive channel's internal data FIFOs (1). When the frame is being received, a frame descriptor entry is generated by the FSI core and stored inside the internal intermediate command FIFO of the receive channel (2).

When the ring of free buffers is ready in the system memory, the FSI core will read a descriptor from the ring into the internal ring FIFO (3). When a buffer descriptor exists inside the FSI core, receive data is output into the system memory (4). When one receive buffer is filled up, the indication is generated by the FSI core and placed in the descriptor of this buffer. The last step of the receive process is to write the indications of the buffer to the system memory (5). Note, however, that the internal frame descriptor entry is cleared from the intermediate command FIFO after the entire frame has been transferred.

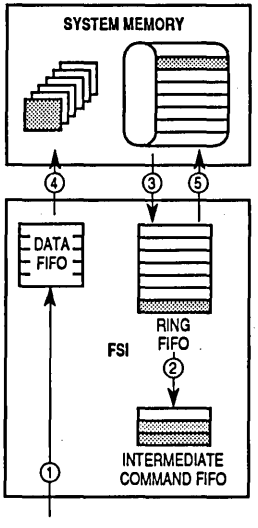


Figure 5-14. Normal Reception by Ring Operation

5.1.6.4.2 Reception Options. During the reception process, frames are first moved from the MAC core into the FSI's hardware FIFO, next into FSI core internal memory, and then

into the system memory external to the IFDDI. This progressive movement is controlled by hardware signals from the MAC core and the FSI core reception logic, the IFDDI main controller, the FSI port control logic, and the external system bus control logic.

Received frames are checked for their type using the FC field. The frame may be directed to one of two receive data FIFOs according to each frame's FC. This sorting is based on the values specified in the two receive frame type registers (RFRs), which are programmed by the user. If the received frame has its respective type bit cleared inside both of these registers, the frame will not be received, and the internal RABORT signal is generated to the MAC to abort reception.

The MAC core interface receiver logic has the following options of operation:

a. Normal operational mode—receive only good frames that are directed to this station. Bad frames (frames discarded by the MAC core) will not normally be transferred to the system. Only when more than one watermark's worth of information has been transferred into the FSI core internal memory will a frame go to the next step, being transferred to the FSI core system memory with the appropriate indication in the bad frame's last buffer descriptor. If a receive frame is aborted during reception and the amount of data received is less than one watermark, then the frame is discarded from the FSI core internal memory, and its receive buffer descriptor and data buffer are reused for another frame.

b. Receive all the data transferred from the MAC core to the FSI core—includes partial frames, frames with errors, etc., as defined by MAC programming.

c. Receive one buffer mode—this special mode of MAC core interface operation receives only the first buffer of each frame. This mode may be very useful in network analyzers when the MAC is programmed to receive all the frames (promiscuous mode). In this mode, only a portion of each frame is received starting at the beginning of the frame. This portion is defined by the length of the receive data buffer in the appropriate ring as specified by the FC of the frame being received. Trailing flags and a complete frame status indication, including the total frame length and the CRC status, are generated by the FSI core and put into the frame's indication.

d. Split header mode—this mode provides for the extraction and separation of the beginning of a frame from the remainder. The initial portion is called the header, and the remainder of the frame is referred to as the data portion. When split mode is used, the header portion is always directed to ring 4, and the data portion is directed according to the setting of the RFRs.

In addition to the data reception function, the FSI core may be programmed to provide an indication of the following events:

a. FRAME SENT BY THIS STATION. This indication will include flags (E, A, C, and any others) as they are received at the end of the frame. This indication of frames sent by this station is directed to one of the receive descriptor rings according to each frame's FC field.

b. TOKEN CYCLE END. This indication is issued by the MAC core when token, claim, or beacon frames are received and the FSI core has accomplished at least one frame transmission during the current token cycle. The TOKEN CYCLE END indication is directed to one of the rings by the definition of the TE bit in the RFR.

The indication for these events is written into a receive buffer descriptor/indication ring in one of the two receive rings. Note that the receive data buffer described by this receive buffer descriptor entry will not be used. The FSI core leaves the buffer pointer field unchanged to allow reuse of the buffer.

5.1.7 Local Memory Operation

The local memory may be connected to any port of the IFDDI. Logically, the local memory appears as an expansion of the internal FSI core memory and is transparent to the host processor. The FSI core continues to support all the data structures and the entire set of transmit and receive rings in the system memory (four transmit rings and two receive rings). The user may choose the ring mode for each one of the rings during ring definition (by using the DEFINE RING command) to be one of the following:

a. Normal Mode—This is the mode of operation when the local memory is not used as a temporary storage and frames pass only through the FSI core internal memory FIFOs.

b. Local Memory Mode—This is a mode of operation when local memory is used as an expansion to the FSI core internal memory.

Some possible data flow configurations with local memory are shown in Figure 5-15. In this figure, two channels are defined to operate in a normal mode (1 and 5), and two other channels are operating in a local memory mode (3 and 4).

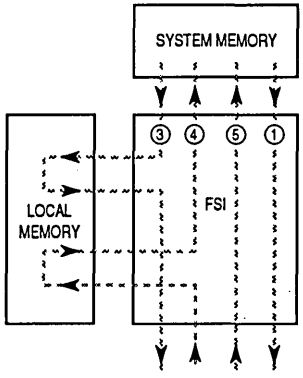


Figure 5-15. Data Flow Configurations

Only in the normal transmit operation are all the commands allowed to be placed inside the transmit (command) rings. These commands will then be executed sequentially. In the local memory transmit mode, only transmit commands can be placed into the FIFOs. Transmit command execution is complete when the indication has been written back to the ring. For transmission, there are two options for the indications. One method is to provide the indication when the frame has been transmitted to the network (or aborted for some reason). The second method allows for the indication to be provided when the frame has been completely transferred to the local memory. In the latter case, an indication will

not be given when the frame has been successfully or unsuccessfully transmitted to the network. Frame reception is complete when the frame data and the associated indication(s) have been written to the host memory.

5.1.7.1 LOCAL MEMORY ASSIGNMENT. The local memory is considered to be the IFDDI private memory. Therefore, it should be able to meet the FSI core requirements for access. The FSI core sees the local memory as six, large, cyclic buffers. Each area is assigned to one FSI core channel that operates in local memory mode. An example of channel assignment into the local memory is shown in Figure 5-16.

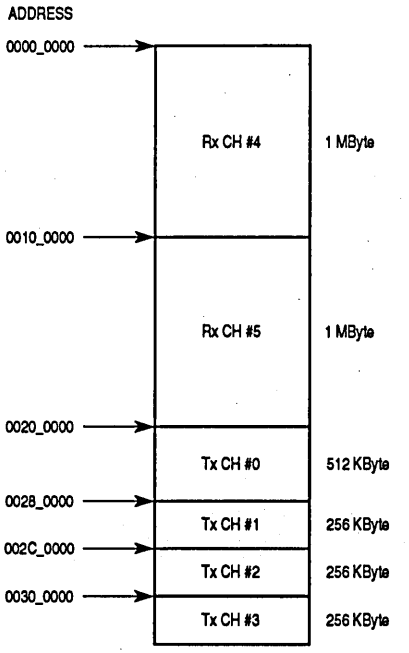


Figure 5-16. Example Local Memory Map

The starting address in local memory for each channel is defined separately using the SET LOCAL MEMORY SPACE command. The channel's assigned area space may vary from 8 Kbyte to 1 Mbyte and is specified for each channel by the LML parameter in the PER. Figure 5-16 shows a method of assigning addresses to each channel's area. The technique assigns the large areas first, starting at the bottom of memory, followed by equal or smaller areas. This technique ensures that successive areas are properly aligned and that the memory space is efficiently used.

In each area space, the frames are written one after another without any gaps between them. In the case of multibuffer transmit frames, the data buffers are assembled and realigned by the FSI core such that the frame is stored as a continuous stream of bytes starting on 32-bit word boundaries. The FSI core keeps track of how many frames exist

inside each area. The maximum number of transmit frames waiting to be transmitted inside one local memory area is programmable from 1 to 128. When the programmed limit is reached, the FSI core will stop transferring frames for this channel from the system memory to the local memory. The transfer will continue when the number of frames has decremented below the limit (after a frame has been transmitted).

The maximum number of receive frames that can be waiting inside a receive area is also programmable from 1 to 128 via the limit register (LMT). The FSI core will stop receiving frames for a receive channel when the number of frames waiting to be transferred to the system memory in the local memory area of this channel reaches the programmed limit. Note that the IFDDI considers the local memory to be full based on the number of frames in the local memory space, not the number of bytes occupied by the frames in the local memory space. Since the FSI core treats each memory space for a channel as a cyclical memory buffer, the user can avoid wrapping memory and possibly corrupting data by setting the maximum number of frames multiplied by the maximum expected frame size to be less than the assigned memory area for the FIFO in local memory.

5.1.7.2 TRANSMISSION PROCESS USING LOCAL MEMORY. As in normal mode, the transmit process starts when the host processor has finished preparing a frame or a number of frames in system memory and informed the FSI core that the transmit ring is ready (see Figure 5-17). The FSI core will then read a number of descriptors from the ring into its internal memory (1) and will start to transfer the data buffers specified by these descriptors from the system memory into the local memory (2 and 3). This transfer is done in quantities of 256 bytes—i.e., 256 bytes of the frame are transferred from the system memory to the internal FSI core memory through Port_In and then from the internal memory to the local memory through the Port_Out. Each transfer will occur when its port is available (i.e., not occupied by some other activity). Therefore, one port will not influence the other port because of its speed or latency. After the current frame has been transferred to the local memory, the FSI core will start to transfer the highest priority frame available.

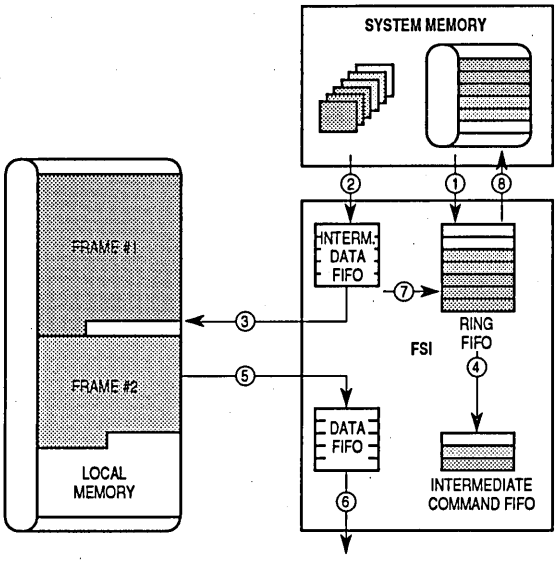


Figure 5-17. Transmit Process

When the entire frame has been transferred to the local memory, the FSI core generates an intermediate transmit command entry and stores it in its internal memory (4). This entry includes the length of the frame and the start address in local memory. It also includes the number of buffers in the original frame and the internal memory address of the last descriptor of the frame in order to put the indication there after frame transmission to the FDDI network. The intermediate command entry is used by the FSI core to transmit the frame from the local memory to the network. It is transparent to the user; however, while calculating the amount of internal memory required by the application, the user must consider that one such entry (8 bytes) will be generated and stored in the FSI core internal memory for each frame in the local memory. When at least one whole frame exists inside the local memory, the FSI core will start to transfer the first available frame from the highest priority channel to the internal data FIFO of this channel (5) until it satisfies the conditions to transmit this frame (either enough data has been transferred in to reach the watermark of the FIFO or an entire frame is in the internal data FIFO). The internal data FIFO treatment algorithm is the same as in normal mode. The user should realize, however, that because of the nature of the local memory (which is essentially the FSI core private memory), the watermarks may be programmed by the user to be minimal (64 bytes). When the frame has been transmitted (6), the indication is generated and placed on top of the last buffer descriptor of the frame (7). The next and last step of the transmit process is to write the indications of the frame to the system memory (8). At this point, all frame descriptors will be cleared from the internal memory.

When using the local memory operational mode, most of the 8 Kbyte internal memory might be used for descriptors, intermediate commands, and a relatively small portion for data because of the small watermarks required with local memory. When indications are

to be written after frame transfer to local memory, less memory is used than when the indications are held until transmission of the frame.

5.1.7.3 RECEPTION PROCESS USING LOCAL MEMORY. The receive process shown in Figure 5-18 starts with the reception of data from the MAC core into one of the receive channel's internal data FIFOs (1) in the same manner as it is accomplished in the normal mode operation. The data will then be written into the local memory area assigned to the receive channel (2). When the entire frame has been received, a frame descriptor entry is generated by the FSI core and stored inside the internal intermediate command FIFO of the receive channel (3).

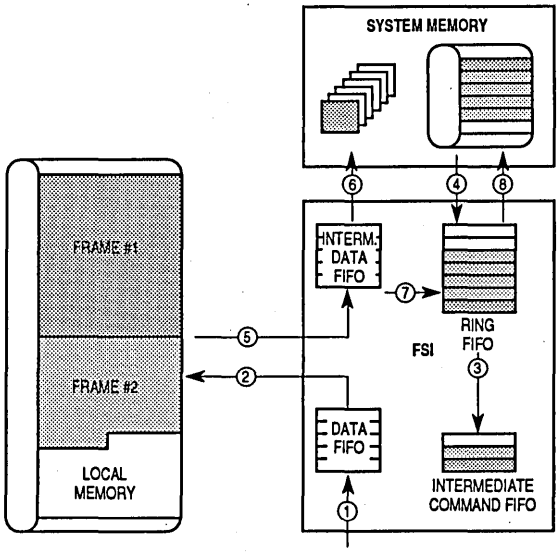


Figure 5-18. Receive Process

When the ring of free buffers is ready in the system memory, the FSI core will read a descriptor from the ring into the internal ring FIFO (4). When a buffer descriptor exists inside the FSI core, the local memory to system transfer is enabled. This transfer is performed through the intermediate data FIFO in quantities of 256 bytes (5 and 6). When one receive buffer is filled up, the indication is generated by the FSI core and placed in the descriptor of this buffer (7). The last step of the receive process is to write the indications of the buffer to the system memory. Note, however, that the internal frame descriptor entry is cleared from the intermediate command FIFO after the entire frame has been transferred.

5.1.8 Port to Port DMA Operation

Any transmit channel not using local memory (up to a total of four channels) and either command channel (one for each port) may be used to transfer data from one system to

another (e.g., from a host processor to a node processor). It is important to note that DMA operation is not a special operational mode of the channel. Therefore, the same channel may be used for both transmit and DMA operations (see Figure 5-19). However, when mixing transmit and DMA commands on the same channel, the following limitation applies: either single mode should be used for the transmit commands (bit 45 set) or a MAKE INDICATION command should be inserted before each DMA command that follows a transmit command.

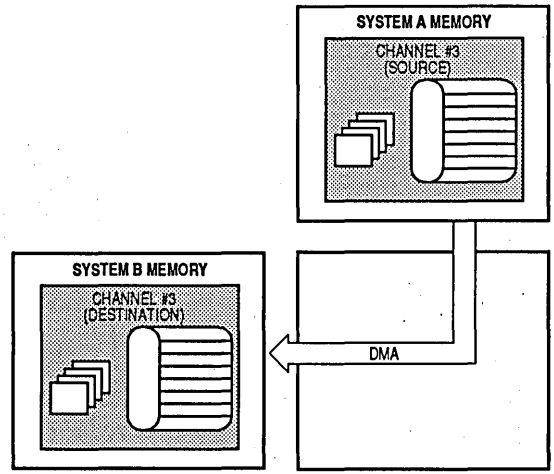


Figure 5-19. Port to Port DMA

An additional property of transmit and command channels, called "destination ring," is defined to support DMA operation. The normal ring of a channel is called the "source ring."

The destination ring may be defined for every transmit channel and every command channel and is used to hold descriptors of DMA destination buffers.

The DMA operation is executed by the IFDDI as a result of a DMA command given through a source ring or through the FSI command register (CMR). The DMA command format is similar to the transmit command and may be seen as a transmit operation directed to a destination memory space specified by the destination ring descriptors.

5.1.8.1 DESTINATION RINGS. A destination ring has the same properties as a source ring:

1. A destination ring may be assigned to any memory space (A or B).
2. A destination ring's data buffers may be placed in any memory space (A or B).
3. A destination ring may have its own ring maximum length, which may be different from the source ring's ring maximum length.

4. Parity may be checked when descriptors are read from the ring. The ring states of the destination ring are simplified as follows: this ring may be "ready" (e.g., there are descriptors inside the ring) or "not ready" (e.g., the FSI core did not find a valid descriptor in this ring). To transfer the destination ring to the "ready" state, the "destination ring ready" control access should be performed by the processor (similar to the "ring ready" control access for source rings).

5.1.8.2 DMA OPERATION. DMA operation flow for a transmit channel (see Figure 5-20) includes the following:

1. When the source ring is ready, a number of descriptors will be read by the FSI core into its ring FIFO (this is the same as in normal operation).
2. When descriptors exist inside the ring FIFO, the FSI core will execute commands specified by these descriptors. If a current descriptor is a first descriptor of the DMA, the FSI core will read one destination buffer descriptor from the destination ring.

3/4. The FSI core will transfer the data from source memory space to a destination memory space through the internal data FIFO of the channel. This transfer is accomplished in quantities of 256 bytes.

5. Once one destination buffer has been used, its descriptor is turned into an indication.
6. This indication will be written back to the Destination Ring. If there is a need for another destination buffer, its descriptor will be read from the Destination Ring and the operation will continue.
7. When the entire frame has been transferred from the source to the destination, the indication is made and placed inside the last source descriptor.
8. All indications waiting inside the ring FIFO are written back to the source ring.

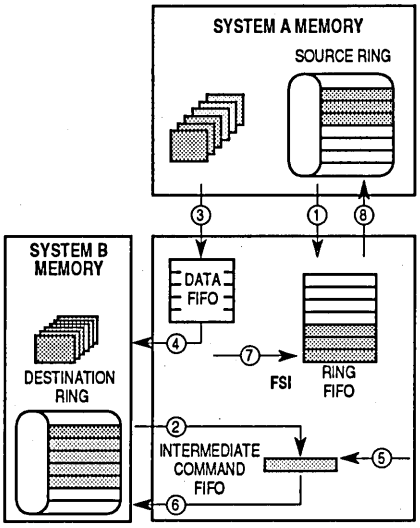


Figure 5-20. DMA Operation for a Transmit Channel

DMA operation using a command channel (see Figure 5-21) is similar:

1. The processor on the source side issues a DMA command. As in the operation of the transmit command from the command register, the DMA command given through the command register should have only one descriptor; therefore, the maximum amount of data transferred by the command channel by one command is 8 Kbytes.
2. The FSI core will read one destination buffer descriptor from the destination ring of this command channel.

3/4. The FSI core will transfer the data from a source memory space to a destination memory space through the internal data FIFO of the channel. This transfer is accomplished in quantities of 256 bytes.

5. Once one destination buffer has been used, its descriptor is turned into an indication.
6. This indication will be written back to the destination ring. If there is a need for another destination buffer, its descriptor will be read from the destination ring, and the operation will continue.
7. When the entire frame has been transferred from the source to the destination, the indication is created and placed inside the command register.
8. The CDN status bit in FSI status register 1 (SR1) is set, causing an interrupt to the source processor (if enabled).

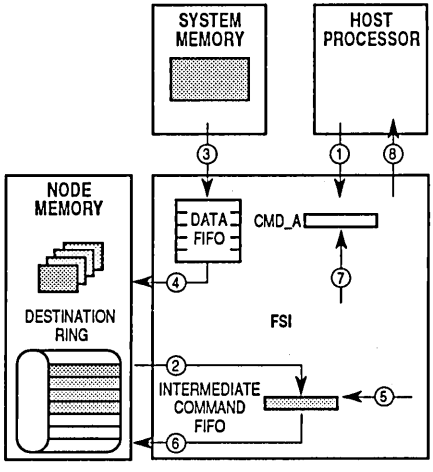


Figure 5-21. DMA Transfers Using Command Register

5.1.9 Double Buffer Mode

There are three types of data transfer tasks that the IFDDI performs in quantities of 256 bytes:

1. System memory to local memory transfers of transmit frames
2. Local memory to system memory transfers of receive frames
3. Port to port transfers of DMA frames

5.1.9.1 Double Buffer Mode Operation. When double buffer mode is disabled (the default after chip reset), the input and output operations of the transfers mentioned above are performed sequentially and do not overlap. First, a quantity of data is read from the source external memory into the IFDDI internal memory. Then, this data is transferred to the destination external memory. This process is repeated until the entire frame has been transferred.

When double buffer mode is enabled, the IFDDI will store up to two buffers of 256 bytes each in its internal memory. Therefore, while the first quantity of data is being transferred to the destination external memory, a second quantity can simultaneously be read into the IFDDI from the source external memory. This will result in greater throughput in the data transfer operations mentioned above (all of which are "off-line" as far as the FDDI ring is concerned) for most configurations.

The cost of the improved performance obtained in Double Buffer mode is the use of more of the IFDDI's internal memory for temporary storage of the data. The additional memory consists of a 256-byte buffer for each receive ring defined, and an additional 256-bute buffer for the transmit rings, totalling up to 768 bytes if both receive rings are defined. This is additional overhead that cannot be used for transmit or received data FIFOs.

The trade off of transfer speed vs. internal memory usage depends on the application. Therefore the use of Double Buffer mode is user-programmable as described below.

5.1.9.2 Double Buffer Mode Control. Bit 0 (zero) of the IFDDI Configuraiton register (ICR) enables the Double Buffer mode of the IFDDI. This bit may be set to configure the IFDDI to operated in Double Buffer mode. Note that this is a configuration bit and should not be modified during operation.

5.1.10 Header Splitting

An additional enhancement to the FSI core is the ability to split received frames into two parts (the length of the header is specified in the header length register (HLR)). The first part (the header) will always be directed to receive channel number 4, regardless of the channel selection decision based on the FC, and will use an entire receive buffer in this channel (see Figure 5-22). The indication for this buffer will have a unique encoding (S-bit set, see **9.2.2.11 RECEIVE FRAME NORMAL Indication and 1.2.2.12 RECEIVE ERROR Indication**) to differentiate this buffer from the normal buffers and will have both the FIRST and LAST bits set. Therefore, the FSI core will treat this first buffer as a whole frame. The remainder of the frame is directed to the receive channel selected by the FC of the frame and will take as many buffers as required, depending upon the buffer length programmed for the channel and the remaining frame length. The FSI core will treat the second part of the frame as a whole frame as well—i.e., the first buffer occupied by this part will have the FIRST bit set in its indication, and the last buffer will have the LAST bit set in the indication. All indications will have an identifier (S-bit set, see **9.2.2.11 RECEIVE FRAME NORMAL Indication and 1.2.2.12 RECEIVE ERROR Indication)** to notify the user that it is the remainder of a frame and not a complete frame.

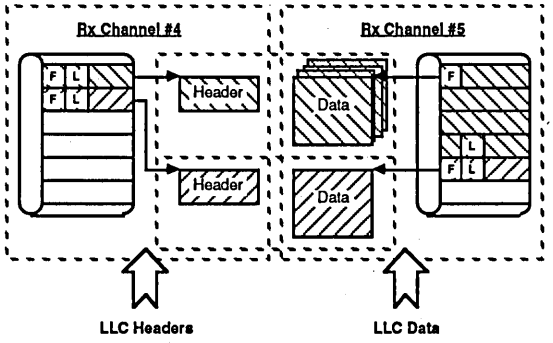


Figure 5-22. Header Splitting

Once the FSI core has been programmed to work in the split mode, the two possible submodes of operation are as follows:

a. The indication for the first buffer (the header buffer) is generated immediately after its reception has been completed.

b. The indication of the header buffer is delayed until the entire frame has been received. Note that when using this mode, the user should program receive channel 4 in so that it is used only for headers.

Due to the possibility that the frame may be aborted during its continued reception after the header buffer has already been received, a split mode data error indication may be generated for the second part of the frame.

Because the FSI core treats the header buffer as a small frame, the receive complete status bit (RXC4) will be set when the indication of this buffer is written back to the ring. This status bit in status register 1 (SR1) will cause an interrupt (if enabled), which may actually be considered as an early receive interrupt when operating in mode A.

5.2 CAMEL FUNCTIONAL OPERATION

The CAMEL block provides the combined functionality of the MAC (MC68838) and the ELM (MC68837) chips. The CAMEL registers may be accessed either directly through port A or indirectly through the FSI control register (FCR).

For more timing-specific information on CAMEL register access, see **Section 13 Electrical Characteristics.**

5.2.1 ELM Functional Operation

The ELM core provides facilities for CMT and link status indications as set forth in the ANSI FDDI SMT document. CMT defines the operation of PHY insertion and removal and the connection of PHY entities to the MAC entities. PCM, a subset of CMT, is the management of a physical connection between the PHY being managed and another PHY.

The logic blocks implementing these features are the line state machine (LSM), the link error monitor (LEM), the data stream generator, and the PCM. These logic blocks are discussed in this subsection.

5.2.1.1 LINE STATE MACHINE OPERATION. The LSM constantly monitors incoming aligned symbol pairs. The current symbol pair is encoded and compared to the encoded value of the previous symbol pair. The symbol pairs are counted until a line state is reached. Once a line state is reached, the counter is stopped, the new line state is stored, and the unknown line state (UNKN_LINE_ST) bit is reset to zero. Anytime a noise symbol is received, the line state is set to noise line state and the UNKN_LINE_ST bit is reset to zero. Upon receiving the first symbol pair that is different from the previous symbol pair, the sumbol pair counter is started and UNKN_LINE_ST is set (i.e. equal to 1) until conditions for a new line state are met.

The recognition of these line states is reported to the PCM, which uses this information for either insertion or removal of the station from the ring, ring recovery, or maintenance. The values of the Line State and the UNKN_LINE_ST bit can be read from the ELM_STATUS_A register. A change in the line state can be reported via the LS_MATCH interrupt bit.

The LSM is reset into the NOT_ACTIVE state with Line State = Noise State, UNKN_LINE_ST = 0, Symbol Pair Count = 0 and Previous Line State = Quiet Line State.

Table 5-1. LSM Line States

Line State Name	Condition
Noise Line State	Any Line State Not Defined
Active line State	JK Symbol Pair
Idle Line State 4	4 I-Symbols
Quiet Line State	16 Q-Symbols
Master Line State	8 Pairs of H/Q or Q/H Symbols
Idle Line State 16	16 I-Symbols
Halt Line State	16 H-Symbols

5.2.1.2 LINK ERROR MONITOR OPERATION. The LEM hardware consists of a detector, accumulator, and threshold element. The detector is a state machine that constantly monitors incoming symbol pairs on the receive data path. When link error events are detected, they are counted by the 8-bit link error event counter register. When the link error event counter register matches the count written to the link error event threshold register, the LE_CTR bit in the ELM interrupt event register is set.

5.2.1.3 DATA STREAM GENERATOR. The data stream generator uses a multiplexer to generate a symbol pair at the request of the PCM or the repeat filter, or by transmitting the symbol pair from TXDATx. When the PCM is in the MAINT state, the symbol sourced is defined by the state of the MAINT_LS field in the ELM control register B. The repeat filter and the PCM state machine can be turned off, but while operating, they generate symbol pairs according to their internal algorithms. For more information on the Data Stream Generator, see **Section 4.3.7 Data Stream Generator.**

Table 5-2. Data Stream Generator Output

LS_REQUEST(2–0)	RF_CNTRL(1–0)	DATA_STRM(9–0)
000	XX	Q-Symbol Pair
001	XX	I-Symbol Pair
010	XX	H-Symbol Pair
011	XX	M-Symbol Pair
100	XX	Q-Symbol Pair
101	XX	Q-Symbol Pair
110	00	Symbol Pair from TXDATx
110	01	I-Symbol Pair
110	10	H-Symbol Pair
110	11	I-Symbol Pair
111	XX	Q-Symbol Pair

X = Don't care

5.2.1.4 PHYSICAL CONNECTION MANAGEMENT. The ELM core implements CMT through a PCM state machine as specified in the ANSI FDDI SMT standard. The ELM also implements part of Configuration Management (CFM) via the Physical Connection Insertion (PCI) state machine.

CMT defines the rules that govern the allowable topologies in an FDDI ring. Fundamental to this task is the management of a connection between two PHYs in adjacent stations. It is the task of the PCM state machines in both stations to cooperate in forming a connection between the two PHYs within the rules established by CMT. The four types of PHY/PMD pairs, called ports, are as follows:

A Primary Ring In/Secondary Ring Out

B Secondary Ring In/Primary Ring Out

S Single Attachment Node

M Concentrator Attachment to an End Station

In addition, CMT defines the type of physical connection between two physical attachments to be determined by the PHY types at each end of the connection. The characteristics of a type of connection determine if that connection will be allowed, if the SMT will be notified of possible connection problems, and the connection mode that will be established. Table 5-3 from the ANSI FDDI SMT document lists the connection rules, and Figure 5-23 illustrates some possible station interconnections to graphically show the types of stations as implemented with concentrators and end stations.

Table 5-3. Connection Rules

		Other PHY			
		A	B	S	M
This PHY	A	V, N	V	V, N	V, P
	B	V	V,N	V, N	V, P
	S	V, N	V, N	V	V
	M	V	V	V	X,N

NOTE:
V = Indicates a valid connection
X = Indicates an illegal connection
N = Indicates that notification to SMT is required
P = Indicates, if active, prevent THRU in CFM and PHY B takes precedence

More details of FDDI network topology can be found in the ANSI FDDI SMT document.

5.2.1.4.1 PCM STATE MACHINE. CMT secures a PCM state machine deterministic ring topology, independent of the sequence of station power-up, etc., by allowing only a specific set of connection types. The primary purpose of PCM is to enforce these allowable connections. The PCM announces its attachment type to the remote PCM and listens for the type of attachment from the remote PCM. If they are compatible, the PCM accepts the connection and reports the type of connection to the station configurator. Once the connection type has been established, the two PCMs share in testing the pair of physical links between them. If this test is successful, the link can then be inserted into the ring. Note that PCM operates between two cooperating PHY entities to determine the viability of the link between them, regardless of the actual ring operation. This fact can be observed by reference to the upper pair of concentrators in Figure 5-23 in which the various PHYs are connected to an input on one ring and an output on the other ring. During power-up, the PHYs will determine the viability of the connection; once both connections are valid, multiplexing between the ELM and MAC cores (if any in the concentrator) will establish the topology of the ring or rings. Using the IFDDI, the multiplexing is done with software control over internal multiplexers.

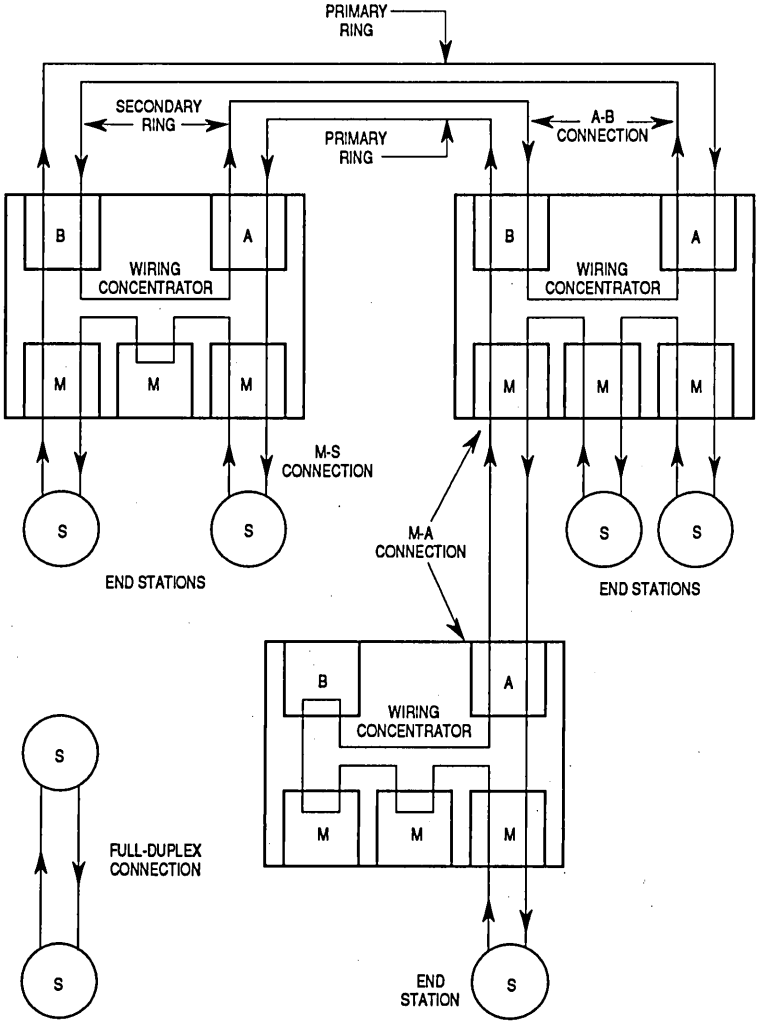


Figure 5-23. Sample FDDI PHY Connections

The architectural model for the PCM consists of ten states: OFF, BREAK, TRACE, CONNECT, NEXT, SIGNAL, JOIN, VERIFY, ACTIVE, and MAINT for both the ANSI standard and the hardware realization. The ELM PCM state machine accomplishes all of the transitions in the ANSI FDDI SMT standard, although transitions between some states are enabled by software in certain situations. Interrupts are provided to furnish indications when actions have been completed that require software to specify actions for continuing the PCM state transitions. It is also possible to accomplish non-ANSI-specified action by operating in the MAINT state. The following PCM machine descriptions, along with the ANSI FDDI SMT standard, can be used to produce ANSI-compliant SMT software.

5.2.1.4.2 BIT SIGNALING MECHANISM. The bit signaling protocol is implemented to reduce the software processing overhead and to allow enough flexibility to change the actual pseudo-code without affecting the silicon implementation.

When the PCM is in the OFF state, all parameter registers and configuration registers are loaded with the appropriate values. The transmit vector length register is written with the value *n*–1 (*n* = the number of bits to be transmitted). The transmit vector register is written with the bit pattern to be transmitted. PC_START is then written into the ELM control register B. The PCM then transitions through the BREAK, CONNECT, and NEXT states. It then transitions back and forth between the NEXT state and the SIGNAL state until all bits in the transmit vector register are transmitted. While the PCM is transmitting all the bits, it also receives the corresponding bits from the remote station and forms a receive vector that is stored in the receive vector register. When all bits are received for the transmitted bits, the PCM_CODE interrupt bit is set in ELM_INTR. The node processor can then read the receive vector register. Note that the PCM is still in the NEXT state.

If for any reason (other than PC_START) the PCM transitions to the BREAK state, then a PC_START has to be issued before the connection process can begin again. This allows the transmit vector length register and the transmit vector register to be reinitialized. Also, any transition to the BREAK state sets the PCM_BREAK interrupt bit and writes the reason for the transition in the ELM status register B BREAK_REASON field.

If the node processor wants to do a Link Confidence Test, it can do so by setting the PC_LOOP bits in ELM control register B. If the PCM is not in the NEXT state, or PC_SIGNALING is set, then setting the PC_LOOP bits will have no effect on the state machine. Normally, these bits should be set after the PCM_CODE interrupt bit is set. The node processor can set the station to do a transmit_PDR function or transmit_Idle function or a remote loopback function. If the LONG bit is not set in ELM control register B, the Link Confidence Test will last for LC_SHORT (a writable parameter) period of time, after which the PCM_CODE interrupt bit is set. If the LONG bit is set, then the ELM will be in Link Confidence Test mode continuously until the software interrupts it by giving one of the control commands (e.g., write XMIT_VECTOR, PC_Start, etc.). When PC_LOOP is set, the PCM sets the TDF flag internally.

After the Link Confidence Test has completed (e.g., after LC_SHORT or after Halt or Master Line State is received), the PCM_CODE interrupt bit is set. If the node processor decides to transmit more signaling bits, it should load the transmit vector length register with a new value of *n* and then load the transmit vector register with the bit pattern to be transmitted. The PCM again starts transmitting these bits and alternates between the NEXT and SIGNAL states until all bits have been transmitted, as indicated when the PCM_CODE interrupt bit is set again.

This sequence continues until all the bits have been transmitted and the node processor writes PC_JOIN in ELM_CNTL_B register. The PCM then leaves the NEXT state and enters the JOIN state. Setting PC_JOIN has no effect when the PCM is not in the NEXT state or when PCM_SIGNALING is set. However, if PC_JOIN is set when the Link Confidence Test is in progress, then the Link Confidence Test will be aborted, and the PCM JOIN state will be initiated.

5.2.1.4.3 NOISE DETECTION MECHANISM. The TNE timer in the PCM times the period between Idle Line State receptions. This timer is loaded with the noise time register parameter when the LSM leaves the Idle Line State. The TNE timer keeps counting noise until Idle Line State is again detected. If this timer expires while in the ACTIVE state, the PCM will break the link and transition to the BREAK state. In the ACTIVE state, the TNE timer starts counting noise only after the LSF bit is set. If PC_TRACE is received and the TNE timer expires in the same cycle, then the transition to the TRACE state is taken. This timer is ignored in all PCM states except the ACTIVE state.

5.2.1.4.4 NOISE IN MAINT STATE. If the NOISE_TIMER bit in ELM control register A is cleared, then the node processor can write to the TNE timer if the PCM is in MAINT state. If the NOISE_TIMER bit is set, the TNE timer is used in the MAINT state to time the noise as previously described. If the TNE timer expires, then the TNE_EXPIRED interrupt bit is set.

5.2.1.4.5 OPERATION IN TRACE STATE. If trace propagation (transition 88c) is detected in the ACTIVE state, then the TRACE_PROP interrupt bit is set. In the TRACE state, if PC_TRACE is received and a transition is made to the TRACE state, the station remains inserted, Master Line State is sourced on the TDATAx port, and no scrubbing is performed. If Master Line State is detected in the TRACE state, the TRACE_PROP interrupt bit is set. If Quiet Line State or Halt Line State is detected (transition 22a), then the SELF_TEST interrupt bit is set.

5.2.1.4.6 PHYSICAL CONNECTION INSERTION. The ELM core implements PCI to intelligently bring a new connection into a ring and to remove an existing connection from a ring. The PCI state machine works in conjunction with the PCM state machine to control the PRCDATx and TXDATx paths of the ELM.

There are three primary functions of the PCI state machine:

1. Provide a bypass path between TXDATx and PRCDATx.
2. Provide a scrubbing function upon the insertion and removal of a station from the ring.
3. Provide a direct data path between the fiber and the MAC core.

The operation of the PCI state machine depends on whether the CLASS_S bit in ELM control register B is set and whether the PCM state machine is in the MAINT state.

5.2.1.4.7 PCI OPERATION FOR NON-CLASS-S TYPE STATION. After a reset, the PCI state machine will be in the REMOVED state. If the station is not a class-S type, the ELM core will be in the bypass mode whereby data input on TXDATx is directly output on PRCDATx.

When the PCM state machine enters the ACTIVE state and asserts the SC_JOIN flag, the PCI state machine enters the INSERT_SCRUB state and I-symbol pairs are sourced on PRCDATx. At the same time, the PCM state machine causes I-symbols to be output on TDATAx.

The PCI state machine remains in the INSERT_SCRUB state for T_SCRUB length of time, after which it enters the INSERTED state. Upon entering the INSERTED state, the PCM_ENABLED interrupt bit is asserted. In this state, a direct path exists from TXDATx to TDATAx.

If the connection is broken and the PCM state machine enters the BREAK state, the PCI state machine enters the REMOVE_SCRUB state, and I-symbol pairs are sourced on PRCDATx. Because the PCM state machine is in the BREAK state, Q-symbols are sourced on TDATAx. While scrubbing is being performed, the PCM state machine will not restart the connection process. The PCI state machine remains in the REMOVE_SCRUB state for T_SCRUB length of time and then enters the REMOVED state.

5.2.1.4.8 PCI OPERATION FOR CLASS-S TYPE STATION. For a class-S type station, the PCI operation is identical to the non-class-S type with one exception—whenever the PCI state machine would normally be in the REMOVED state, it will be in the INSERTED state. Thus, before entering INSERT_SCRUB or after leaving REMOVE_SCRUB, rather than putting the ELM in the bypass mode, PHY_INVALID is output on PRCDATx.

5.2.1.4.9 PCI OPERATION IN MAINT STATE. When the PCM state machine is in the MAINT state, the PCI state machine does not control the previously mentioned functions. In the MAINT state, all data paths are under the control of software via several control bits in ELM control register A. Software can also override the PCI functions when the PCM state machine is not in the MAINT state by setting the CONFIG_CNTRL bit in ELM control register B.

5.2.2 MAC-PHY Interface Functional Operation

The MAC will never generate a halt (H), quiet (Q), or violation (V) symbol nor signal a PHY_INVALID to the PHY layer (ELM). The response of the MAC is undefined when it is passed a code that is not listed in the data link code column of Table 5-5. This interface consists of two 10-bit unidirectional buses. The PRCDATx/MRCDATx bus is used to transfer a received symbol pair from the ELM to the MAC. TXDATx is used to transfer from the MAC to the ELM a symbol pair to be transmitted on the ring. These buses are similar in that: a) both can be decomposed into two independent halves of 5 bits, each representing one symbol, and b) both use the same encoding for symbols.

The most significant 5 bits on RCDATx/TXDATx, bits 9 to 5, correspond to the first symbol received or transmitted on the physical medium, where as the least significant 5 bits, 4 to 0, correspond to the second symbol of the symbol pair. The encoding is listed in Table 5-4.

This subsection is provided in the case of the MRCDATx/PRCDATx bus being used in conjunction with an external ELM. The MAC-PHY interface links the MAC core to the ELM core. These buses are synchronous with BYTCLK.

Table 5-4. PRCDATx/TXDATx Encoding

Symbol	Data Link Code	Symbol	Data Link Code
0	00000	D	01101
1	00001	E	01110
2	00010	F	01111
3	00011	H	10100
4	00100	I	10111
5	00101	J	11100
6	00110	K	10011
7	00111	Q	10000
8	01000	R	10001
9	01001	S	11001
A	01010	T	11101
B	01011	V	11000
C	01100	PHY Invalid	11111

5.2.3 CAM Interface Functional Operation

The CAM interface is a BYTCLK-synchronous interface used to connect the IFDDI to one or more CAMs or other frame-recognition logic for the purpose of detecting an arbitrary number of 48-bit individual or multicast (group) destination and/or source addresses. This interface consists of one output (DA), 2 inputs (MATCH, REJECT), and one bidirectional signal (LDADDR/TR_BR_FWD). The data for the CAM comes from the PRCDATx/MRCDATx bus before it enters the MAC.

5.2.3.1 CAM INTERFACE. The CAM interface presents frame information to a CAM for address comparison purposes one byte at a time from the ELM core over the PRCDATx lines at the same time the information is being furnished to the MAC core. Since the byte-wide data on PRCDATx is two data symbols, only the lower four bits of each symbol need be used; the upper bit of each symbol is always zero for data. Thus, the CAM uses PRCDAT8–PRCDAT5 and PRCDAT3–PRCDAT0.

If the presented information matches an address stored in the CAM, the $\overline{\text{MATCH}}$ line to the MAC block should be asserted by the CAM. LDADDR and DA are additional signals driven by the MAC block that indicate to external logic or a CAM the position of data on the PRCDATx lines relative to the frame. BYTCLK is used to clock the CAM interface and to latch the address into the CAM.

5.2.3.2 NORMAL (NONEXTENDED) MATCH MODE. Referring to Figure 5-24, the following sequence of events should occur between the MAC and a CAM when an address is presented on the PRCDATx lines and the MAC has EXT_DA_MATCH = 0 in MAC control register B:

1. LDADDR is pulsed for one BYTCLK cycle just before the first byte of both the DA and SA fields is given to the MAC and the CAM from the ELM. The DA signal from the MAC differentiates whether the current address is the DA or SA. LDADDR is negated by the MAC within one BYTCLK cycle after its assertion.
2. The first address byte of the SA or DA is valid on PRCDATx during the BYTCLK cycle following the assertion of LDADDR. This byte and the next five bytes of each address are then consecutively loaded into the CAM. One BYTCLK cycle is then allowed for the compare operation. The $\overline{\text{MATCH}}$ signal must then be asserted to the MAC before the eighth rising edge of BYTCLK after LDADDR has been negated to indicate that the address is recognized and that the MAC should copy the frame.
3. DA is asserted with the frame's FC field, which means that DA has the same functional timing as LDADDR, but remains asserted and valid up to the last byte of the destination address.

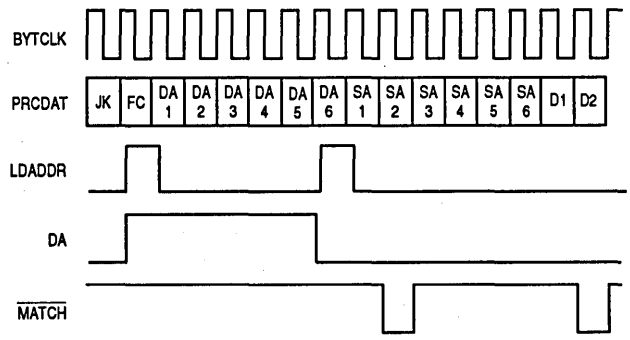


Figure 5-24. CAM Interface Signals (EXT_DA_MATCH = 0)

5.2.3.3 EXTENDED MATCH MODE. Figure 5-25 shows the sequence of events on the CAM interface when EXT_DA_MATCH is programmed to one (for delayed address matching), which also causes the LDADDR output signal to become $\overline{\text{TR\_BR\_FWD}}$, an additional input signal for bridge address matching logic. The $\overline{\text{TR\_BR\_FWD}}$ signal has the same timing as the $\overline{\text{MATCH}}$ signal and is used to specifically indicate an address match with the effect on the A and C frame status indicators as shown in Figure 5-25. Both the $\overline{\text{TR\_BR\_FWD}}$ and $\overline{\text{MATCH}}$ signal can be used on the same implementation. Either input can be asserted at any time from the second byte of the SA up to and including the fourth byte of the FCS. The $\overline{\text{MATCH}}$ signal has the effect of setting the A-bit and overrides the $\overline{\text{TR\_BR\_FWD}}$ input.

$\overline{\text{REJECT}}$ can be asserted at any time during frame reception to cause the currently received frame to be flushed.

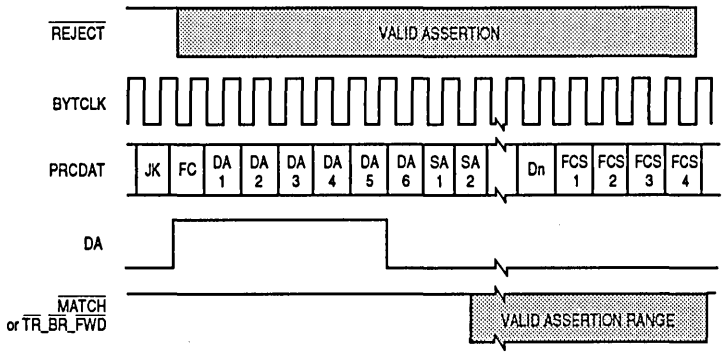


Figure 5-25. CAM Interface Signals (EXT_DA_MATCH = 1)

Figure 5-26 shows the sequence of events on the CAM interface when receiving a token frame for both normal and extended match mode. DA is asserted for one or two BYTCLK cycles when a token is received. Only a two BYTCLK assertion is shown.

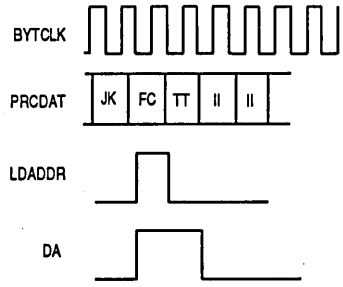


Figure 5-26. CAM Interface Timing (Receiving Token Frame for Normal and Extended Match Mode)

5.2.3.4 EXTENSIONS TO A AND C BIT HANDLING. The MAC_MODE_CTL bit in MAC control register B and the $\overline{\text{TR\_BR\_FWD}}$ input signal give the MAC extended capabilities for handling the A and C bit fields at the end of received frames (see Table 5-5). These extensions are particularly useful for certain bridging protocols.

Table 5-5. MAC A and C Bit Control

Inputs	Outputs			
	Ax	Cx		
	Always	Non-Abort	Abort (see Note 1)	
Function	A	C (Frame Copied)	C (MAC_MODE_CTL = 0, Frame Flushed)	C (MAC_MODE_CTL = 1, Frame Flushed)
End Station MLA/MSA/EXT MATCH Ar = R Cr = x Ar = S Cr = x EXT_DA_MATCH = 0 (Normal Match Mode)	 S S	 S S	 RPT RPT	 R RPT
Bridge Mode Source Routing MATCH Ar = R Cr = x Ar = S Cr = x EXT_DA_MATCH = 1 (Extended Match Mode)	 S S	 S S	 RPT RPT	 R RPT
Bridge Mode Promiscuous Ar = x Cr = x EXT_DA_MATCH = 0 (Normal Match Mode)	RPT	RPT	RPT	RPT
Bridge Mode Transparent_Bridge_Forward Ar = R Cr = x Ar = S Cr = x EXT_DA_MATCH = 1 (Extended Match Mode)	 RPT RPT	 S RPT	 RPT RPT	 RPT RPT

NOTES:
1. Abort is the assertion of $\overline{\text{REJECT}}$ or FSI-generated abort.
2. Ar = A Received
   Ax = A Transmitted
   Cr = C Received
   Cx = C Transmitted
3. MAC_MODE_CTL = MAC_CNTRL_B Bit 2
   EXT_ DA_MATCH = MAC _CNTRL_B Bit 4

To implement the various types of bridges and the resultant end stations discussed in the ANSI and IEEE standards, the MAC can be used as follows:

1. $\overline{\text{TR\_BR\_FWD}}$, for a transparent bridge, and $\overline{\text{MATCH}}$, for an end station, provide a MAC status-setting functionality.
2. Promiscuous mode provides a MAC status-repeating functionality.
3. A MAC status-clearing functionality for an end station is provided when MAC_MODE_CNTRL=1.
4. Both $\overline{\text{MATCH}}$ and $\overline{\text{TR\_BR\_FWD}}$ provide the necessary timing extensions required to perform source address routing and transparent bridging table lookup when EXT_DA_MATCH = 1.
5. $\overline{\text{REJECT}}$ can be used with the $\overline{\text{MATCH}}$ or $\overline{\text{TR\_BR\_FWD}}$ pin, or in promiscuous mode, to stop the transfer of a frame from the CAMEL to the FSI and to properly set the C-bit for a rejected or incompletely received frame.

5.3 TWISTED PAIR FUNCTIONAL OPERATION

Twisted pair support can be broken into three areas; streaming cipher scrambling, FOTOFF control and signal detect filtering

5.3.1 Streaming Cipher

The current draft of the proposed TP-PMD standard uses ciphering to randomize the transmission data bits in order to reduce high frequency concentrations of the radiated spectrum at certain frequencies (i.e. to flatten and widen the spectrum). The streaming cipher block provides that ciphering.

The scrambler is enabled and disabled by the CIPHER_EN bit in the cipher control register (CIPHER_CNTRL). When ciphering is enabled, the transmitted data is scrambled using the polynomial: $X^{11} + X^9 + 1$. When the scrambler is disabled, the transmitted data is output directly from the PHY layer.

The descrambler is enabled and disabled by the CIPHER_EN bit in CIPHER_CNTRL. When ciphering is enabled, the descrambler looks for any of the repeating line states (Idle Line state (ILS), Halt Line state (HLS), Quiet Line state (QLS), or Master Line state (MLS)) to aquire synchronization with the remote scrambler. Once synchronized, the descrambler will properly descramble the received data using the ploynomial: $X^{11} + X^9 + 1$.

5.3.2 FOTOFF Control

When using fiber media, whenever the physical connection machine (PCM) in the PHY block enters either the BREAK or OFF states, no light energy is to be transmitted onto the fiber. The IFDDI insures this through use of the FOTOFF output pin by turning off the transmit data output of the FCG (clock generation and recovery). This bypasses any information the FCG's NRZI encoder is attempting to output to the fiber.

When using twisted-pair media, a different mechanism must be used. The proposed standard requires that a scrambled-quiet be transmitted for a minimum time (currently

50uS). At any point following this minimum time, true-quiet can be sent. The FOTOFF control provided by the IFDDI is flexible. A user can choose to send scrambled-quiet during the BREAK state and true-quiet during the OFF state. A user is also allowed to transmit only scrambled-quiet at all times, never transmitting true-quiet. Table 5-6 decodes the operation of the FOTOFF_CNTRL bits in the CIPHER_CNTRL register.

Table 5-6. FOTOFF Control Modes

FOTOFF_CNTRL (1:0)	Function
00	Delayed FOTOFF
01	Reserved
10	FOTOFF Forced Inactive (Scrambled Quiet)
11	FOTOFF Forced Active (True Quiet)

5.3.3 Signal Detect

The Signal Detect block filters the Signal Detect (SD) input from the clock recovery device for presentation to the PHY/ELM device. In fiber-optic systems, this block's three filters should be left disabled. In twisted-pair systems, the SD input is optionally filtered by data path transistion sensors. In addition, SD can be delayed to allow the descrambler to initialize.

The Signal Detect block performs three types of filtering: a turn-off filter; a turn-on filter; and an NRZ filter. The turn-off filter enables the SD input to the PHY block only when the RDATA(4:0) bus is active. The turn-on filter delays the assertion of SD for a fixed amount of time to allow the descrambler to aquire lock. The NRZ filter enables SD only when the de-scrambled NRZDATA(4:0) bus is active. The Signal Detect ouput to the PHY layer is called Signal Detect Filtered (SDF).

Because of the increased functionality of signal detect, the SD input is now synchronous to the RSCLK input and not the BYTCLK input.

The turn-on, turn-off and NRZ filters are enabled using the CIPHER_CNTRL register. See **Section 7.4.1 Cipher Control Register** for more details.

5.4 MODES OF OPERATION

Three modes of operation are discussed in the following paragraphs.

5.4.1 MC68840 as an FSI

The MC68840 also operates as a stand-alone FSI. During reset, the operational mode used by the MC68840 is determined by the state of the LDADDR/$\overline{\text{TR}}$_$\overline{\text{BR}}$_$\overline{\text{FWD}}$ pin. The

exposed FSI is compatible to the MC68839 REVD. Please see **Section 10 IFDDI as an FSI** for a detailed guide on using the MC68840 as an FSI REVD.

The mode of operation is selected when the IFDDI samples the LDADDR/$\overline{TR}$_$\overline{BR}$_$\overline{FWD}$ pin coming out of reset on the rising edge of $\overline{RESET}$. If the LDADDR/$\overline{TR}$_$\overline{BR}$_$\overline{FWD}$ is sampled as Vcc on the rising edge of $\overline{RESET}$, the chip will operate as an IFDDI device. If the LDADDR/$\overline{TR}$_$\overline{BR}$_$\overline{FWD}$ pin is sampled as GND on the rising edge of $\overline{RESET}$, the chip will operate as an FSI device. The LDADDR/$\overline{TR}$_$\overline{BR}$_$\overline{FWD}$ pin has an internal pullup during reset; thus, if it is not connected to GND, the chip will operate as an IFDDI device.

5.4.2 Bypass Mode

The FSI chip/block operates using FSICLK, which may be connected either to SYMCLK or to another separate clock. When the FSI operates using SYMCLK, the port synchronization function (PSF) is not needed, and the user should bypass the PSF by selecting the bypass mode. The bypass mode is selected when the IFDDI samples the DA pin coming out of reset on the rising edge of $\overline{RESET}$. If the DA pin is sampled as V_{CC} on the rising edge of $\overline{RESET}$, the chip will operate in non-bypass mode, and the PSF will be used. If the DA pin is sampled as GND on the rising edge of $\overline{RESET}$, the chip will operate in bypass mode, and the PSF will not be used. The DA pin has an internal pullup; therefore, if the pin is not connected to GND before the rising edge of $\overline{RESET}$, the chip will operate in non-bypass mode.

5.4.3 CAMEL Test Mode

This mode is used to expose the CAMEL block I/O pins for testing purposes *only*. The IFDDI enters this mode by setting the CTE bit in the IFDDI configuration register (ICR).

SECTION 6
PORT OPERATION

Combined with an external programmable array logic (PAL) type of device, the IFDDI is able to function as a bus master or slave in a wide variety of bus protocols. The IFDDI only requests to be read or written; whereas, the external logic accesses the IFDDI according to the IFDDI requests with regard to the specific bus protocol's timing. The IFDDI provides the external logic with hooks such as new page indication and also provides internal address generation, byte swapping, and parity checking.

The IFDDI has two major operational modes, normal and pipeline. When using pipeline mode, the port can easily be connected to a wide variety of burst/pipelined buses.

6.1 PORT DATA TRANSFERS

The four types of transfers supported by the IFDDI are:

1. Command or descriptor reads from command/descriptor rings,
2. Indication writes to command/descriptor rings,
3. Data buffer reads of transmit/DMA frames,
4. Data buffer writes for received/DMA frames.

The first transfer type is performed by the port interface unit using the ring read pointer (RRP) as defined in the DEFINE RING command. The OWN bit (the most significant bit of the first long word in each command) is sensed during the memory read access and, if it is set (i.e., the IFDDI is the owner of this entry), the RRP is updated to point to the next entry. If the OWN bit is reset, the RRP will not be updated and this ring is empty. Note that because of the pipeline nature of the ring accesses, an extra access is required by the IFDDI, but the accessed data will be ignored. No further accesses for this ring are requested by the IFDDI until the ring state becomes ready again. The RRP is incremented within the limits defined by ring maximum length (RML).

The other transfer types occur based upon the information in the first transfer type, such as receive and transmit buffer pointers, or upon internal information such as current receive or transmit indication pointers.

If it is enabled, parity checking for a transmit data buffer will occur on the full 32 bits or 64 bits of data on each byte, regardless of actual data byte alignment. The user should ensure that buffer memory has had data with valid parity written to locations that might be accessed by the IFDDI in such situations.

6.1.1 Port Signals

All port signals are described in detail in **Section 8 Signal Description**. The following subsections provide functional descriptions of these signals.

6.1.1.1 PORT REQUEST. The port request signals, $\overline{\text{AREQ3}}$–$\overline{\text{AREQ0}}$ and $\overline{\text{BREQ3}}$–$\overline{\text{BREQ0}}$, are used to indicate both the internal port control logic state and the next required cycle type—read, write, or idle. Since the IFDDI is a slave DMA device, external logic is used to decode the $\overline{\text{REQx}}$ signals to determine the type of access required by the IFDDI.

Each port has three basic states. The port control unit can transition through the three states shown in Figure 6-1.

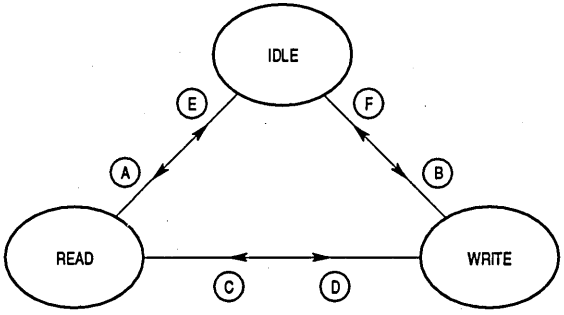


Figure 6-1. Port Control Unit States

Idle: In this state, the port has completed the pending transfers it had to execute. The port will remain in this state until a new transfer is requested from the main controller. The port will transition to this state ("E" and "F") when there is no active internal request. This transition is performed during the last data access. In the idle state, all requests to the external bus control logic are negated.

Read: The port will transition to this state when a read operation (descriptors or Tx DMA read) is required by the main controller. Transition "A" from the idle state is performed synchronously or asynchronously with the chip select input signal, depending on the mode of port operation.

Write: The port will transition to this state when a write operation (indications or Rx DMA write) is required by the main controller. Transition "B" from the idle state is performed synchronously or asynchronously with the chip select input signal, depending on the mode of port operation.

The "C" and "D" transitions between the read and write states occur according to the internal priority when both read and write operations are required by the main controller. These transitions are always synchronized with chip select input signals.

In addition, page information is available to allow efficient use of external dynamic memory. The page output signal on the request lines ($\overline{\text{AREQ0}}$ for port A and $\overline{\text{BREQ0}}$ for port B) is asserted when all the following conditions are satisfied:

1. The address of the next data access is on the same memory page with the current data access.
2. The next data access has the same direction (both read or both write) as the current one.
3. The next data access is the same type as the current one (both are data DMA or descriptor accesses). If it is a descriptor access, read descriptors or write indications, the page signal is negated when the ring is wrapped.

Since the request lines are changed to indicate the next required cycle during the access to the data register or during a NOP access, the next required cycle type may be sampled by external control logic on the negation of chip select. If the system bus is operating in an asynchronous mode, the request lines will change from idle to read or write asynchronously to chip select.

6.1.1.2 PORT CONTROL. Internal to the IFDDI, the CNTLx input lines are used to access the IFDDI. These lines are sampled by the IFDDI on chip select assertion. The CNTLx lines access internal control registers, data address registers, or internal data FIFOs or perform special port functions as with abort and NOP.

The internal address generator is also updated each data access (access to the data register (DTR) where CNTLx = 0010) on chip select assertion. If an address register (ADR) is accessed, the IFDDI will latch the value of the address generator on the assertion of chip select, and this value will appear on the data pins of the selected port. When a multiplexed address/data configuration is used, the address of this port is accessed by CNTLx = 0100. When a nonmultiplexed configuration is used and the address appears at a different port from the data, the address is accessed by CNTLx = 1000 to indicate that the address of the other port is required.

Access to ports A and B is enabled through the IFDDI external control pins (ACNTL3–ACNTL0 and BCNTL3–BCNTL0) when one of the chip selects is asserted as defined in Table 6-1.

Table 6-1. IFDDI Port Direct Access Map

CNTLx	Read		Write	
3210	Port A	Port B	Port A	Port B
0000	NOP			
0010	DTR(A)	DTR(B)	DTR(A)	DTR(B)
0100	ADR(A)	ADR(B)	Reserved	Reserved
1000	ADR(B)	ADR(A)	Reserved	Reserved
1110	Abort			

Since the IFDDI is a slave DMA device, the external logic can, at any time, drive the CNTLx and R/$\overline{W}$ lines to access any of the IFDDI internal registers. These accesses can be interleaved with the DMA accesses requested by IFDDI. That is, the IFDDI does not have to be in idle state when the registers are accessed.

There are two approaches for adding wait states to the IFDDI. In the first approach, the chip select assertions of $\overline{ACSx}$ and $\overline{BCSx}$ are controlled to assert only when the external logic is ready for the data transfer holding CNTLx for either the data or address transfer. The second approach is used if the chip selects are a clock. In this method, the CNTLx signals must be controlled to provide for the data or address exchange on a single clock cycle and are then NOPed so as to not request new address or data on the following chip select clock.

6.1.1.2.1 Abort Access. This access will cause the port to abort its current operation. It may be used during transmit data reads, descriptor reads, or receive data writes, and it will have the following affect:

a. During transmit data reads, an abort access will abort the transfer of data into the IFDDI internal memory, potentially making the frame into a fragment. This is similar to the situation that would occur if a parity error was detected during a transmit data read. If the frame does become a fragment, the abort indication is provided in the IFDDI transmit indication.

b. During descriptor reads, an abort access will cause the IFDDI to treat the ring as empty—i.e., as if it had no additional valid descriptors.

c. During receive data writes, an abort access will abort the transfer of the frame being received from the IFDDI internal memory. The frame data is discarded.

Note that the abort access has no effect on indication writes. In 64-bit mode, the abort access should be given to port A and port B concurrently. Note that the abort access has no effect on the request lines of the port. Therefore, an additional data access is required after the abort access to complete the current data transfer operation.

6.1.1.2.2 NOP Access. This access has no effect on the internal port operation and is used only in port synchronous operation mode to assert the request output signals synchronously with the chip select input signals. Note that if R/$\overline{W}$ is asserted for a NOP access, the IFDDI will drive the port data bus.

6.1.1.3 PORT CHIP SELECT. The IFDDI provides two chip selects for each port, ACS0, ACS1 and BCS0, BCS1. Only one of the chip select signals may be asserted for each access on each port. Two chip selects per port are provided for the configuration in which more than one master device is operating with the IFDDI. The chip select signals provide a clocking mechanism for the port control unit.

Asynchronous or synchronous bus request operation (i.e., the change from idle to a new read or write state) is selected via bit 5 in each port's port control register (PCR). The function of the mode selection is to determine whether the port state in the $\overline{\text{REQx}}$ signals will change synchronously or asynchronously to chip select. In asynchronous operation, the $\overline{\text{REQx}}$ signals will change from idle to read or write asynchronously to chip select. In synchronous operation, the change from idle to read or write will occur synchronously to chip select assertion when the CNTLx input lines are performing a NOP access. After the read or write cycle has started (i.e., the interface goes from idle to read or write), subsequent state changes are synchronous to chip select.

6.2 PORT OPERATION—NORMAL MODE

The external processor(s) and external bus control logic will normally operate on their own clock as defined by the external bus timing. All transfers to and from the IFDDI are synchronized inside the chip; therefore, no external synchronization is required. The external timing is defined by $\overline{\text{ACS0}}$ and $\overline{\text{ACS1}}$ for port A and by $\overline{\text{BCS0}}$ and $\overline{\text{BCS1}}$ for port B. All port bus activities are defined with reference to these chip select signals. Note that port A may use an external clock that is different from that of port B.

If asynchronous port operation is selected in the PCR, note that the bus control logic has to synchronize the transition of request lines from the idle state to the active state in conjunction with the system clock in order to correctly recognize the next required cycle. The chip select signals are used to acknowledge readiness for the external logic to perform the required cycle.

If synchronous port operation is selected in the PCR, the request output lines of each port are changed only during the active cycle of one of the chip selects. The chip select signals are the clock signals for synchronous operation, and all timing is relative to these signals. Therefore, if the port's current state is idle, the transition to the active state will occur synchronously with the chip select line and may be sampled by the bus control logic when the chip select line is negated. Note that this transition will occur only when the external bus control logic performs a NOP access to the port.

6.2.1 Port Address Generation

Each transfer of data or command/indication information has its address generated internally by the IFDDI port control unit. This address is updated each time the data transfer is performed. The next address to be accessed is compared with the current access address to decide if the next access is on the same memory page as the current access. The address may be accessed at any time between data accesses.

6.2.2 Interport Operation

When used as two separate 32-bit ports, the IFDDI offers the system designer the flexibility to provide capability such as FIFO descriptors and indications on one port and FIFO data on a separate port. In these modes, the following considerations should be taken into account:

a. It is possible to interrupt a processor on port B, for example, with events happening on port A by using the general status register (SR1). This is possible since the SR1 is common to both ports and accesses to this register will show the same information except for command done (CDN), control register free (CRF), and host error (HER) bits, which reflect the individual status for each port. As an example, a ring that transitions from the ready state for a ring in the port B memory space may cause an interrupt to the port A processor if the RNE bit for this ring is set in the interrupt mask register (IMR1) of port A.

b. Directly accessed registers, such as the interrupt mask register (IMR), command register (CMR), command extension register (CER), port status register (PSR) etc., are divided into port A and port B registers and cannot be accessed from the other port.

c. All internal parameters, including the port control and ring parameters for each of the rings, are accessible by accessing the FSI control register (FCR) and may be set from either or both ports.

6.2.3 Port Operational Errors

The port operational error (POE) condition occurs when a system bus error is detected by bus control logic. For example, when the bus control logic requests a data access and the data access acknowledgment has not been received by the bus control logic quickly enough, POE can provide an interrupt to the host processor. In bus error situations, the bus control logic needs to abort the current data access and create an abort. To accomplish this, the bus control logic should assert the second chip select on the offending port, causing both chip selects to be active at the same time. In response to this abnormal operation, the IFDDI will set the POE status bit and reset the port enable control bit. The IFDDI will ignore the new request it has generated and will transition to an idle indication on the request lines a delay time after the second chip select has been asserted. Depending upon the current transfer type, the effect is as follows:

a. For transmit or receive descriptor reads, the ring state is changed to empty with the OER status bit set in the appropriate ring state register. The RNR and RER bits in the SR1 are set.

b. For transmit data reads, setting the POE bit will abort the transfer of a transmit frame into the IFDDI, potentially making the frame into a fragment. This is similar to the situation encountered if a parity error is detected during transmit frame data reads. The indication generated by the IFDDI will contain the abort indication if it occurs.

c. When the IFDDI is performing receive data writes, setting the POE bit will abort the transfer of a received frame to the memory. The indication generated by the IFDDI will contain the abort indication.

d. Setting the POE during an indication write operation has no effect on the operation.

6.2.4 Programmed I/O Operation

The IFDDI can operate in a programmed I/O mode. In this method of interfacing to a system, the data is transferred by either the high-speed processor itself or by another high-performance peripheral in the system.

The IFDDI itself does not perform any bus control functions and is totally indifferent to how, and by what device, bus control cycles are implemented. Therefore, the host processor may actually handle all transfers without any bus control logic. In this case, the host processor may choose to give commands to the FSI through a command/descriptor ring or through the command (CMR) and command extension (CER) registers. These methods are described in the following paragraphs. To execute transmission by issuing commands through the command/descriptor rings, the host processor performs the following procedure:

a. The host sets the ring to ready using a write operation to the FSI control register (FCR). As a result, the IFDDI will transition this ring state to ready and will request a read descriptor cycle.

b. Then the host writes the transmit buffer descriptor entry into the IFDDI when it senses the port request, either by reading the port status register (PSR) or by getting an interrupt based on the external request signal inputs. Using these write cycles, the host processor may enter a number of descriptors inside the internal transmit ring FIFO. To stop the read descriptor cycle of the IFDDI, the host processor writes the last entry with the OWN bit reset. When the port interface reads this entry and determines that is does not belong to the IFDDI, it stops the read descriptor cycle, and this ring transitions to empty. The next IFDDI cycle is the read data cycle to fill the internal transmit data FIFO up to its watermark.

c. The host writes the data into the IFDDI when it senses the port request. When the internal transmit data FIFO reaches its watermark, the port request is negated. The host processor may use the idle state condition on the request signals to get an interrupt. The read data cycle request is asserted again by the port when the IFDDI's internal FIFO starts to transmit the data to the MAC. The host processor should then continue to write the data into the IFDDI. After the frame is transmitted, the port initiates a write indication cycle.

d. Finally, the host reads the indications from the IFDDI when it senses an appropriate port request. As in the case of data, the port may go to the idle state after a number of indications. Additional attempts to read the indications are ignored by the IFDDI. When the last indication of a frame is transferred to the host, the RCC status bit in the SR1 is asserted.

NOTE

The user should not perform data accesses to the IFDDI when the IFDDI is in idle state—i.e., when the $\overline{\text{REQx}}$ lines indicate idle.

To execute a slave transmission using commands given directly to the IFDDI, the host processor performs the following procedure:

a. Writes the transmit buffer descriptor into the FSI command register (CMR). When this command is written, the IFDDI executes it by trying to get the data for this frame. Therefore, the port initiates a read data cycle.

b. Writes the data into the IFDDI after it senses the port request. When the internal FIFO has reached its watermark, the port request is negated. The host processor may use the idle state condition on the external $\overline{REQx}$ lines to get an interrupt. All additional data write accesses executed by the host processor after reaching the port idle state are ignored by the IFDDI, and the host processor should repeat them after handling the interrupt. The host processor may read the internal IFDDI address or transfer counter to determine what data it should repeat. The read data cycle request is reasserted by the IFDDI when the IFDDI's internal FIFO starts to transmit the data to the MAC. The host processor should then continue to write data into the IFDDI. After the frame has been transmitted, the IFDDI writes the indication inside the CMR, resetting the OWN bit of the command. This, in turn, may cause assertion of the command done interrupt if it was previously enabled. Note that the CMR remains occupied until the completion of transmission. Therefore, no other command can be issued from the CMR during this period.

Transmission using transmit commands issued directly to the IFDDI may be used to meet the FDDI standard requirement for immediate frame transmission. Such frames must be able to supersede all other frames queued for transmission. This procedure may also be used to transmit other types of frames in addition to immediate frames. These frames may be of high priority or may not properly belong in one of the four transmission rings.

6.2.5 Functional Port Operation Examples—Normal Mode

The following figures illustrate some functional timing examples for the IFDDI in normal mode. Figure 6-2 describes CAMEL direct access read and write cycles. The user places the register address on the control lines (ACNTL8 = 1 and ACNTL7–ACNTL0 = CAMEL register address) a setup time before the clock negation. The $\overline{CS}$ is then asserted for at least Tx ns. For the definition of Tx, refer to Table 13.7, Note 1. On write operations, the data should be valid during the entire $\overline{CS}$ assertion.

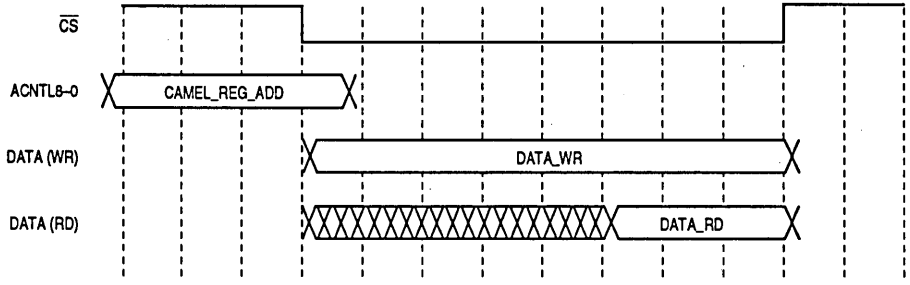


Figure 6-2. Read and Write CAMEL Internal Register Timing

Figure 6-3 shows four accesses of data from the same external memory page. A preceding access to the address register is shown to determine the start of this burst. In this figure, as well as in the next two figures, the ambiguous time frames for the CNTLx and R/$\overline{W}$ lines, as indicated around the rising edge of chip select, are meant to emphasize that the IFDDI samples the CNTLx and R/$\overline{W}$ lines on chip select assertion.

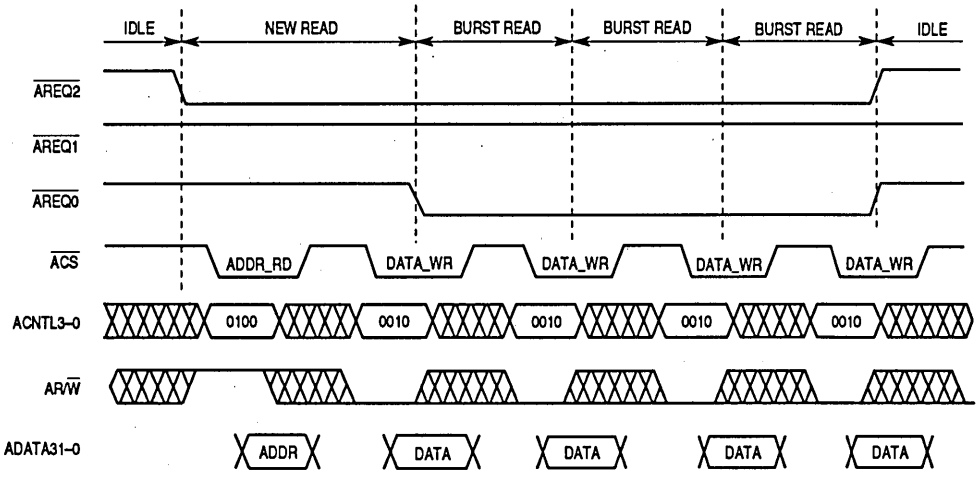


Figure 6-3. Address/Data Multiplexed Operation

In Figure 6-4, port A is used for 32-bit data and port B is used for 32-bit address. Note that the address read is done prior to the data access to reduce the cycle time. The address is changed on the falling edge of the data access, and it may be read out on the following address access.

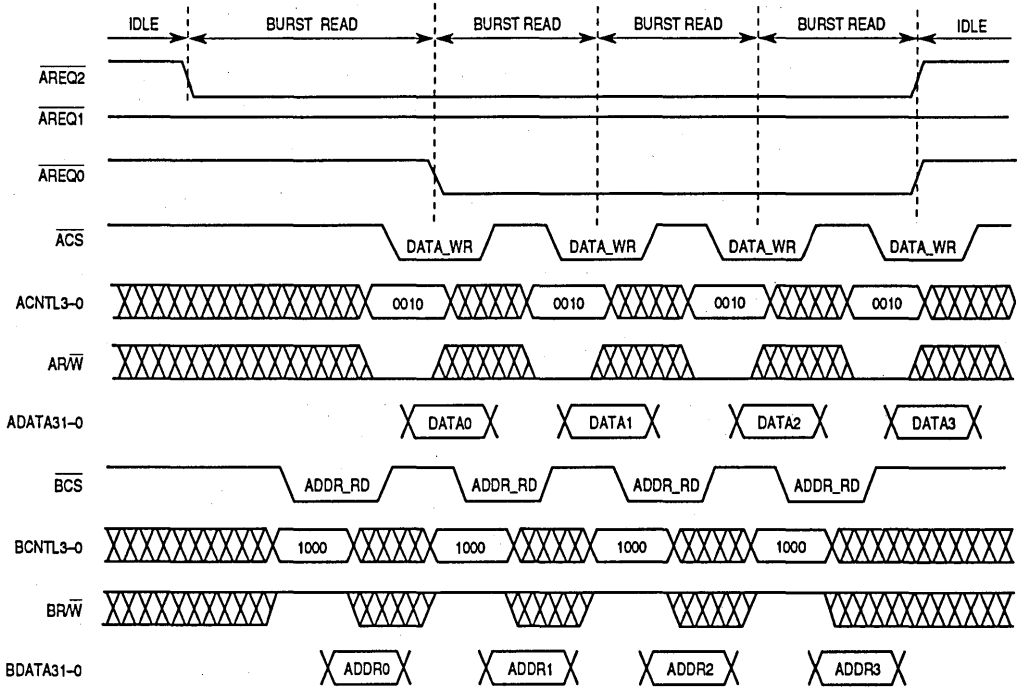


Figure 6-4. Nonmultiplexed Address/Data Operation

Figure 6-5 demonstrates synchronous port operation. The $\overline{\text{REQx}}$ lines are changed from idle to new read when $\overline{\text{ACS}}$ is asserted and ACNTLx = NOP.

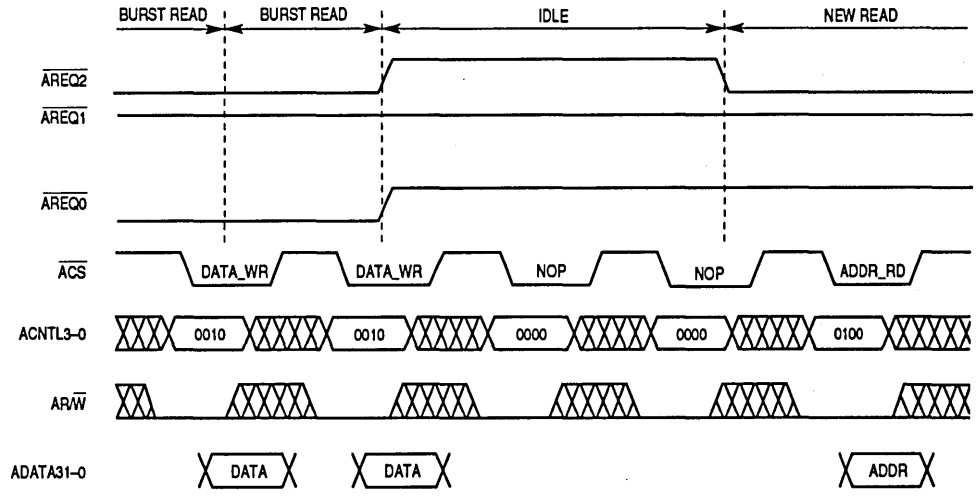


Figure 6-5. Synchronous Operation

Figure 6-6 shows an asynchronous bus cycle—that is, the change of state from idle to new read is done asynchronously with respect to ACSx. In this case, the bus logic would have had to synchronize the change of state on the REQx lines and then present ACS when it was ready to proceed with the remainder of the bus tenure. The initial access is a new read in which an address is presented, followed by a single data transfer for that address, the burst read, and then another new read cycle due to a change of page.

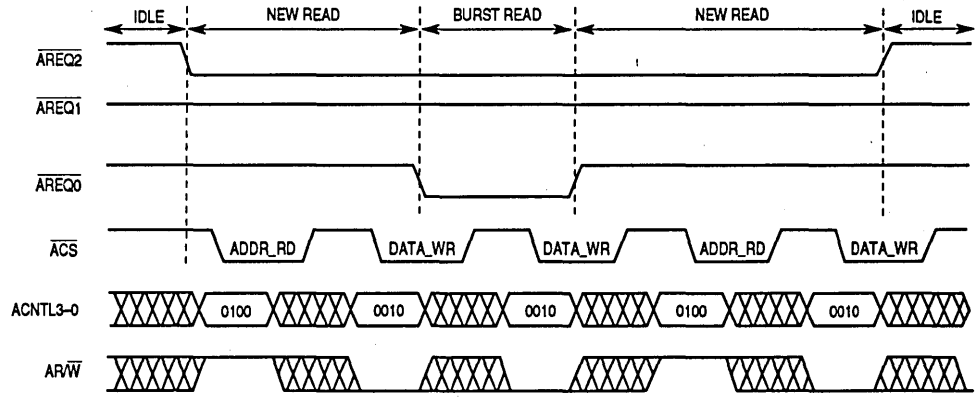


Figure 6-6. Data Read Operation with a Change of Page

Figure 6-7 illustrates the interleaving of reads and writes on a single bus tenure. It shows an asynchronous bus tenure in which a single read access is followed by a write access followed by a read access.

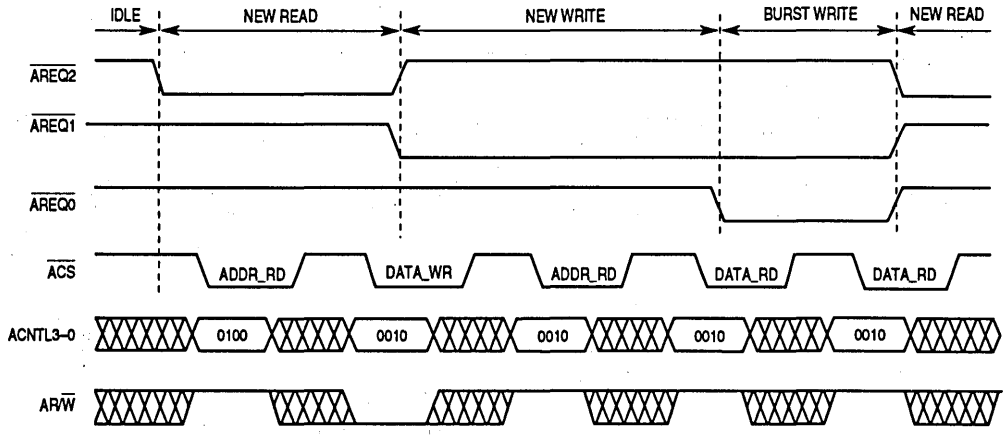


Figure 6-7. Data Read Followed by a Data Write in the Same Bus Tenure

Figure 6-8 shows that in asynchronous mode there is a minimum of 5 clocks between the IFDDI releasing its request lines to become idle and the next time the IFDDI asserts its request lines.

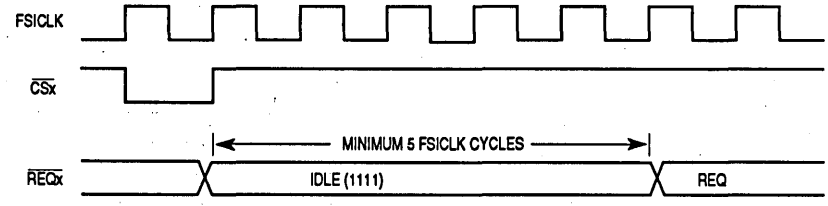


Figure 6-8. Request Negation to Request Assertion in Asynchronous Mode

6.3 PIPELINE MODE AND BURST OPERATION

The high-performance buses use burst accesses to speed up the data transfers. The clock cycle of these buses is short (33, 40 ns), and to use them at maximum performance, the data should be placed on the data bus for the entire clock cycle.

The control logic should be able to operate on a clock-cycle basis to act immediately rather than with a delay of a clock cycle.

The IFDDI was designed to support easy interface with burst buses. The following subsections explain how to use the pipeline mode and the burst operations.

6.3.1 Port A and Port B Data Pins

Figure 6-9 describes the basic structure of the port A and port B data pins. Note that port A and port B operational modes (normal or pipeline) are independent of one another.

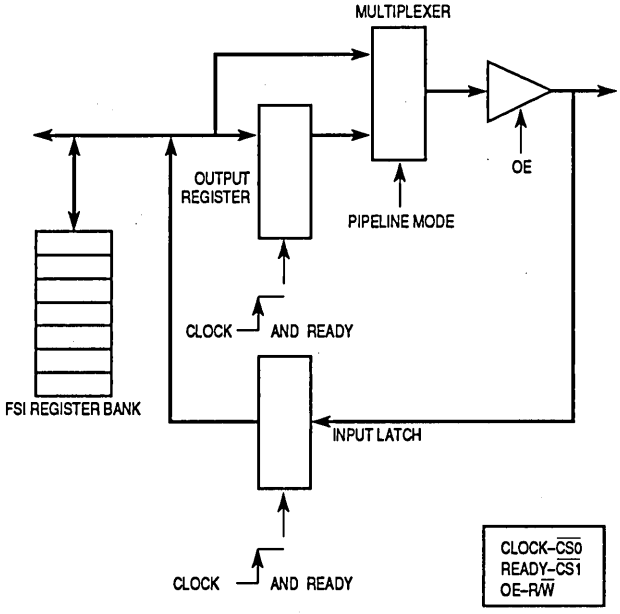


Figure 6-9. IFDDI I/O Basic Structure

The pipeline register removes any need for external latches. On data-out cycles, the data bus may be driven using the output register. On data-in cycles, the IFDDI samples the data bus on the rising edge of the clock. This improvement enables the data bus to be driven for the entire clock cycle instead of only on the low phase (as in normal mode).

The user has total control of the pipeline register, including control over when data is loaded into the register and when to output its contents to the external pins. From the IFDDI's viewpoint, the pipeline register belongs to the user; data that was loaded into it during a data read access cannot be re-accessed, even if the user did not read it to the bus.

The IFDDI data pins are connected to a multiplexer output that selects between the IFDDI internal register bank bus and the output register. The IFDDI internal register bank is selected in normal mode, and the output register is selected in pipeline mode. The output register will sample the FSI block's internal register bank bus on the rising edge of the

clock with READY asserted. The multiplexer output is connected to an output driver controlled by OE.

The IFDDI data-in path is connected to the internal register bank bus through an input latch that reduces the data in hold time. The input latch latches the data in on the rising edge of the clock with READY asserted.

6.3.2 Pin Assignment in Pipeline Mode

The pin assignment in pipeline mode is discussed in the following paragraphs.

6.3.2.1 SYSTEM CLOCK. In pipeline mode, the IFDDI/FSI operation is synchronized with the external bus clock by connecting the external bus clock to $\overline{CS0}$.

6.3.2.2 READY. The READY signal extends the cycle (wait state) by disabling the rising edge of the internal clock. This continues the drive from the pipeline register during a read operation and delays the data sampling during a write operation.

The READY signal is connected to $\overline{CS1}$. The READY signal is sampled by the IFDDI on the rising edge of the clock.

Do *not* change the CNTLx lines in phase high after READY negation (0) except in the case of multiple cycle accesses.

6.3.2.3 OUTPUT ENABLE (OE). This line enables the pipeline register to drive the data bus. OE is connected to R/$\overline{W}$ line.

6.3.3 Pipeline Mode Selection

A port will enter the pipeline mode upon detecting a PNOP access. The port will exit from the pipeline mode upon detecting a NOP access. Note that port A and port B operational modes (normal or pipeline) are independent of each another.

In most applications, the port will operate in one of the two modes (normal or pipeline) by choosing NOP in normal mode, or PNOP in pipeline mode, as the regular no operation access. However, switching between pipeline mode and normal mode can be easily performed to enable the user to change the port mode dynamically (see Figure 6-10).

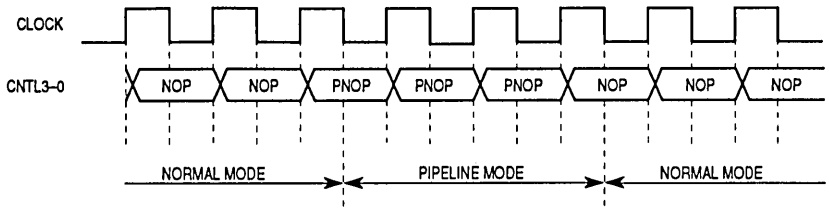


Figure 6-10. Pipeline Mode Selection

During pipeline mode, the port must operate in synchronous operation. The port is set to synchronous operation by setting the SO bit in the FSI port control register (PCR).

6.3.4 Burst Operation

A complete burst cycle consists of two phases: 1) declaration cycle and 2) data cycle. The declaration cycle starts with a burst address (BADR) access. The port drives the data bus with the address as it does in an ordinary address access and drives the $\overline{\text{REQ3}}$–$\overline{\text{REQ1}}$ line with the burst size. This access will "lock" the port for the whole burst (i.e., no switching between read and write). The port will be "locked" until the limit counter expires or a PNOP access is detected. The burst size is limited to a maximum of the setup value in the burst limit register (BLR). Note that the IFDDI calculates the actual burst size based on the minimum of one of three values : the value in the BLR; the external address; and the amount of actual data to be transferred (see **Section 6.3.5 Size Calculation**).

6.3.4.1 $\overline{\text{REQx}}$ LINES IN BURST MODE. The $\overline{\text{REQx}}$ lines in burst mode may be defined as follows:

$\overline{\text{REQ3}}$—Limit
$\overline{\text{REQ2}}$—Read Request
$\overline{\text{REQ1}}$—Write Request
$\overline{\text{REQ0}}$—Data Valid

In burst mode, $\overline{\text{REQ3}}$ shows the limit counter state. After a BADR access, $\overline{\text{REQ3}}$ is asserted until the limit counter expires. The limit counter counts down only on data accesses. The state when $\overline{\text{REQ3}}$ is negated is called external end of burst. This state is used to inform the user that from now on the port is free to switch between read and write (no "lock").

In burst mode, $\overline{\text{REQ0}}$ identifies the internal end of burst. $\overline{\text{REQ0}}$ is asserted only if the next access address is consecutive and on the same page. The state when $\overline{\text{REQ0}}$ is negated is called internal end of burst. In this state, the port will ignore all data accesses. Note that $\overline{\text{REQ0}}$ will generally be negated on longer bursts when an unanticipated condition causes the IFDDI to stop in the middle of a burst.

During the data cycle, the port will perform data accesses. The user can drive the data bus for more than one clock cycle per data by negating the READY control line. The port

will continue transfers until the end of the burst. If the port finishes the burst internally before externally, the port negates $\overline{REQ0}$. The user may then end the burst session by a PNOP access or continue accessing the IFDDI until the limit counter expires (negation of $\overline{REQ3}$). If the port finishes the burst externally before internally, the port negates $\overline{REQ3}$. The user may then start a new burst session, or the port can simply continue its accesses. The port is then free to switch from read to write or vice versa.

The burst may be stopped at any time using a PNOP access. To enable switching between read and write, at least two PNOP accesses must be issued before the next BADR access.

In indication/DMA out, if the port finishes the burst internally before externally and if the accesses are continued until the end of the burst, the port will drive the entire data bus to one to set the OWN bit and to identify these indications.

6.3.4.2 IFDDI STATES DURING A BURST CYCLE. The IFDDI states during a burst cycle are shown in Figure 6-11.

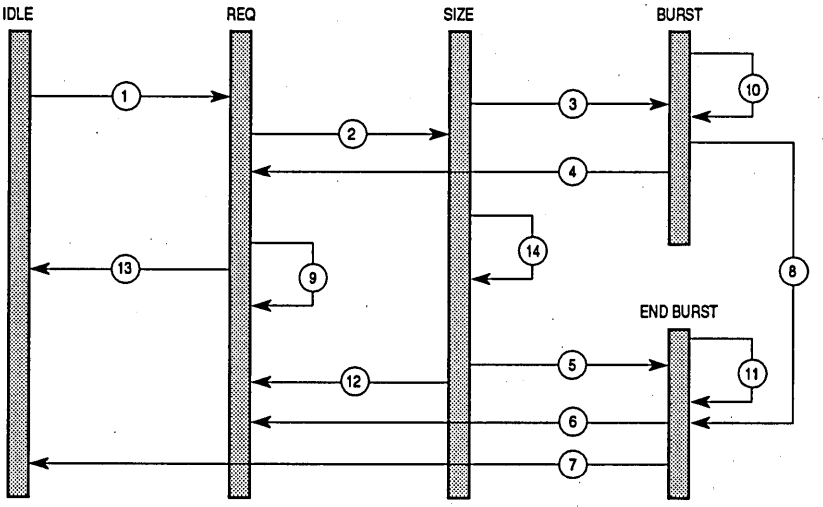


Figure 6-11. IFDDI States During Burst Cycle

The IFDDI states are defined as follows:

IDLE — The port is in the idle state—no request is asserted.

REQ — A PNOP access has been done—the port requests read or write using the normal request line mode.

SIZE — A BADR access has been done—the port drives the address on the data bus and the size on $\overline{REQ3}$–$\overline{REQ1}$. The limit counter is loaded with the

size value, and the port will be "locked" for the whole burst (i.e., no switching between read and write).

BURST — The port will be "locked" until the limit counter expires. The user may then continue the data accesses until the port switches between read and write or to idle. The port $\overline{REQx}$ lines are in burst mode.

END_BURST — The port ignores all data accesses. The port $\overline{REQx}$ lines are in burst mode.

The state transition conditions are as follows:

1. PNOP access
2. BADR access
3. Data access and limit counter not expired
4. PNOP access
5. Data access and the port switched between read and write or to idle
6. PNOP access and the port needs an additional data access
7. PNOP access and the port does not need an additional data access
8. Data access and the port switched between read and write or to idle
9. Data access and the port needs an additional data access (The data access was done without using a previous BADR access.)
10. Data access and limit counter not expired or limit counter expired and data access in the same direction as the burst
11. Data access
12. PNOP access
13. Data access and the port does not need an additional data access (The data access was done without using a previous BADR access.)
14. BADR access

6.3.5 Size Calculation

The size calculation is based on three factors:

1. Burst Limit Register (BLR)
2. Transfer Counter
3. External Address

The size is set to the lowest value among the three factors (burst limit register, transfer counter, and external address, if enabled). During a BADR access, the size value is placed on $\overline{REQ3}$–$\overline{REQ1}$ and is loaded into the limit counter. The limit counter counts down until zero, and then it negates $\overline{REQ3}$.

Burst Size (Bytes)	$\overline{REQ3}$	$\overline{REQ2}$	$\overline{REQ1}$
4	1	1	0
8	1	0	1
16	1	0	0
32	0	1	1
64	0	1	0
128	0	0	1
256	0	0	0

6.3.6 Multiple Cycle Access

A multiple cycle access is composed of two consecutive clock cycles with READY negated before the rising edge of the second clock. The control lines are also changed in the second clock cycle. The action in a multiple cycle access is affected by the two consecutive control line codes.

The multiple cycle access mechanism is used by:

REGWR—Write Registers During Pipeline Mode

POE—Port Operation Error During Pipeline Data Access

NOTE

REGWR and POE are codes placed on the control lines of the IFDDI by the glue logic. See Table 7-3 for the definition of these codes.

6.3.7 Register Write During Pipeline Mode

In pipeline mode, register write is performed using multiple cycle access. On the first cycle, the user places the register address on the control lines; on the second cycle, the user places the REGWR code (0010). READY will be negated between the two cycles. The data will be written to the register on the clock rising edge of the second cycle.

6.3.8 Port Operation Error During Pipeline Data Access

If a port operation error occurs during pipeline data access, the user should inform the IFDDI by changing the CNTLx lines from 0010 (DTR) to XXX1 during the data access (DTR) cycle with READY negated. This should always be done except in the case when the IFDDI has already negated its request.

6.3.9 Non-Multiplexed Address/Data Operation in Pipeline Mode

The update of the Other Port Address register is now controlled by the READY signal in Pipeline mode. This description of non-multiplexed address/data operation in Pipeline

mode assumes that the data transfers are through Port A of the IFDDI and the address is read through Port B. If the opposite is true, the A and B prefixed of the signal names should be interchanged. Both ports use Pupeline mode with the same clock. One or more wait states are inserted on the data transfers of Port A by using the AREADY IFDDI input pin (ACS1). This discussion relates to the use of BREADY to obtain the address on BDATA at the proper time relative to the data on ADATA. When performing DMA transfers from RAM to the IFDDI, negating BREADY one clock cycle after AREADY and asserting it together with AREADY will provide the address on BDATA at the same time that the data is on ADATA as shown in Figure 6-12. If AREADY is only negated for one clock cycle, BREADY should not be neageted at all for these results as shown in Figure 6-13. When performing DMA transfers from the IFDDI to RAM, BREADY should be identical to AREADY in order to provide the address on BDATA at the same time that the data is on ADATA as shown in Figure 6-14 and Figure 6-15.

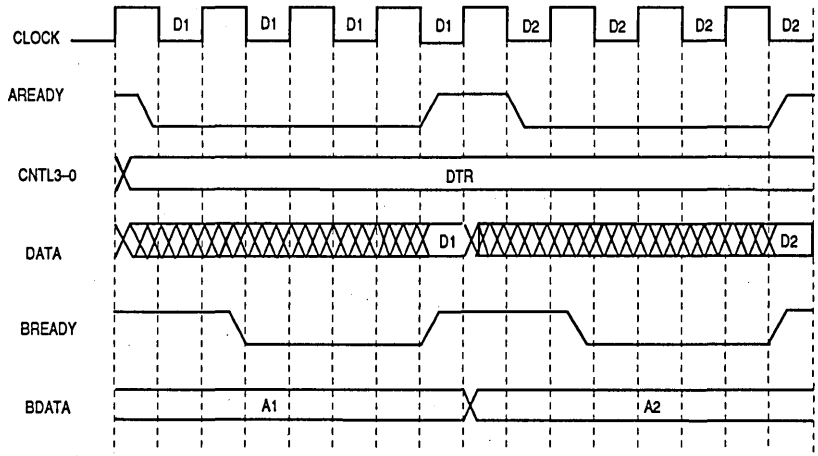


Figure 6-12. DMA Write to IFDDI - Read from DRAM (Multiple Wait States)

NOTE

BREADY is negated one clock after AREADY and asserted together with AREADY

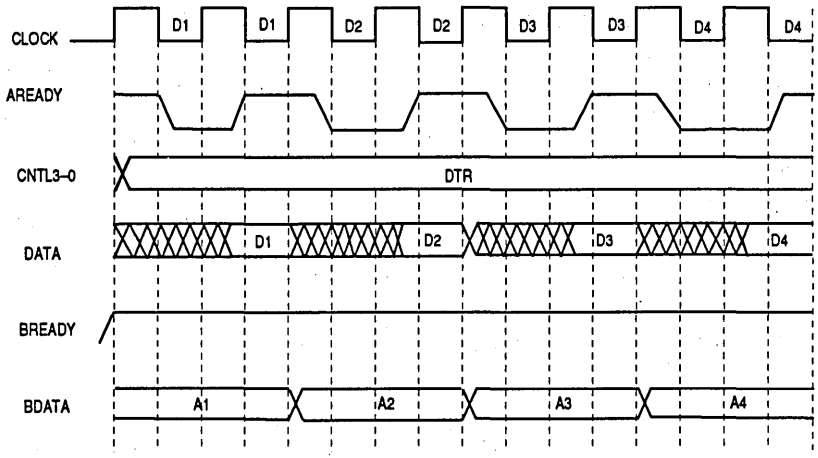


Figure 6-13. DMA Write to IFDDI - Read from DRAM (One Wait State)

NOTE

BREADY is asserted with AREADY and remains asserted for the entire access.

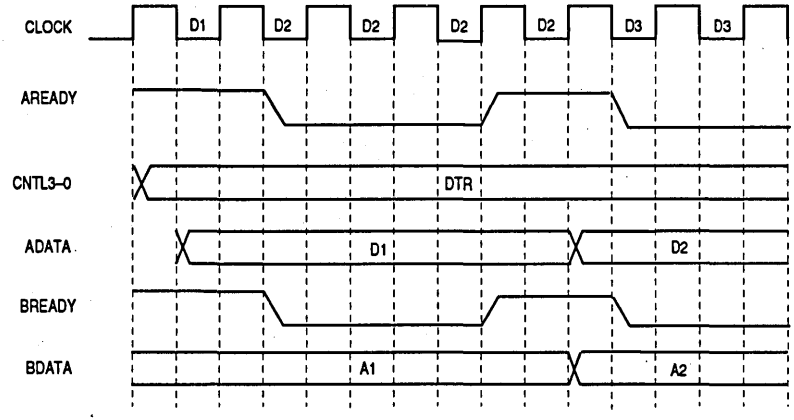


Figure 6-14. DMA Read from IFDDI - Write to DRAM (Multiple Wait States)

NOTE

BREADY is identical to AREADY

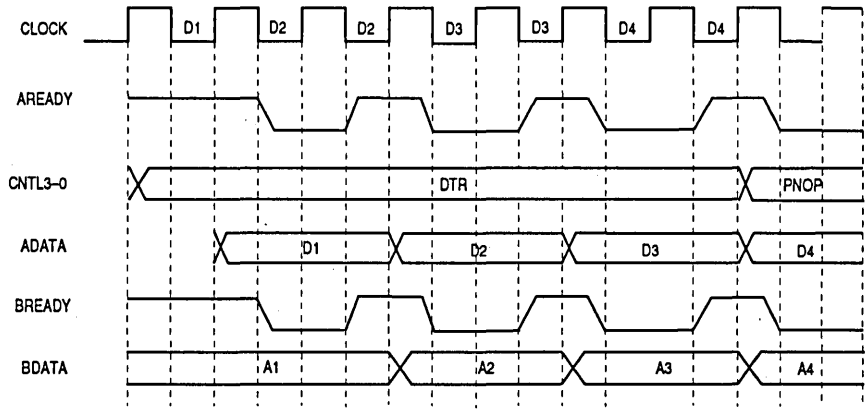


Figure 6-15. DMA Read from IFDDI - Write to DRAM (One Wait State)

NOTE

BREADY is identical to AREADY.

6.3.10 Pipeline Mode and Burst Operation Timing Examples

Figures 6-16 to 6-19 illustrate pipeline mode operation timing examples. Figure 6-16 describes a pipeline read operation without wait state(s). Data out is sampled by the output register on the rising edge of the clock. The data is placed on the data bus as OE is asserted during the entire cycle. There is no wait state since READY is asserted during the entire cycle.

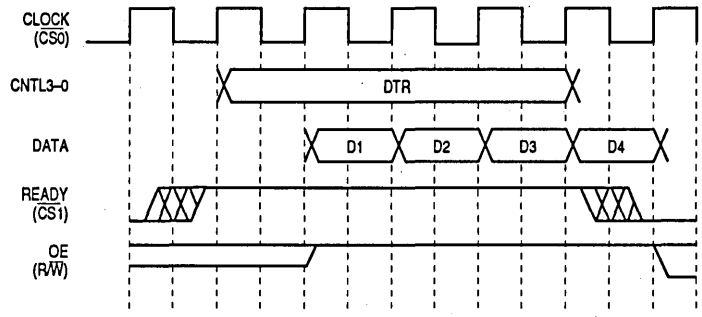


Figure 6-16. Pipeline Read without Wait State Timing

Figure 6-17 describes a pipeline read operation with wait state(s). Data out is sampled by the output register on the rising edge of the clock. The negation of READY during D3 disables the output register from sampling D4. D3 is placed on the data bus for two clock

cycles. The user may negate READY for more than one cycle to extend D3 for more than two clock cycles.

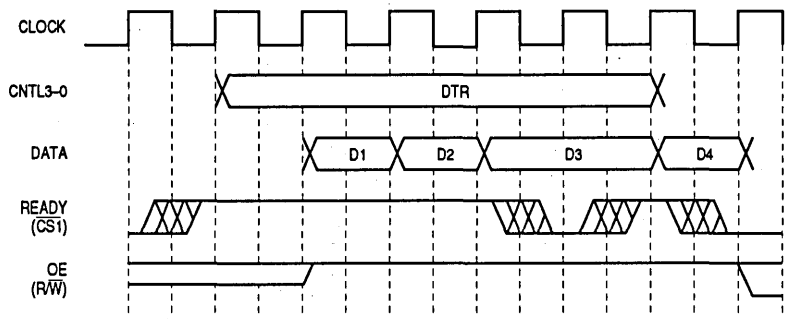


Figure 6-17. Pipeline Read with Wait State Timing

Figure 6-18 describes a pipeline write operation without wait state(s). Data in is sampled by the input latch on the rising edge of the clock. There is no wait state because READY is asserted during the entire cycle.

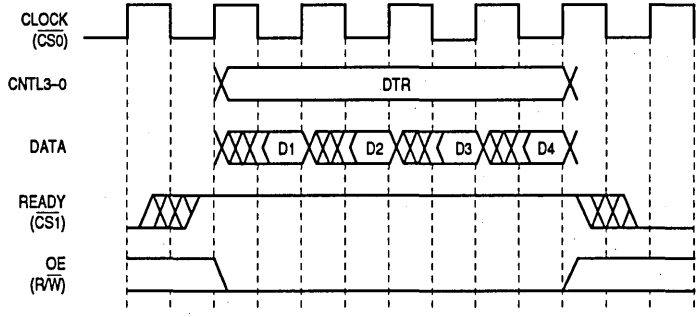


Figure 6-18. Pipeline Write without Wait State Timing

Figure 6-19 describes a pipeline write operation with wait state(s). Data out is sampled by the input latch on the rising edge of the clock. The negation of READY during D2 disables the input latch from latching D2. D2 is latched on the next clock cycle. The user may negate READY for more than one cycle to extend D2 for more than two clock cycles.

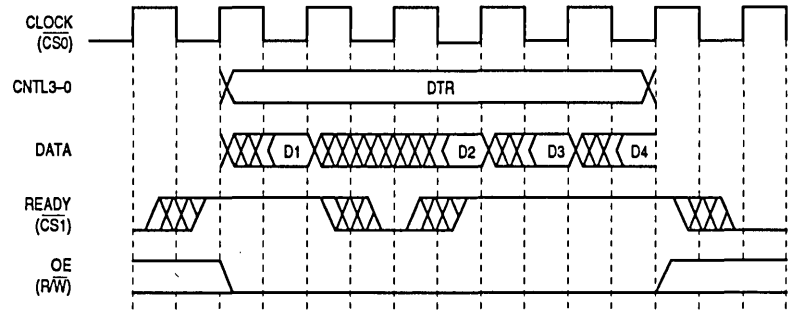


Figure 6-19. Pipeline Write with Wait State Timing

Figure 6-20 describes a register read operation. The user places the register address on the CNTLx lines. On the following cycle, the register contents are placed on the data bus. The user may negate READY to extend the data for more than one clock cycle.

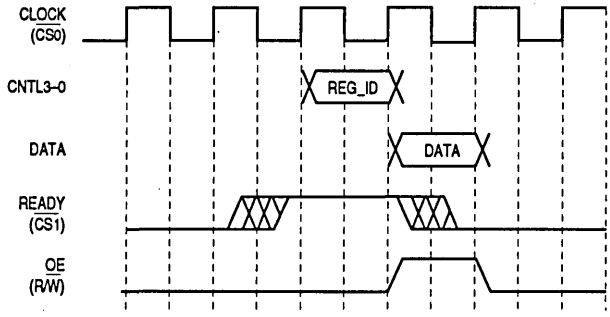


Figure 6-20. Register Read Timing

Figure 6-21 describes a register write operation. This operation uses the multiple cycle mechanism. On the first cycle, the user places the register address on the control lines, and on the second cycle, the user places the REGWR code. READY is negated between the two cycles. The data is written to the register on the clock rising edge of the second cycle. The first cycle may be stretched by not changing the control lines to REGWR. The second cycle may be extended by negating READY for more then one clock cycle.

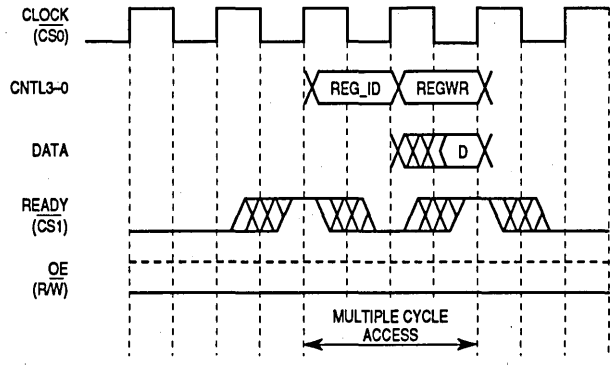


Figure 6-21. Register Write Timing

6.3.10.1 CAMEL DIRECT ACCESS IN PIPELINE MODE. Figure 6-22 describes a CAMEL internal register read operation in pipeline mode. The user places the register address on the control lines (ACNTL8 = 1, ACNTL7–ACNTL0 = CAMEL register address) a setup time before the clock negation, and extends the cycle to at least Tx ns by negating the READY line ($\overline{CS1}$). On the cycle following READY assertion, the register contents are placed on the data bus.

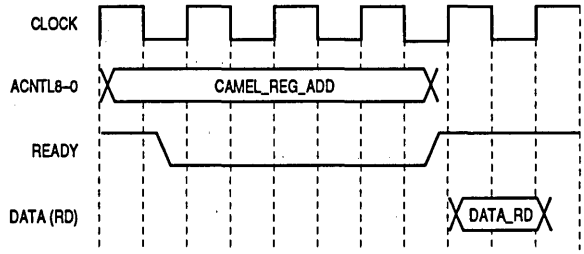


Figure 6-22. Read CAMEL Internal Register In Pipeline Mode

Figure 6-23 describes a CAMEL internal register write operation in pipeline mode. This operation uses the multiple cycle mechanism. On the first cycle, the user places the register address on the control lines (ACNTL8 = 1, ACNTL7–ACNTL0 = CAMEL register address) a setup time before the clock negation, and extends the cycle to at least Tx ns by negating the READY line. On the second cycle, the user places the REGWR code on the CNTLx lines, and again extends the cycle to at least Tx ns by negating READY. Note that the data should be valid during the REGWR cycle.

For CAMEL direct access in normal mode, see **6.2.5 Functional Port Operation Examples—Normal Mode.**

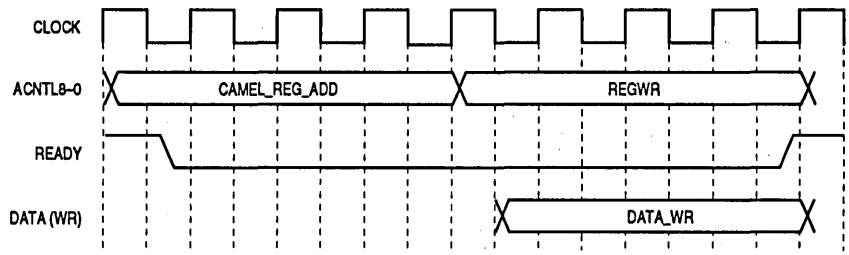


Figure 6-23. Write CAMEL Internal Register In Pipeline Mode

6.3.10.2 ACCESS TIME OF THE CAMEL DIRECT ACCESS. The access time of the CAMEL direct access mode is long because synchronization to BYTCLK is needed.

If only the CAMEL direct access is used to access the CAMEL, the access time will be:

Tx = ~4 x BYTCLK_PERIOD

Tx = ~4 x 80 = 320 ns @ BYTCLK = 12.5 MHz

If the user chooses to use CAMEL direct access and also the FSI control register (FCR) (16-bit configuration) or CONTROL REGISTER WRITE command (issued by the host processor through one of the CMRs or placed inside one of the transmit buffer descriptor rings), the access time will be:

Tx = ~5 x BYTCLK_PERIOD

Tx = ~5 x 80 = 400 ns @ BYTCLK = 12.5 MHz

6.3.10.3 BURST READ CYCLE TIMING. Figure 6-24 describes a burst read cycle operation. During a PNOP access, the port changes the $\overline{\text{REQx}}$ lines to a new write. A BADR access is then issued by the user's control logic. In the BADR access cycle, the port places the size on $\overline{\text{REQ3}}$–$\overline{\text{REQ1}}$. The limit counter is loaded with the size value, and the port is "locked" for the entire burst (i.e., no switching from read to write). On the next clock rising edge, the port drives the address on the data bus. The user then issues data accesses (DTRs) until the limit counter expires on the fourth access. The user will recognize this state by the negation of limit ($\overline{\text{REQ3}}$).

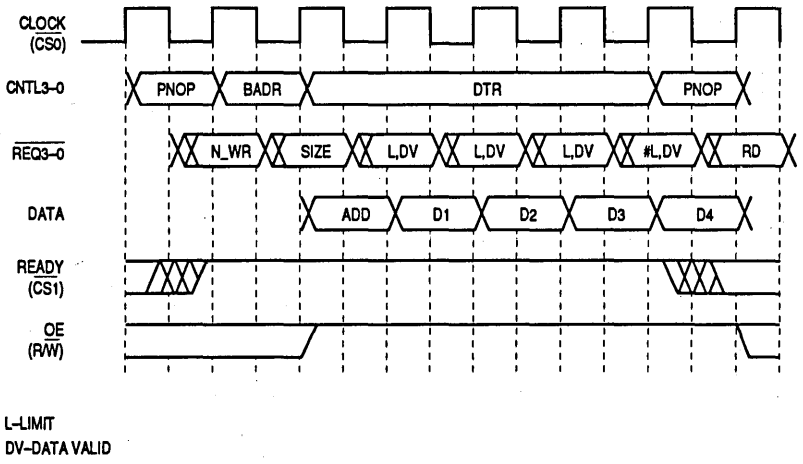


Figure 6-24. Burst Read Cycle Timing

During the data accesses, the port drives the data bus on each rising edge of the clock. On the first data access (DTR), the $\overline{\text{REQx}}$ lines will be switched from normal mode to burst mode. They will remain in burst mode until the user issues a PNOP access.

6.3.10.4 BURST WRITE CYCLE TIMING. Figure 6-25 describes burst write cycle operation. During a PNOP access, the port changes the $\overline{\text{REQx}}$ lines to a new read. The user then issues a BADR access. In the BADR access cycle, the port places the size of the transfer on $\overline{\text{REQ3}}$–$\overline{\text{REQ1}}$. The limit counter is loaded to the size value, and the port is "locked" for the entire burst (i.e., no switching from write to read). On the next clock rising edge, the port drives the address on the data bus. The user then issues a data access (DTR), and the limit counter expires. The user will recognize this state by the negation of limit ($\overline{\text{REQ3}}$). The READY line is negated to extend the DTR access since the address is driven on the bus during the first DTR cycle, which occurs whenever the bus direction is switched from read to write. This is because the data of a read access is on the bus in the clock following CNTL3-0, while the data of a write access in on the bus in the same clock cycle as CNTL3-0.

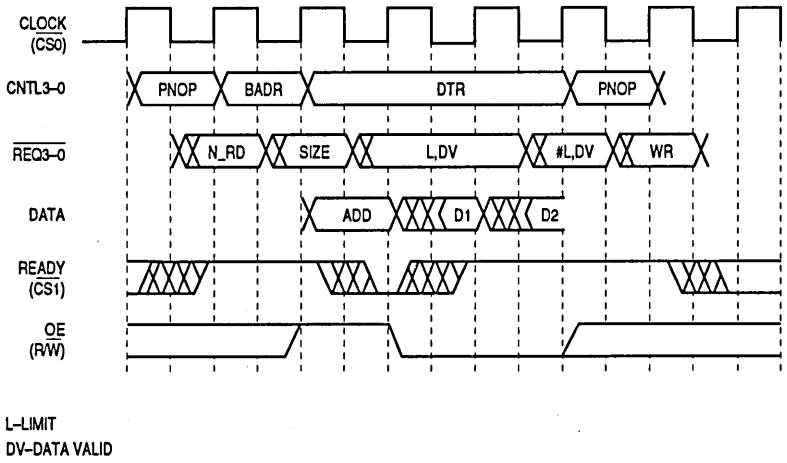


Figure 6-25. Burst Write Cycle Timing

During the data accesses, the port samples the data bus on each rising edge of the clock. On the first data access (DTR), the REQx lines will be switched from normal mode to burst mode. They will remain in burst mode until the user issues a PNOP access.

6.3.11 S-Bus Timing Examples

Figures 6-26 and 6-27 illustrate two examples of IFDDI operation in burst mode on the S-bus.

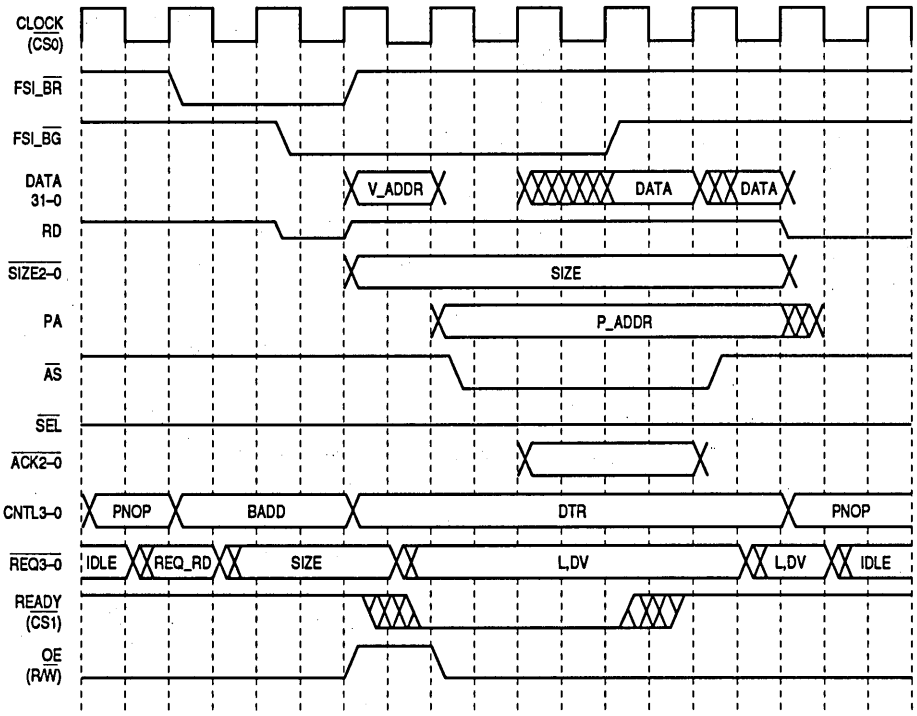


Figure 6-26. S-Bus Read Cycle (FSI Write Cycle)

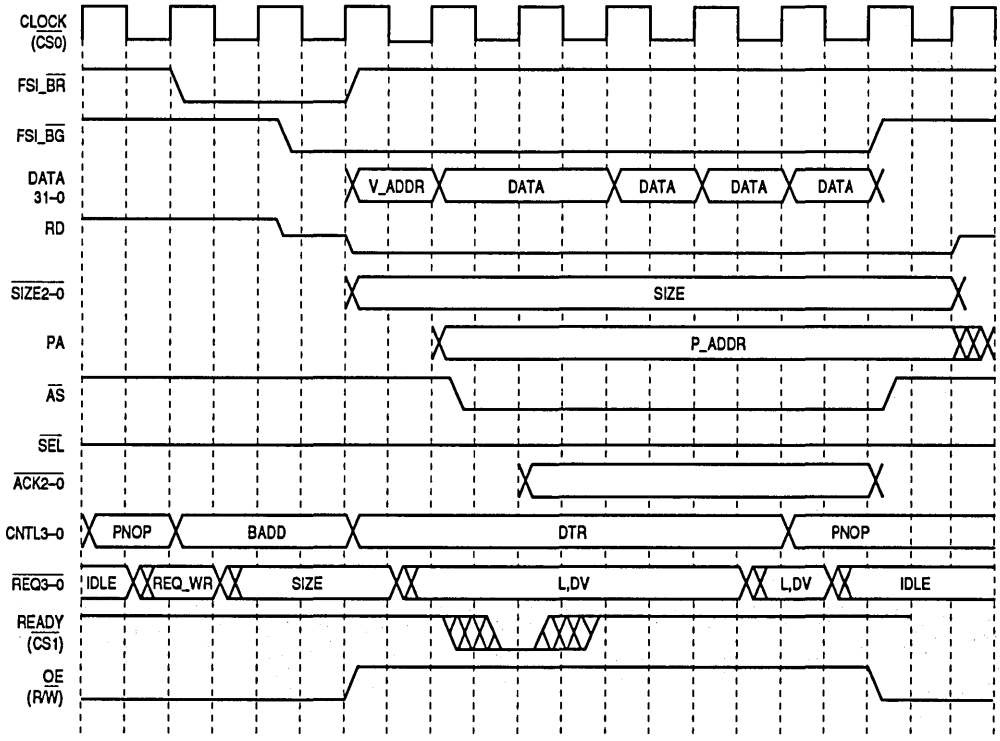


Figure 6-27. S-Bus Write Cycle (FSI Read Cycle)

SECTION 7
REGISTER DESCRIPTION

The MC68840 IFDDI contains several registers that may be accessed by the host processor and the bus control logic through both the port A and port B data buses. The selection, programming, and/or reading of the registers is accomplished through either port.

7.1 REGISTER ACCESSING METHODS

There are two register accessing methods provided by the IFDDI: direct and indirect. The IFDDI registers are divided into three groups:

1. FSI Directly Accessed Registers
2. FSI Indirectly Accessed Registers
3. CAMEL Registers

The direct access method can be used to access both the IFDDI directly accessed registers and the CAMEL registers. The indirect access method can be used to access the CAMEL registers and IFDDI indirectly accessed registers. There are two indirect access methods available to the user. First, the registers can be accessed indirectly through the FSI control register (FCR). Second, the registers can be accessed indirectly through the command register (CMR/CER) for a write operation only. Figure 7-1 illustrates the relationship between the accessing methods and the registers they relate to.

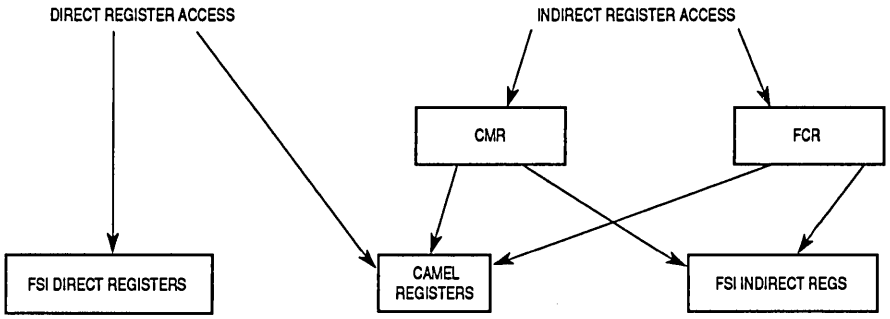
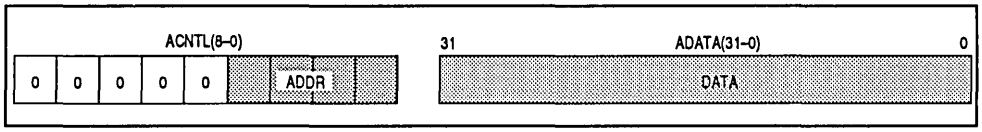


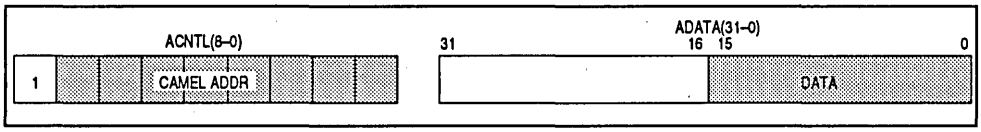
Figure 7-1. Register Access Methods

Each register is provided with an address. The FSI directly accessed register addresses are contained in the register map provided in Tables 7-1 and 7-2. IFDDI multiple cycle access codes are listed in Table 7-3. The FSI indirectly accessed register addresses are provided in Table 7-5. The CAMEL register addresses can be found in Tables 7-6 to 7-8.

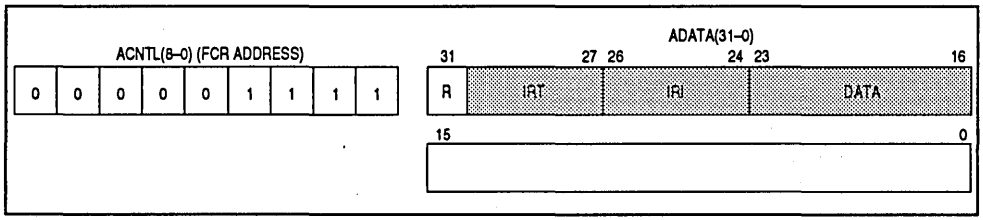
To access one of the registers, locate the register address in the appropriate register map and insert the address in the template that describes the access method desired. The templates are provided in Figure 7-2.



(a) FSI Direct Register Access

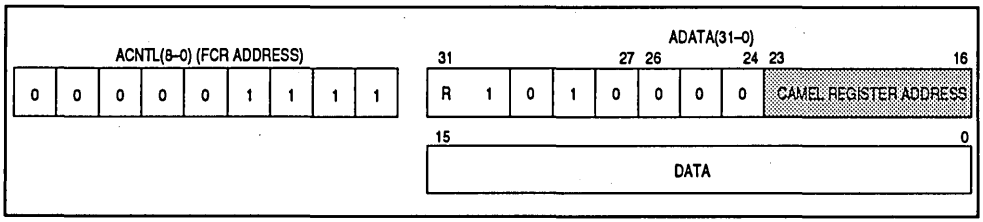


(b) CAMEL Direct Register Access



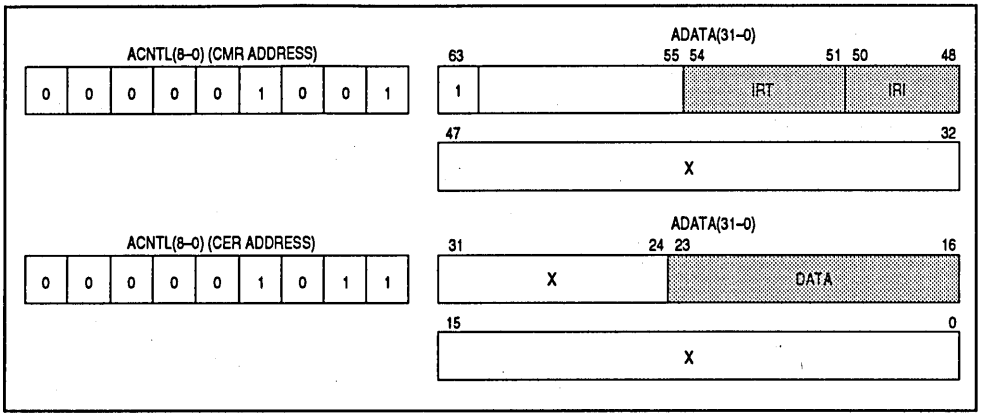
NOTE: IN FCR READ OPERATIONS, THE DATA TO BE READ IS VALID ON ADATA(23-16) WHEN THE CONTROL REGISTER FREE BIT IN STATUS REGISTER 1 IS SET TO ONE AFTER A READ OPERATION IS REQUESTED. BIT 31 SHOULD BE EQUAL TO ONE WHEN REQUESTING A READ FROM THE FCR AND SHOULD BE ZERO WHEN PERFORMING A WRITE OPERATION.

(b) FCR Indirect Register Access

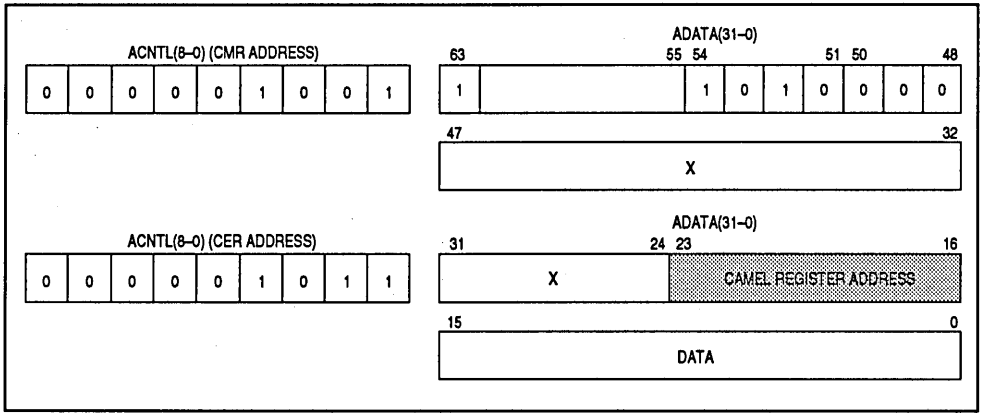


(c) CAMEL Indirect Register Access

Figure 7-2. Register Access Templates



(d) FCRs Through the CMR



(e) CAMEL Registers Through the CMR

Figure 7-2. Register Access Templates (Continued)

NOTE

Please note that the accesses shown in Figure 7-2 relate only to port A. Port B accesses are identical to port A in that BCNTL(3–0) are identical to ACNTL(3–0). There are no BCNTL(8–4).

Table 7-1. Directly Accessed Register Map—Normal Mode

ACNTL (8–0)	BCNTL (3–0)	Read	Write
0 0000 0000	0000	NOP	NOP
0 0000 0001	0001	SR1	SR1
0 0000 0010	0010	DTR	DTR
0 0000 0011	0011	IOR	IOR
0 0000 0100	0100	ADR	Reserved
0 0000 0101	0101	IMR1	IMR1
0 0000 0110	0110	Reserved	Reserved
0 0000 0111	0111	PSR	PSR
0 0000 1000	1000	Other Port's Address	Reserved
0 0000 1001	1001	CMR	CMR
0 0000 1010	1010	PNOP	PNOP
0 0000 1011	1011	CER	CER
0 0000 1100	1100	IMR2	IMR2
0 0000 1101	1101	SR2	SR2
0 0000 1110	1110	ABORT	ABORT
0 0000 1111	1111	FCR	FCR
0 0001 XXXX	—	Reserved	Reserved
0 001X XXXX	—	Reserved	Reserved
0 01XX XXXX	—	Reserved	Reserved
0 1XXX XXXX	—	Reserved	Reserved
1 YYYY YYYY	—	CAMEL Direct Access	CAMEL Direct Access

NOTES:

1. YYYY YYYY = CAMEL block register address (see **7.3 CAMEL Register Set**).
2. The port A CNTL lines are set only when accessing through port A; likewise, for port B.

Table 7-2. Directly Accessed Register Map—Pipeline Mode

ACNTL (8–0)	BCNTL (3–0)	Read	Write
0 0000 0000	0000	NOP	NOP
0 0000 0001	0001	SR1	SR1
0 0000 0010	0010	DTR	DTR
0 0000 0011	0011	IOR	IOR
0 0000 0100	0100	BADR	BADR
0 0000 0101	0101	IMR1	IMR1
0 0000 0110	0110	Reserved	Reserved
0 0000 0111	0111	PSR	PSR
0 0000 1000	1000	Other Port's Address	Reserved
0 0000 1001	1001	CMR	CMR
0 0000 1010	1010	PNOP	PNOP
0 0000 1011	1011	CER	CER
0 0000 1100	1100	IMR2	IMR2
0 0000 1101	1101	SR2	SR2
0 0000 1110	1110	Reserved	Reserved
0 0000 1111	1111	FCR	FCR
0 0001 XXXX	—	Reserved	Reserved
0 001X XXXX	—	Reserved	Reserved
0 01XX XXXX	—	Reserved	Reserved
0 1XXX XXXX	—	Reserved	Reserved
1 YYYY YYYY	—	CAMEL Direct Access	CAMEL Direct Access

NOTES:

1. YYYY YYYY = CAMEL block register address (see **7.3 CAMEL Register Set**).
2. The port A CNTL lines are set only when accessing through port A; likewise, for port B.

Table 7-3. IFDDI Multiple Cycle Access Codes

ACNTL (8–0)	BCNTL (3–0)	Write
X XXXX XXX1	XXX1	POE
0 XXXX 0010	0010	REGWR

7.2. FSI ASSOCIATED REGISTER SET

The register set associated with the FSI portion of the IFDDI consists of two types: a directly addressable set and an internal, indirectly addressable set. The latter set is accessed through the FSI control register (FCR). Figure 7-3 depicts the register set relevant to the system interface block of the IFDDI.

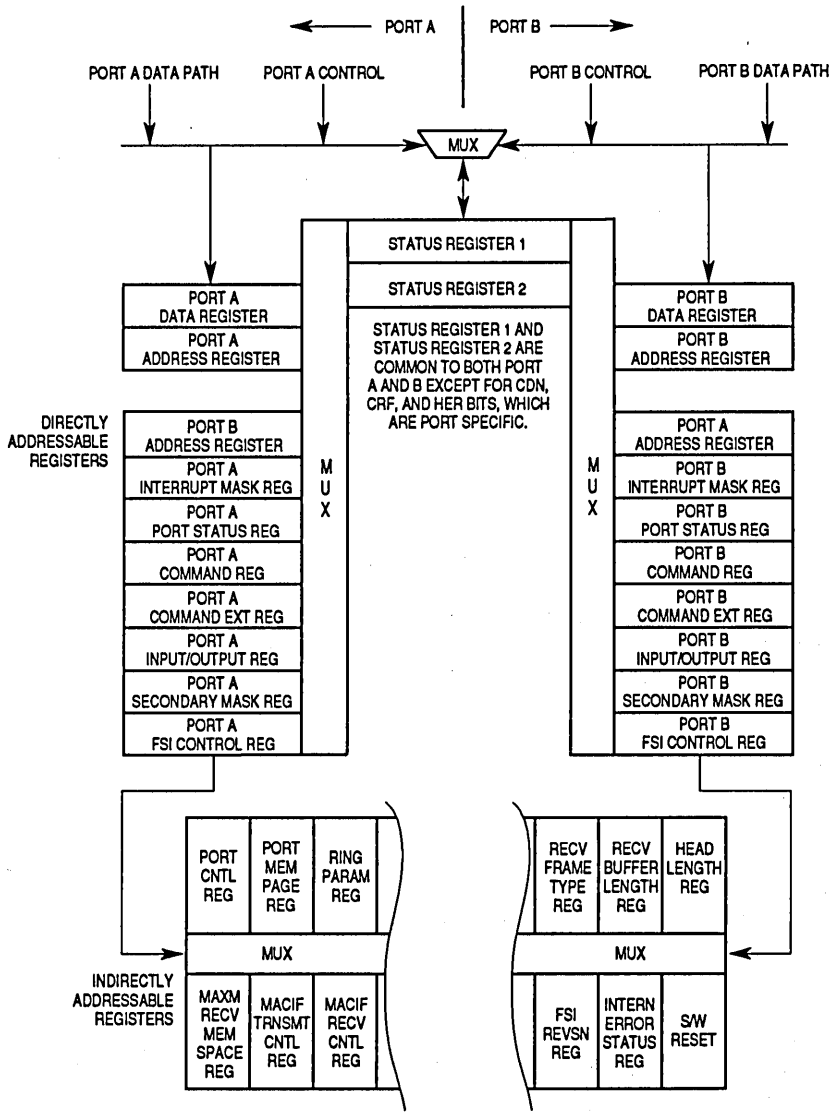


Figure 7-3. System Interface Block Internal and Common Registers

7.2.1 FSI Associated Directly Accessed Registers

The directly accessed registers are discussed in the following subsections.

7.2.1.1 STATUS REGISTER 1 (SR1) xCNTL3–0 = 0001. This register includes all the FSI relevant general status information that may cause an interrupt if the appropriate interrupt

mask register 1 bit is set. There is only one SR1 in the FSI portion of the IFDDI, and it may be accessed from either port A or port B. The CDN, CRF, and HER bits are port specific and indicate the status of the port that is reading the SR1.

The bits in this register are divided between real status bits, which are set and cleared by the IFDDI according to the status of the item, and sticky bits, which may only be set by the IFDDI when the appropriate conditions occur. These bits are cleared by a host processor write access to the SR1 with the corresponding bit set.

The bit numbers following the bit descriptions correspond to the appropriate ring number—receive rings 5 and 4 and transmit rings 3, 2, 1, and 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CDN	CRF	HER	IOE	ROV5	ROV4	POEB	POEA	CIN	STE	RER5	RER4	RER3	RER2	RER1	RER0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		RXC5	RXC4	RCC3	RCC2	RCC1	RCC0	0		RNR5	RNR4	RNR3	RNR2	RNR1	RNR0

CDN—Command Done

When the SR1 is read from one of the ports, the CDN bit references the command register of this port. When a command is written to the command register, its OWN bit should be set, which causes a reset of the CDN bit. When command execution has been completed, the IFDDI writes the indication to the command register with the OWN bit clear, which causes the CDN to be set. CDN is a real-time status bit and is not affected by writing to the SR1.

CRF—Control Register Free

When the SR1 is read from one of the ports, the CRF bit indicates the status of the respective port's FSI control register (FCR). When the CRF bit is set, the FCR of the port may be written. Note that there are many internal control registers that may be set through a control register access. These internal control registers can be accessed through either port. The CRF bit is a real-time status bit and is not affected by writing to the SR1.

HER—Host Error

The HER bit is set when a parity error is detected (if the parity checking is enabled) during a write access to a directly accessible register. This bit is a sticky bit and must be cleared by the host.

IOE—Internal Operation Error

The IOE bit is set when an internal operation error is detected. This bit is a sticky bit and must be cleared by the host.

Internal errors may be detected in the following situations:

a. The MAC interface recognizes a parity error on transmit data. An internal parity error detected during data reads from the internal memory will always indicate a problem in an internal FSI core data path. In this case, the transmit parity error (TPE) bit in

the internal error status register (IER) is set, the frame currently being transmitted is aborted with an appropriate indication, and the transmit enable (TE) bit in the MAC interface transmit control register (MTR) is reset.

b. The port interface recognizes a parity error on received data being transferred to the system bus. The port internal error (PAE/PBE) bit is set in the internal error status register (IER).

c. The MAC interface experiences an internal underrun or an internal overrun. During normal operation, the main controller ensures a maximum response time for data transfers between the MAC interface and internal memory. If this response time exceeds these limits, this may indicate that the main controller may not be performing properly. As a result, the internal overrun error (IOE) bit or the internal underrun error (IUE) bit is set in the internal error status register (IER). If an internal underrun occurred, the transmitting frame is aborted with the appropriate indication, and the transmitter is disabled (TE reset). If an internal overrun occurred, the receiver is disabled (RE4 and RE5 cleared).

d. The MAC interface receiver recognizes a protocol handshake error between the MAC and the FSI. In this case, the MAC error (MER) bit in the internal error status register (IER) is set. The MAC interface will generate an RABORT signal to the MAC and will try to resynchronize with the MAC on the next frame. This error may be caused by temporary problems (e.g., noise on control lines); therefore, no special recovery functions are required. If there are permanent problems in the FSI to MAC interface (e.g., control lines are disconnected or forced to some constant value), the host processor should perform an external diagnostic. Note that RABORT and the FSI/MAC bus are internal buses only.

e. The FSI core has experienced an internal memory overrun causing the memory overrun (MOV) bit in the internal error status register to be set. This error may be caused by an incorrect definition of an internal FIFO watermark. Therefore, the FSI relevant parameter definitions should be checked, and the FSI should be reinitialized.

ROV5–ROV4—Receive Overrun Error

There is one ROV bit for each receive ring. Each bit is set when an overrun condition occurs for its ring. These bits are sticky and must be cleared by a host write to the SR1. Receive overrun is a temporary ring state in which there is no space for incoming frames and they are then discarded. The host does not have to take any action on this condition since the IFDDI will recover from the condition and begin inputting frames to the IFDDI as soon as there is available space for them.

POEB—Port Operation Error B

The POEB bit is set when an operation error is recognized during a port B operation. The POEB bit is a sticky bit and must be cleared by the host.

POEA—Port Operation Error A

The POEA bit is set when an operation error is recognized during a port A operation. The POEA bit is a sticky bit and must be cleared by the host.

CIN—CAMEL Interrupt

The CIN bit will be set to notify an external processor of the occurrence of a CAMEL interruptible event. The CIN bit will remain set until the appropriate CAMEL interrupt register is read to clear the interrupting event(s) or until the interrupt is masked by writing a zero to the appropriate CAMEL interrupt mask register bit. The CIN is a real status bit, and it is not affected by writing to the SR1.

STE—SMT Counter Expired

The STE bit is set when the SMT counter expires. This bit is a sticky bit and must be cleared by the host.

RER5–RER0—Ring Error

The RER bit is set if a parity error, first or last bit error, or port operation error has been recognized during an access to the entry of a ring. These bits are sticky bits and must be cleared by the host.

RXC5–RXC4—Receive Complete

Each RXC bit refers to one of the receive rings. The RXC bit is set when a buffer indication is written with the last bit set to the ring (i.e., the receive frame has been completely transferred to the external memory and passed to the system processor). These bits are sticky bits and must be cleared by the host.

RCC3–RCC0—Ring Command Complete

Each RCC bit is referenced to one of the transmit rings. The RCC bit is set in two cases:

1. The indication of a transmit frame with the last bit set has been written back into the ring (i.e., a frame has been completely transmitted).
2. A command from a transmit ring has been executed and its indication has been written back into the ring.

These bits must be cleared by the host.

RNR5–RNR0—Ring Not Ready

Each RNR bit is set when any condition in the FSI core causes the ring to become not ready, even if the ring is currently not ready. These bits are sticky bits and must be cleared by the host.

7.2.1.2 INTERRUPT MASK REGISTER 1 (IMR1) xCNTL3–0 = 0101. There are two IMR1s, one for each port. There is an interrupt enable bit for each status bit in the SR1. The FSI core will furnish an interrupt when both the SR1 bit and its respective interrupt enable bit in that port's IMR1 are set. If the same interrupt enable bit is set in both IMR1s, the interrupt is generated on both ports as a result of an SR1 bit becoming set.

The IMR1 may be read at any time; therefore, a read-modify-write operation may be performed by the host processor to change the interrupt mask. The IMR1s are cleared on power-up reset and are unaffected by a software reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CDE	CFE	HEE	IEE	RVE5	RVE4	PEEB	PEEA	CIE	SIE	REE5	REE4	REE3	REE2	REE1	REE0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		RXE5	RXE4	RCCE3	RCCE2	RCCE1	RCCE0	0		RNE5	RNE4	RNE3	RNE2	RNE1	RNE0

CDE—Command Done Interrupt Enable
When the CDE and CDN are both set, an interrupt is generated.

CFE—Control Register Free Interrupt Enable
When the CFE and CRF are both set, an interrupt is generated.

HEE—Host Error Interrupt Enable
When the HEE and HER are both set, an interrupt is generated.

IEE—Internal Operation Error Interrupt Enable
When the IEE and IOE are both set, an interrupt is generated.

RVE5–RVE4—Receive Overrun Error Interrupt Enable
When the RVEx and ROVx are both set, an interrupt is generated.

PEEB—Port Operation Error B Interrupt Enable
When the PEEB and POEB are both set, an interrupt is generated.

PEEA—Port Operation Error A Interrupt Enable
When the PEEA and POEA are both set, an interrupt is generated.

CIE—CAMEL Interrupt Enable
When the CIE and CIN are both set, an interrupt is generated.

SIE—SMT Timer Expired Interrupt Enable
When the STE and SIE bits are both set, an interrupt is generated.

REE5–REE0—Ring Error Interrupt Enable
When the REEx and RERx are both set, an interrupt is generated.

RXE5–RXE4—Receive Complete Interrupt Enable
When the RXEx and RXCx are both set, an interrupt is generated.

RCCE3–RCCE0—Ring Command Complete Interrupt Enable
When the RCCEx and RCCx are both set, an interrupt is generated.

RNE5–RNE0—Ring Not Ready Interrupt Enable

When the RNEx and RNRx are both set, an interrupt is generated.

7.2.1.3 STATUS REGISTER 2 (SR2) xCNTL3–0 = 1101. This register includes the FSI relevant status information that may cause an interrupt if the appropriate interrupt mask bit is set. There is only one SR2 in the FSI. This register may be accessed from either port A or port B.

All the bits are sticky bits that may only be set by the FSI core when the appropriate conditions occur. They are cleared by a host processor write access to the SR2 with the corresponding bit set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	DRE7	DRE6	0	0	DRE3	DRE2	DRE1	DRE0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DXC7	DXC6	0	0	DXC3	DXC2	DXC1	DXC0	DNR7	DNR6	0	0	DNR3	DNR2	DNR1	DNR0

INT7–INT0—User-Defined Interrupt

Each status bit may be set by writing to the SIG internal control register. These interrupts may be used for signaling between host and node processors.

DRE (7, 6, 3–0)—Destination Ring Error

The DRE bit is set if a parity or port operation error has been recognized during an access to the entry of a destination ring.

DXC (7, 6, 3–0)—Destination Transfer Complete

Each DXC bit is referenced to one of the destination rings. The DXC bit is set when the last indication (with the LAST bit set) has been written back into the destination ring (i.e., a frame has been completely transferred).

DNR (7, 6, 3–0)—Destination Ring Not Ready

Each DNR bit is set when its destination ring has changed its state from ready to empty.

7.2.1.4 INTERRUPT MASK REGISTER 2 (IMR2) xCNTL3–0 = 1100. There are two IMR2s, one for each port. There is an interrupt enable bit for each status bit in the SR2. The FSI core will furnish an interrupt when both the SR2 bit and its respective interrupt enable bit in that port's IMR2 are set. If the same interrupt enable bit is set in both IMR2s, the interrupt is generated on both ports as a result of an SR2 bit becoming set.

The IMR2 may be read at any time; therefore a read-modify-write operation may be performed by the host processor to change the interrupt mask. The IMR2s are cleared on power-up reset and are unaffected by a software reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INE7	INE6	INE5	INE4	INE3	INE2	INE1	INE0	DREE7	DREE6	0	0	DREE3	DREE2	DREE1	DREE0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DXCE7	DXCE6	0	0	DXCE3	DXCE2	DXCE1	DXCE0	DNRE7	DNRE6	0	0	DNRE3	DNRE2	DNRE1	DNRE0

INE7–INE0—User-Defined Interrupt Enable

When the INEx and INTx are both set, an interrupt is generated.

DREE (7, 6, 3–0)—Destination Ring Error Interrupt Enable

When the DREEx and DREx are both set, an interrupt is generated.

DXCE (7, 6, 3–0)—Destination Ring Transfer Complete Interrupt Enable

When the DXCEx and DXCx are both set, an interrupt is generated.

DNRE (7, 6, 3–0)—Destination Ring Not Ready Interrupt Enable

When the DNREx and DNRx are both set, an interrupt is generated.

7.2.1.5 COMMAND REGISTER (CMR) xCNTL3–0 = 1001. There are two CMRs, one for each port. The CMR is used for direct command access to the FSI core, as opposed to a command that is presented in the transmit descriptor ring. The CMR may be written by the host processor anytime the CMR is available—i.e., when its CDN bit is set in the SR1 and the required port (indicated by the CDA bit in the CPR) is enabled (PE bit in the corresponding PCR is set to one). A write access to the CMR when the respective command done (CDN) bit is cleared is not allowed. If both CMRs are used for commands, the following cases may occur:

a. If both instructions require the same operation (i.e., two transmit frame commands or two ring handling commands to the same ring), then these commands are executed sequentially.

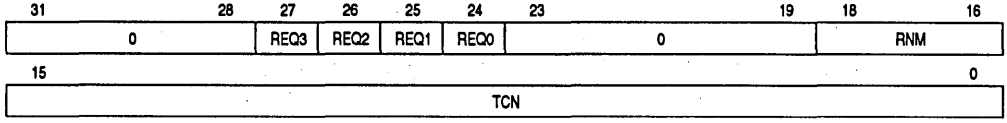
b. If the commands can be executed together (i.e., a transmit command and CAM command, etc.), the FSI core will execute them concurrently.

The command written to the CMR is normally associated with the command extension written to the command extension register (CER), which contains the second long word of the command entry. This second long word is usually the address of a data buffer or a ring. A write access to the CMR will cause the FSI core to take this command and begin execution. Therefore, the CER should be written prior to the CMR write and should not be changed until the COMMAND DONE indication has been received. Similar to command reads from rings, direct commands written directly to a CMR have an OWN bit that must be written as a one by the host processor. The OWN bit is cleared by the FSI core when the command has been executed, and an indication is written to the CMR and to the CER, if applicable. The CMR may be read by the host at any time.

7.2.1.6 COMMAND EXTENSION REGISTER (CER) xCNTL3–0 = 1011. There are two CERs, one for each port. Each is related to its respective CMR and contains the second

long word, if required, of a command entry. The CER should be changed only when the CDN bit associated with the appropriate CMR is set, (i.e., the previous command is done). The CER should be written prior to writing its associated command to the CMR. If a parity error occurs (indicated by the HER bit in the port's SR1) during the CER access, the CER should be rewritten prior to writing to the CMR.

7.2.1.7 PORT STATUS REGISTER (PSR) xCNTL3–0 = 0111. There are two PSRs, one for each port. Each PSR indicates the current port status and may be read by the host processor at any time. The PSRs are read-only registers.



7

REQ3–REQ0—Request

These four status bits show the actual state of the request output lines of this port. REQ3 differentiates between descriptors or indications and data transfers:

0 = Descriptors or Indications
1 = Data Transfers

The encoding of REQ2–REQ0 is as follows:

000 = Idle State
100 = Read Cycle
101 = Read Cycle with Address on the Same Memory Page
010 = Write Cycle
011 = Write Cycle with Address on the Same Memory Page

RNM—Ring Number

RNM indicates the number of the ring currently being accessed.

000 = Transmit Ring 0
001 = Transmit Ring 1
010 = Transmit Ring 2
011 = Transmit Ring 3
100 = Receive Ring 4
101 = Receive Ring 5
110 = Command Register A
111 = Command Register B

TCN—Transfer Counter

The TCN has different meanings in the following situations:

a. For a transmit data read operation, the TCN contains the number of bytes remaining to be transferred in the current transmit data buffer.

b. In the case of a receive data write operation, the TCN contains the number of bytes remaining to be transferred to the external memory for the frame currently being received.

c. During a ring FIFO read operation, the TCN will indicate the number of bytes that the FSI core expects to read from the ring.

d. The TCN has no meaning during a ring write operation.

Note that the operation of the port may be switched between various states according to internal priorities and algorithms. Therefore, in cases where several activities are occurring inside the FSI core, the port context may be switched before the transfer counter reaches zero.

7.2.1.8 THIS PORT'S ADDRESS REGISTER (ADRx) xCNTL3–0 = 0100. Each port has an address register that is used by the port control unit to access external memory. The ADRx contains the contents of the port's address generator.

The address generator has two modes of operation, depending on the nature of the access: descriptor ring access or data access. During a descriptor ring access, the ring read address or the ring write address is used as defined in the DEFINE RING command. The ring address is then incremented, as required, up to the user-defined ring maximum length (RML) value before wrapping at the specified limit.

During a data access, the address is generated from the address field of a buffer descriptor and is incremented after every data access. When the transfers are to or from local memory, the generated address will wrap at the boundary of the local memory space. Accesses to the address generator have no influence on the address itself. Only a successful data access causes a change in the address. The address is advanced according to the data access width (32 or 64 bits). The address is not updated in the following situations:

a. When reading the entries of the ring, the ring read address will not be incremented if the read entry does not belong to the FSI (i.e., the ring became empty).

b. When reading the entries of the ring, the ring read address will not be incremented if a parity error or port operation error is recognized. The address will continue to point to the descriptor that caused the error.

c. When a FIRST or LAST bit error situation is encountered (see **Appendix E Error Discussion**).

The address register contents have no meaning when the external request lines are in the IDLE state.

This register description for this address is valid only in normal mode. For pipeline mode, see **6.5.4 Burst Operation.**

7.2.1.9 THE OTHER PORT'S ADDRESS REGISTER (ADRx) xCNTL3–0 = 1000. The address generator of one port may be accessed through the other port to allow simultaneous access of address and data (i.e., as in the case of a nonmultiplexed 32-bit address and data external bus).

7.2.1.10 DATA REGISTER (DTR) xCNTL3–0 = 0010. Each port has its own 32-bit data register used to transfer either data for transmit and receive frames or commands, descriptors, and indications from/to transmit and receive rings.

The allowable access type (read/write) to the data register depends upon the current port state and the operation desired. If an IFDDI port is in the read state, the system needs to furnish transmit data or transmit commands and descriptors and is only allowed to perform a write access. Note that in all descriptions, port states (read or write) are relative to the FSI. Conversely, if an IFDDI port is in the write state, the system is expected to process receive data or indications and is only allowed to perform a read access. Any attempt to access the data register in a different manner may result in conflicting behavior of the IFDDI.

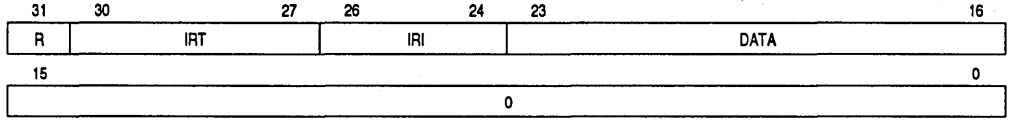
The data register is 32-bit. When the port's data registers are concatenated in 64-bit operation mode, the port A data register holds the most significant portion of the data, and the port B data register holds the least significant portion of the data. The data register access will update the address of the corresponding port. In 64-bit mode, the address is generated in port A but can be accessed through either port.

7.2.1.11 INPUT/OUTPUT REGISTER (IOR) xCNTL 3–0 = 0011. There are two 32-bit IORs, one for each port. These registers may be written and read by the host processor at any time. The IOR has no meaning for the IFDDI, and it is not changed as a result of IFDDI operation.

The IOR may be used by the host processor to hold an interrupt vector that could be read and used by the host on interrupt detection. Another potential use for the IOR is to extend the address space to 64 bits. On bus accesses, the IOR could be read and used as the upper 32 bits of address as desired.

7.2.1.12 FSI CONTROL REGISTER (FCR) xCNTL3–0 = 1111. The FCR is used to access the internal register set of the IFDDI. The FCR may be written by the host processor at any time when the CRF bit in the SR1 is set.

When the FCR is written with the R-bit (bit 31) as zero, its data portion (bits 23–16) is transferred into one of the internal control registers according to the register identification fields (IRT and IRI) after synchronization to the FSI system clock (FSICLK). When data has been transferred, the relevant CRF bit is set again. To read data from the internal control registers, the FCR should be written with the R-bit set. The FSI system will transfer the value of the relevant internal control register to the FCR, and the appropriate CRF bit is set to indicate that the FCR is ready to be read by the host processor.



R—Read/Write Operation

1 = Read
0 = Write

IRT—Internal Control Register Type

IRI—Internal Control Register ID

DATA—Internal Control Register Data

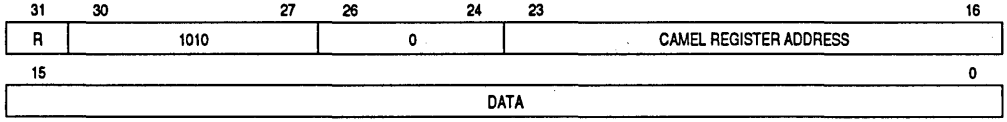
This field contains the data for a register write and the returned data on a register read.

The Table 7-4 lists the internal control registers and their associated IRT and IRI number in hexadecimal:

Table 7-4. Internal Control Registers

Register Name	IRT	IRI	Number of Registers
Port Control Registers (PCR)	B	6, 7	2
Port Memory Page Registers (PMP)	8	6, 7	2
Ring Parameters Registers (RPR)	6	0–5	6
Parameter Extension Registers (PER)	7	0–7	8
Command Parameters Registers (CPR)	6	6, 7	2
Ring State Registers (RSR)	E	0–7	8
FIFO Watermark Registers (FWR)	4	0–7	8
Limit Registers (LMR)	5	0–5	6
Receive Frame Type Registers (RFR)	1	4, 5	2
Receive Buffer Length Registers (RBR)	3	4, 5	2
Header Length Register (HLR)	2	4	1
Maximum Receive Memory Space Registers (RMR)	C	4, 5	2
MACIF Transmit Control Register (MTR)	1	0	1
MACIF Receive Control Register (MRR)	1	1	1
Ring Ready (RDY)	0	0–5	6
Destination Ring Ready (DRY)	9	0–3, 6, 7	6
Signal Register (SIG)	D	0–7	8
User Register (USR)	F	4–5	2
Internal Error Status Register (IER)	F	0	1
Software Reset(SWR)	F	7	1
Burst Limit Registers (BLR)	3	6, 7	2
IFDDI Configuration Register (ICR)	1	2	1
SMT Timer Registers (STR)	2	2, 3	2
SMT Timer Load Value Registers (STL)	2	0, 1	2
IFDDI Revision Register (IREV)	F	3	1
FSI Revision Register (REV)	F	2	1

7.2.1.13 FSI CONTROL REGISTER (FCR)—INDIRECT CAMEL ACCESS—xCNTL 3–0 = 1111. The FCR is also used for an indirect CAMEL access.



R—Read/Write operation
1 = Read
0 = Write

7

CAMEL Register Address—Address of CAMEL Register to Read/Write
The CAMEL register set may also be directly accessed via port A. See Figure 7-2(a) for more information.

DATA—Internal Control Register Data
This field contains the data for a register write and the returned data on a register read.

7.2.2 FSI Indirectly Accessed Registers

The following subsections detail the indirectly accessed internal control registers that are used to control FSI relevant operation and to define various parameters (see Table 7-5). These control registers are 8 bits in width and are accessed indirectly through either port's FSI control register (FCR). The addresses shown refer to the IRT and IRI values in the respective FCR fields.

Table 7-5. FSI Associated Indirectly Accessed Registers

IRT	IRI							
	000	001	010	011	100	101	110	111
0000	Ring Ready Reg 0	Ring Ready Reg 1	Ring Ready Reg 2	Ring Ready Reg 3	Ring Ready Reg 4	Ring Ready Reg 5	—	—
0001	MACIF Transmit Cntl Reg	MACIF Receive Cntl Reg	IFDDI Config Reg	—	Receive Frame Type Reg 4	Receive Frame Type Reg 5	—	—
0010	SMT Timer Load Value Reg 0	SMT Timer Load Value Reg 1	SMT Timer Reg 0	SMT Timer Reg 1	Header Length Reg	—	—	—
0011	—	—	—	—	Rx Buffer Length Reg 4	Rx Buffer Length Reg 5	Burst Limit Reg A	Burst Limit Reg B
0100	FIFO Watermark Reg 0	FIFO Watermark Reg 1	FIFO Watermark Reg 2	FIFO Watermark Reg 3	FIFO Watermark Reg 4	FIFO Watermark Reg 5	FIFO Watermark Reg 6	FIFO Watermark Reg 7
0101	Limit Reg 0	Limit Reg 1	Limit Reg 2	Limit Reg 3	Limit Reg 4	Limit Reg 5	—	—
0110	Ring Parameter Reg 0	Ring Parameter Reg 1	Ring Parameter Reg 2	Ring Parameter Reg 3	Ring Parameter Reg 4	Ring Parameter Reg 5	Command Parameter Reg A	Command Parameter Reg B
0111	Parameter Extension Reg 0	Parameter Extension Reg 1	Parameter Extension Reg 2	Parameter Extension Reg 3	Parameter Extension Reg 4	Parameter Extension Reg 5	Parameter Extension Reg 6	Parameter Extension Reg 7
1000	—	—	—	—	—	—	Port Memory Page Reg A	Port Memory Page Reg B
1001	Destination Ring Ready Reg 0	Destination Ring Ready Reg 1	Destination Ring Ready Reg 2	Destination Ring Ready Reg 3	—	—	Destination Ring Ready Reg 6	Destination Ring Ready Reg 7
1010	—	—	—	—	—	—	—	—
1011	—	—	—	—	—	—	Port Control Reg A	Port Control Reg B
1100	—	—	—	—	Maximum Rx Memory Space Reg 4	Maximum Rx Memory Space Reg 5	—	—
1101	Signal Reg 0	Signal Reg 1	Signal Reg 2	Signal Reg 3	Signal Reg 4	Signal Reg 5	Signal Reg 6	Signal Reg 7
1110	Ring State Reg 0	Ring State Reg 1	Ring State Reg 2	Ring State Reg 3	Ring State Reg 4	Ring State Reg 5	Ring State Reg 6	Ring State Reg 7
1111	Internal Error Status Reg	—	FSI Rev Reg	IFDDI Rev Reg	User Reg 0	User Reg 1	—	Software Reset

7.2.2.1 RING READY REGISTER (RDY) IRT = 0; IRI = 0–5. Ring ready is used to transition the ring specified by the IRI to the ready state. The ring ready action is caused by an access to the register indicated by the IRT and IRI values. If this ring was already in the ready state, the ring ready control will have no effect. There are six ring ready registers. The ring ready access cannot be performed on a ring that does not exist in the system.

7.2.2.2 MACIF TRANSMIT CONTROL REGISTER (MTR) IRT = 1; IRI = 0. This register controls the MACIF transmission.

7	6	5	4	3	2	1	0
TD3	TD2	TD1	TD0	TD7	TD6	0	TE

TD (3–0, 7, 6)—Transmit Disable

There are six TD bits, one for each transmit and command channel. When TD is set, transmit operation of this channel is disabled. When TD is zero, transmit operation of this channel is enabled. Note that disabling one channel transmit operation will not affect the other channels.

TE—Transmit Enable

When the TE bit is set, transmit operation is enabled. When TE is zero, transmit operation is disabled. During the transmit operation, if this bit is reset, the transmitter is disabled after the current transmission has completed. This is a global transmit enable bit.

7.2.2.3 MACIF RECEIVE CONTROL REGISTER (MRR) IRT = 1; IRI = 1. This register controls the MACIF reception.

7	6	5	4	3	2	1	0
RPE	SLF	RMI	RAL	RCM	ROB/IHI	RE5	RE4

RPE—Receive Parity Enable

When RPE is set, the MACIF checks the incoming data parity according to the parity mode specified in the port control register (PCR). If a parity error is detected, the MACIF generates the RABORT signal to the MAC, and an appropriate indication is made for the affected frame. If RPE is zero, the MACIF does not check the incoming data parity. In this case, the parity is generated by the MACIF and is stored in the IFDDI internal memory with the associated data.

SLF—Split Frame

When SLF is set, the MACIF will split all received frames. The header portion of a frame will be directed to channel 4. The rest of the data for this frame will be directed to one of the receive channels (4 or 5) according to its assignment specified in the receive frame

type register (RFR). (RFR) The length of a header is specified in the header length register (HLR).

RMI—Receive My Frames Indications

When RMI is set, the MACIF receives an indication for frames that have been previously transmitted by this station. These indications are transferred to one of the receive rings according to its frame type.

RAL—Receive All

When RAL is set, the MACIF transfers all the data it receives to the internal memory, including remnant frames. This is useful in implementing a network analyzer. When RAL is zero, only valid frames are passed to the internal memory.

RCM—Receive CRC Mode

When RCM is set, the MACIF furnishes the received CRC with the received frame, and the frame length includes four bytes of CRC. When RCM is zero, the CRC is not transferred to the internal memory, and the frame length does not include the four bytes for the CRC. When SLF or RAL is set, the CRC will always be furnished, and the RCM bit has no effect.

ROB/IHI—Receive One Buffer/Immediate Header Indication

This bit has different meanings in split and no-split operations.

In no-split operation, this bit is receive one buffer (ROB). When ROB is set, the MACIF transfers only the first buffer of the frame into the internal memory. The buffer length is defined by the RBR of the ring. However, the indication is furnished as in normal reception. When ROB is zero, the MACIF will transfer all the data of a received frame to the FSI internal memory.

In split operation, this bit is immediate header indication (IHI). When IHI is set, the header of the frame will be released immediately after its reception. When IHI is reset, the header of the frame will not be released until the entire frame is received. Therefore, if the frame was aborted, the header indication will be an error indication.

RE5–RE4—Receive Enable

When REx is zero, the reception of data into the receive FIFOx is disabled. In this case, the MACIF generates the RABORT signal to the MAC if the frame currently being received is directed to receive FIFOx.

7.2.2.4 IFDDI CONFIGURATION REGISTER (ICR) IRT = 1; IRI = 2. This register is used for IFDDI configuration setup.

7	6	5	4	3	2	1	0
0	0	CTE	CDS	COM	CSM	CEO	DBM

CTE—CAMEL Test Enable

This bit is used for testing. In normal use this bit should be zero.

CDS—CAM Disable

This bit disables the CAM operation. When this bit is one, the CAM ignores the LDADDR input pin, and the MATCH output pin will not be asserted .

COM—CAM Operation Mode

When this bit is zero, the CAM is used for 48-bit address comparison. When this bit is one, the CAM may be used for non-FDDI applications (see **Appendix A System Configurations**).

CSM—Counter Segmentation Mode

The CSM bit is connected to the CAMEL CTR_SEG_MODE input pins. This pin is used to select between CAMEL normal operation (CSM = 0) mode and the counter segmentation test mode (CSM = 1).

CEO—CAMEL ENCOFF

This bit is connected to the CAMEL ENCOFF input pin. This pin controls the encoding/decoding function of the ELM. When ENCOFF is one, the encoding and the decoding are disabled, allowing for the transmission of any symbols, including INVALID symbols (for diagnostic purposes). When ENCOFF is zero, the encoding and the decoding are enabled.

DBM—Double Buffer Mode

This bit controls the use of the Double Buffer mode of the IFDDI. When this bit is one, burst transfers will be performed using a double buffer in the IFDDI internal memory. For more information, see **Section 5.1.9 Double Buffer Mode**.

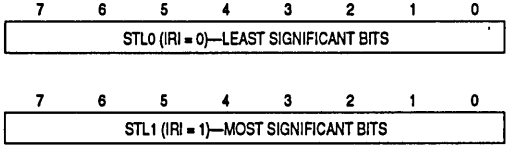
7.2.2.5 RECEIVE FRAME TYPE REGISTERS (RFR) IRT = 1; IRI = 4, 5. There are two RFRs, one for each receive internal data FIFO. The RFR defines the types of receive frames that can be received into each FIFO.

7	6	5	4	3	2	1	0
TE	0	OT	LS	LA	MF	SF	VF

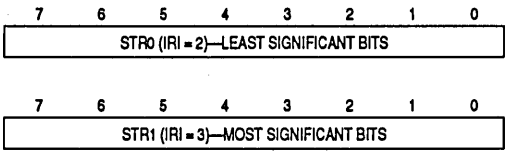
With the RFRs, users may direct incoming frames to one of two receive FIFOs—e.g., LLC frames to one FIFO and MAC and SMT frames to the second FIFO. Each control bit in these registers defines whether a frame type is to be received into this FIFO (when the bit is set), or not received into this FIFO (when the bit is reset). If the bit is reset in both RFRs, frames with the indicated type will not be received by the FSI. If the bit is set in both RFRs, the frame is received only into receive data FIFO 4. The bit assignments are listed in the following table.

Bit Num	Bit Name	Frame Type
0	VF	Void Frame
1	SF	SMT Frame
2	MF	MAC Frame
3	LA	LLC Asynchronous Frame
4	LS	LLC Synchronous Frame
5	OT	Other Frames
6	—	Reserved
7	TE	TOKEN CYCLE END indication from MAC. This mode cannot be used when using split-header operation with not immediate header indication.

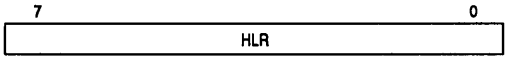
7.2.2.6 SMT TIMER LOAD VALUE REGISTERS (STL) IRT = 2; IRI = 0, 1. STL0 and STL1 together contain the SMT timer load value. This value is the intended expiration time of the SMT timer measured in units of 512 BYTCLKs (40.96 µs at 12.5 MHz). Since writing to STL1 causes the timer to start running, STL0 should be written first. The STLs may be read at any time without affecting the operation of the SMT timer.



7.2.2.7 SMT TIMER REGISTERS (STR) IRT = 2; IRI = 2, 3. STR0 and STR1 together contain the current value of the SMT timer. These are read-only registers that may be read at any time. Note that reading either of these registers will stop the SMT timer so that the two reads together provide a consistent value.



7.2.2.8 HEADER LENGTH REGISTER (HLR) IRT = 2; IRI = 4. This register defines the length of the frame header and is used in header splitting operation.



Only one bit in the HLR may be set to represent one of the following eight header length choices:

Bit Num	Buffer Length in Bytes
0	32
1	64
2	128
3	256
4	512
5	1K
6	2K
7	4K

The minimum allowed header length is 32 bytes. Note that if the HLR is used, the receive buffer length registers should not be programmed.

7.2.2.9 BURST LIMIT REGISTER (BLR) IRT = 3; IRI = 6, 7. There are two burst limit registers, one for each port (IRI = 6 for port A and IRI = 7 for port B). The BLR specifies the longest burst that is enabled on each port. The BLR also determines whether the address affects the size calculations.

7	6	5	4	3	2	1	0
ADE	0	BLM5	BLM4	BLM3	BLM2	BLM1	BLM0

ADE—Address Enable

When this bit is zero, the address does not affect the size calculations. When this bit is set, the address affects the size calculations. If the ADE bit is set, the transfer count will be limited by the address according to this rule: the transfers must always be aligned to an address whose $\log_2$ (size of the transfer in bytes) least significant bit(s) is (are) zero. The following table will clarify this rule.

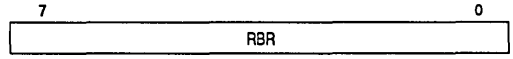
Address	Transfer Count in 32-Bit Transfers
xxxxxxx100	1
xxxxxx1000	2
xxxxx10000	4
xxxx100000	8
xxx1000000	16
xx10000000	32
x100000000	64

BLM5–BLM0—Burst Limit

These bits specify the maximum burst length. The following values of burst are defined:

BLM (5–0)	Limit in Bytes	Transfers (32 Bit)	Transfers (64 Bit)
000000	4	1	1
000001	8	2	1
000011	16	4	2
000111	32	8	4
001111	64	16	8
011111	128	32	16
111111	256	64	32

7.2.2.10 RECEIVE BUFFER LENGTH REGISTER (RBR) IRT = 3; IRI = 4, 5. There are two RBRs, one for each receive ring. The RBR defines the length of the receive data buffers for each ring in external memory.

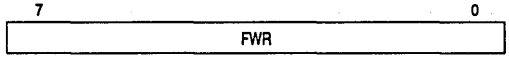


If the RBR is used, the length is fixed for all buffers in the same ring. If the buffer length is provided in the receive buffer descriptor, then this register should be set to zero. If receive one buffer mode is used, then the RBR must be programmed. Only one bit in the RBR may be set to represent one of the following eight buffer length choices:

Bit Num	Bit Name	Buffer Length in Bytes
0	RBR0	64
1	RBR1	128
2	RBR2	256
3	RBR3	512
4	RBR4	1K
5	RBR5	2K
6	RBR6	4K
7	RBR7	8K

The largest receive data buffer can hold the entire frame (the FDDI standard maximum frame length is 4500 bytes). In 64-bit mode, the minimum allowed receive buffer length is 128 bytes. Note that if the RBRs are used, the header length register (HLR) should not be programmed.

7.2.2.11 FIFO WATERMARK REGISTER (FWR) IRT = 4; IRI = 0–7. There are eight FWRs, one for each ring and one for each command register. The watermark for the transmit command issued directly to the IFDDI has the same meaning as in normal transmit operation.

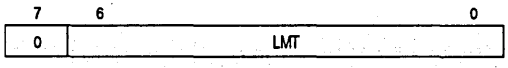


The watermark has a different meaning for transmit and receive operations. In transmit operations, the internal transmit data FIFO should reach the watermark or should hold an entire frame to be able to transmit data from a FIFO to ensure the successful transmission of the entire frame. Therefore, this number should be calculated according to the latency of the IFDDI external bus.

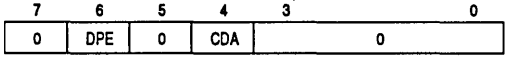
The value for the FIFO watermark is calculated as follows: Watermark = (FWR) x 32 bytes. The minimum value allowed for the FIFO watermark is 64 bytes (FWR = 2), and the maximum value is 7360 bytes (FWR = 230).

In receive operations, the internal receive data FIFO will either reach the watermark or hold an entire frame before starting to transfer received data into external memory. Therefore, the calculation of the settings for the RWRs should normally be according to the number of memory cycles the system designer wants the IFDDI to use in DMA burst operation.

7.2.2.12 LIMIT REGISTER (LMT) IRT = 5; IRI = 0–5. There are six LMTs, one for each transmit and receive channel. The LMT specifies the maximum number of frames allowed inside local memory space for each channel. The minimum number of frames is 1 (LMT = 000 0000), and the maximum value is 128 (LMT = 111 1111).



7.2.2.13 COMMAND PARAMETER REGISTERS (CPR) IRT = 6; IRI = 6, 7. There are two CPRs, one for each port's command register (IRI = 6 for port A and IRI = 7 for port B). The CPR is used to define the source port for the transmit command when it is to be issued directly to the IFDDI. The CPR operates similarly to the ring parameter register with a command data assignment (CDA) bit rather than a ring data assignment (RDA) bit.



DPE—Data Parity Enable

When DPE is set, parity checking is enabled for data transfers (DTRs) from a descriptor in this command register.

CDA—Command Data Assignment

When CDA is zero, the data for transmit commands issued directly to the FSI is read through port A. When CDA is set, it is read through port B. In 64-bit operation this bit should be zero.

7.2.2.14 RING PARAMETER REGISTER (RPR) IRT = 6; IRI = 0–5. There are six RPRs, one for each ring (four transmit and two receive rings). These registers define the maximum ring length and select the ports from which the buffer descriptor rings and data buffers operate.

7	6	5	4	3	0
RPE	DPE	RPA	RDA	RML	

RPE—Ring Parity Enable

When RPE is set, parity checking is enabled for ring entry transfers from this ring.

DPE—Data Parity Enable

When DPE is set, parity checking is enabled for data transfers of this ring.

RPA—Ring Port Assignment

When RPA is zero, the ring entries are accessed through port A. When RPA is set, the ring entries are accessed through port B. Note that a ring may be assigned one port space and its data may be assigned the other port space. In 64-bit operation this bit should be zero.

RDA—Ring Data Assignment

When RDA is zero, transmit data for rings 0, 1, 2, 3 or receive data for rings 4 and 5 are transferred through port A. When RDA is set, the data is transferred through port B. In 64-bit operation this bit should be zero.

RML3–RML0—Ring Maximum Length

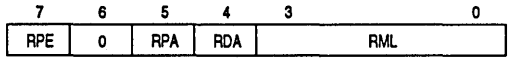
These four bits specify the maximum length of the ring, (the maximum number of entries with each entry being 64 bits). The values of maximum length (in bytes) are defined in the following table.

For example, with an RML set to 3 (64 bytes), using 32-bit transfers, a current ring address of 1BF8 (hex) will be incremented to 1BFC (hex), and on another increment, will wrap to 1BC0 (hex).

RML	Num of Entries	Maximum Length in Bytes
0	1	8
1	2	16
2	4	32
3	8	64
4	16	128
5	32	256
6	64	512
7	128	1K
8	256	2K
9	512	4K
10	1024	8K
11	2048	16K
12	4096	32K
13	8192	64K
14	16384	128K
15	32768	256K

7.2.2.15 PARAMETER EXTENSION REGISTER (PER) IRT = 7; IRI = 0–7. There are eight PERs, one for each channel. The PER has two exclusive definitions, depending on whether the ring's secondary usage is as a destination ring (for port to port DMA operations) or as a ring using the local memory option.

When the user desires to set up destination rings for transmit channels (0–3) and for DMA command channels (6, 7), these registers define the destination ring maximum length and its ring and data assignment (similar to RPR).



(Destination Ring)

RPE—Ring Parity Enable

When RPE is set, parity checking is enabled for ring entry transfers from this ring.

RPA—Ring Port Assignment

When RPA is zero, the ring entries are accessed through port A. When RPA is set, the ring entries are accessed through port B. Note that a ring may be assigned one port space and its data may be assigned the other port space. In 64-bit operation this bit should be zero.

RDA—Ring Data Assignment

When RDA is zero, DMA data is transferred through port A. When RDA is set, the data is transferred through port B. In 64-bit operation this bit should be zero.

RML—Ring Maximum Length

These four bits specify the maximum length of the ring (the maximum number of entries with each entry being 64 bits). The following values of maximum length (in bytes) are defined:

RML	Num of Entries	Maximum Length in Bytes
0	1	8
1	2	16
2	4	32
3	8	64
4	16	128
5	32	256
6	64	512
7	128	1K
8	256	2K
9	512	4K
10	1024	8K
11	2048	16K
12	4096	32K
13	8192	64K
14	16384	128K
15	32768	256K

For example, with an RML set to 3 (64 bytes), using 32-bit transfers, a current ring address of 1BF8 (hex) will be incremented to 1BFC (hex), and on another increment, will wrap to 1BC0 (hex).

For transmit and receive channels (0–5) in local memory mode, these registers define local memory space and its port assignment and parity control as shown in the following register. Note that starting addresses for each ring's memory space are defined by the DEFINE LOCAL MEMORY command.

7	6	5	4	3	2	0
0	LPE	0	LPA	0	LML	

(Local Memory Parameters)

LPE—Local Memory Parity Enable

When LPE is set, parity checking is enabled for this local memory space.

LPA—Local Memory Port Assignment

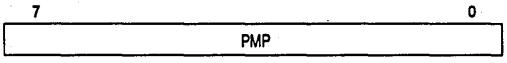
When LDA is zero, the local memory space is accessed through port A. When LDA is set, the local memory space is accessed through port B.

LML2–LML0—Local Memory Length

These three bits specify the length of the local memory space for this channel. The following values of length (in bytes) are defined:

LML	Length In Bytes
0	8K
1	16K
2	32K
3	64K
4	128K
5	256K
6	512K
7	1M

7.2.2.16 PORT MEMORY PAGE REGISTER (PMP) IRT = 8; IRI = 6, 7. There are two PMPs, one for each port (IRI = 6 for port A and IRI = 7 for port B). The PMP specifies the external memory page length size utilized on a port, allowing the IFDDI to decide whether or not the next access is on the same page as the current access.



The page indication generated by the IFDDI may be very useful when the external memory is implemented by nibble or page mode dynamic memory to implement burst access and reduce bus bandwidth utilized by the IFDDI. The bits of the PMP (7–0) specify the page length. The minimum supported page length is 16 bytes (PMP = 0000 0000), and the maximum supported page length is 16 Kbytes, (PMR = 1111 1111). The port memory page length (in bytes) is specified in the following table.

PMP (7–0)	Length In Bytes
0000 0000	16
0000 0001	32
0000 0011	256
0000 0111	512
0000 1111	1K
0001 1111	2K
0011 1111	4K
0111 1111	8K
1111 1111	16K

7.2.2.17 DESTINATION RING READY (DRY) IRT = 9; IRI = 0–3, 6, 7. There are six DRYs, four for each transmit channel or ring and two for the DMA command rings. The destination rings offer expanded capabilities and are programmed via the parameter extension register (PER). DRY is used to transition the destination ring specified by the IRI to the ready state. The DRY action is caused by an access to the register indicated by the IRT and IRI values. If this ring was already in the ready state, the DRY control will have no effect. The DRY access cannot be performed on a ring that does not exist in the system.

7.2.2.18 PORT CONTROL REGISTER (PCR) IRT = B; IRI = 6, 7. There are two PCRs, one for each port. (IRI = 6 for port A and IRI = 7 for port B.)

7	6	5	4	3	2	1	0
HPE	PC	SO	DO	0	DW	0	PE

HPE—Host Parity Enable

This bit indicates whether there is parity checking for port accesses that involve the host processor directly—i.e., SR1, SR2, FCR, CMR, CER, IMR1, IMR2, or IOR write accesses. If this bit is set, parity checking is enabled.

PC—Parity Control

This bit in the PCR of port A defines the parity treatment mode for the entire FSI portion of the IFDDI. If this bit is zero, the parity is odd. If this bit is set, the parity is even.

SO—Synchronous Operation

If this bit is reset, the assertion of the request output signals when the port is in the idle state is asynchronous. If this bit is set, the assertion of the request output signals when the port is in the idle state is synchronous to chip select.

DO—Data Order

This bit defines the structure of the data bus and the order of bytes inside the 32-bit or 64-bit data structures. If DO = 0, then the byte order is most significant byte first (Motorola or IBM style). If DO = 1, then the byte order is least significant byte first (Intel or Digital style).

DW—Data Width

This bit selects the port data width as follows:

0 = 32 Bit
1 = 64 Bit (port A only)

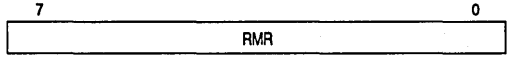
NOTE

If port A is programmed for 64-bit data width, its control word is used for both port A and port B, and port A handles the most significant portion of the data (bits 63–32).

PE—Port Enable

When this bit is zero, the operation of the port is disabled, and the port remains in the idle state. When this bit is set, the port's request operation is enabled.

7.2.2.19 MAXIMUM RECEIVE MEMORY SPACE REGISTER (RMR) IRT = C; IRI = 4, 5. There are two RMRs, one for each receive ring. The RMR defines the maximum amount of internal memory to be allocated for the handling of the receive ring.



Included in the RMR value are both the receive internal ring FIFO and the receive internal data FIFO. The value for the maximum receive memory space is calculated as follows: Max_Receive_Memory_Space = (RMR + 1) × 32 bytes. Therefore, the largest is 8 Kbytes (RMR = 255). The minimum value of RMR should be 10 (Max_Receive_Memory_Space = 352 bytes).

7.2.2.20 SIGNAL REGISTER (SIG) IRT = D; IRI = 0–7. These write-only registers are used to set one of the INTx status bits inside status register 2 (SR2). Each SIG register is related to one INTx bit when the IRI of this register defines the index of the bit. The INTx bit is set by writing the SIG register with predefined data. Note that INTx bits in SR2 are sticky bits and may be cleared only by a host processor write access to the SR2 with the corresponding bit set.

7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0

7.2.2.21 RING STATE REGISTER (RSR) IRT = E; IRI = 0–7. There are eight RSRs, one for each ring. For rings 6 and 7, only the DRY bit is provided. The Ring State Registers are Read-Only registers.

7	6	5	4	3	2	1	0
EX	PER	OER	DRY	STP	RDY	EMP	CPL

EX—Exists

When EX is reset, this ring does not exist in the system.

PER—Parity Error

PER is set when a parity error is detected by the FSI portion of the IFDDI during a ring entry's read. This bit is reset when the ring is redefined or transitions to the ready state by a ring ready control access (access to the FCR).

OER—Operation Error

OER is set when a port operation error or a FIRST or LAST bit error is detected by the FSI portion of the IFDDI during a ring entry's read. This bit is reset when the ring is redefined or transitions to the ready state by a ring ready control access (access to the FCR).

DRY—Destination Ring Ready

When DRY is set, this destination ring is in ready state. This bit has meaning only if the ring is used for DMA operations. If it is used in local memory mode, then this bit may be set or reset and has no meaning.

STP, RDY, EMP, CPL

The following table shows the meaning of bits in the RSR in various ring states:

RSR Bit Values for Ring States

EX	STP	RDY	EMP	CPL	Ring State
0	x	x	x	x	Ring Does Not Exist
1	0	1	0	x	READY
1	1	0	x	x	STOP
1	0	0	1	0	EMPTY
1	0	0	x	1	COMPLETE

There are situations when more than one bit is set. For example, the RDY and CPL bits may be set if this ring has been transitioned from the complete state to the ready state by a ring ready control access, but a descriptor read access has not yet been performed. Also, if a parity or operation error is detected during a descriptor read operation, the ring is empty (no more valid descriptors), and the PER or OER bit is set in conjunction with the EMP bit. If EX is zero, the other bits have no meaning.

7.2.2.22 INTERNAL ERROR STATUS REGISTER (IER) IRT = F; IRI = 0. This read-only register is used to identify FSI relevant internal operation errors. All bits are zero after

reset and are cleared, except MOV, by a write access to the IER, regardless of the data being written. MOV is cleared only by a hardware or software reset.

7	6	5	4	3	2	1	0
IOE	IUE	TPE	MER	MOV	0	PBE	PAE

IOE—Internal Overrun Error

This bit indicates that the MACIF has experienced an internal overrun. During normal operation, the main controller ensures a maximum response time for data transfers between the MACIF and the internal memory. If this response time exceeds these limits, it might indicate that the main controller is not performing properly. As a result, the IOE may be set, the receiver is disabled (both RE4 and RE5 reset), and the RABORT signal is generated in the case of FSI operational mode only.

IUE—Internal Underrun Error

This bit indicates that the MACIF has experienced an internal underrun. During normal operation, the main controller ensures a maximum response time for data transfers between the MACIF and the internal memory. If this response time exceeds these limits, it might indicate that the main controller is not performing properly. As a result, the IUE bit may be set, the current transmit frame is aborted with the appropriate indication, and the transmitter is disabled (TE reset).

TPE—Transmit Internal Parity Error

The MACIF sets this bit when a parity error is detected on the data read from internal memory. In this situation, the TPE bit is set, the current transmit frame is aborted with an appropriate indication, and the transmit enable (TE) bit in the MACIF transmit control register (MTR) is reset.

MER—MAC Error

This bit indicates that the FSI portion of the IFDDI has detected a protocol handshake error between the MAC and the FSI during the reception of a frame. The MACIF will generate the RABORT to the MAC and will try to resynchronize with the MAC on the next frame. This error may be caused by temporary problems (e.g., noise on control lines); therefore, no special recovery functions are required. If there are permanent problems in the FSI to MAC interface (e.g., control lines are disconnected or forced to some constant value), the host processor should perform an external diagnostic. Note that this is relevant in the case of FSI operation mode (not IFDDI) only.

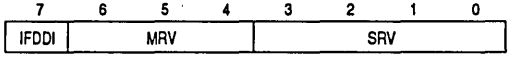
MOV—Memory Overrun

The FSI portion of the IFDDI has experienced an internal memory overrun. This error may be caused by an incorrect definition of an internal FIFO's watermark. Therefore, the FSI parameter definitions should be checked, and the FSI portion of the IFDDI should be reinitialized.

PBE, PAE—Port B or Port A Internal Error

The port A or port B interface unit sets the appropriate bit when it detects a parity error while transferring data from the internal FSI memory buffer to the port.

7.2.2.23 FSI REVISION REGISTER (REV) IRT = F; IRI = 2. This read-only register is used to identify the revision number of the FSI.



IFDDI

When IFDDI = 0, the MC68840 is operating as a FSI standalone device. When IFDDI = 1, the MC68840 operates as the IFDDI device.

MRV—Major Revision

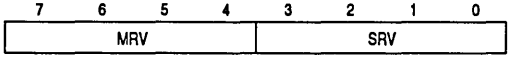
This field indicates the major revision number of the FSI.

SRV—Sub Revision

This field indicates the sub revision number of the FSI:

FSI Revision	FSI Revision Register Value
Revision A	0_000_0000
Revision B	0_001_0000
Revision C	0_001_0001
Revision D Standalone (IFDDI in FSI Mode)	0_010_0000
Revision D in the IFDDI (IFDDI in IFDDI Mode)	1_010_0000
Revision E Standalone (IFDDI in FSI Mode)	0_011_0000
Revision E in the IFDDI (IFDDI in IFDDI Mode)	1_011_0000

7.2.2.24 IFDDI REVISION REGISTER (IREV) IRT = F; IRI = 3. This read-only register is used to identify the revision number of the IFDDI.



MRV—Major Revision

This field indicates the major revision number of the IFDDI.

SRV—Sub Revision

This field indicates the sub revision number of the IFDDI:

IFDDI Revision	IFDDI Revision Register Value
Revision A	0000_0000
Revision B	0001_0000

7.2.2.25 USER REGISTER (USR) IRT = F; IRI = 4, 5. There are two USR read/write registers that may be used for processor-to-processor communication. The USR has no meaning for the IFDDI, and it is not changed as a result of IFDDI operation.

7.2.2.26 SOFTWARE RESET—IRT = F; IRI = 7. Accessing the pseudo-register location "software reset" with the R-bit of the FCR will accomplish the same functions as activating the hardware $\overline{\text{RESET}}$ pin except that the interrupt mask registers (IMR1 and IMR2) are unaffected by software reset. The CRF bit of status register 1 (SR1) will be set when the software reset is complete, generating an interrupt if the CFE bit of interrupt mask register 1 (IMR1) is set.

7.3 CAMEL REGISTER SET

The CAMEL registers (MAC,ELM and Global registers) can be accessed directly via Port A and indirectly through the FCR or CMR registers. The CAMEL core contains 76 registers. These registers have addresses ranging from 00–94 hex. Not all of the addresses in this range access valid registers.

The CAMEL contains 5 global registers (not including the "zero" registers) in addresses 80–94 hex, 29 ELM registers in addresses 00–1A hex and 36 MAC registers in addresses 40–74 hex.

For Direct Register Access, refer to Figure 7-2a.

For Indirect Register Access, refer to Figure 7-2c.

Read-only registers are defined as those that the system can access for status or control information but cannot write to. Read/write registers are defined as those that the system can access for both read and write operations. Read-only clear registers are those that are cleared upon a read access. Read/CNTL Write MAC registers can be read at any time, but can only be written when the Rx and Tx FSMs are turned off (MAC_ON=0 in MAC_CNTL_A).

7.3.1 CAMEL Core General Registers

The CAMEL general registers are listed in Table 7-6.

Table 7-6. CAMEL Core General Registers

Address	Name	Mnemonic	Type
80	CAMEL Control Register	CAMEL_CNTRL	Read/Write
81	CAMEL Interrupt Mask Register	CAMEL_MASK	Read/Write
84	CAMEL Interrupt Event Register	CAMEL_INTR	Read-Only Clear
85	CAMEL Zero Register A	—	Read-Only
86	CAMEL Interrupt Location Register	CAMEL_INT_LOC	Read-Only
87	CAMEL Revision Number	CAMEL_REV_NO	Read-Only
90	CAMEL Zero Register B	—	Read-Only
91	CAMEL Zero Register C	—	Read-Only
92	CAMEL Zero Register D	—	Read-Only
93	CAMEL Zero Register E	—	Read-Only
94	CAMEL Zero Register F	—	Read/Write

Reading CAMEL zero registers A–F will always return 0x0000. Writing CAMEL zero register F will not change the register's value.

7.3.2 ELM Core Registers

The ELM registers are listed in Table 7-7. Note that three new registers have been added and are no longer invalid addresses. The three registers are the CIPHER_CNTRL, FOTOFF_ASSERT and FOTOFF_DEASSERT registers. For information concerning the functionality of these registers, see **7.4 Stream Cipher Registers.**

Table 7-7. ELM Core Registers

Address	Name	Mnemonic	Type
00	ELM Control Register A	ELM_CNTRL_A	Read/Write
01	ELM Control Register B	ELM_CNTRL_B	Read/Write
02	ELM Interrupt Mask Register	ELM_MASK	Read/Write
03	Transmit Vector Register	XMIT_VECTOR	Read/Write
04	Transmit Vector Length Register	VECTOR_LENGTH	Read/Write
05	Link Error Event Threshold Register	LE_THRESHOLD	Read/Write
06	Maximum PHY Acquisition Time Register	A_MAX	Read/Write
07	Maximum Line State Change Time Register	LS_MAX	Read/Write
08	Minimum Break Time Register	TB_MIN	Read/Write
09	Signaling Time-Out Register	T_OUT	Read/Write
0A	Cipher Control Register	CIPHER_CNTRL	Read/Write
0B	Short Link Confidence Test Time Register	LC_SHORT	Read/Write
0C	Scrub Time Register	T_SCRUB	Read/Write
0D	Noise Time Register	NS_MAX	Read/Write
0E	TPC Load Value Register	TPC_LOAD_VALUE	Write-Only
0F	TNE Load Value Register	TNE_LOAD_VALUE	Write-Only
10	ELM Status Register A	ELM_STATUS_A	Read-Only
11	ELM Status Register B	ELM_STATUS_B	Read-Only
12	TPC Timer Register	TPC	Read-Only
13	TNE Timer Register	TNE	Read-Only
14	Clock Divider Register	CLK_DIV	Read-Only
15	ELM BIST Signature Register	ELM_BIST	Read-Only
16	Receive Vector Length Register	RCV_VECTOR	Read-Only
17	ELM Interrupt Event Register	ELM_INTR	Read-Only Clear
18	Violation Symbol Counter Register	VIOL_SYM_CTR	Read-Only Clear
19	Minimum Idle Counter Register	MIN_IDLE_CTR	Read-Only Clear
1A	Link Error Event Counter Register	LINK_ERR_CTR	Read-Only Clear
1E	FOTOFF Assert Timer	FOTOFF_ASSERT	Read/Write
1F	FOTOFF De-assert Timer	FOTOFF_DEASSERT	Read/Write

7.3.3 MAC Core Registers

The MAC registers are listed in Table 7-8.

Table 7-8. MAC Core Registers

Address	Register Name	Mnemonic	Type
40	MAC Control Register A	MAC_CNTRL_A	Read/Write
41	MAC Control Register B	MAC_CNTRL_B	Read/Write
42	MAC Interrupt Mask Register A	MAC_MASK_A	Read/Write
43	MAC Interrupt Mask Register B	MAC_MASK_B	Read/Write
44	MAC Interrupt Mask Register C	MAC_MASK_C	Read/Write
50	My Short Address Register	MSA	Read/Cntl Write
51	My Long Address Register A	MLA_A	Read/Cntl Write
52	My Long Address Register B	MLA_B	Read/Cntl Write
53	My Long Address Register C	MLA_C	Read/Cntl Write
54	Requested Target Token Rotation Time	T_REQ	Read/Cntl Write
55	TVX Timer Initial Value & Maximum TRT	TVX_VALUE & T_MAX	Read/Cntl Write
5C	MAC Revision Number	MAC_REV_NO	Read-Only
5D	MAC Interrupt Event Register C	MAC_INTR_C	Read-Only Clear
5E	Void Time Register	VOID_TIME	Read-Only
5F	Token Count Register	TOKEN_CT	Read-Only Clear
60	Frame Count Register	FRAME_CT	Read-Only Clear
61	Lost and Error Count Register	LOST_CT & ERROR_CT	Read-Only Clear
62	MAC Interrupt Event Register A	MAC_INTR_A	Read-Only Clear
63	MAC Interrupt Event Register B	MAC_INTR_B	Read-Only Clear
64	MAC Receive Status Register	MRX_STATUS	Read-Only
65	MAC Transmit Status Register	MTX_STATUS	Read-Only
66	Negotiated TTRT A	T_NEG_A	Read-Only
67	Negotiated TTRT B	T_NEG_B	Read-Only
68	Information Field Register A	INFO_REG_A	Read-Only
69	Information Field Register B	INFO_REG_B	Read-Only
6A	MAC BIST Signature Register	MAC_BIST	Read-Only
6B	TVX Timer Register	TVX_TIMER	Read-Only
6C	TRT Timer Register A	TRT_TIMER_A	Read-Only
6D	TRT Timer Register B	TRT_TIMER_B	Read-Only
6E	THT Timer Register A	THT_TIMER_A	Read-Only
6F	Sent Count Registers & THT Timer B	SENT_COUNT &THT_TIMER_B	Read-Only

Table 7-8. MAC Core Registers (Continued)

Address	Register Name	Mnemonic	Type
70	Packet Request Register	PKT_REQUEST	Read-Only
71	Receive CRC Register A	RX_CRC_A	Read-Only
72	Receive CRC Register B	RX_CRC_B	Read-Only
73	Transmit CRC Register A	TX_CRC_A	Read-Only
74	Transmit CRC Register B	TX_CRC_B	Read-Only

7.3.4 Control and Status Registers

The control and status information is contained in one global CAMEL register, four ELM registers, and four MAC registers.

7.3.4.1 CAMEL CONTROL REGISTER (CAMEL_CNTRL) ACNTL = 1 1000 0000. The CAMEL control register is a read/write register. All bits of this register are cleared with the assertion of $\overline{\text{RESET}}$. This register is used to control the internal bypass MUX on the receive data path to the MAC core. The 15 most significant bits are reserved and should be programmed to zero to maintain upward compatibility.

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	0	0	0	0	0	0	BY_CNTRL

Bits15–1—Reserved

BY_CNTRL—Bypass MUX Control

0 = The MAC receives its data and parity internally on the PRCDATx bus from the ELM core.

1 = The MAC receives its data and parity on the MRCDATx input pins. This allows the use of a second ELM to configure a dual attachment station (DAS).

This bit is zero at power-up reset.

7.3.4.2 ELM CONTROL REGISTER A (ELM_CNTRL_A) ACNTL = 1 0000 0000. ELM_CNTRL_A is a read/write register. All bits of this register are cleared with the assertion of $\overline{\text{RESET}}$. ELM_CNTRL_A is used for the following functions:

- Timer Configuration
- PCM MAINT State Options Specification
- Counter Interrupt Frequency Selection
- ELM Data Path Configuration
- ELM BIST Execution
- Physical Layer Media Dependent Control

Note that several bits of this register can only be written if the PCM is in the OFF or MAINT state.

15	14	13	12	11	10	9	8
0	NOISE_TIMER	TNE_16BIT	TPC_16BIT	REQ_SCRUB	ENA_PAR_CHK	VSYM_CTR_ INTRS	MINI_CTR_INTRS
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCG_LOOP_ CTRL	FOT_OFF	EB_LOC_LOOP	LM_LOC_LOOP	SC_BYPASS	REM_LOOP	RF_DISABLE	RUN_BIST

Bit 15—Reserved

This bit is reserved and should be set to zero.

NOISE_TIMER—Noise Timer

The NOISE_TIMER bit allows the noise timing function of the PCM to be used when the PCM is in the MAINT state. This function causes the TNE timer to be loaded with the value in the noise time register when the LSM transitions from Idle Line State to Noise Line State, Active Line State, or Unknown Line State. If the timer expires before Idle Line State is recognized, the TNE_EXPIRED bit in ELM_INTR is set.

TNE_16BIT—TNE 16-Bit Timer

When TNE_16BIT is set, it causes the TNE timer to operate as a 16-bit timer. In this mode, the two bits of the TNE clock divider are bypassed, and the TNE timer is incremented every 80 ns. TNE_16BIT can only be written if the PCM is in the OFF or MAINT state.

TPC_16BIT—TPC 16-Bit Timer

When TPC_16BIT is set, it causes the TPC timer to operate as a 16-bit timer. In this mode, the eight bits of the TPC clock divider are bypassed, and the TPC timer is incremented every 80 ns. TPC_16BIT can only be written if the PCM is in the OFF or MAINT state.

REQ_SCRUB—Request Scrub

The REQ_SCRUB bit allows limited access to the scrub capability of the ELM core. If the PCM is in the MAINT state or if the CONFIG_CNTRL bit is set in ELM_CNTRL_B, then REQ_SCRUB controls the scrub MUX. If REQ_SCRUB is set, then I-symbols are sourced at the PRCDATx port. The output at the TDATAx port is controlled separately by the MAINT_LS field in ELM control register B (ELM_CNTRL_B). This bit may be written at any time, but only takes effect when the PCM is in the MAINT state or the CONFIG_CNTRL bit in the ELM_CNTRL_B is set.

ENA_PAR_CHK—Enable Parity Check

The ENA_PAR_CHK bit controls parity checking in the ELM core logic on the TXDATAx inputs from the MAC core logic.

0 = Parity checking is disabled.
1 = Parity checking is enabled.

VSYM_CTR_INTRS—Violation Symbol Counter Interrupt

The VSYM_CTR_INTRS bit controls when the VSYM_CTR interrupt bit in the ELM interrupt event register (ELM_INTR) is set. When VSYM_CTR_INTRS is set, the interrupt is generated only when the violation symbol counter register overflows (reaches 256). When VSYM_CTR__INTRS is cleared, the interrupt is generated every time the violation symbol counter register (VIOL_SYM_CTR) is incremented (occurs whenever a V-symbol pair is detected).

MINI_CTR_INTRS—Minimum Idle Counter Interrupt

The MINI_CTR_INTRS bit controls when the minimum idle gap counter interrupt bit in ELM_INTR is set by the minimum idle gap counter portion of MIN_IDLE_CTR. This bit does not affect interrupts caused by the idle counter minimum detector portion of MIN_IDLE_CTR.

0 = The MINI_CTR interrupt is generated every time the Minimum Idle Gap counter is incremented (whenever a minimum length idle gap is detected).

1 = The MINI_CTR interrupt is generated when the minimum idle gap counter overflows (reaches 16).

FCG_LOOP_CNTRL—FCG Loopback Control

Setting FCG_LOOP_CNTRL causes the $\overline{\text{LOOPBACK}}$ output pin to be asserted low, which, in turn, causes data to be looped back from the FCG's transmit output to its receive input.

FOT_OFF—Fiber-Optic Transmitter Off

Setting FOT_OFF is one of several conditions that assert the $\overline{\text{FOTOFF}}$ output pin.

EB_LOC_LOOP—Elasticity Buffer Local Loopback

When EB_LOC_LOOP is set, a loopback path is set up in the CAMEL just prior to the CAMEL-to-FCG interface. Data from the CAMEL transmit path is looped back to the input of the framer at the elasticity buffer local loopback MUX. This bit also controls which clock the framer and elasticity buffer use. When EB_LOC_LOOP is cleared, the recovered byte clock derived from RSCLK is used. When EB_LOC_LOOP is set, the local byte clock (BYTCLK) is used. Thus when this is set, a clock glitch could be created that could cause receive data to be indeterminate for a clock cycle, spurious interrupts and unknown values in the event counters. EB_LOC_LOOP can only be written if the PCM is in the OFF or MAINT state.

LM_LOC_LOOP—LM Local Loopback

When LM_LOC_LOOP is set, a loopback path is set up in the ELM core so that data from TXDATx is passed through the transmit path and looped back to the input of the receive path at the LM local loopback MUX. LM_LOC_LOOP can only be written if the PCM is in the OFF or MAINT state.

SC_BYPASS—Scrub/Bypass

The SC_BYPASS bit provides limited control over the data path by furnishing a physical bypass of the ELM core. If the PCM is in the MAINT state or if the CONFIG_CNTRL bit in ELM_CNTRL_B is set, the SC_BYPASS bit controls the bypass MUX. If REQ_SCRUB is set, then PRCDATx is driven with I-symbols. If SC_BYPASS is set and REQ_SCRUB is cleared, then PRCDATx is driven by the data entering the ELM at the TXDATx input. Otherwise, PRCDATx is driven by the data entering the ELM at the RDATAx input. This bit may be set or cleared at any time, but it only takes effect when the PCM is in the MAINT state or the CONFIG_CNTRL bit is set in ELM_CNTRL_B.

SC_BYPASS	REQ_SCRUB	PRCDATx
0	0	RDATAx
0	1	I-Symbols
1	0	TXDATx
1	1	I-Symbols

When used in concentrator applications, the SC_BYPASS bit provides for isolation of the PHYs. The data is latched only once; therefore, there is only a 1-BYTCLK delay through a bypassed ELM.

REM_LOOP—Remote Loopback

When REM_LOOP is set, a remote loopback path is set up inside the ELM whereby symbols from the receive data path are looped back onto the transmit data path, traversing both paths except for the scrub MUX, bypass MUX, PRCDATx latch, and the TXDATx latch. If the PCM is in the MAINT state or if the CONFIG_CNTRL bit in ELM_CNTRL_B is set, the REM_LOOP bit controls the remote loopback MUX. This loopback is used by the PCM to control the configuration and can be used to monitor the ring or otherwise control configuration during normal operation. This bit has no effect if the LM_LOC_LOOP bit or the EB_LOC_LOOP bit is set. This bit may be set or cleared at any time, but only takes effect when the PCM is in the MAINT state or the CONFIG_CNTRL bit is set in ELM_CNTRL_B.

RF_DISABLE—Repeat Filter Disable

When RF_DISABLE is set, it disables the ELM repeat filter state machine.

RUN_BIST—Run Built-In Self-Test

When RUN_BIST is set, it causes the ELM to begin running BIST. The completion of BIST is indicated via an interrupt. BIST can be stopped before completion by clearing this bit. Once BIST has completed, this bit must be cleared and set again before BIST will restart.

7.3.4.3 ELM CONTROL REGISTER B (ELM_CNTRL_B) ACNTL = 1 0000 0001. ELM_CNTRL_B is a read/write register. All bits of this register are cleared with the

assertion of $\overline{\text{RESET}}$. ELM_CNTRL_B contains signals and requests to direct the PCM process. It is also used to control the line state match interrupt.

15	14			11	10		8
CONFIG_CNTRL	MATCH_LS				MAINT_LS		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CLASS_S	PC_LOOP		PC_JOIN	LONG	PC_MAINT	PCM_CNTRL	

CONFIG_CNTRL—Configuration Control

The CONFIG_CNTRL bit allows control over the scrub, bypass, and remote loopback data path MUXs while the PCM is in normal operation.

0 = The REQ_SCRUB, SC_BYPASS, and REM_LOOP bits only have an effect if the PCM is in the MAINT state.
1 = The REQ_SCRUB, SC_BYPASS, and REM_LOOP bits in ELM control register A (ELM_CNTRL_A) have an effect regardless of the state of the PCM.

MATCH_LS—Match Line State.

The MATCH_LS field specifies the line state to be compared with the currently detected line state (defined by LINE_ST in ELM_STATUS_A). When a match occurs, the LS_MATCH bit in the ELM_INTR register is set. Each bit of MATCH_LS corresponds to a line state. If more than one bit is set, the interrupt is signaled if any of the line states match the current line state. If no bits are set, the interrupt is signaled on any change in the LINE_ST field or the UNKN_LINE_ST bit. MATCH_LS is defined as follows:

0000 = Interrupt on any change in LINE_ST or UNKN_LINE_ST
1XXX = Interrupt on Quiet Line State
X1XX = Interrupt on Master Line State
XX1X = Interrupt on Halt Line State
XXX1 = Interrupt on Idle Line State

In the above list, X means don't care. Also, Idle Line State refers to ILS16, which is signaled only after 16 I-symbols (eight I-bytes) have been received.

MAINT_LS—MAINT Line State

The MAINT_LS field defines the line state the data stream generator will source while the PCM is in the MAINT state. MAINT_LS is defined as follows:

000 = Transmit_Quiet Line State
001 = Transmit_Idle Line State
010 = Transmit_Halt Line State
011 = Transmit_Master Line State
100 = Transmit_Quiet Line State
101 = Transmit_Quiet Line State
110 = Transmit_PDR (Transmit PHY_DATA request. The symbol pair at TXDATx is transmitted.)
111 = Transmit_Quiet Line State

CLASS_S—Class Slave

When CLASS_S is set, signifying that the PHY is a single attachment station (SAS), the station will not be bypassed before the PCM goes to the ACTIVE state (e.g., during configuration)—i.e., data coming from the MAC to the ELM will not be looped back to the MAC. Normally, this bit would not be set for A, B, and M type PHYs, in which case the ELM will be bypassed anytime the PCM is not in the ACTIVE or TRACE state. This bit has an effect when the PCM is in normal operation. When the PCM is in the MAINT state, the REQ_SCRUB and SC_BYPASS bits in ELM control register A control the scrubbing and bypass operation. This bit can only be changed when the PCM is in the OFF state. If this register is set or cleared when the PCM is in any other state, this bit will remain unchanged.

PC_LOOP—Physical Connection Loopback

PC_LOOP controls the loopback used in the Link Confidence Test. When it is set to a value other than zero and the PCM is in the NEXT state, the PCM will set TD_Flag (defined in the ANSI SMT standard) and perform the Link Confidence Test in one of three ways. The action taken is according to the value of these two bits:

00 = No Link Confidence Test is performed.
01 = The PCM sets Transmit_PDR, which assumes that protocol data units will be input at TXDATx.
10 = The PCM sets Transmit_Idle, which causes the ELM to source I-symbols.
11 = The PCM sets Transmit_PDR and sets up a remote loopback path in the ELM.

PC_LOOP should only be set or cleared after the PCM_CODE interrupt has been generated. If the PCM is not in the NEXT state or if PCM_SIGNALING is set, then any value written to the field is ignored. Once PC_LOOP has been written, it must be cleared and written again to perform another Link Confidence Test.

PC_JOIN—Physical Connection Join

When PC_JOIN is set and the PCM is in the NEXT state, the PCM will transition to the JOIN state and the PCM join sequence will be started. PC_JOIN should only be written after the PCM_CODE interrupt has been generated. If the PCM is not in the NEXT state or if PCM_SIGNALING is set, then any value written to this bit is ignored. After this bit has been set, it must be cleared and then set again to cause another transition from the NEXT state to the JOIN state. Note that if PC_JOIN is set after the Link Confidence Test has been started but before it has completed, the test will be aborted and the PCM join sequence will be initiated.

LONG—Long Link Confidence Test

When LONG is set, the PCM will perform a long Link Confidence Test—that is, it will continue the test until the processor issues a PC_SIGNAL, PC_JOIN, or other command. Otherwise, it will perform a short Link Confidence Test—that is, it will stop the test after the length of time indicated in the LC_SHORT time parameter. In either case, the Link Confidence Test will stop whenever MLS or HLS is detected, indicating that the neighboring PHY has completed its Link Confidence Test and started signaling.

PC_MAINT—PCM MAINT State

When PC_MAINT is set, the PCM state machine transitions to the MAINT state if it is currently in the OFF state. If the PCM is not in the OFF state when this bit is set, it will immediately transition to the MAINT state when the OFF state is reached.

PCM_CNTRL—PCM Control

PCM_CNTRL controls the PCM state machine. When this bit field is set to a value other than zero, it causes the PCM to immediately transition to the BREAK, TRACE, or OFF state. The transition to the BREAK or OFF state occurs regardless of the PCM state at the time. The transition to the TRACE state only occurs if the PCM is in the ACTIVE state; otherwise, PCM_CNTRL is ignored. This field must first be cleared and then written with another value to cause another transition. The following action is taken according to the value of these two bits:

00 = The PCM state is not affected.
01 = The PCM goes to BREAK state (PC_Start).
10 = The PCM goes to the TRACE state (PC_Trace).
11 = The PCM goes to the OFF state (PC_Stop).

Note that if the PCM goes to the BREAK state for a reason other than writing PCM_CNTRL (e.g., Quiet Line State is received or a time-out occurs), the PCM will not go to the CONNECT state and will remain in the BREAK state until PCM_CNTRL is written with the PC_Start value. If the PCM goes from the ACTIVE state to the BREAK state, it will scrub the ring before leaving the BREAK state. If the PC_Start value is written to PCM_CNTRL while scrubbing is being performed, the scrubbing will complete before the PCM goes to the CONNECT state.

7.3.4.4 MAC CONTROL REGISTER A (MAC_CNTRL_A) ACNTL = 1 0100 0000. The MAC never modifies this register. It is cleared on power-up reset and unaffected by a MAC_Reset. MAC_CNTRL_A contains some bits (most named COPY_xxx) that control which frames are received and passed to the FSI. Most of these bits do not affect the behavior of the MAC relative to the ring—e.g., whether the MAC repeats or strips the frame, the transmitted A and C indicators, etc. When all these bits are zero, the MAC copies all LLC, SMT, implementor, and reserved FC frames (a) whose length is valid, (b) whose DA is matched, (c) whose SA is not matched or is an NSA frame, and (d) whose frames are not a secondary NSA frame.

"DA matched" means that DA = MSA or MLA or broadcast address (all ones) or in the CAM. "SA matched" means that SA = MSA or MLA or in the CAM or bridge strip algorithm match (see BRIDGE_STRIP bit description).

A primary NSA frame is an SMT NSA frame (FC = 0L00 1111) whose receive A-frame status indicator is an R-symbol. The N_Flag is cleared for these frames. A secondary NSA frame is an NSA frame whose A indicator is missing or is an S-symbol. The N_Flag is set for these frames.

15	14	13	12	11	10	9	8
MAC_ON	SET_BIT_5	SET_BIT_4	REVERSE_ADDR	FLUSH_SA47	COPY_ALL		COPY_OWN
7	6	5	4	3	2	1	0
COPY_EXTRA_SMT		COPY_IND_LLC	COPY_GRP_LLC	DISABLE_BRDCST	RUN_BIST	RX_PARITY	NOTE_ALL_ FRAMES

MAC_ON—MAC On

This bit turns the receiver/transmitter finite state machine on or off.

0 = Both the receiver and transmitter finite state machine are turned off; no MAC timers are running. When in this state, the MAC receiver/transmitter ignores all inputs and stays in this state until MAC_ON is set. When they are operating, the receiver and transmitter can be turned off at any time by clearing this bit.

1 = When set to one, the receiver finite state machine transitions to the listen state (R0), and the transmitter finite state machine transitions to the Tx_Idle state (T0), after which the receiver and transmitter finite state machines can be in any of the states R0–R5 or T0–T5, respectively, or in the FDX states.

SET_BIT5—Set Bit 5

Repeat fifth control indicator as an S-symbol.

0 = The fifth control indicator is repeated exactly as received.

1 = The repeating function always causes the fifth control indicator received to be transmitted as an S-symbol. If the fifth control indicator received is already an S-symbol, then the BIT5_IS_S interrupt is signaled. Nothing happens if there are not five control indicators.

SET_BIT4—Set Bit 4

Repeat fourth control indicator as an S-symbol.

0 = The fourth control indicator is repeated exactly as received.

1 = The repeating function always causes the fourth control indicator received to be transmitted as an S-symbol. If the fourth control indicator received is already an S-symbol, then the BIT4_IS_S interrupt is signaled. Nothing happens if there are not four control indicators.

REVERSE_ADDR—Reverse the DA and SA Fields of All Frames

This bit is used by both the receiver and transmitter portion of the MAC. This bit reversal only occurs across the FSI/MAC internal bus; hence, it does not affect the CRC checking/generation nor the interpretation of the my long address (MLA) or my short address (MSA) registers, etc. The order of the octets is not affected by this bit. Octets are passed to and from the FSI in the same order as they appear on the fiber.

0 = No bit reversal will occur.

1 = The MAC core will reverse the bit order of data octets passed to and from the FSI (i.e., across RPATHx and TPATHx for all octets that make up the DA and SA fields of all frames to be sent or received from the FSI). Therefore, for DA and SA octets, bit (7–x) is passed to the FSI core on RPATHx, and bit (7–x) is obtained from the FSI core on TPATHx.

This feature is useful for implementing bridges between IEEE 802.3 and 802.4 protocols to FDDI.

FLUSH_SA47—Flush Source Routing Frames

0 = The MAC will copy frames when the individual/group bit of the SA (bit 47 of 48-bit addresses or bit 15 of 16-bit addresses) is one.

1 = The MAC will not copy frames when the individual/group bit of the SA (bit 47 of 48-bit addresses or bit 15 of 16-bit addresses) is one.

In Source Routing frames, the I/G bit of the SA is one. FLUSH_SA7 controls whether this MAC copies Source Routing frames.

This bit has no effect on the ring protocols. In particular, it does not change the frame stripping algorithms nor affect which station wins the claim process. When comparing the SA of a received frame to the MLA/MSA register, the I/G bit of the SA is ignored, i.e., always considered matched regardless of the value of FLUSH_SA7. On the other hand, in the DA, the I/G bit is used in address comparisons.

FLUSH_SA7 has no effect when COPY_ALL is 10 or 11, and does not effect NSA frames.

COPY_ALL—Copy Extra Frames and Fragments

COPY_ALL is intended for use in monitor and analyzer stations. Even when COPY_ALL=11 or 10, the MAC still flushes the secondary NSA frames if COPY_EXTRA_SMT=00.

00 = The MAC copies all LLC, SMT, implementor, and reserved FC frames with valid length, DA matched, SA not matched (subject to COPY_OWN) or is an NSA frame and is not a secondary NSA frame.

01 = The MAC additionally copies MAC and void frames whose length is valid, whose DA is matched, and whose SA is not matched (subject to COPY_OWN).

10 = The MAC additionally copies tokens and frames whose length is too short (but with an even number of data symbols), whose DA is not matched, or whose SA is matched.

11 = The MAC additionally copies fragments, format errors, and frames with an odd number of data symbols.

COPY_OWN—Copy Frames Sent by This Station

The MAC ignores this bit when COPY_ALL = 11 or 10 or if the received frame is an NSA frame. The COPY_OWN mode of operation is not intended for normal operation but is reserved for special monitor stations and for ring loopback tests applicable to all stations.

0 = The MAC does not copy frames that it is currently sending nor frames that it believes it previously sent, even if the MAC is requested to copy all frames with a certain FC or DA (see the following register fields).

1 = The MAC copies frames whose SA is matched if the frame would be copied otherwise and copies other frames only if the frame is directly addressed to the station. (DA = broadcast, MLA register, or MSA register.)

COPY_EXTRA_SMT—Copy Certain Extra SMT Frames

When COPY_ALL = 11 or 10, this bit field is ignored (except that secondary NSA frames are still flushed when COPY_EXTRA_SMT = 00). This bit field does not affect how the MAC sets the A and C control indicators.

00 = The MAC copies valid SMT frames whose DA is matched, and the SA is not matched (subject to COPY_OWN) or are primary NSA frames (even if matched).

01 = The MAC additionally copies all valid secondary NSA frames whose DA is matched (even if SA is matched).

10 = The MAC additionally copies all valid SMT frames whose DA is a 48-bit group address (including broadcast) and whose SA is not matched (subject to COPY_OWN), or are primary or secondary any NSA frames (even if SA matched).

11 = The MAC additionally copies all valid SMT frames whose DA is a 48-bit individual or group address (including broadcast), and SA is not matched (subject to COPY_OWN) OR frames that are primary or secondary NSA frames (even if SA is matched).

COPY_IND_LLC—Copy All Individual LLC Frames

This bit does not affect how the MAC sets the A and C control indicators. COPY_IND_LLC is primarily used for promiscuous bridge implementations. The MAC ignores this bit when COPY_ALL = 11 or 10.

0 = The MAC copies individually addressed LLC implementor and reserved FC frames only if the DA is matched.

1 = In addition to the frames it is already copying, the MAC copies all LLC, implementor, and reserved frames whose DA field indicates a 48-bit individual address. The MAC will not copy LLC frames it sent unless COPY_OWN = 1.

COPY_GRP_LLC—Copy All Multicast LLC Frames

This bit does not affect how the MAC sets the A and C control indicators. COPY_GRP_LLC is primarily used for promiscuous bridge implementations. The MAC ignores this bit when COPY_ALL = 11 or 10.

0 = The MAC copies group addressed LLC, implementor and reserved FC frames only if the DA is matched.

1 = In addition to the frames it is already copying, the MAC copies all LLC, implementor, and reserved frames whose DA field indicates a 48-bit group address. The MAC will not copy group LLC frames it sent unless COPY_OWN = 1.

DISABLE_BRDCST—Disable Broadcast

When DISABLE_BRDCST is set (that is, 1), the MAC treats non-SMT broadcast frames (that is, DA is all 1's) exactly like other multicast frames of the same type. DISABLE_BRDCST affects the DD field of MAC-to-FSI END_DATA transfers for received broadcast frames. This bit DOES affect the A and C flags.

0 = The MAC treats broadcast frames normally (as described in the FDDI standard).

1 = The MAC treats a MAC, LLC, implementor, or reserved broadcast frames (i.e., DA is all ones) exactly as if it were another multicast frame of the same frame type. Hence, an LLC broadcast frame is recognized and copied only if the broadcast address is found in the CAM or COPY_GROUP_LLC = 1 (subject to COPY_OWN). The A_FLAG and C_FLAG are only set if the broadcast address is found in the CAM since COPY_GROUP_LLC has no effect on these indicators.

RUN_BIST—Run Built-In Self-Test

0 = The MAC operates normally.

1 = The MAC runs its internal BIST (see **12.2.2 MAC BIST Operation**).

RX_PARITY—Generate Odd or Even Receive Parity

0 = The MAC generates RPRITY so that RPRITY and RPATHx have odd parity.

1 = The MAC generates RPRITY so that RPRITY and RPATHx have even parity.

Note that RPRITY and RPATHx are internal buses only.

NOTE_ALL_FRAMES—Note All Frames

0 = The MAC sets the FRAME_RCVD bit in the MAC interrupt register A only when the FRAME_CT overflows (every 65536 frames).

1 = The MAC sets FRAME_RCVD in MAC_INTR_A whenever FRAME_CT is incremented (every frame). It is possible, in this case, for the DOUBLE_OVFL interrupt bit not to be set when FRAME_CT overflows (that is, if FRAME_RCVD was 0). Therefore, it is possible to have an incorrect frame count with no warning from the MAC. Because of this, NOTE_ALL_FRAMES should not be set unless the node processor can read FRAME_CT before it overflows or unless precise frame counts are not required.

7.3.4.5 MAC CONTROL REGISTER B (MAC_CNTRL_B) ACNTL = 1 0100 0001. The node processor/system interface can read and write this register at any time. The MAC only modifies the RESET_FIELD and FDX_MODE bits. The register is set to 0018H on power-up reset and is unaffected by MAC_Reset.

15	14	13	12	11	10	9	8
RING_PURGE	FDX_MODE	BRIDGE_STRIP	TXPARITY_ON	REPEAT_ONLY	LOSE_CLAIM	RESET_FIELD	
7	6	5	4	3	2	1	0
FSI_BEACON	DELAY_TOKEN	IGNORE_ SACAM	EXT_DA_ MATCH	X	MAC_MODE_ CTL	RCDAT_PARITY _ON	TXDAT_BAD_ PAR

X=Don't Care

RING_PURGE—Enable Ring Purging Mode

During ring purging mode, the MAC removes (purges) all frames from the ring. This bit is ignored by the MAC when RING_OPERATIONAL is false or when the MAC is in the FDX states. Unlike BRIDGE_STRIP, RING_PURGE does not affect which frames the MAC thinks it sent and hence which packets the MAC asks the FSI block to flush. Consequently, it is useful to set both BRIDGE_STRIP and RING_PURGE. Typically, there is only one purging station per ring. The choice of a ring purger, if any, is beyond the scope of the ANSI FDDI standard.

0 = The MAC operates normally.
1 = The MAC purges the ring upon every token rotation until the node processor resets this bit. The MAC performs the following operations:
Captures every token (unless RING_OPERATIONAL is zero),
Sends any FSI frames for which the token is usable,
Sends two special void frames (see the BRIDGE_STRIP bit description), and
Releases the token.

The kind of token released is specified by the TOKEN_SEND field in the packet request header of the last FSI frame sent with this token or is the kind of token captured if no FSI frames were sent. The MAC then purges all frames until one of the following occurs:

A special void frames returns;
A nonduplicate token returns;
RING_OPERATIONAL becomes false;
The transmitter enters the FDX states; or
The MAC is turned off (MAC_ON = 0).

Purging does not stop if a duplicate token is received (i.e., a token received while transmitting). Purging differs from stripping in that purging creates no frame fragments.

FDX_MODE—Enable Full-Duplex Operation

FDX_MODE is used in implementing point-to-point links and for diagnostics.

0 = The transmitter operates purely in ring mode.
1 = The next time the transmitter enters the Tx_Idle (T0) state, it transitions to the FDX_Idle state. The transmitter then alternates between the FDX_Idle and FDX_Data states, depending on whether or not there is a frame to send. The transmitter leaves FDX_Idle or FDX_Data upon:
Receiving a higher or lower claim frame,
Receiving any beacon frame,
Receiving a MAC_Reset (RESET_FIELD <> 00), or
FDX_MODE = 0 and the transmitter is in the FDX_Idle state.

RING_OPERATIONAL (which could be either zero or one) is frozen while in the FDX states. RING_OPERATIONAL and the FDX_MODE bit are cleared upon leaving the FDX states unless these states are left because FDX_MODE is cleared.

BRIDGE_STRIP—Use Bridge Stripping Algorithm

0 = The normal stripping algorithm is used, based upon matching the SA against MSA register, MLA register, and the CAM entries.
1 = An additional stripping mode is enabled whereby stripping occurs when the count of frames sent minus frames received (called SENT_COUNT) is greater than zero. In the transmitter, each time the MAC captures a token to send an FSI frame, the MAC will send a special void frame just before releasing the token. If RING_PURGE = 1, two special void frames are sent. The MAC transmits a special void frame with the following characteristics:
FC = 48-bit addressed void frame (0100 0000)
DA = MLA

SA = MLA
Zero INFO bytes

Once the token is released, the MAC will then strip frames until SENT_COUNT becomes zero. SENT_COUNT is cleared (hence, stripping also stops) when:

A special void frame is received.
A nonduplicate token is received.
A claim or beacon frame is received.
RING_OPERATIONAL is false.
The transmitter enters the FDX states.
The MAC is turned off (MAC_ON = 0).

SENT_COUNT is not cleared (stripping continues) if a duplicate token is received (i.e., a token received while transmitting).

Note that the stripping algorithm is also used by the MAC to determine (for FSI reception and frame association purposes) whether it believes it sent a frame.

This bit is ignored (treated as zero) when RING_OPERATIONAL is false or when the MAC is in the FDX states.

TXPARITY_ON—Transmit Parity Check On

0 = The TPRITY input in the TPATHx bus is ignored.
1 = TPATHx and TPRITY together must have odd parity (i.e., an odd number of the nine lines must be high), or else the MAC aborts the transmission of this frame and asserts TABORT.

Note that TPRITY and TPATHx are internal buses only.

REPEAT_ONLY—Repeat Only

The transmitter cannot capture the token.

0 = The transmitter operates normally.
1 = The MAC will not start sending any more frames from the FSI, although it can finish any frames it has started to send as well as sending any frames internally generated by the MAC (i.e., claim, beacon, and void frames). Specifically, the conditions USABLE_TOKEN and ANOTHER_FRAME are always zero; thus, the transmitter cannot capture a token. If it has the token, it cannot send any more frames although it can finish sending the frame it is currently sending (plus any associated token). The transmitter cannot start sending frames whose TOKEN_TYPE field in the packet request header allows the frame to be sent without a token. When REPEAT_ONLY = 1 and RING_PURGE = 1, the MAC will still capture the token, send two special void frames, and then release the same kind of token.

LOSE_CLAIM—Lose Claim

The transmitter always loses the claim bidding process.

0 = The transmitter works normally.
1 = The transmitter treats all Lower_Claim frames as Higher_Claim frames and all My_Beacon frames as Other_Beacon frames, and the receiver sets the H and L

bits to reflect this. All other Recovery_Required conditions are ignored (i.e., TVX expiration, TRT expiration when LATE_CT > 0, and T_Opr < T_REQ when RING_OPERATIONAL is one) while the MAC is in any of the states from which the Recovery_Required transitions originate (i.e., Tx_Idle, Tx_Repeat, TX_DATA, Tx_Token, or Tx_Void states). Because the Recovery_Required transitions cannot occur, the transmitter will never enter the Tx_Claim or Tx_Beacon states, except upon a MAC_Reset/CLAIMING, MAC_Reset/BEACONING, or upon TRT expiration while already in Tx_Claim (MAC then goes to Tx_Beacon). However, once it is in these states (e.g., if this bit is set while in Tx_Claim), the MAC may stay in any of these states for longer than usual, but it will eventually recover and enter the normal operational states.

RESET_FIELD—Reset Field

This field, which includes various types of MAC resets, is used to apply general signals to the whole of the core. The idea of a signal is that it only lasts for a single BYTCLK cycle, unlike the regular control bits whose effect continues as long as the bit is set. When this bit field is set to any value other than 00, a form of MAC_RESET occurs for one BYTCLK cycle assuming MAC_ON = 1. The MAC core then resets this field back to 00 on the next BYTCLK.

00 = Normal operation and no MAC_Reset occurs.
01 = A regular FDDI-specified MAC_Reset occurs.
10 = A combined MAC_Reset/BEACONING action occurs (i.e., a MAC_Reset followed by the transmitter going to the Tx_Beacon state). This action is equivalent to an SA_MA_CONTROL request (beacon) service primitive.
11 = A combined MAC_Reset/CLAIMING action occurs (i.e., a MAC_Reset followed by the transmitter going to the Tx_Claim state).

When the MAC is off (MAC_ON = 0), this bit field retains its last written value (i.e., it is not cleared upon the next rising edge of BYTCLK), and it has no effect until the MAC is subsequently turned on.

FSI_BEACON—Transmit Beacons from the FSI

FSI_BEACON controls the source of the beacon frames transmitted when the MAC transmit finite state machine transitions to the Tx_Beacon (T5) state.

0 = If FSI_BEACON is zero, the MAC sends internally created beacon frames with an INFO field consisting of four bytes of zeros (i.e., BEACON_TYPE is unsuccessful claim) if the MAC transmitter is in the TX_Beacon state.
1 = The MAC allows the FSI to send beacon frames generated by upper-level software. When the MAC is in the beacon state (T5)—a frame is available to transmit at the MAC-FSI interface and the BCN_FRAME bit is set in its packet request header—the MAC will send this frame. Otherwise, the MAC will send internally created beacon frames. The MAC will not attempt to skip over FSI-generated frames whose BCN_FRAME is zero to find later frames with a BCN_FRAME of one. Also, a frame with a BCN_FRAME of one will only be sent once. Hence, for the MAC to continue to send FSI-generated beacon frames, new frames need to be continually queued up to the MAC.

The FSI_BEACON has no effect, unless the MAC is in the TX_BEACON state (T5). Also when FSI_BEACON=1, only the BCN_FRAME, APPEND_CRC and EXTRA_FS

fields of the packet request header have any effect. The FC and address fields of the frames sent from the FSI are not checked in any way, although an incorrect CRC will still generate the BAD_CRC_SENT interrupt.

DELAY_TOKEN—Wait for FSI Data While Holding the Token

This function allows for slower delivery of the start of frame data at the FSI-MAC interface.

0 = The MAC ensures that exactly eight I-symbol pairs of preamble are sent between the ending delimiter of the last frame and the starting delimiter of the following frame or token. If the last frame transmission was aborted (i.e., no ending delimiter sent), then zero, one, or two additional I-symbol pairs may be sent, as measured from the last data symbol pair sent.

1 = The MAC waits up to an additional 32 cycles for the FSI to transfer a new TX_START after a TX_END transfer (see **10.4.1.2 Transmit Data Interface)** before releasing the token, if the M-bit of TX_END = 1.

7

IGNORE_SACAM—Ignore Source Address CAM Recognition

If EXT_DA_MATCH is set, then this bit is ignored.

0 = If the $\overline{\text{MATCH}}$ signal is asserted in the second cycle immediately following the last byte of the SA, then (assuming no special copy modes are set) the SA is stripped and the frame is flushed.

1 = The MAC ignores the $\overline{\text{MATCH}}$ signal in determining whether the SA matches or not.

EXT_DA_MATCH—Extended Destination Address Match Control

0 = Normal match mode. The $\overline{\text{MATCH}}$ signal is only examined by the MAC at the second byte following the end of a received DA or SA field. The LDADDR/ $\overline{\text{TR}}$/$\overline{\text{BR}}$_$\overline{\text{FWD}}$ pin functions as the LDADDR output.

1 = Extended DA match allows the user to delay asserting the $\overline{\text{MATCH}}$ or $\overline{\text{TR}}$_$\overline{\text{BR}}$_$\overline{\text{FWD}}$ signals up to and including the last byte of the FCS field. The packet can be flushed at any time by asserting $\overline{\text{REJECT}}$. SA_CAM match is not available with this option. (The LDADDR pin becomes an input signal, $\overline{\text{TR}}$_$\overline{\text{BR}}$_$\overline{\text{FWD}}$.)

This bit is set on power-up. To use the LDADDR pin in normal mode, the user (initialization firmware) must clear this bit.

MAC_MODE_CTL—MAC A and C Frame Status Bit Handling Option

This bit defines how the MAC sets, resets, or repeats the C-indicator bit when RABORT or $\overline{\text{REJECT}}$ occurs while the MAC is receiving a frame addressed to itself or recognized as receivable by itself.

0 = Option 1. Set the A and C bits according to the MAC_MODE_CTL = 0 function as defined in Table 5-6.

1 = Option 2. Set the A and C bits according to the MAC_MODE_CTL = 1 function as defined in Table 5-6.

RCDAT_PARITY_ON—Enable RCDATx Parity Checking

This bit enables parity checking on the RCDATx input to the MAC core.

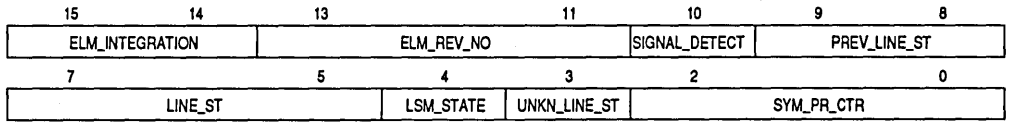
0 = Parity checking disabled.
1 = Parity checking enabled. A parity error sets the RCDAT_PAR_ERR bit in the MAC interrupt event register C. There are no changes in the input data and the MAC core continues to process the input data stream normally.

TXDAT_BAD_PAR—Enable Bad Parity on TX_DATA Bus

This bit enables bad parity to be generated on the TXDATx output lines of the MAC to the ELM for test purposes.

0 = Normal parity generated
1 = Inverted parity generated to test parity detection.

7.3.4.6 ELM STATUS REGISTER A (ELM_STATUS_A) ACNTL = 1 0001 0000. Status register A, which is read-only, is used to report status information about the line state machine (LSM).



ELM_INTEGRATION—ELM Integration Bits

00 = Stand alone ELM Revision Band C
01 = ELM integrated into CAMEL

ELM_REV_NO—ELM Revision Number

000 = ELM Revision B
001 = ELM in CAMEL core
010 = ELM in CAMEL core revision B

SIGNAL_DETECT—Signal Detect Value

This bit contains the inverse of the value on the SD input pin.

0 = Signal Detected
1 = Signal Not Detected

PREV_LINE_ST—Previous Line State

This field contains the value of the previous line state when the line state changes from Quiet, Master, Halt, or Idle (ILS16, where ILS16 is achieved after 16 I-symbols) to another line state. When the line state changes from anything else, this field is not updated. These two bits are defined as follows:

00 = Quiet Line State
01 = Master Line State
10 = Halt Line State
11 = Idle Line State (ILS16 achieved after 16 I-symbols)

LINE_ST—Current Line State

This field contains the most recently recognized line state by the LSM. LINE_ST is further defined as follows:

000 = Noise Line State
001 = Active Line State
010 = Reserved
011 = Idle Line State (ILS4 achieved after 4 I-symbols)
100 = Quiet Line State
101 = Master Line State
110 = Halt Line State
111 = Idle Line State (ILS16 achieved after 16 I-symbols)

LSM_STATE—Line State Machine State

This field contains the state bit of the LSM.

0 = Not Active Line State
1 = Active Line State

UNKN_LINE_ST—Unknown Line State

This bit is the unknown line state indication.

0 = Line State Known
1 = Line State Unknown

SYM_PR_CTR—Symbol Pairs Counter

This field contains the LSM symbol pairs counter. When the count reaches seven, indicating eight consecutive like symbol pairs, then LINE_ST is set with the new line state, and the UNKN_LINE_ST bit is reset. Note that Idle Line State is reached after just two I-symbol pairs.

7.3.4.7 ELM STATUS REGISTER B (ELM_STATUS_B) ACNTL = 1 0001 0001. Status register B, which is read-only, contains signals and status from the repeat filter and physical connection management (PCM) state machine.

15 14	13 12	11	10 8
RF_STATE	PCI_STATE	PCI_SCRUB	PCM_STATE

7 6	5	4	3	2 0
PCM_STATE / PCM_SIGNALING	LSF	RCF	TCF	BREAK_REASON

RF_STATE—Repeat Filter State

This field contains the state bits of the repeat filter state machine. The states are defined as follows:

00 = REPEAT
01 = IDLE
10 = HALT1
11 = HALT2

PCI_STATE—Physical Connection Insertion State

This field contains the state bits of the PCI state machine. The states are defined as follows:

00 = REMOVED
01 = INSERT_SCRUB
10 = REMOVE_SCRUB
11 = INSERTED

PCI_SCRUB—Physical Connection Insertion Scrub

This flag indicates that the scrubbing function is operating—that is, I-symbol pairs are being sourced on the PRCDATx output pins.

PCM_STATE—Physical Connection Management State

This field contains the state bits of the PCM state machine. The states are defined as follows:

0000 = PC0 (OFF)
0001 = PC1 (BREAK)
0010 = PC2 (TRACE)
0011 = PC3 (CONNECT)
0100 = PC4 (NEXT)
0101 = PC5 (SIGNAL)
0110 = PC6 (JOIN)
0111 = PC7 (VERIFY)
1000 = PC8 (ACTIVE)
1001 = PC9 (MAINT)
1010–0111 = Reserved

PCM_SIGNALING—Physical Connection Management Signaling

This PCM flag indicates that the transmit vector register has been written and the PCM is in the process of transmitting these bits to its neighboring PCM. The transmit vector register and transmit vector length registers cannot be written when this flag is set.

LSF—Line State Flag

The PCM uses this bit to indicate that a given line state has been received since entering the current state. It is cleared on every change of PCM state.

RCF—Receive Code Flag

The PCM uses this bit to indicate that the receive logic has started execution. This flag is used to prevent the receive station management PCM code from being started multiple times while in the NEXT state.

TCF—Transmit Code Flag

The PCM uses this bit to indicate that the transmit logic has started execution. This flag is used to prevent the transmit station management PCM code from being started multiple times while in the NEXT state.

BREAK_REASON—Break Reason

This field, which indicates the reason for the PCM state machine's last transition to the BREAK state, is defined as follows:

000 = The PCM state machine has not gone to the BREAK state
001 = PC_Start Issued
010 = TPC Timer Expired after T_OUT
011 = TNE_Timer Expired after NS_MAX
100 = Quiet Line State Detected
101 = Idle Line State Detected
110 = Halt Line State Detected
111 = Reserved

7.3.4.8 MAC RECEIVE STATUS REGISTER (MRX_STATUS) ACNTL = 1 0110 0100. The receive status register holds the status flags, the comparison state (e.g., the H_Flag, L_Flag, and M_Flag), and the receiver FSM state. The node processor can read the receive status register at any time but can never write to this register. It is initialized to E020 by power-up reset and by MAC_Reset. The flags displayed in this register are internal flags used as described in the ANSI FDDI MAC standard.

15	14	13	12	11	10	9	8
RX_FSM_STATE			R_FLAG	E_FLAG	FS_STATE		
7	6	5	4	3	2	1	0
N_FLAG	FR_PARS_STATE			L_FLAG	H_FLAG	M_FLAG	A_FLAG

RX_FSM_STATE—Receiver Finite State Machine State

This state machine controls the overall operation of the MAC receiver.

000 = Await_Sd (R1)—Wait for JK of new frame
001 = Rc_FS (R4)—Receive and decode FS
010 = Rc_FC (R2)—Receive and decode FC byte
011 = Rc_Info (R3)—Receive DA, SA, INFO, and CRC
100 = Chk_TK2 (R5')—State used to repeat TT
101 = Listen (R0)—Wait for first idle
110 = Chk_TK1 (R5)—Receive TT of token
111 = Rc_Off—MAC is turned off

R_FLAG—Current Value of R-FLAG

In general, this bit indicates whether the last token received was a restricted token or a nonrestricted token. This bit is affected by token fragments and duplicate tokens.

0 = A nonrestricted token FC was received, a claim or beacon frame was received, or the MAC is turned off (MAC_ON = 0).
1 = A restricted token FC was received.

E_FLAG—Current Value of E-Flag

FS_STATE—Frame Status Machine State

These bits indicate the state of the state machine that parses the frame status field for all frames (including those that have an odd number of data symbols).

000 = Wait_Ed—Wait for Ending Delimiter(T)
001 = Reserved
010 = Rc_4_5—Expecting 4 and 5 Indicators
011 = Rc_A_C—Expecting A and C Indicators
100 = Rc_5_x—Expecting 5 Indicator
101 = Rc_C_4—Expecting C and 4 Indicators
110 = Wait_FS_end—Wait for End of Frame Status
111 = Rc_E_A—Expecting E and A Indicators

N_FLAG—Current Value of N-Flag

FR_PARS_STATE—Frame Parsing State Machine State

These bits indicate the state of the frame parsing state machine that parses the DA, SA, INFO, and CRC fields to initiate the DA_Actions, SA_Actions, and CT_Actions and control the CAM interface signals.

000 = Parse_DA—Receiving DA
001 = Parse_SA—Receiving SA
010 = Parse_Reset—Not receiving a frame
011 = Reserved
100 = Parse_FCS—Receiving possible FCS
101 = Parse_INFO—Receiving INFO field of MAC frame
110 = Parse_Value—Receiving rest of INFO field
111 = Reserved

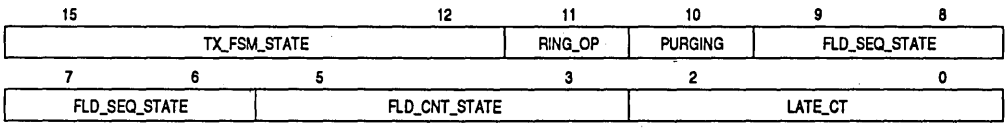
L_FLAG—Current Value of L-Flag

H_FLAG—Current Value of H-Flag

M_FLAG—Current value of M-Flag

A_FLAG—Current Value of A-Flag

7.3.4.9 MAC TRANSMIT STATUS REGISTER (MTX_STATUS) ACNTL = 1 0110 0101. The transmit status register holds the transmit FSM state, LATE_CT, RING_OPERATIONAL, and the state of the frame transmission machine. The node processor can read the transmit status register at any time but can never write to this register. It is initialized to F031 on power-up reset and by a MAC_Reset.



TX_FSM_STATE—Transmit Finite State Machine State

These bits indicate the state of the overall operation of the MAC transmit FSM.

0000 = Tx_Idle (T0)—Constantly transmit I-symbols
0001 = TX_DATA (T2)—Transmit data frames

0010 = Tx_Token (T3)—Transmit token
0011 = Tx_Void—Transmit special void frame
0100 = Tx_Repeat (T1)—Repeat incoming frame/token
0101 = Reserved
0110 = Tx_Beacon (T5)—Constantly transmit beacon frames
0111 = Tx_Claim (T4)—Constantly transmit claim frames
1000 = FDX_Idle —Constantly transmit I-symbols
1001 = Reserved
1010 = Reserved
1011 = Reserved
1100 = FDX_Data—Transmit FSI FDX data frame
1101 = Reserved
1110 = Reserved
1111 = Tx_Off—MAC is turned off

RING_OP—Ring Operational

This value indicates whether or not the ring is operational.

0 = RING_OPERATIONAL is cleared, indicating that the ring is not operational.
1 = RING_OPERATIONAL is set, indicating that the ring is operational.

PURGING—Purging

This bit indicates the current value of purging.

1 = Transmitter is currently purging the ring. The transmitter will not enter repeat mode (Tx_Repeat state). This bit is set upon sending the end of the first of the two special void frames sent as a result of RING_PURGE being set.
0 = Transmitter no longer purging ring. This bit is set to zero when a special void frame is returned or a nonduplicate token is returned, RING_OPERATIONAL becomes zero, the transmitter enters the FDX states, or the MAC is turned off (MAC_ON = 0). This bit is not cleared (stripping continues) if a duplicate token is received (i.e., a token received while transmitting).

FLD_SEQ_STATE—Field Sequence State

These bits indicate the state of the field sequence machine that is responsible for creating tokens, beacon, claim, and special void frames, for adding the preamble, JK, and frame status, and for controlling the addition of the FCS to frames that the FSI passes to the MAC for transmission.

0000 = Pre_State—Transmit preamble (idles)
0001 = Post_State—Transmit postamble (idles)
0010 = Data_FC_State—Transmit FC for data frame
0011 = Data_DA_State—Transmit DA for data frame
0100 = SD_State—Transmit JK for token/all frames
0101 = CRC_State—Transmit FCS (all required frames)
0110 = Ed_State—Transmit TT (token) or TR (otherwise)
0111 = Data_State—Transmit INFO field for data frame
1000 = Data_SA_State—Transmit SA for data frame
1001 = FS_State—Transmit RR + any extra FS required

1010 = Unused
1011 = Unused
1100 = FC_State—Transmit FC for all but data frames
1101 = Info_State—Transmit INFO (claim/beacon frame)
1110 = DA_State—Transmit DA (claim/beacon/void)
1111 = SA_State—Transmit SA (claim/beacon/void)

FLD_CNT_STATE—Field Count State

These bits indicate the state of the field count machine that counts down to determine when each of the various fields have ended and when the field sequence state machine should proceed to its next state.

000 = Field ends after this byte
110 = Field has 8 more bytes
111 = Field has 7 more bytes
001 = Field has 6 more bytes
010 = Field has 5 more bytes
011 = Field has 4 more bytes
100 = Field has 3 more bytes
101 = Field has 2 more bytes

LATE_CT—Current Value of LATE_CT

This 3-bit counter holds the current value of LATE_CT, which is simply the number of times that the TRT timer has expired since the receiver has seen and/or created a token (except for the second rotation of the token). This counter does not wrap around—i.e., if TRT expires when LATE_CT is seven (111), LATE_CT will continue to be seven until a 'Clear LATE_CT' or 'Set LATE_CT=1' action is performed by the transmitter.

7.3.5 Interrupt Registers

The interrupt registers include three CAMEL global-purpose registers, two ELM core registers, and six MAC core registers.

The interrupt event registers report events to the node processor. When an interrupt-causing event occurs, the corresponding bit is set in the appropriate interrupt event register. This bit remains set until the node processor reads this register which clears all bits in the register to zero.

When one of the bits in the interrupt event register is set and the corresponding bit in the matching interrupt mask register is also set, the $\overline{\text{CAMINT}}$ pin is asserted, and the CIN status bit in the FSI status register 1 (SR1) is set. The appropriate bit in the CAMEL interrupt location register (CAMEL_INTR_LOC) is also set. The interrupt can be cleared by reading the interrupt event register(s) or by clearing the appropriate bits in the interrupt mask register(s).

The interrupt event registers are read-only. They are cleared upon power-up reset.

7.3.5.1 CAMEL INTERRUPT LOCATION REGISTER (CAMEL_INT_LOC) ACNTL = 1 1000 0110. The CAMEL interrupt location register is used to determine the source of the

interrupt event that caused the assertion of the $\overline{\text{CAMINT}}$ pin. This register determines if the interrupt event is associated with the MAC core, ELM core, or CAMEL global control logic. This register is read-only and does not need to be read-only clear since it is connected to the individual interrupt lines from the three core sections of the chip. The individual interrupt event registers will be used to clear any of the core logic interrupts. The core logic interrupt signals will be negated as a result of a node processor read operation to the interrupt event registers in the chip. This action will also negate the $\overline{\text{CAMINT}}$ pin.

It is possible and very probable that more than one section of the chip will have an interrupt pending, meaning that more than one bit in the location register will be a one.

The purpose of this register is to preserve present software implementations for doing interrupt service routines done previously for the separate implementations of the MAC and ELM chips. More concise information is not provided in this register for the particular interrupt event register since this would possibly save only two node processor reads at best.

The node processor can read the CAMEL interrupt location register at any time using CSR address 86. Since this is a read only register, any writes to this register will cause a node processor error interrupt even being asserted in the CAMEL_INTR register.This register is initialized to zero on completion of power-up reset.

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	0	0	0	CAMEL_INT	MAC_INT	ELM_INT

7.3.5.2 CAMEL INTERRUPT EVENT REGISTER (CAMEL_INTR) ACNTL = 1 1000 0100. The CAMEL interrupt event register (CAMEL_INTR) is used to report global events. When one of the indicated events occurs, it sets the corresponding bit in this register. This bit remains set until the node processor reads this register, which clears all bits in the register to zero.

When a bit in this register is set and the corresponding bit in the CAMEL interrupt mask register (CAMEL_MASK) is also set, the $\overline{\text{CAMINT}}$ pin is asserted, and the CIN status bit in FSI status register 1 (SR1) is set. The CAMEL_INT bit in the CAMEL interrupt location register (CAMEL_INTR_LOC) is also set. The interrupt can be negated by reading CAMEL_INTR or by clearing the appropriate bits in CAMEL_MASK.

CAMEL_INTR is read-only. It is cleared by power-up reset.

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	NP_ERR

Bits 15–1—Reserved

NP_ERR—Node Processor Error

Node processor errors occur when undefined registers are accessed or when registers are accessed at incorrect times. These cases include:

1. Read/Write to an undefined register
2. Write to a Read-Only Register
3. Read to a Write-Only Register
4. Write to TPC_LOAD_VALUE when PCM_STATE<>MAINT.
5. Write to TNE_LOAD_VALUE when PCM_STATE<>MAINT or NOISE_TIMER = 1.
6. Write to XMIT_VECTOR or VECTOR LENGTH when PCM_SIGNALING = 1.
7. Write to a Read/Control-Write MAC register when MAC_ON = 1.

7.3.5.3 ELM INTERRUPT EVENT REGISTER (ELM_INT) ACNTL = 1 0001 0111. The ELM interrupt event register (ELM_INTR) is used to report PHY events. When one of the indicated events occurs, it sets the corresponding bit in this register. This bit remains set until the node processor reads this register, clearing all bits in the register to zero.

When a bit in this register is set and the corresponding bit in the ELM interrupt mask register (ELM_MASK) is also set, the $\overline{\text{CAMINT}}$ pin is asserted, and the CIN status bit in FSI status register 1 (SR1) is set. The ELM_INT bit in the CAMEL interrupt location register (CAMEL_INTR_LOC) is also set. The interrupt can be cleared by reading ELM_INTR or by clearing the appropriate bits in the ELM_MASK register.

While the RUN_BIST bit in ELM_CNTRL_A is set, all ELM interrupts are masked except BIST_DONE. Since this is the only ELM interrupt that can occur in this situation, BIST_DONE does not occupy a bit in ELM_INTR. This interrupt is cleared by clearing the RUN_BIST bit in ELM_CNTRL_A.

ELM_INTR is read only. It is cleared by power-up reset.

15	14	13	12	11	10	9	8
0	SD	LE_CTR	MINI_CTR	VSYM_CTR	PHYINV	EBUF_ERR	TNE_EXPIRED
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TPC_EXPIRED	PCM_ENABLED	PCM_BREAK	SELF_TEST	TRACE_PROP	PCM_CODE	LS_MATCH	PARITY_ERR

Bit 15—Reserved

In the standalone ELM chip, this was the NP_ERR indicator bit, which is now located in the CAMEL_INTR and CAMEL_MASK registers.

SD—Signal Detect

This bit indicates signal detect—that is, the SD nput pin has been asserted.

LE_CTR—Link Error Counter

This bit indicates that the link error event counter has reached the value contained in the link error event threshold register.

MINI_CTR—Minimum Idle Counter

This bit indicates that either of the following events have occurred in the minimum idle counter register—the idle counter minimum detector has changed to a lower value, or the minimum idle gap counter has incremented or overflowed (depending on the MINI_CTR_INTRS bit in ELM control register A).

VSYM_CTR—Violation Symbol Counter

This bit indicates that the violation symbol counter has incremented or overflowed (depending on the VSYM_CTR_INTRS bit in ELM control register A).

PHYINV—Physical Layer Invalid

This bit indicates that the physical layer invalid signal has been asserted by the PCM.

EBUF_ERR—Elasticity Buffer Error

This bit indicates that the elasticity buffer has experienced an overflow or an underflow. EBUF_ERR is only reset after recognition of Idle or Active Line States.

TNE_EXPIRED—TNE Timer Expired

This bit indicates that the TNE timer has expired—i.e., reached zero.

TPC_EXPIRED—TPC Timer Expired

TPC_EXPIRED indicates that the TPC timer has expired—i.e., reached zero.

PCM_ENABLED—Physical Connection Management Enabled

PCM_ENABLED indicates that the PCM has asserted CF_JOIN (ANSI state transition PC(88b)), has completed scrubbing (for class M, A, or B stations), and is in the ACTIVE state.

PCM_BREAK—Physical Connection Management Break

This bit indicates that the PCM has entered the BREAK state.

SELF_TEST—Self-Test

This bit indicates that a Quiet or Halt Line State has been received while the PCM is in the TRACE state.

TRACE_PROP—Trace Propagate

This bit indicates that a Master Line State has been received while the PCM is in the ACTIVE or TRACE state.

PCM_CODE—Physical Connection Management Code

PCM_CODE indicates that the PCM has completed transmitting the last bit in the vector written to the transmit vector register and has received the corresponding bit of the receive vector register or that the Link Confidence Test has been completed.

LS_MATCH—Line State Match

This bit indicates that the line state detected equals the line state in the MATCH_LS field of ELM control register B (ELM_CNTRL_B).

PARITY_ERR—Parity Error

This bit indicates that a parity error has been detected on the TXDATx input pins. The parity feature was designed for ELMs implemented in a concentrator. Since there is no parity feature between the MAC and the ELM, this bit should be masked when the ELM is used in an end station. The frame data is protected by the FCS field when the data path is between the ELM and the MAC.

7.3.5.4 MAC INTERRUPT EVENT REGISTER A (MAC_INTR_A) ACNTL = 1 0110 0010. The MAC interrupt event register A (MAC_INTR_A) is used to report MAC events. When one of the indicated events occurs, it sets the corresponding bit in this register. This bit remains set until the node processor reads this register, which clears all bits in the register to zero.

When a bit in this register is set and the corresponding bit in the MAC interrupt mask register A (MAC_MASK_A) is also set, the $\overline{\text{CAMINT}}$ pin is asserted, and the CIN status bit in FSI status register 1 (SR1) is set. The MAC_INT bit in the CAMEL interrupt location register (CAMEL_INTR_LOC) is also set. The interrupt can be negated by reading MAC_INTR_A or by clearing the appropriate bits in MAC_MASK_A.

While the RUN_BIST bit in MAC_CNTRL_A is set, all MAC interrupts are masked except BIST_DONE. Since this is the only MAC interrupt that can occur in this situation, BIST_DONE does not occupy a bit in MAC_INTR_A. This interrupt is cleared by clearing the RUN_BIST bit in MAC_CNTRL_A.

MAC_INTR_A is read-only. It is cleared by power-up reset.

15	14	13	12	11	10	9	8
PH_INVALID	U_TOKEN_RCVD	R_TOKEN_RCVD	TKN_CAPTURE	BEACON_RCVD	CLAIM_RCVD	FRAME_ERR	FRAME_RCVD
7	6	5	4	3	2	1	0
DOUBLE_OVFL	RING_OP_CHNG	BAD_T_OPR	TVX_EXPIR	LATE_TKN	RCVRY_FAIL	DUPL_TKN	DUPL_ADDR

PH_INVALID—PHY Invalid Indication Detected

This event is signaled when the ELM passes the MAC a PHY Invalid symbol, indicating that the PHY is in a line state other than Active Line State or Idle Line State. When the receiver FSM is operational, receipt of PHY Invalid symbol causes the FSM to enter the listen (R0) state.

U_TOKEN_RCVD—Unrestricted Token Received

This event is signaled when an unrestricted token is received (i.e., when the receiver FSM signals TK_Received and the R_Flag is cleared), regardless of whether the token is repeated or captured by the transmitter. RING_OPERATIONAL does not affect the setting of this bit.

R_TOKEN_RCVD—Restricted Token Received

This event is signaled when a restricted token is received (i.e., when the receiver FSM signals TK_Received and the R_Flag is set), regardless of whether the token is repeated or captured by the transmitter. RING_OPERATIONAL does not affect the setting of this bit.

TKN_CAPTURE—Token Has Been Captured

This event is signaled by the transmitter FSM when the token is captured. Specifically, this bit is set when the transmitter transitions from Tx_Idle to Tx_Data or Tx_Void due to the receiver signaling TK_Received (FDDI MAC transition T(02), not T(10a) or T(03)).

This event is signaled even if the token was captured only because RING_PURGE is set. This event is not signaled if the transmitter transitions to Tx_Data or Tx_Void without a token because the TOKEN_TYPE field in the packet request header indicates that no token is required.

BEACON_RCVD—MY_BEACON or OTHER_BEACON Frame Received

This bit is set when the receiver FSM signals My_Beacon or Other_Beacon. This MAC requires the received E-indicator to be present and to be an R-symbol (in addition to FDDI requirements) for My_Beacon or Other_Beacon to be signaled and for this bit to be set.

CLAIM_RCVD—MY_CLAIM, HIGHER_CLAIM, or LOWER_CLAIM Frame Received

This bit is set when the receiver FSM signals My_Claim, Higher_Claim, or Lower_Claim. This MAC requires the received E-indicator to be present and to be an R-symbol (in addition to FDDI requirements) for My_Claim, Higher_Claim, or Lower_Claim to be signaled and for this bit to be set.

FRAME_ERR—Frame Format Error or Locatable Frame Error Detected

This event is signaled when LOST_CT or ERROR_CT is incremented (see **7.3.10 MAC Counter Registers** for definition).

FRAME_RCVD—Frame Received

When the NOTE_ALL_FRAMES bit in MAC control register A is one, this event occurs every time FRAME_CT is incremented. When NOTE_ALL_FRAMES is zero, this event occurs every time FRAME_CT overflows.

DOUBLE_OVFL—Double Counter Overflow

This event indicates that some frames, format errors, or locatable errors have been lost for counting purposes. (See **7.3.10 MAC Counter Registers** for descriptions of frames, format errors, and locatable errors.) This event occurs if:

1. ERROR_CT overflows (i.e., wraps around from 63 to 0), or
2. LOST_CT overflows (i.e., wraps around from 63 to 0), or
3. FRAME_CT overflows (i.e., wraps around from 65535 to 0) when the FRAME_RCVD is one.

Note that when NOTE_ALL_FRAMES is one, it is possible for the DOUBLE_OVFL bit not to be set when FRAME_CT overflows (i.e., if FRAME_RCVD is zero). Hence, it is possible to have an incorrect frame count and no warning from the MAC.

During the cycle that this register is read (and hence cleared), FRAME_RCVD is considered cleared. Thus, if NOTE_ALL_FRAMES is zero, FRAME_CT is 65535, and a frame is received the same cycle that this register is read/cleared, the FRAME_RCVD bit is set to one, FRAME_CT wraps around to zero, and the rest of this register is cleared (i.e., DOUBLE_OVFL is read as zero and remains zero), which is the proper behavior.

RING_OP_CHNG—RING_OPERATIONAL Flag Changed

This bit is set when RING_OPERATIONAL changes from zero to one or vice versa.

BAD_T_OPR—T_OPR < T_REQ When Ring Is Operational

This bit is set when the transmitter FSM takes the Recovery_Required transition because RING_OPERATIONAL is one and T_Opr < T_REQ. This can only happen when a MAC_RESET, which clears RING_OPERATIONAL and sets T_Neg = T_Max, is followed by reception of a token (Pass_Actions performed), which sets RING_OPERATIONAL and sets T_Opr = T_Max (since T_Neg still equals T_Max), followed by reception of My_Claim or a Higher_Claim, since that recomputes T_Neg (and hence T_Opr).

If the LOSE_CLAIM control bit is one, BAD_T_OPR is not set since the MAC ignores the "RING_OPERATIONAL is one *and* T_Opr < T_Req" condition in this case (i.e., no transition is taken in this case).

TVX_EXPIR—TVX Timer Expiration

This bit is set when the TVX timer expires and causes a Recovery_Required transition in the transmitter FSM. This bit is not set when the TVX expires if LOSE_CLAIM is true or the transmitter FSM is in the Tx_Claim, Tx_Beacon, or FDX states, since the timer expiration in this case causes no state transition.

LATE_TKN—TRT TIMER Expiration When LATE_CT > 0

This bit is set when the TRT timer expires and causes a Recovery_Required transition in the transmitter FSM. This bit is not set when the TRT expires if LOSE_CLAIM is true or the transmitter FSM is in the Tx_Claim, Tx_Beacon, or FDX states, since the timer expiration in this case causes no state transition.

RCVRY_FAIL—Recovery Failure

This bit is set when the TRT timer expires while the transmitter FSM is in the Tx_Claim or Tx_Beacon state (unless there is some overriding transition caused by, for example, a MAC_Reset or a claim/beacon received). It is possible for this bit to be set even if LOSE_CLAIM is true.

DUPL_TKN—Duplicate Token Detected

This bit is set when the transmitter believes that it is holding the token (i.e., when the transmitter FSM is in the Tx_Data, Tx_Token, Tx_Void, or FDX states) and a token is received.

DUPL_ADDR—Duplicate Address Detected

This bit is set when a frame is received with a matching individual DA and the received A indicator is an S-symbol (meaning that another station also matched this individual address). Specifically, this bit is set when A_FLAG = true, E_FLAG = false, the I/G bit of the DA = 0, and Ar = S-symbol.

7.3.5.5 MAC INTERRUPT EVENT REGISTER B (MAC_INTR_B) ACNTL = 1 0110 0011. The MAC interrupt event register B (MAC_INTR_B) is used to report MAC events. When one of the indicated events occurs, it sets the corresponding bit in this register. This bit remains set until the node processor reads this register, which clears all bits in the register to zero.

When a bit in this register is set and the corresponding bit in the MAC interrupt mask register B (MAC_MASK_B) is also set, the $\overline{\text{CAMINT}}$ pin is asserted, and the CIN status bit in FSI status register 1 (SR1) is set. The MAC_INT bit in the CAMEL interrupt location register (CAMEL_INTR_LOC) is also set. The interrupt can be negated by reading MAC_INTR_B or by clearing the appropriate bits in MAC_MASK_B.

While the RUN_BIST bit in MAC_CNTRL_A is set, all MAC interrupts are masked except BIST_DONE. Since this is the only MAC interrupt that can occur in this situation, BIST_DONE does not occupy a bit in MAC_INTR_B. This interrupt is cleared by clearing the RUN_BIST bit in MAC_CNTRL_A.

MAC_INTR_B is read-only. It is cleared by power-up reset.

15	14	13	12	11	10	9	8
MY_BEACON	OTHER_BEACON	HIGHER_CLAIM	LOWER_CLAIM	MY_CLAIM	BAD_T_BID	PURGE_ERR	BRIDGE_STRIP_ERR
7	6	5	4	3	2	1	0
WON_CLAIM	0	SI_ERR	NOT_COPIED	FDX_CHANGE	BIT4_IS_S	BIT5_IS_S	BAD_CRC_SENT

MY_BEACON—My Beacon

This bit is set when the receiver FSM signals the MY_BEACON event. The conditions that cause the receiver FSM to signal this condition are described in the ANSI FDDI MAC standard in the MAC receiver FSM text and state diagram. Although the receiver

only asserts this signal for one clock cycle, as with all MAC interrupts, this interrupt remains set until it is read by the external processor.

OTHER_BEACON—Other Beacon

This bit is set when the receiver FSM signals the OTHER_BEACON event. The conditions that cause the receiver FSM to signal this condition are described in the ANSI FDDI MAC standard in the MAC receiver FSM text and state diagram. Although the receiver only asserts this signal for one clock cycle, as with all MAC interrupts, this interrupt remains set until it is read by the external processor.

HIGHER_CLAIM—Higher Claim

This bit is set when the receiver FSM signals the HIGHER_CLAIM event. The conditions that cause the receiver FSM to signal this condition are described in the ANSI FDDI MAC standard in the MAC receiver FSM text and state diagram. Although the receiver only asserts this signal for one clock cycle, as with all MAC interrupts, this interrupt remains set until it is read by the external processor.

LOWER_CLAIM—Lower Claim

This bit is set when the receiver FSM signals the LOWER_CLAIM event. The conditions that cause the receiver FSM to signal this condition are described in the ANSI FDDI MAC standard in the MAC receiver FSM text and state diagram. Although the receiver only asserts this signal for one clock cycle, as with all MAC interrupts, this interrupt remains set until it is read by the external processor.

MY_CLAIM—My Claim

This bit is set when the receiver FSM signals the MY_CLAIM event. The conditions that cause the receiver FSM to signal this condition are described in the ANSI FDDI MAC standard in the MAC receiver FSM text and state diagram. Although the receiver only asserts this signal for one clock cycle, as with all MAC interrupts, this interrupt remains set until it is read by the external processor.

BAD_T_BID—Bad T_Bid

This bit is set when a claim frame is received with SA = MLA register and the receiver FSM signals HIGHER_CLAIM or LOWER_CLAIM, indicating that T_Bid <> T_Req.

PURGE_ERR—Purge Error

This bit indicates that an error has been detected in the purging process. PURGE_ERR is set when the MAC receives a token and the MAC is not transmitting data, void, or token frames and is actively purging the ring (i.e., removing all frames) while awaiting the return of one of its special void frames. A token received while the MAC is transmitting (i.e., a duplicate token) will not set this interrupt bit nor terminate the purging process. On the other hand, such a token will invoke the DUPL_TKN interrupt.

BRIDGE_STRP_ERR—Bridge Strip Error

This bit indicates that an error has been detected in the bridge strip process. BRIDGE_STRIP_ERR is set when the MAC receives a token and the MAC is not transmitting data, void, or token frames and the SENT_COUNT register > 0. A token received while the MAC is transmitting (i.e., a duplicate token) will not set this interrupt bit nor clear the SENT_COUNT register. On the other hand, such a token will invoke the DUPL_TKN interrupt.

WON_CLAIM—Won Claim

This bit is set when the MAC starts to issue a token as a result of winning the claim process (i.e., upon receiving a MY_CLAIM while in the Tx_Claim state). Note that because of subsequent events, the MAC may not actually finish sending the token.

Bit 6—Reserved

In previous versions of the standalone MAC chip, this was the NP_ERR indicator bit, which is now located in the CAMEL_INTR and CAMEL_MASK registers.

SI_ERR—System Interface Error

This bit is set when the MAC detects an error in the FSI/MAC transmit interface. For example, this bit is set when the TPRITY signal indicates a parity error and TXPARITY_ON is enabled or when the TXCTLx lines do not progress through their required cycle. The TPRITY and TXCTLx lines are internal buses only.

NOT_COPIED—Addressed Frame Not Copied

This bit is set when a frame is addressed to this station but cannot be copied. Specifically, this bit is set when:

1. The receiver FSM signals FR_Received (i.e., FDDI MAC receiver transition R(41f) or R(40b)),
2. The A_FLAG is set (i.e., DA matches MSA register, MLA register, or CAM or DA is broadcast and DSABL_BRDCST is zero),
3. The E_FLAG is cleared (i.e., valid data length and valid CRC or implementor frame and E-indicator must be an R-symbol),
4. The C_FLAG is cleared (i.e., FSI aborted reception via RABORT),
5. The N_FLAG is cleared (i.e., not a secondary NSA frame), and
6. The frame is not a MAC or void frame.

Therefore, this bit is never set for secondary NSA frames but can be set for primary NSA frames.

FDX_CHANGE—FDX Mode Change

This bit is set when the transmitter enters the FDX mode of operation (i.e., first enters either the FDX_Idle state or FDX_Data state) or when it leaves FDX mode (i.e., first leaves both of these states but does not go to the off state).

BIT4_IS_S—Bit 4 Indicator S-Symbol Received

This bit is set when the fourth control indicator received is an S-symbol and SET_BIT4 = 1 in MAC control register A.

BIT5_IS_S—Bit 5 Indicator S-Symbol Received

This bit is set when the fifth control indicator received is an S-symbol and SET_BIT_5=1 in MAC control register A.

BAD_CRC_SENT—Bad CRC Sent

This bit indicates that a packet with bad CRC has been transmitted. This bit is set when the transmitter is requested to send a packet without adding an FCS field onto the end of it (presumably because the CRC has already been computed and added to the end of the packet) and the transmitter detects that the precomputed CRC is incorrect. The transmitter still sends the packet as if it did not detect a problem. This bit is set even for reserved for implementor frames. On the other hand, this bit is not set if the MAC aborts the transmission of the frame (i.e., sends a fragment).

7.3.5.6 MAC INTERRUPT EVENT REGISTER C (MAC_INTR_C) ACNTL = 1 0101 1101. The MAC interrupt event register C (MAC_INTR_C) is used to report MAC events. When one of the indicated events occurs, it sets the corresponding bit in this register. This bit remains set until the node processor reads this register, which clears all bits in the register to zero.

When a bit in this register is set and the corresponding bit in the MAC interrupt mask register C (MAC_MASK_C) is also set, the $\overline{\text{CAMINT}}$ pin is asserted, and the CIN status bit in FSI status register 1 (SR1) is set. The MAC_INT bit in the CAMEL interrupt location register (CAMEL_INTR_LOC) is also set. The interrupt can be negated by reading MAC_INTR_C or by clearing the appropriate bits in MAC_MASK_C.

While the RUN_BIST bit in MAC_CNTRL_A is set, all MAC interrupts are masked except BIST_DONE. Since this is the only MAC interrupt that can occur in this situation, BIST_DONE does not occupy a bit in MAC_INTR_C. This interrupt is cleared by clearing the RUN_BIST bit in MAC_CNTRL_A.

Note that when the void timer count exceeds 64K, the timer wratps around and the void time register will not be loaded. This bit can indicate configuration problems in the ring.

MAC_INTR_C is read-only. It is cleared by power-up reset.

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	0	0	0	RCDAT_PAR_ERR	VOID_TIME_REG_RDY	VOID_TIMER_OVF	TKN_CNT_OVF

Bits 15–4—Reserved

RCDAT_PAR_ERR—RCDATx Parity Error

This bit is set when a parity error occurs on the RCDATx input lines and the RCDAT_PARITY_ON bit in MAC control register B is set.

VOID_TIME_REG_RDY—Void Time Register Ready

This bit is set when the void timer loads the void time register with a new count. VOID_TIME_REG_RDY indicates that a new timing of the ring latency was done and the void time register contains the updated latency time.

VOID_TIMER_OVF—Void Timer Overflow Bit

This bit is set when the void timer count exceeds 64K, causing the timer to wrap around. In this case, the void time register will not be loaded. This bit can indicate configuration problems in the ring.

TKN_CNT_OVF—Token Counter Overflow Bit

This bit is set when the token counter exceeds 64K. The token counter wraps around to zero and continues to count. This bit indicates that the token count value is not accurate because there is no way of determining how many times the counter has wrapped.

7.3.5.7 CAMEL INTERRUPT MASK REGISTER (CAMEL_MASK) ACNTL = 1 1000 0001. The CAMEL interrupt mask register controls the assertion of the $\overline{\text{CAMINT}}$ pin. When the bit in this register and the corresponding bit in the CAMEL interrupt event register are both set, the $\overline{\text{CAMINT}}$ pin is asserted.

7.3.5.8 ELM INTERRUPT MASK REGISTER (ELM_MASK) ACNTL = 1 0000 0010. This register corresponds bit for bit with the ELM interrupt event register. When a bit in this register is set and the corresponding bit in the ELM interrupt event register is also set, an interrupt is generated, and the ELM_INT bit in the CAMEL interrupt location register is set. This register can be read and written by the node processor at any time. It is cleared on power-up reset.

7.3.5.9 MAC INTERRUPT MASK REGISTER A (MAC_MASK_A) ACNTL = 1 0100 0010. This register corresponds bit for bit with the MAC interrupt event register A. When a bit in this register is set and the corresponding bit in the MAC interrupt event register A is also set, an interrupt is generated, and the MAC_INT bit in the CAMEL interrupt location register is set. This register can be read and written by the node processor at any time. It is cleared on power-up reset.

7.3.5.10 MAC INTERRUPT MASK REGISTER B (MAC_MASK_B) ACNTL = 1 0100 0011. This register corresponds bit for bit with the MAC interrupt event register B. When a bit in this register is set and the corresponding bit in the MAC interrupt event register B is also set, an interrupt is generated, and the MAC_INT bit in the CAMEL interrupt location register is set. This register can be read and written by the node processor at any time. It is cleared on power-up reset.

7.3.5.11 MAC INTERRUPT MASK REGISTER C (MAC_MASK_C) ACNTL = 1 0100 0100. This register corresponds bit for bit with the MAC interrupt event register C. When a

bit in this register is set and the corresponding bit in the MAC interrupt event register C is also set, an interrupt is generated, and the MAC_INT bit in the CAMEL interrupt location register is set. This register can be read and written by the node processor at any time. It is cleared on power-up reset.

7.3.6 PCM Timers

The PCM utilizes two ELM timers, TPC timer and TNE timer, to track PCM timing parameters. Both timers have a clock divider circuit to reduce the frequency at which they are incremented.

7.3.6.1 TPC TIMER ACNTL = 1 0001 0010. The TPC timer is a 16-bit timer. In normal operation it is read-only by the node processor. The TPC timer value is read at address 12 (hex). When the PCM is in the MAINT state, a value can be written to the TPC load value register at address 0E. The TPC timer is incremented by the output of an 8-bit clock divider circuit and is therefore incremented every 20.48 ms, ($2^8 \times 80$ ns). The value in the TPC clock divider is contained in bits 7–0 of the clock divider register, which can be read at address 14 (hex).

The TPC timer is used to ensure that state transitions proceed at the desired rate while the PCM is attempting to establish a physical connection with a neighboring PCM. The timer is loaded with a two's complement value and counts up until it reaches zero. In normal operation, the timer is loaded by the PCM from the relevant timing parameter register, which contains the two's complement of the time value in 20.48-ms units. At the same time the TPC timer is loaded, the TPC clock divider is initialized to zero.

In MAINT state, the TPC timer can be explicitly loaded by the node processor and used to time the scrub function (i.e., REQ_SCRUB = 1). When the TPC_EXPIRED interrupt occurs, the REQ_SCRUB function may be deactivated (REQ_SCRUB = 0) to end scrubbing. If the PCM is not in the MAINT state when a write is attempted to this register, the NP_ERR bit in the CAMEL interrupt event register will be set, and the timer will not be loaded.

For test purposes, the timer can also be used in 16-bit mode, in which the TPC clock divider is bypassed and the timer is incremented every 80 ns when in operation. In this mode, the value loaded into the timer is the two's complement of the remaining time in 80-ns units. This feature is controlled by TPC_16BIT in ELM control register A.

7.3.6.2 TNE TIMER ACNTL = 1 0001 0011. The TNE timer is a 16-bit timer. In normal operation it is read-only by the node processor. The value of the TNE timer can be read at address 13 (hex). When the PCM is in the MAINT state and the NOISE_TIMER bit in ELM control register A is not set, a value can be written to the TNE register by writing TNE load value register at address 0F. The TNE timer, which is incremented by the output of a 2-bit clock divider circuit, is incremented every 0.32 ms (2^2 x 80 ns). The value in the TNE clock divider is contained in bits 9 and 8 of the clock divider register, which can be read at address 14 (hex).

The TNE timer is used to time the length of (potential) noise while the PCM is in the ACTIVE state. The TNE timer is started whenever the LSM transitions from Idle Line State to Noise Line State, Active Line State, or Unknown Line State. If the timer expires before the LSM recognizes Idle Line State again, the PCM transitions to the BREAK state.

The timer is loaded with a two's complement value and counts up until it reaches zero. In normal operation, the timer is loaded by the PCM from the noise time register (NS_MAX), which contains the two's complement of the time value in 0.32-ms units, when the LSM leaves Idle Line State. At the same time the TPC timer is loaded, the TNE clock divider is initialized to zero.

When the PCM is in the MAINT state and the NOISE_TIMER bit in ELM control register A is not set, the TNE timer can be loaded directly with a 16-bit value from the node processor (the TNE clock divider is still loaded with zero). If the PCM is not in the MAINT state or NOISE_TIMER is not set when a write is attempted, the NP_ERR bit in the CAMEL interrupt event register will be set, and the timer will not be loaded.

For testing purposes, the timer can also be used in 16-bit mode, in which the TNE clock divider is bypassed and the timer is incremented every 80 ns when in operation. In this mode, the value loaded into the timer is the two's complement of the remaining time in 80-ns units. This feature is controlled by TNE_16BIT in ELM control register A.

7.3.7 PCM Timing Parameter Registers

The PCM uses a number of different timing parameter registers when forming a physical connection. These registers, which are readable at any time, are programmable and must be written by the node processor. TPC-based timing parameter registers hold the two's complement of the time in 20.48-ms ($2^8 \times 80$ ns) units. They can have a maximum value of about 1.34 sec ($2^{16} \times 20.48$ ms).

In addition to the TPC timing parameters, there is one timing parameter used by the TNE timer, noise time register (NS_MAX), which holds the two's complement of the time in 0.32-ms ($2^2 \times 80$ ns) units. It can have a maximum value of about 20.97 ms ($2^{16} \times 0.32$ ms).

All the PCM timing parameter registers are cleared on power-up reset.

The ANSI FDDI SMT document contains a set of default values for these parameters. Table 7-9 summarizes these values.

Table 7-9. Register Values

Parameter	Default Value (ms)	Register Value (Two's Comp/Hex)	Timer
A_MAX	0.2	FFF6	TPC
LS_MAX	0.025	FFFF	TPC
TB_MIN	5	FF0C	TPC
T_OUT	100	ECED	TPC
LC_SHORT	50	F676	TPC
T_SCRUB	3.5	FF55	TPC
NS_MAX	1.3	F022	TNE

7.3.7.1 MAXIMUM PHY ACQUISITION TIME REGISTER (A_MAX) ACNTL = 1 0000 0110. The A_MAX register value represents the maximum time required to achieve signal acquisition. This register is used for timing the length of time to remain in the Connect State to ensure correct timing with the neighboring PCM (C_MIN).

7.3.7.2 MAXIMUM LINE STATE CHANGE TIME REGISTER (LS_MAX) ACNTL = 1 0000 0111. The LS_MAX register value is the maximum time required for line state recognition. This register is used to set the time required to transmit a given line state before advancing to the next PCM state (TL_MIN).

7.3.7.3 MINIMUM BREAK TIME REGISTER (TB_MIN) ACNTL = 1 0000 1000. The TB_MIN register holds the allowable length of time for the PCM to be in the BREAK state before a response is seen on the inbound physical link. This time allows for the possibility of a bypass failure mode in this station or a neighboring station that could cause four PHYs to be connected in a loop and produce an invalid response to the break. In this case, the minimum break time guarantees that the response to the break will propagate around the loop and be seen on the inbound link.

7.3.7.4 SIGNALING TIME-OUT REGISTER (T_OUT) ACNTL = 1 0000 1001. The T_OUT register is the minimum time that the PCM will remain in a state waiting for a response from a neighboring PCM. When a response is expected and no transition is made in a period equal to T_OUT, the PCM goes to the BREAK state.

7.3.7.5 SHORT LINK CONFIDENCE TEST TIME REGISTER (LC_SHORT) ACNTL = 1 0000 1011. The LC_SHORT register specifies the time duration of the Link Confidence Test. It limits the loopback to prevent deadlock.

7.3.7.6 SCRUB TIME REGISTER (T_SCRUB) ACNTL = 1 0000 1100. T_SCRUB is the time that the ring continuity is broken to remove old PDUs from the ring. It's use is described in the PCI process (see **5.2.1.4.6 Physical Connection Insertion**).

7.3.7.7 NOISE TIME REGISTER (NS_MAX) ACNTL = 1 0000 1101. The NS_MAX register holds the maximum length of time that noise is tolerated before a connection is broken down and is restarted.

7.3.8 PCM Bit Signaling Registers

The PCM uses three ELM registers to perform bit signaling. Bit signaling is the mechanism the PCM uses to transfer information to the PCM in the neighboring station.

7.3.8.1 TRANSMIT VECTOR REGISTER (XMIT_VECTOR) ACNTL = 1 0000 0011. All bits of the read/write transmit vector register are cleared with the assertion of $\overline{\text{RESET}}$. The transmit vector register is writable only when the PCM_SIGNALING bit in ELM status register B is cleared; otherwise, the register will not be written, and the NP_ERR bit in the CAMEL interrupt event register will be set. This register is readable at any time.

The transmit vector register contains from 1 to 16 bits of data to be transmitted from the PCM to its neighboring PCM. Bits are transmitted one at a time by the bit signaling mechanism. A one is represented by the transmission of Halt Line State and a zero by the transmission of Master Line State. Bit 0 of this register is the first bit to be transmitted, then bit 1, etc., up to the number of bits specified in the transmit vector length register.

The transmit vector length register should be written before this register is written.

7.3.8.2 TRANSMIT VECTOR LENGTH REGISTER (VECTOR_LENGTH) ACNTL = 1 0000 0100. All bits of the read/write transmit vector length register are cleared with the assertion of $\overline{\text{RESET}}$. The transmit vector length register is writable only when the PCM_SIGNALING bit in ELM status register B is cleared; otherwise, the register will not be written, and the NP_ERR bit in the CAMEL interrupt event register will be set. This register is readable at any time.

Bits 15–4 are unused and will always be read as zeros. Any value written to these bits will be ignored.

Bits 3–0 contain the number of bits to be transmitted. The value in this field (0 to 15) is actually one less than the number of bits to transmit (1 to 16).

7.3.8.3 RECEIVE VECTOR LENGTH REGISTER (RCV_VECTOR) ACNTL = 1 0001 0110. The read-only receive vector length register contains from 1 to 16 bits of data received from the neighboring PCM. Bits are received at the same time bits are being transmitted. As bit n is received, it is placed in the receive vector length register. If Halt Line State is received, bit n is a one; if Master Line State is received, bit n is a zero. Bit 0 of this register is the first bit received, then bit 1, etc., up to the number of bits specified in the transmit vector length register.

Although this register is readable at any time, if PCM_SIGNALING is asserted when this register is read, the data may be incomplete.

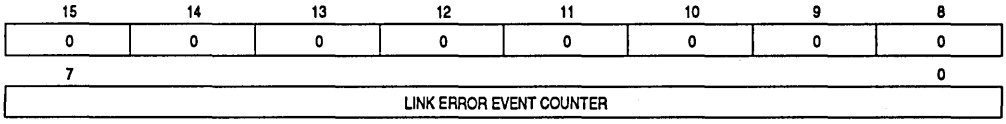
7.3.9 ELM Counter Registers

The ELM contains three event counter registers and one threshold value register used for gathering information about errors occurring on its associated physical link and for monitoring I-symbol gaps between packets.

7.3.9.1 VIOLATION SYMBOL COUNTER (VIOL_SYM_CTR) ACNTL = 1 0001 1000. The violation symbol counter has address 18 (hex). It is read-only and is cleared whenever it is read as well as when $\overline{\text{RESET}}$ is asserted. The VIOL_SYM_CTR high-order 8 bits always read as zeros; the low-order 8 bits contain the counter value. The VSYM_CTR bit in the ELM interrupt event register is set whenever the counter increments or whenever the counter overflows (reaches 256), depending on the setting of the VSYM_CTR_INTRS bit in ELM control register A. When the counter overflows (reaches 256), it wraps to zero and continues to count.

The violation symbol counter is incremented whenever the 4B/5B decoder in the ELM decodes a V-symbol. See Table 5-5 for the symbols considered to be V-symbols by the decoder.

7.3.9.2 LINK ERROR EVENT COUNTER (LINK_ERR_CTR) ACNTL = 1 0001 1010. The link error event counter has address 1A (hex). It is read-only and is cleared whenever it is read as well as when $\overline{\text{RESET}}$ is asserted. An 8-bit counter is contained in bits 7–0. Bits 15–8 of the register always read as zeros. The LE_CTR bit in the ELM interrupt event register is set whenever the counter reaches the value contained in the link error event threshold register (LE_THRESHOLD). The counter will continue to count past this point. When the counter overflows (reaches 256), it wraps to zero and continues to count.



Before the PCM is active, the link error event counter is used by the internal PCM hardware to perform the Link Confidence Test. The number of errors that the user wants the Link Confidence Test to accept should be initialized into the link error event threshold register. If the Link Confidence Test is performed and the link error event threshold is not reached, then the test passed. The test result is given to the software, which then makes the decision as to the next step.

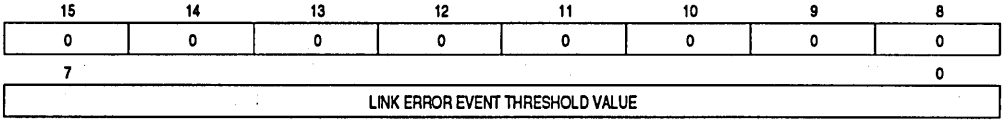
The link error event counter is part of the link error monitor. The link error monitor monitors the bit error rate of an active link and detects and isolates physical links having an inadequate bit error rate, possibly due to a marginal link quality, link degradation, or connector unplugging.

In addition to the counter, the ELM also contains logic to detect link error events. Link error events are defined as:

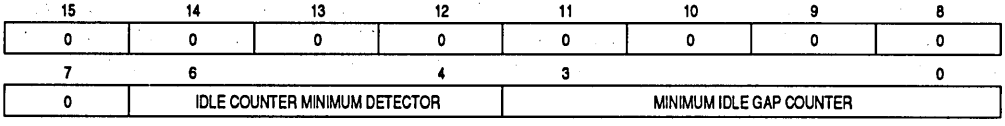
- Transitions from Idle Line State to Unknown Line State or Noise Line State.
- Transitions from Active Line State to Unknown Line State or Noise Line State with the duration of Unknown Line State or Noise Line State exceeding eight symbol times (320 ns).

The link error event counter is only incremented by the link error monitor when a link error occurs and the PCM state machine is in NEXT or ACTIVE state.

7.3.9.3 LINK ERROR EVENT THRESHOLD REGISTER (LE_THRESHOLD) ACNTL = 1 0000 0101. The read/write link error event threshold register is cleared when $\overline{\text{RESET}}$ is asserted. Bits 7–0 of this register contain a value that controls when the LE_CTR bit in the ELM interrupt event register is set. Whenever the value in the link error event counter reaches the value contained in this register, the LE_CTR bit is set. Bits 15–8 always read as zeros.



7.3.9.4 MINIMUM IDLE COUNTER (MIN_IDLE_CTR) ACNTL = 1 0001 1001. The read-only minimum idle counter is cleared whenever it is read as well as when $\overline{\text{RESET}}$ is asserted. Bits 15–7 of the register always read as zeros.



Bits 6–4 of the counter contain the value in the idle counter minimum detector. This is the minimum number of interpacket I-symbol pairs seen since the counter was last reset. It gets reset to 7. Whenever the value changes to a lower value, the MINI_CTR bit in the ELM interrupt event register is set. The counter is a gray code counter. The I-symbol pair count definitions are given in Table 7-10.

Table 7-10. I-Symbol Pair Count

MIN_IDLE_CTR (6–4)	I-Symbol Pair Count
100	7 or more
101	6
111	5
110	4
010	3
011	2
001	1
000	0

Bits 3–0 of the counter contain the value in the minimum idle gap counter. This is the number of times the minimum number of interpacket idles has been seen since the last reset. It gets reset to 1 (coded as 0000). The MINI_CTR bit in the ELM interrupt event register is set whenever the counter increments or whenever the counter overflows

(reaches 16), depending on the setting of the MINI_CTR_INTRS bit in the ELM control register A. When the counter overflows, it remains at 16. The minimum idle occurrence count definitions are given in Table 7-11.

Table 7-11. Minimum Idle Occurrence Count

MIN_IDLE_CTR (3–0)	Minimum Idle Occurrence Count
0000	1
1000	2
1100	3
0100	4
0101	5
0111	6
1111	7
1110	8
1010	9
0010	10
0011	11
0001	12
1001	13
1101	14
0110	15
1011	16

This counter can be used to monitor the activity of the smoother. The number of idles should not go below 7. If they do, it may be desirable to monitor this counter.

NOTE

This block observes the data stream as output from the elasticity buffer and smoother, which may insert or delete idle symbols.

7.3.10 MAC Counter Registers

The three counter registers, FRAME_CT, LOST_CT, and ERROR_CT, are implemented similarly to how they are described in the FDDI MAC standard.

The following definitions apply to these registers.

A "frame" is defined as:

1. JK
2. A Nontoken FC (i.e., two data symbols)
2. Zero or More Data Symbols (including an odd number)

3. A T-symbol

A "fragment" is defined as:

1. JK
2. Zero or More Data Symbols (including an odd number)
3. An Idle Symbol

OR

1. JK
2. A Token FC
3. A T-Symbol
4. An Idle Symbol

A "token" is defined as:

1. JK
2. A Token FC
3. Two T-Symbols
4. Anything OtherThan PHY_INVALID

FRAME_CT counts all frames.

LOST_CT counts format errors, where a format error is defined as a JK followed by zero or more data symbols and is not a frame, fragment, or token.

ERROR_CT counts locatable errors, where a locatable error is defined according to the received E-indicator (Er). Er is the symbol following the T-symbol that ends a frame. ERROR_CT is incremented if Er = an R-symbol and the frame has an invalid data length (odd number of symbols or too short for this FC) or has an invalid FCS. (Note that the FCS for implementor frames is always considered correct.) ERROR_CT is also incremented if Er is missing (i.e., not R or S). Note that there is no locatable error if there is no frame.

Token count is incremented anytime a new valid token is received. The token may be restricted or unrestricted as long as it has been received properly (repeated or captured). Note that once the ring becomes operational, this register should be read and the contents discarded. The counter will now keep a valid count as long as the ring remains operational.

The counter registers are always cleared when read. Also, if these registers are read and cleared in the same cycle that they are incremented, the register will have a one value instead of a zero value at the end of the cycle. Hence, these registers can be read at any time by the node processor without losing count.

These registers are not intended to hold the complete counts of events for network management purposes. They are far too short (e.g., the full frame count register should be at least 48 bits long). Instead, SMT software should keep the full counters. These counters

are used to eliminate the real-time requirements of the software. Instead of requiring the software to guarantee an interrupt latency of less than 5 µs due to possible event frequency, these counters keep track of the number of events that occur during a much larger interrupt latency time. These registers are cleared by power-up reset.

7.3.10.1 FRAME COUNT REGISTER (FRAME_CT) ACNTL = 1 0110 0000. The frame count register is a 16-bit unsigned integer register. The frame count register always wraps from 65535 to 0 even when a double overflow occurs (i.e., when the counter overflows and the FRAME_RCVD interrupt bit is still one from a previous overflow). This register is cleared when read and is not otherwise writable.

7.3.10.2 LOST COUNT AND ERROR COUNT REGISTER (LOST_CT & ERROR_CT) ACNTL = 1 0110 0001. The 6-bit LOST_CT and 6-bit ERROR_CT fields are stored in one register so that they can be read and cleared with one node processor read operation. The ERROR_CT field occupies bits 5–0 of this register, and the LOST_CT field occupies bits 13–8. Bits 0 and 8 are the least significant, and bits 5 and 13 are the most significant bits of each count. Each counter is in its own byte. Bits 6, 7, 14, and 15 are always read as zero and are not part of the counters. This register is always cleared when read and is not otherwise writable.

15	14	13	8
0	0	LOST_CT	

7	6	5	0
0	0	ERROR_CT	

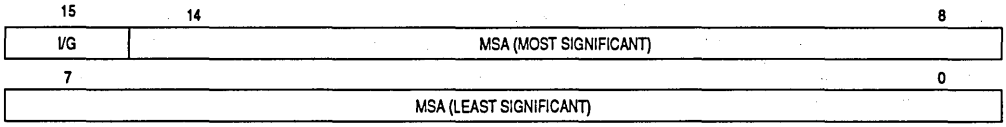
7.3.10.3 TOKEN COUNT REGISTER (TOKEN_CT) ACNTL = 1 0101 1111. The token count register is a 16-bit unsigned count of the number of tokens (restricted or unrestricted) received since the last time this counter was read.

Once the ring becomes operational, this register should be read and the contents discarded. This action will clear and start the counter. The counter will now keep a valid count as long as the ring remains operational.

7.3.11 MAC Station Parameter Registers

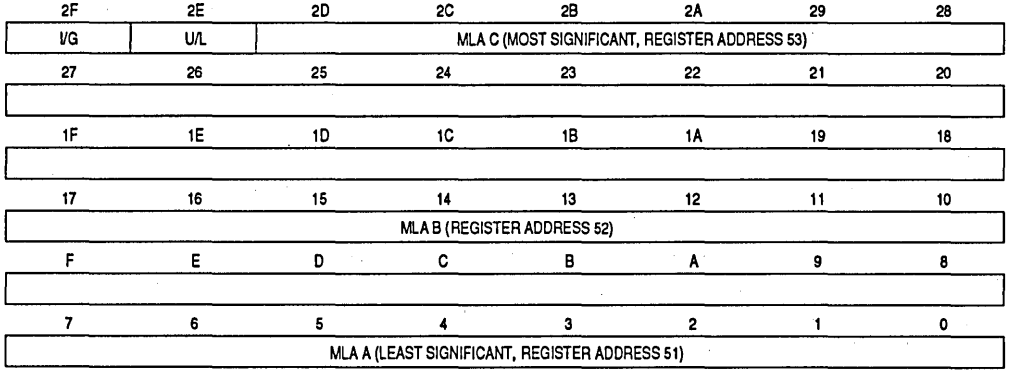
The station parameter registers are normally written when the MAC is first powered up. Sometimes it may be necessary to change these values, in which case the MAC operation must first be disabled via the MAC_ON bit in MAC control register A. Then their values can be changed, and the MAC operation can be re-enabled. These registers can be read during normal MAC operation, but they cannot be changed. These registers are cleared by power-up reset.

7.3.11.1 MY SHORT ADDRESS REGISTER (MSA) ACNTL = 1 0101 0000. My short address register is contained in one 16-bit register. The I/G bit is always bit 15 of this register and is the first bit received. This bit ordering is unaffected by the value of REVERSE_ADDR in MAC control register A as that bit reversal occurs only across the FSI bus.



7.3.11.2 MY LONG ADDRESS REGISTER (MLA_A, MLA_B, MLA_C) ACNTL = 1 0101 0001; ACNTL = 1 0101 0010; ACNTL = 1 0101 0011. My long address register is contained in three 16-bit registers. The least significant 16 bits (MLA_A) have register address 51; the middle 16 bits (MLA_B) have register address 52; the most significant 16 bits (MLA_C), which contain the I/G and U/L bits, have register address 53. Within each register, bit 0 is the least significant bit and bit 15, the most significant bit. This bit ordering is unaffected by the value of REVERSE_ADDR in MAC control register A as that bit reversal occurs only across the FSI bus. The MLA is transmitted and received most significant bit first.

The node processor requires three read/write operations to completely read/write the 48-bit my long address register. The read/write operations do not have to be consecutive since these registers can only be changed by the node processor.



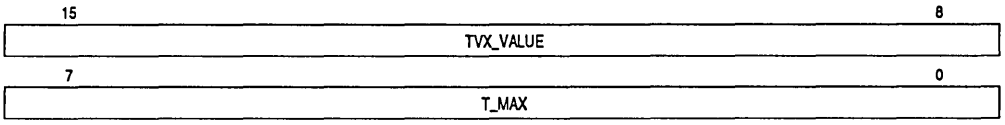
7.3.11.3 REQUESTED TTRT REGISTER (T_REQ) ACNTL = 1 0101 0100. T_REQ is a 16-bit register that holds the two's complement of this station's desired target token rotation time (TTRT) in 20.48-μs units (20.48 μs = 256 × 80 ns) to a maximum of 1342.1568 ms.

T_REQ should normally indicate a time between 1 and 10 ms and must indicate a time smaller than T_MAX (this is not checked by the chip and, if false, the FDDI ring protocol may be violated). A typical value would be 4.014080 ms, obtained by assigning –196 (FF3C in hex) to T_REQ. This would cause the MAC to send claim frames with an INFO field of FF FF 3C 00.

7.3.11.4 TVX TIMER INITIAL VALUE AND MAXIMUM TRT REGISTERS (TVX_VALUE & T_MAX) ACNTL = 1 0101 0101. The 8-bit TVX_VALUE and 8-bit T_MAX register fields are contained in one 16-bit addressable register. T_MAX occupies bits 7–0 of this register, and TVX_VALUE occupies bits 15–8; bits 8 and 0 are the least significant, and bits 15 and 7 are the most significant.

T_MAX holds the two's complement of the TRT time-out to be used when the ring is not operational (i.e., value of T_OPR when RING_OPERATIONAL is false) in 5.242880-ms units (where 5.24288 ms = 2^{16} times 80 ns) to a maximum of 1336.9344 ms. T_OPR is 24-bits wide and represents time in octets (80 ns). When the MAC is required to assign T_MAX to T_OPR, the eight bits of the T_MAX register byte are assigned to bits 16–23 of T_OPR, and bits 0–15 of T_OPR are cleared. A suggested default value for T_MAX is -32 or E0H, representing 167.77216ms. These registers are cleared on power-up reset.

TVX_VALUE is used to load the TVX timer when that timer is reset. The TVX_VALUE register field holds the two's complement of the time remaining in 20.48-µs units (where 20.48 µs = 256 × 80 ns) to a maximum of 5.2224 ms. Since the TVX timer is 16 bits and is incremented every 80 ns, the 8-bit TVX_VALUE equals the most significant 8 bits of the TVX timer (the lower 8 bits are zero) when it is reset. A suggested default value is –128 (80 in hex) representing 2.62144 ms.



7.3.12 MAC Protocol Timing Registers

The following registers are not normally of interest to the node processor (except for the VOID_TIME register), although they can be read at any time (subject to the fact that it can take more than one read operation to obtain their value and that they can change values between these reads). These registers cannot be directly written by the node processor.

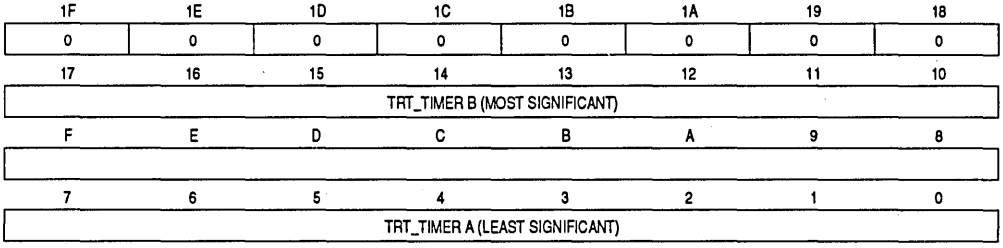
The protocol timing registers include the three FDDI-defined timers (TVX_Timer, TRT_Timer, and THT_Timer), and the negotiated TTRT register, the information field register, the sent count register, and the void time register.

7.3.12.1 TVX TIMER REGISTER (TVX_TIMER) ACNTL = 1 0110 1011. The TVX timer register is a 16-bit counter. It holds the two's complement of the time remaining in 80-ns units. This register is cleared by a power-up reset.

7.3.12.2 TRT TIMER REGISTER (TRT_TIMER_A, TRT_TIMER_B) ACNTL = 1 0110 1100; ACNTL = 1 0110 1101. The TRT timer register is a 24-bit counter that has the addresses 6C and 6D (hex). The node processor may need to read it in two consecutive register reads (only when T_Opr ≥ 5.242880 ms). Because this register can (and usually will) change between the two read operations, care must be taken to get a consistent value.

The least significant 16 bits of the TRT timer occupy register address 6C, and the most significant 8 bits of the TRT timer occupy bits 7–0 of register 6D. Bits 15 and 7 are the most significant, and bit 0 is the least significant in each register. The upper 8 bits of register 6D are always read as zeros even though the timer itself is stored in two's complement value. The TRT counter holds the two's complement of the time remaining in 80-ns units. For example, if register 6C held FF3C (–196 in decimal) and register 6D held FFFE (–1 in decimal with a weighting of 2^{16}), then the time remaining would be (196 × 80ns) + (65536 × 80ns) = 15.68 µs + 5242.88 µs = 5258.56 µs.

These registers are cleared by a power-up reset.

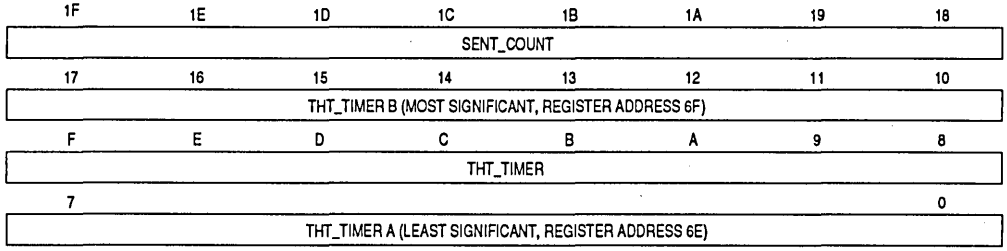


7.3.12.3 THT TIMER AND SENT COUNT REGISTER (THT_TIMER_A, THT_TIMER_B, & SENT_COUNT) ACNTL = 1 0110 1110; ACNTL = 1 0110 1111. Since the THT timer is 24 bits, it may need to be read in two consecutive reads (only when T_Opr is ≥ 5.242880 ms). Because this register can (and usually will) change between the two read operations, care must be taken to get a consistent value.

The 16 least significant bits of the THT timer occupy register address 6E, and the eight most significant bits of the THT timer occupy bits 7–0 of register address 6F. Bits 15 and 7 are the most significant, and bit 0 is the least significant in each register. The THT counter holds the two's complement of the time remaining in 80-ns units.

The upper eight bits of register 6F hold the eight most significant bits of the 10-bit SENT_COUNT register as an unsigned integer. The SENT_COUNT is the count of outstanding frames—i.e., the number of frames transmitted (incremented two BYTCLK cycles after the frame's FS is sent) minus the number of frames received (decremented when a frame's T-symbol has been received). The granularity of this register is four, so the number of outstanding frames is the value of SENT_COUNT times four. SENT_COUNT is used in BRIDGE_STRIP mode to determine whether or not a frame was sent by this station. All data frames as well as special void frames are counted. Claim and beacon frames are not counted. This count is cleared upon receipt of a claim, beacon, or special void frame, or a nonduplicate token. It is also cleared when RING_OPERATIONAL

becomes false, the transmitter enters the FDX states, or the MAC is turned off (MAC_ON = 0). The THT_TIMER_A register (6E) is 0001 after power-up reset. The SENT_COUNT/THT_TIMER_B register (6F) is 0000 after power-up reset.



7.3.12.4 NEGOTIATED TTRT REGISTER (T_NEG_A, T_NEG_B) ACNTL = 1 0110 0110; ACNTL = 1 0110 0111. T_NEG is a 24-bit read-only register that the node processor may need to read in two consecutive reads. Because this register can (but usually will not) change between the two read operations, care must be taken to get a consistent value.

The least significant 16 bits of T_NEG occupy register address 66, and the most significant eight bits occupy bits 7–0 of register 67. The upper eight bits of register 67 are always read as zeros. T_NEG holds the two's complement of the negotiated target token rotation time (TTRT) in 80-ns units. It is up to SMT software to compare this value against T_MIN and T_MAX.

The T_NEG_A register (66) is 7777 (hex) after power-up reset. The T_NEG_B register (67) is 0077 (hex) after power-up reset.

7.3.12.5 INFORMATION FIELD REGISTER (INFO_REG_A, INFO_REG_B) ACNTL = 1 0110 1000; ACNTL = 1 0110 1001. The information field register is a 32-bit register that the node processor may need to read out in two consecutive reads. This register holds the first four octets of the INFO field of the last MAC frame received (normally a claim or beacon frame). Therefore this register holds the last received TTRT bid when the last MAC frame received was a claim frame, and holds the beacon type if the last MAC frame received was a beacon frame. The 16 most significant bits of this register (corresponding to the first four INFO field symbols received) have register address 68, and the 16 least significant bits of this register (corresponding to the fifth through eighth INFO field symbols received) have register address 69. This register is loaded based upon the FC (i.e., FC = MAC frame) before the FCS is checked, so the register can contain the information field of a frame with an incorrect CRC.

These registers are 7777 (hex) after power-up reset.

7.3.12.6 VOID TIME REGISTER (VOID_TIME) ACNTL = 1 0101 1110. This 16-bit register holds the value of the last void time count. This count is the time between the end

of the MAC transmitted void frame (TE symbol pair) and the end of reception of a valid void frame with SA = MLA.

Any void frame created and sent by the MAC core will be timed, and after the completion of the timing, the proper count will be loaded in the void time register. The timer will start to count when the MAC has transmitted the TE symbol pair and will stop counting when My Void Frame is received. This count is in BYTCLK increments. In BRIDGE_STRIP or Purger mode, this timer will time the latency of the first void frame transmitted. If a second void frame is transmitted before receiving the previously transmitted void frame (i.e., the timer is still counting), the second void frame will not be timed and will have no effect on the timer. If the timer overflows, the VOID_TIMER_OVF bit in the MAC INTR_C register will be set, indicating that the ring may have latency problems. This register is cleared on power-up reset.

7.3.13 Miscellaneous Internal Registers

These registers are not normally of interest to the node processor, although they all can be read at any time. They are used for ATE testing and diagnostic software. These registers cannot be directly written by the node processor. These registers are not further defined for user operation.

7.3.13.1 CAMEL REVISION NUMBER REGISTER (CAMEL_REV_NO) ACNTL = 1 1000 0111. This 16-bit read-only register contains the revision number of the CAMEL chip. The current value is 1114 (hex).

7.3.13.2 MAC REVISION NUMBER REGISTER (MAC_REV_NO) ACNTL = 1 0101 1100. This 16-bit read-only register contains the revision number of the MAC core within the CAMEL chip. The current value is 0013 (hex).

7.3.13.3 ELM BUILT-IN SELF-TEST SIGNATURE REGISTER (ELM_BIST) ACNTL = 1 0001 0101. This 16-bit read-only register contains the resultant signature after execution of the ELM core self-test. Refer to **Section 12 Test Operation** for further details.

7.3.13.4 MAC BUILT-IN SELF-TEST SIGNATURE REGISTER (MAC_BIST) ACNTL = 1 0110 1010. This 16-bit read-only register contains the resultant signature after execution of the MAC core self-test. Refer to **Section 12 Test Operation** for further details.

7.3.13.5 PACKET REQUEST REGISTER (PKT_REQUEST) ACNTL = 1 0111 0000. The packet request register is intended for factory testing only to allow test programs to access the internal packet request state. This register contains the last, or current, packet request header control bytes presented to the MAC by the FSI. See **Section 10.4.1.2.5 Packet Request Header** for definitions of these bits.

15	14	13	12	11	10	9	8
0	BCN_FRAME	USE_R_FLAG	PREV_SEND_LAST	TOKEN_TYPE		SYNCH_FRAME	IMMED_MODE

7	6	5	4	3	2	0
SEND_FIRST	SEND_LAST	APPEND_CRC	TOKEN_SEND			EXTRA_FS

BCN_FRAME—Beacon Frame

This bit has the inverse value of the BCN_FRAME bit contained in the last packet request header control bytes.

USE_R_FLAG—Use R-FLAG

This bit contains the value of the internal USE_R_FLAG flip/flop.

PREV_SEND_LAST—Previous SEND_LAST

This bit has the value of the SEND_LAST bit in the previous (one before the last) packet request header.

TOKEN_TYPE—Type of Token Required To Send This Frame

This field has the inverse value of the TOKEN_TYPE field contained in the last packet request header control bytes.

SYNCH_FRAME—Send Synchronous Frame

This bit has the inverse value of the SYNCH_FRAME bit contained in the last packet request header control bytes.

IMMED_MODE—Ignore RING_OPERATIONAL for This Frame

This bit has the inverse value of the IMMED_MODE bit contained in the last packet request header control bytes.

SEND_FIRST—Always Send This Frame First

This bit has the inverse value of the SEND_FIRST bit contained in the last packet request header control bytes.

SEND_LAST—Release Token after This Frame Is Sent

This bit has the inverse value of the SEND_LAST bit contained in the last packet request header control bytes.

APPEND_CRC—Generate and Add an FCS Field to the Frame

This bit has the inverse value of the APPEND_CRC bit contained in the last packet request header control bytes.

TOKEN_SEND—Type of Token To Send after This Frame Is Sent

This field has the inverse value of the TOKEN_SEND field contained in the last packet request header control bytes.

EXTRA_FS—Send Extra Frame Status Indicators

This field has the inverse value of the Extra_FS field contained in the last packet request header control bytes.

7.3.13.6 RECEIVE CRC REGISTER (RX_CRC) ACNTL = 1 0111 001; ACNTL = 1 0111 0010. Receive CRC is a 32-bit register whose least significant 16 bits have address 71 (hex) and whose most significant 16 bits have address 72 (hex). Because this register can change (and normally will) between two read operations, it may be impossible to get a consistent value without stopping the BYTCLK. This register is only expected to be read in test mode because test programs do not need consistent values. Reading this register provides the inverse (bitwise NOT) of the internal CRC register.

7.3.13.7 TRANSMIT CRC REGISTER (TX_CRC) ACNTL = 1 0111 0011; ACNTL = 1 0111 0100. Transmit CRC is a 32-bit register whose least significant 16 bits have address 73 (hex) and whose most significant 16 bits have address 74 (hex). Because this register can change (and normally will) between two read operations, it may be impossible to get a consistent value without stopping the BYTCLK. This register is only expected to be read in test mode because test programs do not need consistent values. Reading this register provides the internal CRC register, not its inverse, although the inverse would be transmitted.

7.4 TWISTED-PAIR OPERATION REGISTERS

There are three registers that affect the use of the streaming cipher functionality of the IFDDI. Table 7-12 contains recommended register configurations when using fiber media and Table 7-13 contains recommended register configurations when using twisted pair media. For more information on the use of these registers, see **5.4 Twisted Pair Functional Operation.**

Table 7-12. Recommended Register Configuration - Fiber Media

Address	Name	Suggested Value
0A	CIPHER_CNTRL	0000
1E	FOTOFF_ASSERT	0000
1F	FOTOFF_DEASSERT	0000

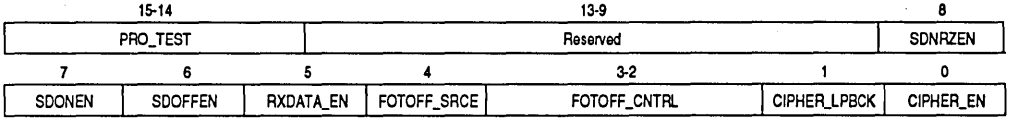
Table 7-13. Recommended Register Configuration - Twisted Pair Media

Address	Name	Suggested Value
0A	CIPHER_CNTRL	00C1
1E	FOTOFF_ASSERT	FD76
1F	FOTOFF_DEASSERT	0000

7.4.1 Cipher Control Register (CIPHER_CNTRL)

The Cipher control register is a Read/Write register. All bits of this register are cleared with the assertion of RESET. Cipher control reigster is used for the following functions:

- Cipher Enable/Disable
- Cipher Loopback
- Signal Detect Filter Control
- FOTOFF (Quiet) Control
- Production Test



PRO_TEST - Production Test

These bits are reserved for production testing of the device and should be set to zero.

Reserved

These bits are reserved and should be set to zero.

SDNRZEN - Signal Detect NRZ Data Filter Enable

When SDNRZEN is set, it causes the received descrambled data to be monitored for activity. If 1000ns elapse without activity, the SD input to the PHY-layer is forced low (inactive).

SDONEN - Signal Detect ON Timer Enable

When SDONEN is set, it causes the assertion of the SD input to the PHY layer to be delayed by 1000ns. This allows the descrambler time to properly synchronize before the PHY layer expects valid data.

SDOFFEN - Signal Detect OFF Timer Enable

When SDOFFEN is set, it causes the received scrambled data to be monitored for activity. If 1000ns elapse without activity, the SD inputs to both the PHY layer and descrambler are forced low (inactive).

RXDATA_EN - Receiver Data Enable

When RXDATA_EN is set, it causes the received scrambled data to be forced to all zeros whenever either the FOTOFF control block's $\overline{\text{FOTOFF}}$ input or output are in their active low state.

FOTOFF_SRCE - FOTOFF Source Select

When FOTOFF_SRCE is set, it causes the FOTOFF control block to use the PCM state machine's variable PCM<> OFF as its input instead of the PHY layer's normal $\overline{\text{FOTOFF}}$ output.

FOTOFF_CNTRL - FOTOFF Mode Selection

The FOTOFF_CNTRL selects the mode of operation of the FOTOFF control block. FOTOFF_CNTRL is defined as follows:

00 = FOTOFF_DELAY. The FOTOFF_ASSERT and FOTOFF_DEASSERT timers are active. In this mode various length of scrambled and true quiet can be transmitted.

01 = Reserved

10 = FOTOFF is forced inactive. In this mode the transmit data is always scrambled and applied to the network.

11 = FOTOFF is forced active. In this mode the transmit data is never scrambled and no energy (true quiet) is applied to the network.

CIPHER_LPBCK - Cipher Loopback Enable

When CIPHER_LPBCK is set, it causes an internal loopback of the scrambled transmit data to be applied to the descrambler input. This allows a loopback similar to that provided by LM_LOC_LOOP, but is closer to the edge of the device. This loopback may be used independent of whether the scrambler and descrambler are active.

CIPHER_EN

When CIPHER_EN is set, it causes the scrambler and descrambler to be active. When CIPHER_EN is cleared, the scrambler and descrambler are inactive and the transmit and receive data pass to/from the pins directly from/to the PHY layer.

7.4.2 FOTOFF Timers

Two additional timers were added to the ELM for use of the IFDDI on twisted-pair networks. These timers control the assertion and de-assertion delays associated with the $\overline{\text{FOTOFF}}$ pin.

7.4.2.1 FOTOFF Assert Timer (FOTOFF_ASSERT). The FOTOFF_ASSERT timer should be loaded with the two's complement of the desired assertion delay in 80ns units. It can have a maximum value of approximately 5.24ms (2^{16} x 80ns). It is initialized to zero upon power-up reset, indicating zero assertion delay.

7.4.2.2 FOTOFF De-Assert Timer (FOTOFF_DEASSERT). The FOTOFF_DEASSERT timer should be loaded with the two's complement of the desired de-assertion delay in 80ns units. It can have a maximum value of approximately 5.24ms (2^{16} x 80ns). It is initialized to zero upon power-up reset, indicating zero de-assertion delay.

SECTION 8
SIGNAL DESCRIPTION

The IFDDI signals are discussed in the following subsections and illustrated in Figure 8-1.

8.1 PORT A INTERFACE

The IFDDI has two identical ports for interfacing to a host system. These two ports allow the IFDDI to be used in various configurations (see **Appendix A System Configurations** and **Section 3 Features Summary**).

Port A Data Bus (ADATA31–ADATA0)

This TTL-level, bidirectional, three-state bus is used to read or write 32 bits of address, control, status, or data from/into the IFDDI. This bus is asynchronous with chip clock and its timing is defined with reference to chip select signals ACS0, ACS1.

Port A Parity (APRTY3–APRTY0)

These TTL-level bidirectional signals indicate the parity of information presented on the ADATAx bus. APRTY0 is the parity of bits ADATA7–ADATA0, and APRTY3 is the parity of bits ADATA31–ADATA24.

Port A Interrupt Request ($\overline{\text{AINT}}$)

This open-drain, active-low, output signal is used to notify an external processor of the occurrence of interrupting events. The interrupt level is asserted when both a status bit in the status register (SR1 or SR2) and its corresponding enable bit in the interrupt mask registers (IMR1 or IMR2) are set. The $\overline{\text{AINT}}$ line changes its logical value synchronously with the internal clock. The user must supply a pullup resistor for this signal to operate correctly.

CAMEL Interrupt ($\overline{\text{CAMINT}}$)

This CMOS-level output signal is used to notify an external processor of the occurrence of a CAMEL interruptible event. The CAMEL operates in a nonvector interrupt mode and must have the appropriate interrupt register read to determine the interrupt(s) that caused the event. This output will remain asserted low until the appropriate interrupt is read to clear the interrupting event(s) or the interrupt is masked by writing a zero into the appropriate interrupt mask register bit.

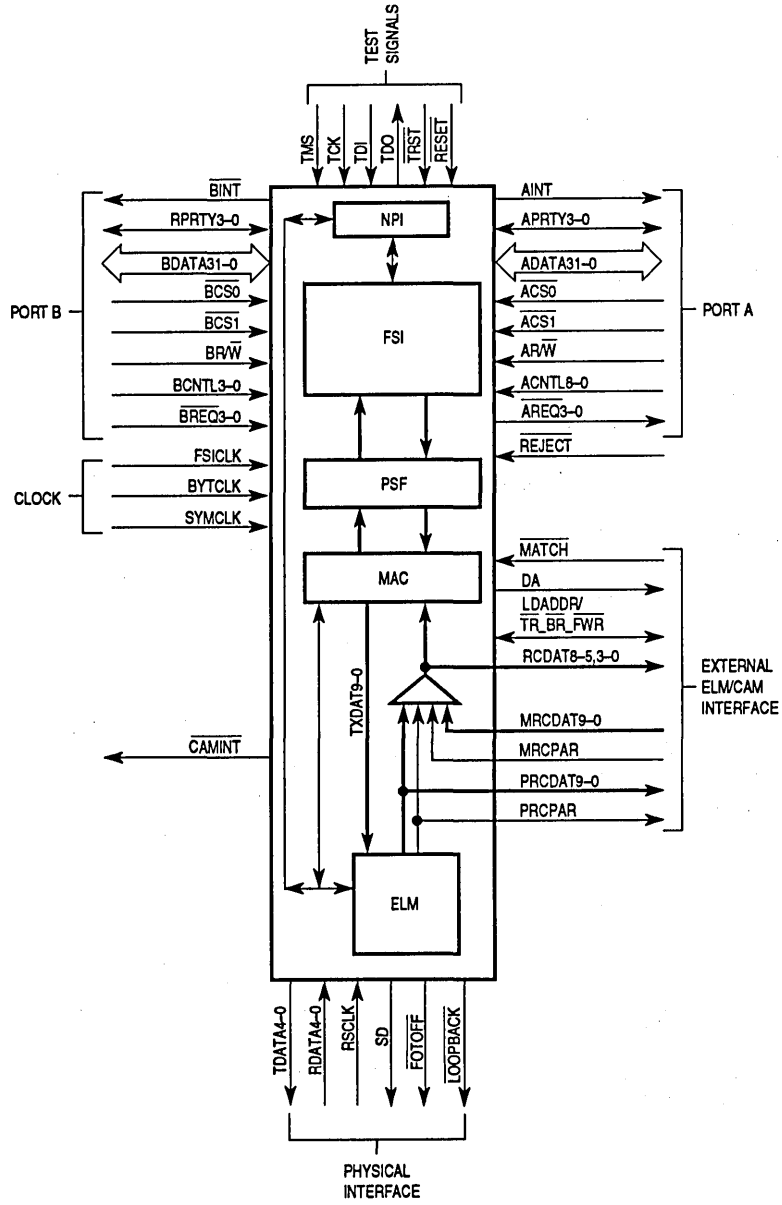


Figure 8-1. IFDDI Signals

Port A Request Lines ($\overline{AREQ3}$–$\overline{AREQ0}$)

These TTL-level, active-low, output signals are used to indicate what type of data transfer is required by the IFDDI. $\overline{AREQ3}$ defines the nature of the data. When $\overline{AREQ3}$ is asserted, receive or transmit data transfers are requested. When $\overline{AREQ3}$ is not asserted, then descriptor or indication information is to be transferred. The encoding of $\overline{AREQ2}$–$\overline{AREQ0}$ in normal mode is listed in Table 8-1.

Table 8-1. $\overline{AREQ}$ Encoding in Normal Mode

$\overline{AREQ2}$	$\overline{AREQ1}$	$\overline{AREQ0}$	Encoding
1	1	1	Idle State
0	1	1	Read Cycle
0	1	0	Read Cycle with Address on the Same Memory Page
1	0	1	Write Cycle
1	0	0	Write Cycle with Address on the Same Memory Page

In burst mode, the $\overline{AREQ3}$–$\overline{AREQ0}$ lines represent the actions listed in Table 8-2.

Table 8-2. $\overline{AREQ}$ Encoding in Burst Mode

Line	Encoding
$\overline{REQ0}$	Data Valid
$\overline{REQ1}$	Write Request
$\overline{REQ2}$	Read Request
$\overline{REQ3}$	Limit Counter State

Port A Control Lines (ACNTL8–ACNTL0)

These TTL-level input signals are used to access a register inside the IFDDI (see **Section 7 Register Description**). These lines should be asserted a setup time before chip select assertion, and should remain stable at least a hold time after chip select assertion in normal mode. When not using CAMEL direct register access, ACNTL8–ACNTL4 are all zeros. When using CAMEL direct register access, ACNTL8 is set to a one, and ACNTL7–ACNTL0 represent the CAMEL register address to be accessed.

Port A Chip Selects ($\overline{ACS1}$, $\overline{ACS0}$)

Two chip selects are provided for the configuration in which more than one master device is operating with the IFDDI. Normally, only one of these signals may be asserted for each access. These two TTL-level input signals are used to select one of the internal IFDDI registers according to the ACNTLx combination. These signals are identical in their operation in 32- and 64-bit mode. $\overline{ACS0}$ and $\overline{ACS1}$ are the only reference signals to define the timing for other input or output signals for this port. When both chip selects are asserted, the port operation error (POE) status bit is set to indicate the illegal

operation. The external bus control logic may use this capability to indicate an external bus error.

In pipeline mode, ACS0 is connected to the external bus clock, and ACS1 serves as a ready signal. For more information, refer to **Section 5 Functional Operation.**

Port A Read/Write (AR/$\overline{W}$)

This TTL-level input signal is used to determine the direction of an access relative to the system bus. When this line is low, data is written to the IFDDI. When this line is high, data is read from the IFDDI. This line should be asserted a setup time before chip select assertion, and should remain stable at least hold time after the chip select assertion.

In pipeline mode, this signal serves as the ADATA31–ADATA0 bus output enable. For more information, see **Section 6 Port Operation.**

8.2 PORT B INTERFACE

All signals except BCNTLx are exactly the same as port A but have the prefix "B."

Port B Control Lines (BCNTL3–BCNTL0)

These TTL-level input signals are used to access a register inside the IFDDI (see **Section 7 Register Description**). These lines should be asserted a setup time before chip select assertion, and should remain stable at least a hold time after chip select assertion in normal mode.

8.3 CLOCK SIGNALS

These signals are used to clock and power up the chip.

FSI Clock (FSICLK)

FSI internal logic clock pin can either be connected to SYMCLK or to another external clock. If SYMCLK is used, the IFDDI should be placed in bypass mode since the port synchronization function (PSF) block between the FSI and the CAMEL is not necessary for proper operation. For more information on bypass mode, see **Section 5 Functional Operation.**

Byte Clock (BYTCLK)

This 12.5-MHz, TTL-level, input signal has a cycle time of 80 ns during normal operation. The rising edge (0 to 5 V) of BYTCLK defines the beginning of a new cycle and is used to sample some input lines and to allow the chip to start presenting new values on all output lines.

Symbol Clock (SYMCLK)

This 25-MHz, TTL-level, input signal is used to clock TDATAx to the FCG and, along with BYTCLK, is used to sample MRCDATx, MRCPAR, and certain internal MAC and ELM inputs.

Recovered Symbol Clock (RSCLK)

RSCLK is a 25-MHz, TTL-level, input signal driven from a clock recovery circuit or chip. RSCLK is used to latch data received on RDATAx and to operate the elasticity buffer.

Reset ($\overline{\text{RESET}}$)

This TTL-level input signal is internally sampled and synchronized by the FSI internal clock. When $\overline{\text{RESET}}$ is asserted, the FSI is initialized and placed in the reset state. When the IFDDI is in non-bypass mode, $\overline{\text{RESET}}$ must be low for at least 10 FSICLKs to be recognized. When the IFDDI is in bypass mode, $\overline{\text{RESET}}$ must be low for at least 10 SYMCLKs.

8.4 PHYSICAL INTERFACE SIGNALS

These signals are used to interface to a clock recovery and generation device, which implements the lower portion of the FDDI physical layer.

Fiber-Optic Transmitter Off ($\overline{\text{FOTOFF}}$)

$\overline{\text{FOTOFF}}$ is a CMOS-level output signal used to control the fiber-optic transmitter. When asserted low, this signal causes the FCG transmitter to turn off (no light output) the fiber-optic transmitter. This signal is asserted if any of the following conditions occur:

- The FOT_OFF bit, the FCG_LOOP_CONTROL bit , or the LM_LOC_LOOP bit is set in ELM control register A, or
- The EB_LOC_LOOP bit is set in ELM control register A, or
- The MAINT_LS field = Transmit_QUIET in ELM control register B, and PC_MAINT = ON, or
- The PCM logic transmits Quiet line state, or
- BIST is active.

In addition to this functionality, the $\overline{\text{FOTOFF}}$ signal is also controlled by the FOTOFF_CNTRL bits in the CIPHER_CNTRL register and the FOTOFF_ASSERT and FOTOFF_DEASSERT timers when in twisted-pair mode. When delayed FOTOFF is selected, the FOTOFF_ASSERT and FOTOFF_DEASSERT timers control the duration and manner in which scrambled quiet and/or true quiet are transmitted on the media.

Signal Detect (SD)

This input signal is an indication from the FCG receiver of the presence of a signal on the physical medium. The inverse of this signal is held in ELM status register A, and the ELM interrupt event register SD bit is set whenever SD becomes asserted. This TTL-level signal is asserted high. Because of the increased functionality of signal detect in twisted pair mode, the SD input is now synchronous to the RSCLK input and not the BYTCLK input. Signal detect input is forced active whenever the remote loopback is enabled. This prevents the signal detect filters from arbitrarily squelching the data. When in twisted-pair mode, the operation of the signal is controlled by the SDONEN, SDOFFEN and SDNRZEN bits in the CIPHER_CNTRL register. When in fiber optic mode, the signal detect filtering should remain disabled.

FCG Control Output (LOOPBACK)

This CMOS-level output signal controls the receive symbol multiplexer in the FCG. If LOOPBACK = 0, the multiplexer selects its input from the FCG transmitter. If LOOPBACK = 1, the multiplexer selects its input from the fiber-optic receiver.

FCG Receive Data Bus (RDATA4–RDATA0)

This TTL-level, parallel, input bus receives unframed data from the FCG receiver section synchronously with the rise of RSCLK.

FCG Transmit Data Bus(TDATA4–TDATA0)

This CMOS-level, parallel, output bus transmits data to the FCG transmitter section on the rising edge of SYMCLK.

8.5 EXTERNAL ELM/MAC INTERFACE SIGNALS

The external ELM/CAM interface is a BYTCLK-synchronous interface that exposes the receive data path between the internal ELM and MAC cores. This interface may be used to connect the MAC to an external ELM device for the purpose of implementing a dual attach station. The interface may also be used simultaneously for external address or frame routing control recognition logic such as CAM chips, or source routing control logic.

MAC Receive Data Bus (MRCDAT9–MRCDAT0)

This 10-bit CMOS bus is used to receive a symbol pair from an external PHY layer device (ELM). The BY_CNTRL bit in the CAMEL control register determines whether the MAC receives its data from the internal ELM section or from the MRCDATx bus. This bus supplies an ANSI standard PM_UNITDATA.indication interface for the MAC section.

For the accepted symbol encodings, see Table 5-5.

The 10 bits are clocked to the MAC on the falling edge of BYTCLK sampled at the falling edge of SYMCLK (Sample_CLK). Sample_CLK is identical to BYTCLK except that it lags BYTCLK by approximately 20ns. Bits 9–5 of the bus contain the first symbol of the FDDI data stream, and bits 4–0 contain the second symbol.

MAC Receive Data Parity (MRCPAR)

This CMOS input signal is the odd parity of the MRCDATx bus.

PHY Receive Data Bus (PRCDAT9–PRCDAT0)

This CMOS-level output bus exposes the data bus between the internal ELM and MAC. The bus is exposed to allow for the implementation of a dual attach station and/or for use by external address recognition logic. For dual attach station implementations, the PRCDATx bus supplies the ANSI standard PH_UNITDATA.indication interface to an A-type PHY for the secondary ring.

PRCDAT9–PRCDAT5 corresponds to the first symbol of the pair received from the fiber. PRCDAT4–PRCDAT0 corresponds to the last symbol of the pair received from the fiber.

PHY Receive Data Parity (PRCPAR)

This CMOS-level output signal is the odd parityof the PRCDATx bus.

8.6 CAM INTERFACE SIGNALS

The CAM interface signals are discussed in the following paragraphs.

Load Address/Transparent Bridge Forward (LDADDR/$\overline{TR}$_$\overline{BR}$_$\overline{FWD}$)

This pin is bidirectional. It has two modes of operation as controlled by the EXT_DA_MATCH bit in the MAC control register B. If EXT_DA_MATCH = 1, this pin is a transparent bridge forward signal ($\overline{TR}$_$\overline{BR}$_$\overline{FWD}$), a TTL-level input. $\overline{TR}$_$\overline{BR}$_$\overline{FWD}$ can be asserted from the secod byte following the DA field until the fourth byte of the FCS field as presented on the PRCDATx or MRCDATx bus. If EXT_DA_MATCH = 0, this pin is LDADDR, a CMOS-level output. As LDADDR, this pin is asserted high for one BYTCLK in the cycle immediately preceding the first byte of the DA or SA field on the PRCDATx or MRCDATx bus. EXT_DA_MATCH = 1 after a power-up reset.

This pin also controls the mode the MC68840 operates in. If LDADDR/$\overline{TR}$_$\overline{BR}$_$\overline{FWD}$ is sampled as V_{CC} on the rising edge of $\overline{RESET}$, the chip operates as an IFDDI device. If this pin is sampled as GND on the rising edge of $\overline{RESET}$, the chip operates as an FSI device. There is an internal pullup on this pin, so unless driven to GND, the chip will operate as an IFDDI device.

Destination Address (DA)

This CMOS-level output signal is used to indicate whether the address being presented to the CAM is the DA or SA field of the received packet. This signal is low when the MAC is receiving fields other than address fields. If this pin is driven low on the rising edge of $\overline{RESET}$, the IFDDI will operate in bypass mode, and the PSF block is not used. If this pin is driven high on the rising edge of $\overline{RESET}$, the IFDDI operates in non-bypass mode, and the PSF block is used. There is an internal pullup on this pin.

Address Match ($\overline{MATCH}$)

This CMOS-level input indicates to the MAC whether the address presented to the CAM matches an individual or multicast group address stored in the CAM or whether the bridge routing logic wants to accept the frame based on its algorithm. $\overline{MATCH}$ is asserted low upon a match. The MAC ignores this signal when receiving frames with 16-bit addresses.

In normal match mode, this signal is examined by the MAC core only at the second byte following the end of a received DA or SA field. In extended match mode, SA CAM match is not available, and the DA match input is valid from the second byte following the end of the DA field until the fourth byte of the FCS field in the frame. This signal need only be valid for one BYTCLK cycle. When this signal is asserted low, the received

frame matches the station's receive criteria, and the MAC takes appropriate SA or DA actions. For the exact timing of the $\overline{\text{MATCH}}$ signal, see **Section 13 Electrical Characteristics.**

Reject Input Line ($\overline{\text{REJECT}}$)

This is a CMOS-level input signal to the IFDDI from external logic or an external CAM indicating that the FSI should reject the incoming frame. This signal would typically be used when the MAC section of the CAMEL is in promiscuous mode or extended match mode.

8.7 TEST SIGNALS

The test interface is used to support JTAG boundary scan testing of the IFDDI.

Test Clock (TCK)

This TTL-level input pin is the JTAG clock. The TDO, TDI, and TMS pins are synchronized by this pin. When TCK is not used, it must be connected to an external pullup

Test Mode Select (TMS)

This TTL-level input pin is sampled by the IFDDI on the rising edge (from 0 to 1) of TCK. This input signal is responsible for the state change in the test access port state machine. TMS is connected to an internal pullup resistor (one of the rules of the IEEE Std 1149.1–1990). When the TMS signal is not used, there is no need to connect it to an external pullup resistor.

Test Data Input (TDI)

This TTL-level input pin is sampled by the IFDDI on the rising edge of the TCK. This input is the data to be shifted toward the TDO output.

When the test logic is enabled to serially shift the data, the information received at TDI will appear at TDO following a number of rising and falling edges of TCK determined by the length of the register selected (instruction, bypass, or boundary scan register). TDI is connected to an internal pullup resistor (one of the rules in the IEEE Std 1149.1–1990). When the TDI signal is not used, there is no need to connect it to an external pullup resistor.

Test Data Output (TDO)

This TTL-level three-state output pin changes its logical value on the falling edge of the TCK. The capability of TDO to switch to a three-state drive allows parallel connection of board-level test data paths in cases where this is required.

Test Reset ($\overline{\text{TRST}}$)

This TTL-level input pin is the JTAG asynchronous reset. When asserted low, the test access port is forced to the Test_Logic_Reset state, and the IFDDI chip will work in its normal mode of operation. When $\overline{\text{TRST}}$ changes from zero to one, the TMS must be held at one to ensure deterministic operation of the test logic. When it is not

implemented, this signal should be tied to $\overline{\text{RESET}}$ or hard-wired to ground. According to IEEE 1149.1–1990, this pin is connected to an internal pullup resistor.

SECTION 9
COMMANDS AND INDICATIONS

The function of the FSI block is to coordinate the transfer of data between its system interface ports and its MAC interface. Transfers of data, transmission, and reception are independent processes but rely upon similar principles and data structures. All data transfers are initialized and controlled by user-supplied commands. The term "descriptor" is reserved for a command that is a data buffer descriptor for either transmit or receive data buffers. The term "command" refers to all other descriptors or directly written commands used to alter FSI activity.

Both ports have command and command extension registers; therefore, up to two commands may be given directly to the FSI simultaneously. Note that receive buffer descriptor rings can contain only receive buffer descriptors. Issuing commands through the use of transmit buffer descriptor rings allows multiple sequential commands to be provided to the FSI.

Indications are generated by the FSI after frame transmission or reception and after command execution. These indications are then written back to the buffer descriptor ring or to the command register. For transmit and receive frames, the frame indication status is placed inside the last buffer descriptor. No other descriptors will include frame status information.

9.1 COMMAND/DESCRIPTOR/INDICATION

Each command, descriptor, or indication fits into a 64-bit long word. A template for a generic descriptor entry is shown in Figure 9-1.

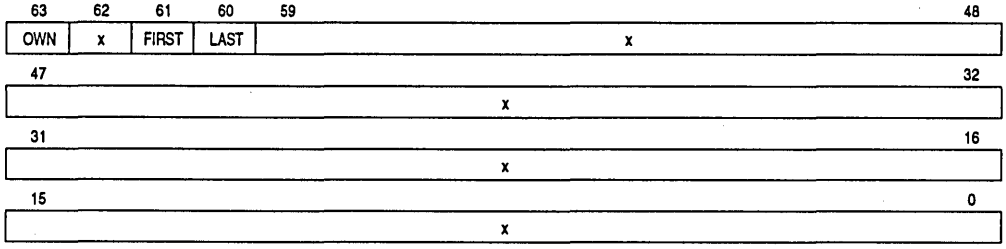


Figure 9-1. Generic Descriptor Entry

All commands have the following bits in common:

OWN—Own Bit

0 = The descriptor is not available for use by the FSI.
1 = The descriptor is available for use by the FSI.

Bit 62—User-Defined Bit

Bit 62 may be written zero or one by the user. This bit has no meaning for the FSI and will not be changed when the indication is written back unless it is a COMMAND ERROR indication or a RECEIVE ERROR indication. Therefore, the bit can be used as a semaphore for user-defined applications.

FIRST—First Bit

0 = Indicates that this is not the first buffer of a frame.
1 = Indicates that this is the first buffer of a frame; always set on nondata buffer descriptors.

LAST—Last Bit

0 = Indicates that this is not the last buffer of a frame.
1 = Indicates that this is the last buffer for this frame; always set on nondata buffer descriptors.

Bits 59–0 are unique for each type of command. All commands that are not data buffer descriptors are single entries and therefore have the FIRST and LAST bits set. Data buffer descriptors that point to buffers intended to hold an entire frame will also have the FIRST and LAST bits set.

Not all commands, indications, or descriptors use the entire 64-bit field for information and/or data. However, the entire field must be provided when the command is issued as a descriptor ring entry. All indications are written out as 64-bit entities. When a 64-bit command is issued directly to the FSI, bits 31–0 are first written to the command extension register (CER), and then bits 64–32 are written to the command register (CMR).

9.2 COMMAND AND INDICATION DESCRIPTION

In cases where commands are given directly to the FSI, the resulting indications are written into the respective port's CMR. Note that a new command cannot be issued until the corresponding indication has been written.

When a command read from a buffer descriptor ring is complete, its indication is written to that ring and the command done (RCC) bit in status register 1 (SR1) is set, which causes an interrupt to the host if this interrupt is enabled. The next command from this ring is executed after this previous command has completed its execution (regardless of whether the indication of the previous command's execution has been written to external memory).

The indications will always have the OWN bit (bit 63) reset to indicate that the FSI has passed control of the descriptor entry to the host processor. Additionally, the FIRST (bit 61) and LAST (bit 60) bits are appropriately set as listed in Table 9-1.

Table 9-1. FIRST and LAST Bit Settings

FIRST Bit	LAST Bit	Condition
1	1	Single Buffer Frame
1	0	First Buffer of a Multiple Buffer Frame
0	1	Last Buffer of a Multiple Buffer Frame
0	0	Intermediate Buffers of a Multiple Buffer Frame

9.2.1 Ring Handling Commands and Indications

The ring handling commands are used to define the ring, to change ring parameters, and to change the ring state. All commands of this type are single-entry commands; therefore, both the FIRST and LAST bits must be set.

The ring pointer values are not necessarily the start of the external memory block that contains the ring. The ring pointer values indicate the next descriptor address to be accessed. Since the address wraps at the modulo value given in the ring parameter register (RPR), the descriptor ring memory block is defined as the pointer address modulo, the RML value in the RPR (see **Section 7 Register Description**).

9.2.1.1 DEFINE RING COMMAND. This command may be issued at any time by the host processor through one of the port command registers or may be placed inside one of the transmit buffer descriptor rings. The DEFINE RING command is used to define a new ring or to change the definition of an existing ring. The format of the DEFINE RING command is shown in Figure 9-2.

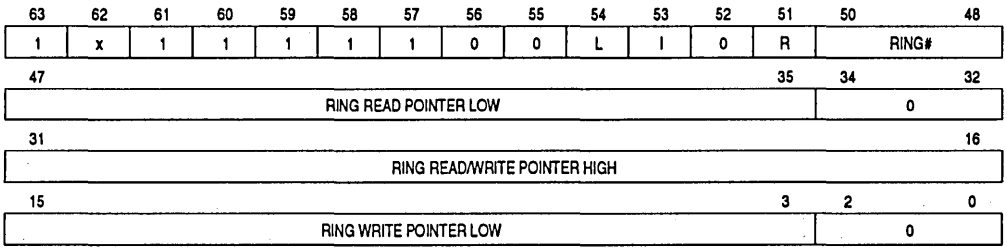


Figure 9-2. DEFINE RING Command

L—Local Memory Mode

This bit is set to indicate local memory use—i.e., it is expected that the user has implemented local memory on one of the ports to buffer data. Table 9-2 lists the valid combinations of the L and I bits.

I—Indication Generation

This bit is set when the user wants indications to be generated after the frame has been transferred to local memory for transmit rings 0, 1, 2, or 3. These indications do not indicate the transmission of the data, only the transfer of data prior to transmission.

Table 9-2. L and I Bit Settings

L-Bit	I-Bit	Condition
0	0	Define Normal Ring (without using local memory)
0	1	Invalid Combination
1	0	Define Ring That Uses the Local Memory. The indications will be generated after the transmission.
1	1	Define Ring That Uses the Local Memory. The indications will be generated after the transfer of the frame to the local memory. This option may be used only for transmit rings (ring # = 0, 1, 2, and 3)

R—Ready State Transfer

If this bit is set, then the ring will transition to the ready state and attempt to read the first descriptor. If this bit is reset, this ring will transition to the complete state—i.e., the ring is empty.

RING#—Ring Number

This field contains the number (0–5) of the ring to be defined. The ring number cannot be the ring number of the ring containing the command.

RING READ POINTER LOW

This field contains the ring read pointer low-order start address in external memory.

RING READ/WRITE POINTER HIGH

This field contains the high-order bits for the start address of both read and write pointers.

RING WRITE POINTER LOW

This field contains the ring write pointer low-order start address in external memory.

The other parameters of a ring, such as ring maximum length (RML), ring port assignment (RPA), and ring data assignment (RDA), are defined by the ring parameter register. To prevent uncertain operation, these parameters may be updated only when the ring is not defined.

The DEFINE RING command may be issued independently of the ring's current state. However, the execution of this command is suspended until the ring under definition is in the STOPPED, COMPLETE, or NOT_DEFINED states.

The region of the address space used for the ring does not necessarily begin with the ring pointer. Rather, the region is defined such that the most significant bits of the pointer remain constant. For example, ring pointer = 3445 4FF8 and an RML = 128 bytes (0100) provides an address range of 3445 4F80 to 3445 4FFC.

In normal ring operation, the read and write pointers will have the same address at the start of the ring. However, the capability to define different read and write pointers can be effectively used to enhance FDDI system operation. If, for example, the host processor wants to transmit a series of identical beacon frames without the processor having to service each beacon frame buffer descriptor, the user can define different read and write pointer low addresses and set the FIFO length so that they will not overlap. In this manner, the updates for read pointers accessed and frames transmitted will not update the buffer descriptors, but are written to a different memory space, and the OWN bit remains set in the transmit descriptors.

An example would be to set the ring read/write pointer high address to FFFF, the ring read pointer low to FF00, the ring write pointer low to FF80, and the RML to 32 bytes (four buffer descriptors). A series of four buffer descriptors containing the frames to be continuously transmitted are placed starting at FFFF FF00, and the frames are continuously transmitted until the host processor stops the ring.

If the user had defined the transmit ring with the option to provide an indication after the transfer of data from system to local memory, then the transmit indication has meaning only for that transfer. In this case, the success or failure of data transmission from local memory to the network will not be reported.

9.2.1.2. SET DESTINATION RING COMMAND. This command may be issued at any time by the host processor through one of the port command registers or may be placed inside one of the transmit buffer descriptor rings. The DEFINE RING command (define normal ring, L & I = 00) sets up the source ring, and the set destination ring sets up the destination ring for the DMA transfers. The format for the SET DESTINATION RING command is shown in Figure 9-3.

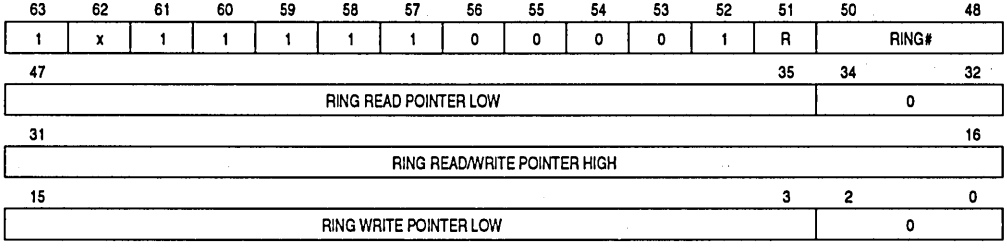


Figure 9-3. SET DESTINATION RING Command

R—Ready

If this bit is set, the destination ring will transition to the ready state. If this bit is reset, the destination ring will remain in the not ready state.

RING#—Ring Number

This field contains the number (0–3, 6–7) of the ring to be defined. The ring number cannot be the ring number of the ring containing the command.

RING READ POINTER LOW

This field contains the ring read pointer low-order start address in external memory.

RING READ/WRITE POINTER HIGH

This field is the high-order address bits for the start address of both read and write pointers.

RING WRITE POINTER LOW

This field contains the ring write pointer low-order start address in external memory.

The other parameters of the ring, such as ring maximum length (RML), ring port assignment (RPA), and ring data assignment (RDA) are defined by the ring parameter extension register (PER). To prevent uncertain operation, these parameters may be updated only when the destination ring is not defined.

The region of the address space used for the ring does not necessarily begin with the ring pointer. Rather, the region is defined so that the most significant bits of the pointer remain constant. For example, ring pointer = 3445 4FF8 and an RML = 128 bytes (16 buffer descriptors) provides an address range of 3445 4F80 to 3445 4FFC.

9.2.1.3 SET LOCAL MEMORY START ADDRESS COMMAND. This command may be issued at any time by the host processor through one of the port command registers or may be placed inside one of the transmit buffer descriptor rings. The SET LOCAL MEMORY START ADDRESS command specifies the start address in local memory, and the DEFINE RING command (define ring that uses the local memory, L & I = 10 or 11) sets up the transmit or receive ring. The format for this command is shown in Figure 9-4.

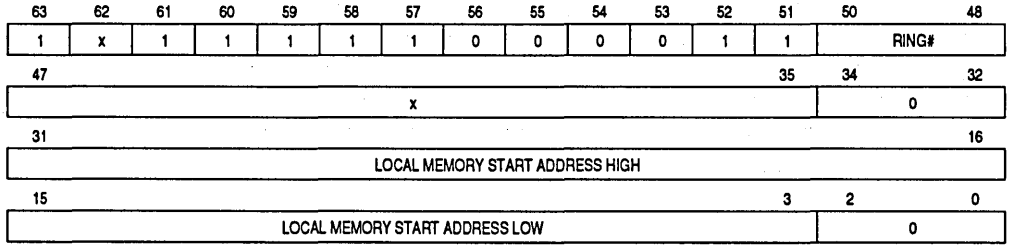


Figure 9-4. SET LOCAL MEMORY START ADDRESS Command

RING#—Ring Number

This field contains the number (0 - 5) of the ring to be defined. The ring number cannot be the ring number of the ring containing the command.

LOCAL MEMORY START ADDRESS HIGH

This field contains the high-order address bits for the start address in external memory

LOCAL MEMORY START ADDRESS LOW

This field contains the low-order address bits for the start address in external memory.

The other parameters of the ring, such as local memory length (LML) and local memory port assignment (LPA), are defined by the ring parameter extension register (PER). To prevent uncertain operation, these parameters may be updated only when the ring is not defined.

The SET LOCAL MEMORY START ADDRESS command should be issued before the DEFINE RING command for the ring number under consideration. In normal practice, the registers pertaining to a given ring should be set up by the initialization routines, and then the SET LOCAL MEMORY START ADDRESS command can be issued to further delineate the memory area. Individual DEFINE RING commands can then be issued as required by the application code.

The region of the address space used for the local memory does not necessarily begin with the local memory start address. Rather, the region is defined so that the most significant bits of the address remain constant. For example, local memory start address = 9454 3578 and an LML = 32K bytes provides an address range of 9454 0000 to 9454 7FFC.

9.2.1.4. USING DEFINE RING AND SET DESTINATION RING COMMANDS. The use of these commands is described in the following list:

1. Define normal ring is done using the DEFINE RING command with L & I = 00.
2. Define port to port DMA channel is done by two commands:
   (a) SET DESTINATION RING command
   (b) DEFINE RING command with L & I = 00
3. Define port to port DMA and normal transmission channel is done by two commands:
   (a) SET DESTINATION RING command
   (b) DEFINE RING command with L & I = 00
4. Define channel using the local memory is done by two commands:
   (a) SET LOCAL MEMORY START ADDRESS command
   (b) DEFINE RING command with L & I = 10 or 11

NOTE

The two commands should be issued in the order given.

9.2.1.5 RING RESET COMMAND. This command may be issued by the host processor through one of the port command registers or placed inside one of the transmit buffer descriptor rings. Execution of this command causes the ring to become NOT_DEFINED. All memory space occupied by this ring in the FSI internal memory is released. The format of the RING RESET command is shown in Figure 9-5.

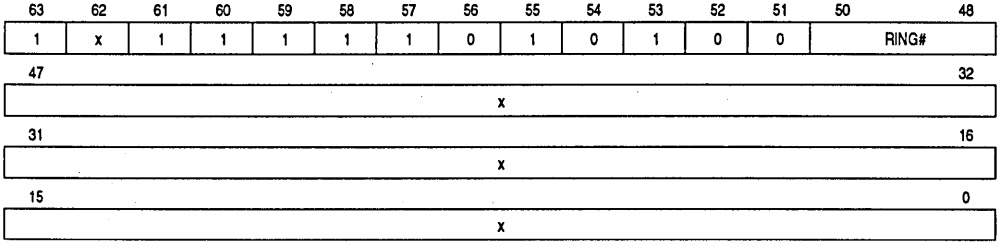


Figure 9-5. RING RESET Command

RING#—Ring Number

This field contains the number (0–5) of the ring to become NOT_DEFINED.

9

9.2.1.6. STOP RING COMMAND. The STOP RING command may be issued at any time by the host processor through one of the port's command registers or may be placed inside one of the transmit buffer descriptors rings. The format of the STOP TRANSMIT RING command is shown in Figure 9-6.

The execution of this command is as follows:

a. When a frame is currently being transmitted from this ring, its transmission is completed. If the current operation is a DMA data move, the transfer is continued until the end of the frame.

b. No more frames are transmitted from this ring, even if they are already inside the FSI internal memory.

c. All indications prepared inside the FSI internal memory are written back to the ring.

d. The STOP RING Command is confirmed.

After this command has been completed, the host processor has an accurate status of all frames that have been transmitted from this ring. However, the FSI internal memory related to a stopped ring may include one or more frames inside the FSI internal transmit data FIFO corresponding to this ring. If the host processor enables this ring again, the FSI will continue operation from its current state.

In situations where the host processor decides to alter the ring that has been stopped, the DEFINE RING command for this ring should be issued to provide new values for the ring read/write pointers. All internal data previously associated with this ring will be cleared and lost.

Note that self stop (i.e., the target ring for the STOP RING command is its source ring) is also allowed. Although the ring is in the STOP state, an indication is not generated immediately (status register 1 (SR1) will have the appropriate RNR bit set), but is written when the ring is reenabled. The indication is lost if the ring is redefined.

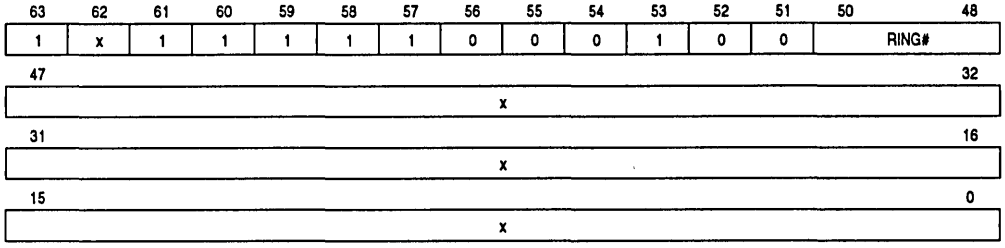


Figure 9-6. STOP RING Command

RING#—Ring Number

This field contains the number (0–3) of the transmit ring to be stopped.

9.2.1.7 READ RING PARAMETERS COMMAND. This command may be issued at any time by the host processor through one of the port command registers (CMRs) or may be placed inside one of the transmit buffer descriptor rings. Execution of this command results in the return of a READ RING PARAMETERS indication. The format of the READ RING PARAMETERS command is shown in Figure 9-7.

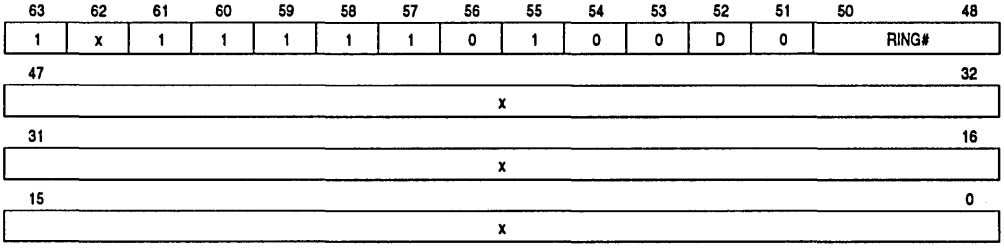


Figure 9-7. RING READ PARAMETERS Command

D—Destination Ring

0 = This command is for the source ring.
1 = This command is for the destination ring.

RING#—Ring Number

This field contains the number (0–7) of the ring whose parameters are to be read.

9.2.1.8 READ RING PARAMETERS INDICATION. This indication is generated in response to a READ RING PARAMETERS command. It includes the current ring read and

write pointers. All other parameters of the ring may be read directly from the FSI internal registers. The format of the READ RING PARAMETERS indication is shown in Figure 9-8.

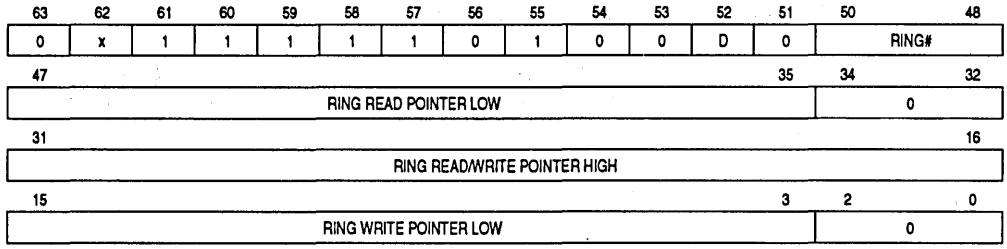


Figure 9-8. READ RING PARAMETERS Indication

D—Destination Ring

0 = This command is for the source ring.
1 = This command is for the destination ring.

RING#—Ring Number

This field contains the number (0–7) of the ring whose parameters are to be read.

RING READ POINTER LOW

This field contains the low-order next descriptor read address in external memory for the ring read pointer.

RING READ/WRITE POINTER HIGH

This field contains the high-order address bits for both the read and write pointers.

RING WRITE POINTER LOW

This field contains the low-order next descriptor write address in external memory for the ring write pointer.

9.2.2 Data Handling Commands and Indications

Data handling commands are used to transmit, receive, and DMA transfer data frames. These commands are buffer descriptors—i.e., a pointer to an external memory buffer. A frame occupies one or more descriptors. Transmit and DMA commands are normally placed inside the transmit buffer descriptor rings, and the receive commands are placed inside the receive buffer descriptor rings. Note that transmit and DMA commands may also be issued through either port command register (CMR).

9.2.2.1 TRANSMIT BUFFER DESCRIPTOR COMMAND. This command is used to describe a buffer of information to be transmitted. A transmit frame is formed from up to 16 data buffers that are pointed to by one of the transmit buffer descriptor ring entries. The format of the TRANSMIT BUFFER DESCRIPTOR command is shown in Figure 9-9.

The first buffer descriptor of the frame has the FIRSt bit set, and the final buffer descriptor for the frame has the LAST bit set inside their respective TRANSMIT BUFFER DESCRIPTOR commands. In all intermediate transmit buffer descriptors, both the FIRST and LAST bits should be zero. In cases where a frame consists of only a single data buffer, its buffer descriptor should have both the FIRST and the LAST bits set.

The data from the buffers is transferred by the FSI into one of its internal data FIFOs according to the source ring number. When this command is issued directly to the FSI using one of the port command registers (PCRs), the frame should consist of only a single data buffer with both the FIRST and the LAST bits set. The watermark must be set for ring 6 when using port A or for ring 7 when using port B. The data is then placed inside the FSI internal command data FIFO without affecting other internal data FIFOs.

All parameters specific to each frame (i.e., usable token type, method of transmission, etc.) must be placed inside the first three bytes of the first data buffer. These parameters are called the packet header. They are transparent to the FSI and are forwarded to the MAC.

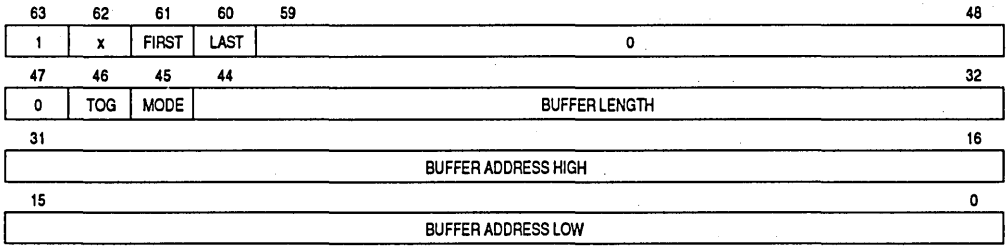


Figure 9-9. TRANSMIT BUFFER DESCRIPTOR Command

TOG—Toggle

This bit indicates a toggle to the other port to be used as a source of data.

0 = The data buffer will be taken from the same port as the previous data buffer.
1 = The data buffer will be taken from the other port. (This port will be used as a source of the data until another Toggle is encountered or until the end of the frame.)

Using this option, a transmit frame may be constructed from buffers that are placed in different memory spaces—e.g., frame header may be prepared by the local processor while the data portion of the frame is taken from the system memory.

MODE—Mode Selection

This bit is appropriate for only the first descriptor of a frame.

0 = Normal frame transmission
1 = Single frame transmission

In normal frame transmission, the FSI will read the data of the next frame in the ring as long as there is sufficient internal memory available for storage. In single frame

transmission, only one frame at a time will be transferred to FSI internal memory and subsequently transmitted out the MAC interface. When the frame is transmitted, a subsequent single frame may be transferred into the FSI. This is valuable for very low-performance bus applications. All the normal internal algorithms, descriptor rings, watermarks, etc., are in operation for the single frame mode.

BUFFER LENGTH

This field contains the number of bytes in this buffer. The buffer length must not be zero.

BUFFER ADDRESS

This field contains the byte address of the data buffer. Note that this address may have any byte alignment.

9.2.2.2 TRANSMIT INDICATION. The TRANSMIT indication is generated after frame transmission and is furnished in the last descriptor of the frame. The format for this indication is shown in Figure 9-10.

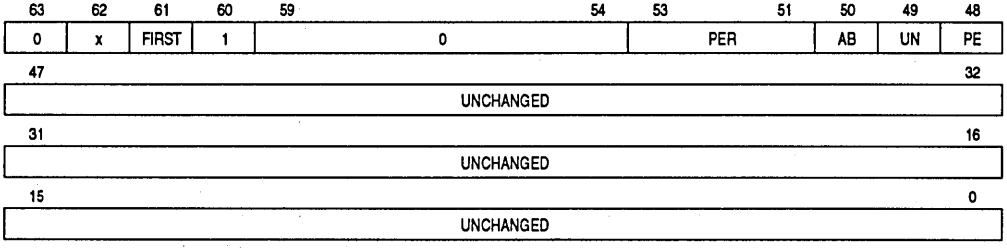


Figure 9-10. TRANSMIT Indication

FIRST—First Bit

0 = Multiple buffer frame
1 = Single buffer Frame

PER—Port Error

This field specifies an error during data transfer through the FSI ports.

001 = Transfer to data FIFO was aborted by either a port operation error or an abort access.
010 = Parity error occurred during data transfer to the data FIFO.
100 = Transfer from the intermediate data FIFO to local memory was aborted by either a port operation error or an abort access.
101 = Transfer to the intermediate data FIFO was aborted by either a port operation error or an abort access.
110 = Parity error occurred during data transfer to the intermediate data FIFO.
111 = Parity error occurred during data transfer from the intermediate data FIFO to local memory.

AB—Abort

0 = Frame transmission completed normally.
1 = Frame transmission has been aborted.

UN—Underrun

0 = An underrun error has not been detected, and transmission continues normally.
1 = An underrun error caused a frame to be aborted. Subsequent transmit frames will, however, continue normally. Only the frame delimited by the descriptors when the underrun occurred is discarded.

PE—Parity Error

0 = A parity error has not been detected, and transmission continues normally.
1 = Transmission of this frame was aborted due to a parity error.

Note that if the AB bit is set and UN, PE, and PER are zero, then the decision to abort frame transmission was caused by the assertion of the TABORT input signal.

9.2.2.3 DMA BUFFER DESCRIPTOR COMMAND. This command is used to transfer a buffer of information from a source ring to a destination ring. A DMA data frame is formed from up to 16 data buffers, which are described by the DMA buffer descriptor ring entries. The buffer length and buffer address fields are the same as those used in the TRANSMIT BUFFER DESCRIPTOR command. Note that a buffer length field equal to zero defines an 8-Kbyte buffer. The format of this command is shown in Figure 9-11.

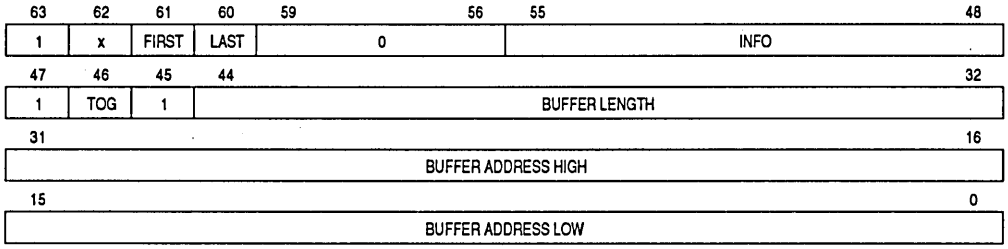


Figure 9-11. DMA BUFFER DESCRIPTOR Command

INFO

This field is transferred into the DMA indication on the destination side. It may be used to attach an ID to transferred data. If the DMA frame is formed from several buffers, the INFO in the last descriptor is used.

9.2.2.4 DMA INDICATION (SOURCE SIDE). The DMA indication is generated after frame transmission and furnished in the last descriptor of the frame. The format for this indication is shown in Figure 9-12.

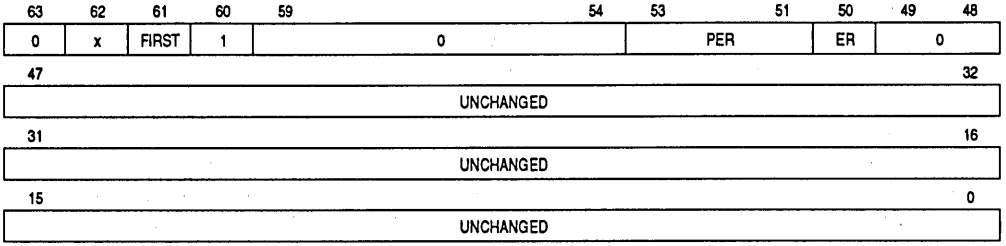


Figure 9-12. DMA Indication (Source Side)

FIRST—First

0 = Multiple buffer frame
1 = Single buffer frame

ER—Error

0 = No error has been detected.
1 = The DMA transfer is aborted due to error.

9

PER—Port Error

The port error field specifies an error during data transfer through the FSI ports.

001 = Transfer to data FIFO was aborted either by a port operation error or an abort access.
010 = Parity error occurred during data transfer to the data FIFO.
100 = Transfer from the data FIFO to a destination buffer was aborted by a port operation error, an abort access, or a discard frame (descriptor with D= 1).
111 = Parity error occurred during data transfer from the data FIFO to destination buffer

9.2.2.5 DESTINATION BUFFER DESCRIPTOR. This descriptor is used to describe a destination data buffer that is available to hold the DMA data (see Figure 9-13). The destination buffer descriptor is placed only in the destination ring. This descriptor is identical to the receive buffer descriptor.

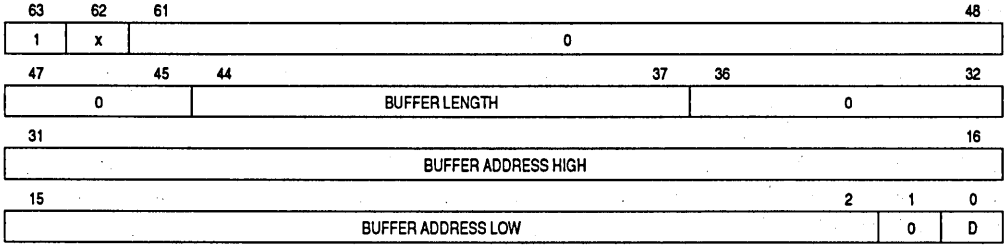


Figure 9-13. Destination Buffer Descriptor

BUFFER LENGTH

The buffer length should be specified for each destination buffer descriptor. There are eight possible lengths:

Buffer Length Field	Buffer Length
0000.0001	64 bytes
0000.0010	128 bytes
0000.0100	256 bytes
0000.1000	512 bytes
0001.0000	1K bytes
0010.0000	2K bytes
0100.0000	4K bytes
1000.0000	8K bytes

BUFFER ADDRESS

This field specifies the address of the first byte in the associated data buffer. Note that this address must be aligned on long-word (64-bit) boundaries for 64-bit accesses and aligned to 32-bit boundaries for 32-bit accesses.

D—Discard Frame

If D is set, then the rest of current DMA frame is discarded—that is, the DMA transfer is aborted for this frame.

9.2.2.6 DMA INDICATION WITHOUT ERROR (DESTINATION SIDE). This indication is generated after a frame transfer and furnished in the last destination descriptor of the frame. The format for this indication is shown in Figure 9-14.

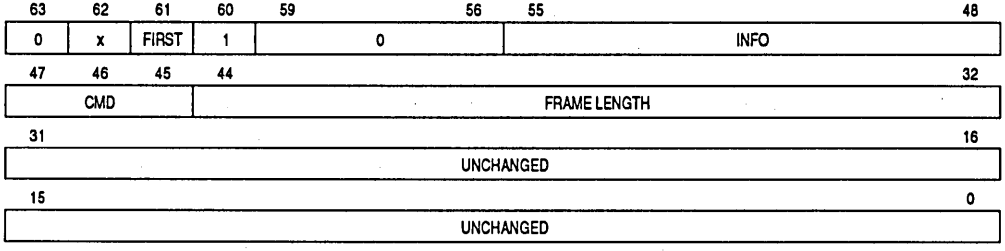


Figure 9-14. DMA Indication Without Error (Destination Side)

INFO

This field is a copy of the SAME field in the last source descriptor.

CMD

This field is a copy of the same field (bits 47, 46, 45) in the last source descriptor.

FRAME LENGTH

This field indicates the length of the entire frame (modulo 8K).

9.2.2.7 DMA ERROR INDICATION (DESTINATION SIDE). This indication is generated after a frame transfer and is furnished in the last destination descriptor of the frame. The format for this indication is shown in Figure 9-15.

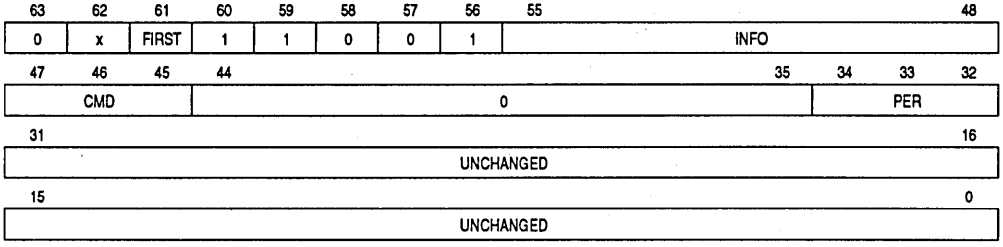


Figure 9-15. DMA ERROR Indication (Destination Side)

INFO

This field is a copy of the same field in last source descriptor.

CMD

This field is a copy of the same field (bits 47, 46, 45) in last source descriptor.

PER—Port Error

This field specifies that an error occurred during a data transfer through the FSI ports:

- 001 = Transfer from a source memory to an internal memory was aborted by either a port operation error or an abort access.
- 010 = Parity error occurred during a data transfer from a source memory to an internal memory.
- 100 = Transfer from an internal memory to a destination buffer was aborted by a port operation error, an abort access, or discard frame (descriptor with D = 1).
- 111 = Parity error occurred during a data transfer from an internal memory to a destination buffer.

9.2.2.8 MAKE INDICATION COMMAND. The MAKE INDICATION command is used to transfer a 24-bit data word, the indication field, to a destination ring. The indication field of this command will be written to the same field inside the descriptor in a destination ring. The format for this command is shown in Figure 9-16.

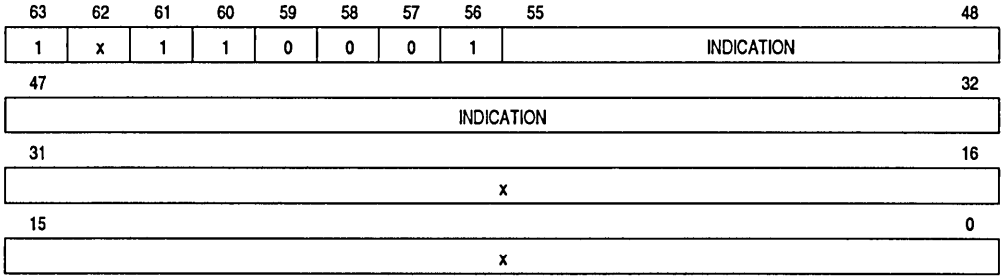


Figure 9-16. MAKE INDICATION Command

INDICATION

The indication field of this command will be written to the same field inside the destination descriptor.

9.2.2.9 INDICATION (DESTINATION SIDE). This indication (see Figure 9-17) is generated on a destination side as a result of a MAKE INDICATION command from source. It is written to a destination buffer descriptor. The user should note that if MAKE INDICATION commands are mixed with DMA commands, the indications are indistinguishable in the case of error.

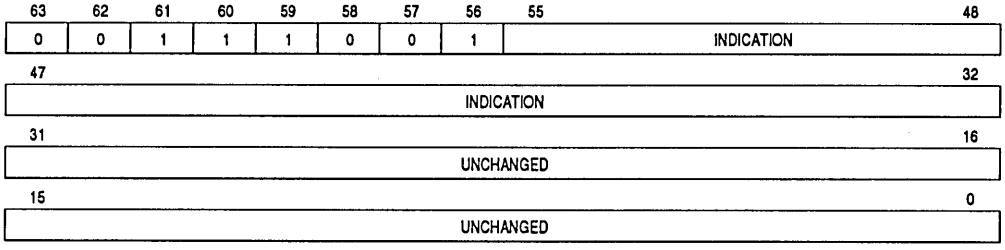


Figure 9-17. Indication (Destination Side)

INDICATION

The indication field is a copy of the same field in the MAKE INDICATION command.

9.2.2.10 RECEIVE BUFFER DESCRIPTOR. This descriptor is used to describe a receive data buffer available to hold the received data. This descriptor is identical to the destination buffer descriptor. The format of the receive buffer descriptor is shown in Figure 9-18.

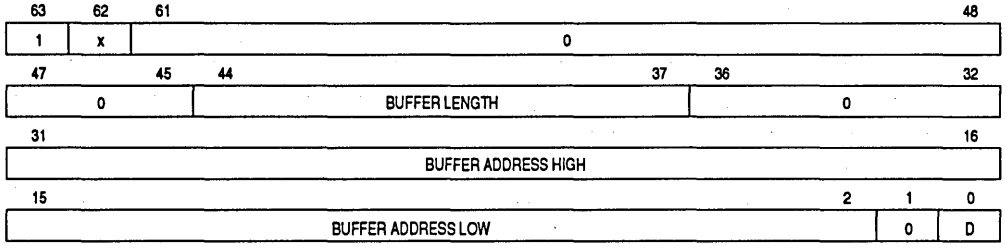


Figure 9-18. Receive Buffer Descriptor

BUFFER LENGTH

The buffer length may be given for each destination descriptor.

Buffer Length Field	Buffer Length
0000.0001	64 bytes
0000.0010	128 bytes
0000.0100	256 bytes
0000.1000	512 bytes
0001.0000	1K bytes
0010.0000	2K bytes
0100.0000	4K bytes
1000.0000	8K bytes

If the buffer length field is zero, then the receive buffer length specified by receive buffer length register is used by the FSI.

BUFFER ADDRESS

This field is the address of the first byte in the associated data buffer. Note that this address must be aligned on long-word (64-bit) boundaries for 64-bit accesses and aligned on 32-bit boundaries for 32-bit accesses.

D—Discard Frame

If D is set, the rest of the current receive frame is discarded.

9.2.2.11 RECEIVE FRAME NORMAL INDICATION. This indication is generated when a frame has been received normally. Note that this frame could still have a CRC error. The FNUM, EF, AF, CF, F0, and F1 fields are taken from frame status information received from the MAC. The CE and DA fields are taken from the END DATA indication that is also provided by the MAC (see Table 10-1). These fields, which are known as the C and DD fields, are further described in the *MC68838 User's Manual* (MC68838UM/AD). The format of the RECEIVE FRAME NORMAL indication is shown in Figure 9-19.

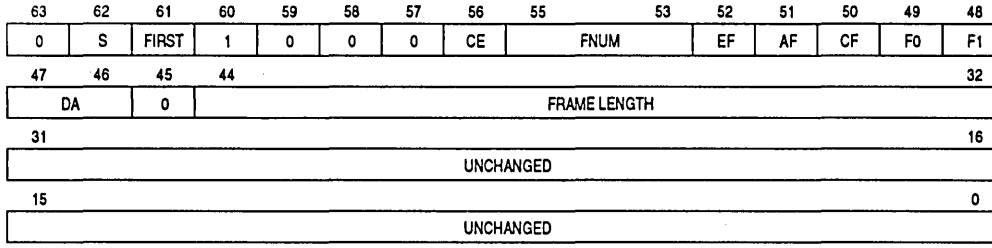


Figure 9-19. RECEIVE FRAME NORMAL Indication

S—Split

When a frame is split into a header portion and a data portion, each of these parts will be treated by the FSI as a whole frame. To distinguish between these frames and normal frames, the S-bit in the indication (for both header and data) will be set. For frames that are not split, this bit will be reset.

FIRST—First Bit

0 = Multiple buffer frame
1 = Single buffer frame

CE—CRC Error

0 = No CRC error detected
1 = CRC error detected

FNUM—Number of Valid Flags

The FNUM field contains the number of valid flags.

EF—E Flag

AF—A Flag

CF—C Flag

F0—First Additional Flag

F1—Second Additional Flag

DA—Destination Address Match Field

00 = Reserved
10 = Promiscuous reception
01 = External CAM match
11 = Local match

FRAME LENGTH

This field holds the length of a frame. When a frame is received in header split mode, the header indication will have a length field equal to the header length, and the data indication will have a length field equal to a length of the data only.

NOTE

When using header split mode, the status fields CE, FNUM, EF, AF, CF, F0, F1, and DA are not valid in the header indication.

9.2.2.12 RECEIVE ERROR INDICATION. The FSI generates a RECEIVE ERROR indication when an error, fragment, or secondary NSA frame (next station address with the A bit set) is received when the FSI is in the RAL mode (MACIF receive control register), when the receive operation has been aborted by the FSI, or when the fragment is longer than the receive watermark. Otherwise, under normal operation, frames with these errors are neither seen by the FSI nor reported in the RECEIVE ERROR indication. The format of the RECEIVE ERROR indication is shown in Figure 9-20.

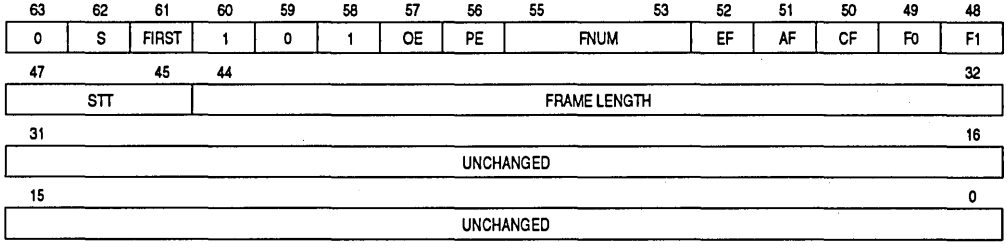


Figure 9-20. RECEIVE ERROR Indication

S—Split

When a frame is split into a header portion and a data portion, each one of these parts will be treated by the FSI as a whole frame. To distinguish between these frames and normal frames, the S-bit in the indication (for both header and data) will be set. For frames that are not split, this bit will be reset.

FIRST—First Bit

0 = Multiple buffer frame
1 = Single buffer frame

OE—Overrun Error

0 = An overrun error has not been detected in the FSI.
1 = An overrun error has been detected in the FSI. As a result of this condition, the receiver for this ring is turned off.

PE—Parity Error

0 = A parity error has not been detected by the MACIF.
1 = A parity error has been detected by the MACIF.

FNUM—Number of Valid Flags

EF—E Flag

AF—A Flag

CF—C Flag

F0—First Additional Flag

F1—Second Additional Flag

STT—Receive Status of MAC

000 = NSA secondary frame
001 = SA matched
010 = DA not matched
011 = FSI abort
100 = Invalid length
101 = Fragment
110 = Format error
111 = MAC reset

FRAME LENGTH

This field holds the length of a frame. When a frame has been split, the header portion of a frame will have a frame length equal to the header length, and the data portion of a frame will have a frame length equal to a length of this portion only

NOTES

When using the split mode, the status fields FNUM, EF, AF, CF, F0, F1, and STT are not valid in the header indication.

In the event of a MAC interface error as reported in the internal error status register (IER), bits 55 to 48 of this indication will be zero, and the STT field is not valid nor meaningful. The FNUM, EF, AF, CF, F0, and F1 fields are taken from frame status information received from MAC. This is further described in **Section 10 IFDDI as an FSI.**

9.2.2.13 RECEIVE PORT ERROR INDICATION. The RECEIVE PORT ERROR indication is provided as a result of an error during data transfer through one of the FSI ports. The format for this indication is shown in Figure 9-21.

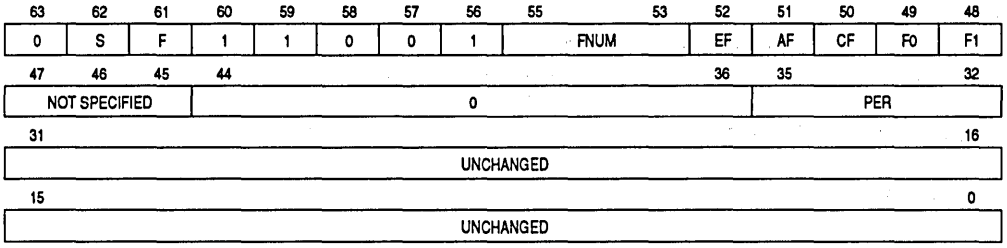


Figure 9-21. RECEIVE PORT ERROR Indication

S—Split

When a frame is split into a header portion and a data portion, each one of these parts will be treated by the FSI as a whole frame. To distinguish between these frames and normal frames, the S bit in the indication (for both header and data) will be set. For frames that are not split, this bit will be reset.

F—First Bit

0 = Multiple buffer frame
1 = Single buffer frame

FNUM—Number of Valid Flags

EF—E Flag

AF—A Flag

CF—C Flag

F0—First Additional Flag

F1—Second Additional Flag

PER—Port Error

This field specifies an error during data transfer through the FSI ports:

1100 = Transfer from a data FIFO was aborted by a port operation error, an abort access, or a discard frame (descriptor with D = 1).
1111 = Parity error occurred during data transfer from the data FIFO.
0100 = Transfer from the intermediate data FIFO (IDF) to system memory was aborted by a port operation error, an abort access, or a discard frame (descriptor with D = 1).
0101 = Transfer to the IDF was aborted by either a port operation error or an abort access.
0110 = Parity error occurred during a data transfer to the IDF.
0111 = Parity error occurred during a data transfer from the IDF to system memory.

9.2.2.14 SPLIT MODE DATA ERROR INDICATION. This indication is generated when split mode is enabled and an error occurs on the data portion of the frame, which causes a discard of the entire data portion. This indication is generated to provide synchronization between the header ring and the data ring so that for every header indication there will be a corresponding data indication, even in the event of an error. If the received data has already been partially or completely transferred when an error occurs, the result will be a RECEIVE ERROR indication as described in the preceding paragraph. The format for the SPLIT MODE DATA ERROR indication is shown in Figure 9-22.

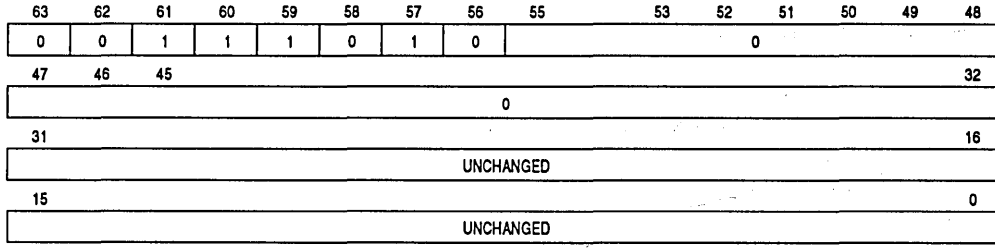


Figure 9-22. SPLIT MODE DATA ERROR Indication

9.2.2.15 RECEIVE MY FRAME INDICATION. This indication is generated if it has been enabled through the RMI bit in the MACIF receive control register (MRR). It indicates that a frame transmitted by this station has been received and stripped. Note that this indication is placed inside one of the receive descriptor rings according to the received frame's frame control field. The FNUM, EF, AF, CF, F0, and F1 fields are taken from frame status information received from the MAC. This field is further described in the **Section 10 IFDDI as an FSI**. The format of the RECEIVE MY FRAME indication is shown in Figure 9-23.

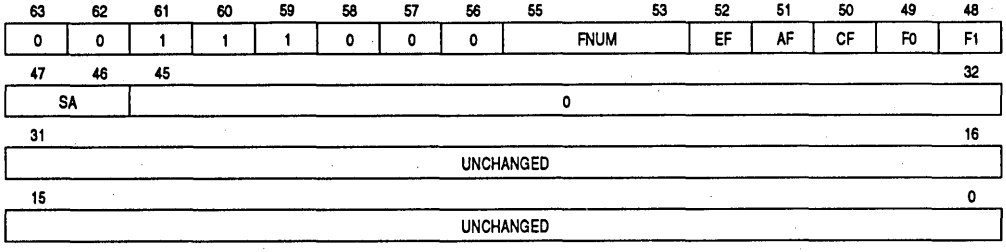


Figure 9-23. RECEIVE MY FRAME Indication

FNUM—Number of Valid Flags

EF—E Flag

AF—A Flag

CF—C Flag

F0—First Additional Flag

F1— Second Additional Flag

SA—Source Address Match Field

00 = MAC aborted the frame prior to an SA match
01 = Bridge match
10 = External CAM match
11 = Local match

9.2.2.16 TOKEN CYCLE END INDICATION. If it has been enabled by the TE bit inside one of the receive frame type registers (RFRs), this indication is generated when the MAC issues a token cycle end event and the FSI has accomplished at least one frame transmission during the current token cycle. The format of this indication is shown in Figure 9-24.

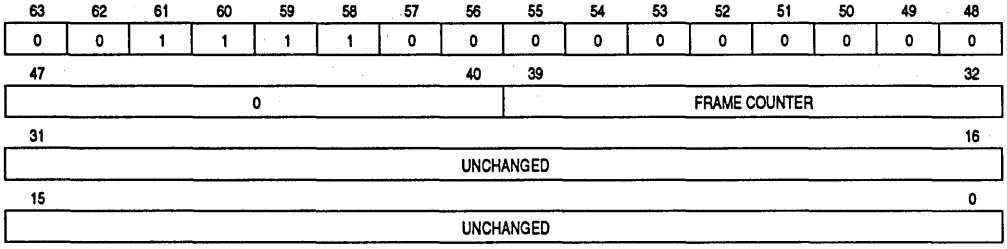


Figure 9-24. TOKEN CYCLE END Indication

FRAME COUNTER

The frame counter indicates the number of frames that were transmitted and not received back.

9.2.3 CAM Commands and Indications

The following paragraphs discuss the CAM commands and indications.

9.2.3.1 SET UP CAM COMMAND. This command may be issued by the host processor through one of the port command registers (CMR) or may be placed inside one of the transmit buffer descriptors rings. The SET UP CAM command writes the CAM ENTRY and a V to the CAM_ID in the CAM. Figure 9-25 illustrates the format for this command.

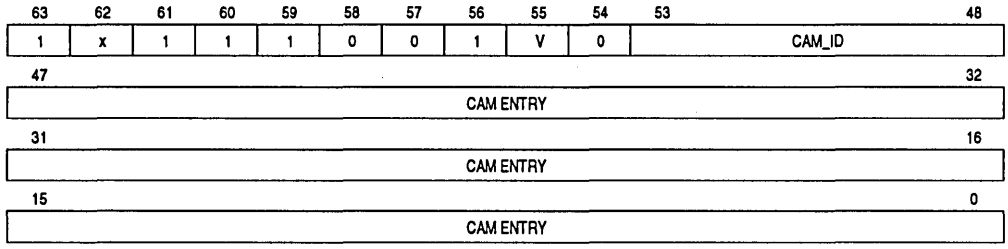


Figure 9-25. SET UP CAM Command

V—Valid

0 = Invalid entry
1 = Valid entry

CAM_ID—CAM Entry Address

CAM ENTRY—Address Entry

This field contains the address entry to place inside the CAM at CAM_ID.

9.2.3.2 READ CAM ENTRY COMMAND. The READ CAM ENTRY command is used to read particular CAM entries for CAM device testing, etc. The value of the CAM entry being read is returned in the READ CAM ENTRY indication. This command may be issued by the host processor through one of the port command registers (CMR) or may be placed inside one of the transmit buffer descriptor rings. The format of the READ CAM ENTRY command is shown in Figure 9-26.

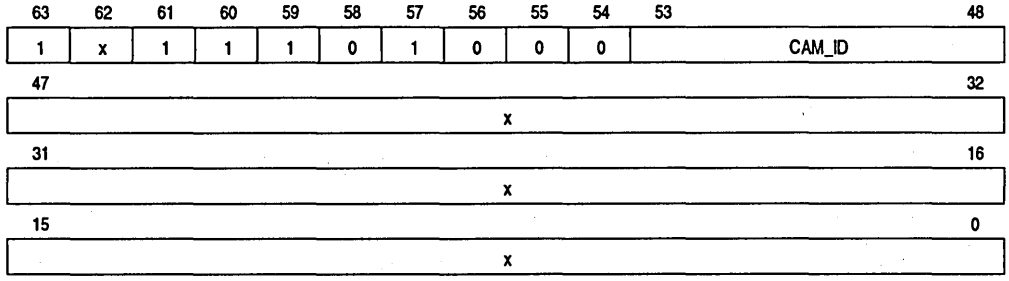


Figure 9-26. READ CAM ENTRY Command

CAM_ID—FSI CAM Entry Address

9.2.3.3 READ CAM ENTRY INDICATION. The READ CAM ENTRY indication is generated in response to the READ CAM ENTRY command. This indication contains the value of the CAM entry being read. The format of the READ CAM ENTRY indication is shown in Figure 9-27.

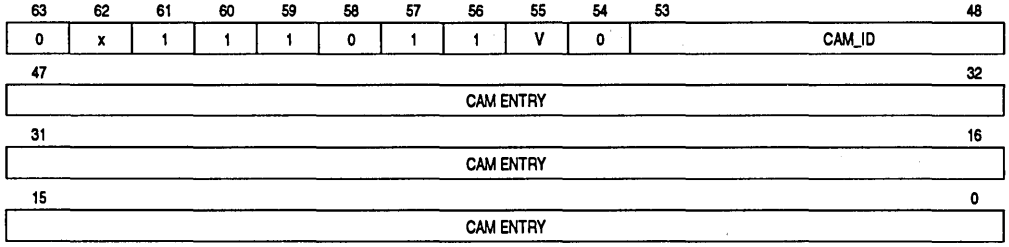


Figure 9-27. READ CAM ENTRY Indication

V—Valid

0 = Invalid entry
1 = Valid entry

CAM_ID—FSI CAM Entry Address

CAM Entry—Address Entry

This field contains the address entry read by this command.

9.2.3.4 COMPARE CAM ENTRY COMMAND. The COMPARE CAM ENTRY command is used for testing the CAM device and/or to determine whether the CAM has been programmed correctly. This command may be issued by the host processor through one of the port command registers or may be placed inside one of the transmit buffer descriptor rings. The format of this command is shown in Figure 9-28.

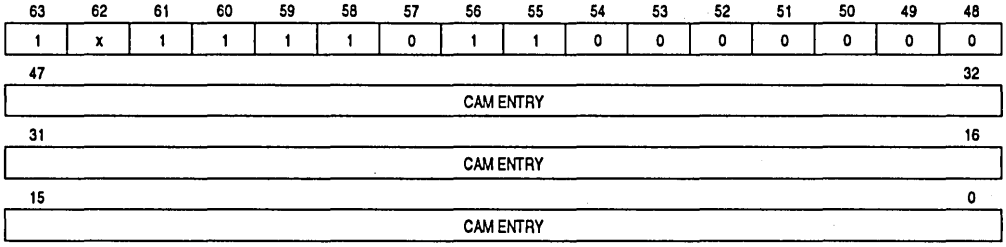


Figure 9-28. COMPARE CAM ENTRY Command

CAM Entry—Address Entry

This field contains a 48-bit address entry that is compared with CAM entries. Bit 47 is the I/G bit.

9.2.3.5 COMPARE CAM ENTRY INDICATION. The COMPARE CAM ENTRY indication is generated in response to a COMPARE CAM ENTRY command. If the M-bit is set, the entry has been found in the FSI CAM and its ID is indicated in the CAM_ID field. If there is no match to any FSI CAM entry, the M-bit is zero and the CAM_ID field is not valid. The format of this indication is shown in Figure 9-29.

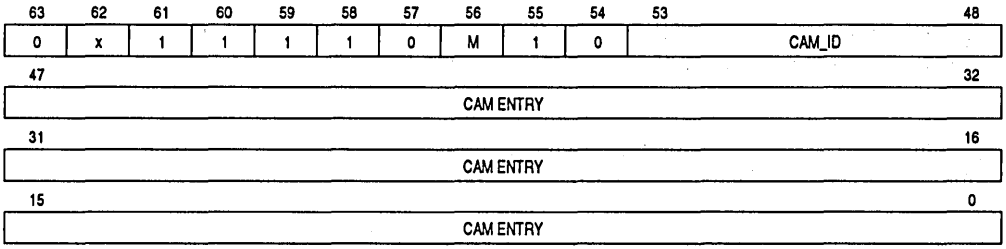


Figure 9-29. COMPARE CAM ENTRY Indication

M—Match Indicator

0 = No match
1 = Match

CAM_ID—CAM Entry Address

This field is valid only if M =1.

CAM ENTRY—Address Entry

This field contains the address entry that this command searched for.

9.2.4 General Commands and Indications

The following paragraphs discuss the general commands and indications.

9.2.4.1 NOP COMMAND. This command might be useful, for example, where frame entries inside the transmit buffer descriptor rings have aged too long. This command has to have both the FIRST and LAST bits set. The FSI will not create any activity executing this command except that the COMMAND DONE indication is given as for all commands. To ensure that entries inside a ring are properly changed, the host processor should ensure that the ring is in the STOPPED state while the host is changing entries. The format for the NOP command is shown in Figure 9-30.

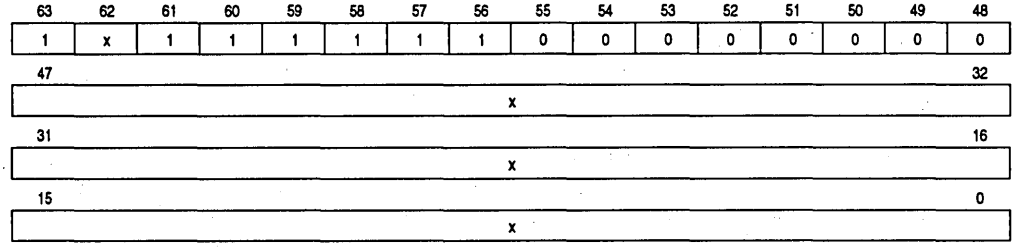


Figure 9-30. NOP Command

9.2.4.2 CONTROL REGISTER WRITE COMMAND. This command allows the user to write to any FSI internal control register. The major advantages of using this command are for FSI initialization and for dynamically changing a control register—e.g., when writing RING READY commands. Instead of writing all the control registers by FCR accesses, the user can prepare an initialization ring that includes control register write commands. The format for the CONTROL REGISTER WRITE command is shown in Figure 9-31.

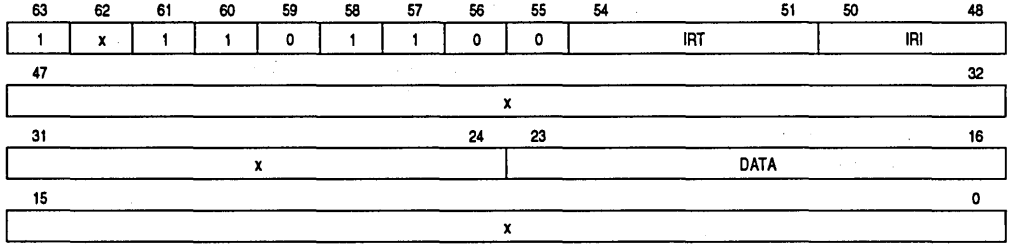


Figure 9-31. CONTROL REGISTER WRITE Command

IRT—Internal Register Type
See Table 7-4 for FCR definitions.

IRI—Internal Register ID
See Table 7-4 for FCR definitions.

DATA—Internal Control Register Data
See Table 7-4 for FCR definitions.

9.2.4.3 16-BIT CONTROL REGISTER WRITE COMMAND FOR CAMEL REGISTER ACCESS. The 16-BIT CONTROL REGISTER WRITE command can be used to write the CAMEL internal register (16-bit registers). The command may be issued by the host processor through one of the port command registers (CMR) or may be placed inside one of the transmit buffer descriptor rings. Figure 9-32 illustrates the format for the 16-BIT CONTROL REGISTER WRITE command for CAMEL indirect access.

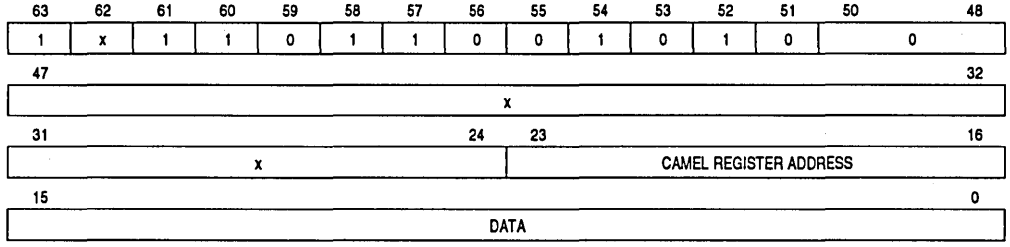


Figure 9-32. 16-BIT CONTROL REGISTER WRITE Command

9

IRT—Internal Control Register Type = A

IRI—Internal Control Register ID = 0

EXA—Extension Address = CAMEL Register Address

9.2.4.4. MOVE COMMAND. This command moves 8 bytes of data between an external absolute memory address and an internal absolute (the 8K of FSI memory) address. The port used for the data transfer is determined by the RPA bit of the ring parameter extension register (PER) along with the C-bit (bit 32). The user should not transfer data to the FSI internal memory without first reserving a block of that memory by use of the GET BLOCK command. Figure 9-33 illustrates the format for the MOVE command.

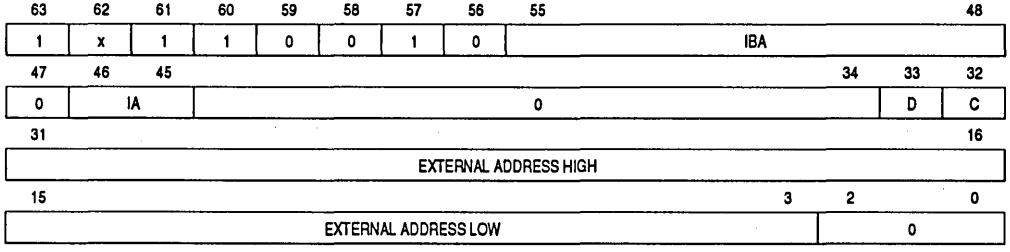


Figure 9-33. MOVE Command

IBA—Internal Block Address

This field contains the address of a block inside the FSI internal memory.

IA—Internal Address

This field contains the address of an 8-byte word inside the block (each block is 32 bytes).

D—Direction

When D is reset, the data transfer occurs from the external memory to the FSI internal memory. When D is set, the data transfer occurs from the FSI internal memory to the external memory.

C—Change Port

When C is reset, the external memory space has the same port assignment as indicated by the RPA bit in the PER. When C is set, the external memory space is assigned to the port opposite that indicated by the RPA bit in the PER.

9.2.4.5 INDIRECT COMMAND. This command causes the FSI to take a command from an external memory and execute it. The external memory command will first replace the INDIRECT command inside the internal memory and then be executed as a normal command. Therefore, the indication of the external memory command will overwrite the original INDIRECT command. The port used to read the external memory command is defined by use of the RPA bit in the PER along with the C-bit (bit 32). Figure 9-34 illustrates the format for the INDIRECT command.

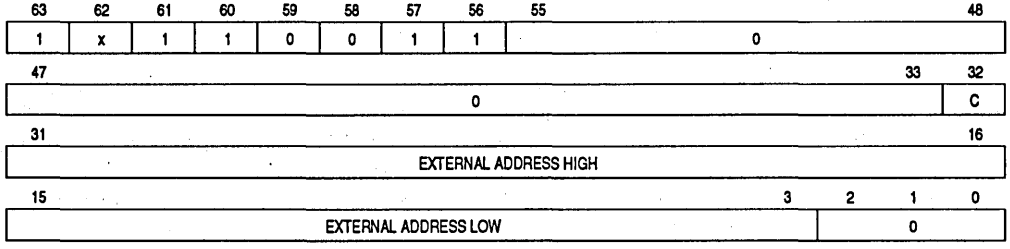


Figure 9-34. INDIRECT Command

C—Change Port

When C is reset, the external memory space has the same port assignment as indicated by the RPA bit in the PER. When C is set, the external memory space is assigned to the port opposite that indicated by the RPA bit in the PER.

9.2.4.6 GET BLOCK COMMAND. This command causes the FSI to extract one 32-byte block of internal memory from the queue of free blocks. Since this block will no longer be used by the FSI, it may be used for temporary storage by the user. The format for this command is shown in Figure 9-35.

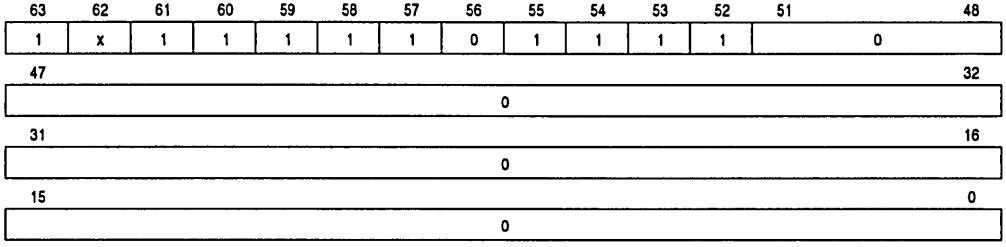


Figure 9-35. GET BLOCK Command

9.2.4.7 GET BLOCK INDICATION. This indication is generated in response to a GET BLOCK command. The format for this indication is shown in Figure 9-36.

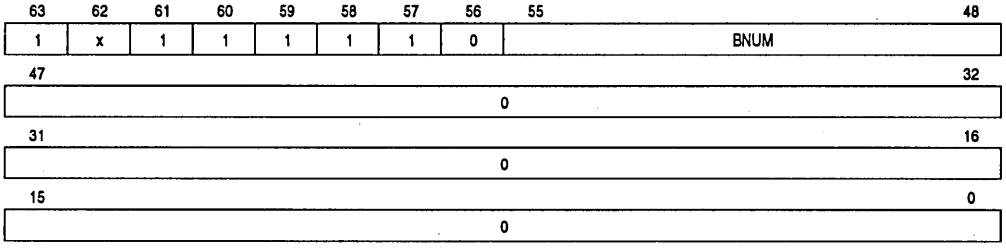


Figure 9-36. GET BLOCK Indication

BNUM—Block Number

This field contains the number of the block that is provided and should be used as the internal block address in MOVE commands.

9.2.4.8 RESOURCE REQUEST COMMAND. The internal RESOURCE REQUEST and RESOURCE RELEASE commands can be used for synchronization between the activities of various channels. The RESOURCE REQUEST command requests the use of a single internal general-purpose resource shared by all the DMA channels or rings. If the resource is occupied or used by another channel, the execution by the requesting channel will be delayed until the channel currently holding the resource releases it. If there are multiple RESOURCE REQUEST commands pending, the resource, when released, will be assigned according to the priority of the requesting channels: 6, 7, 0, 1, 2, then 3. The format of this command is shown in Figure 9-37.

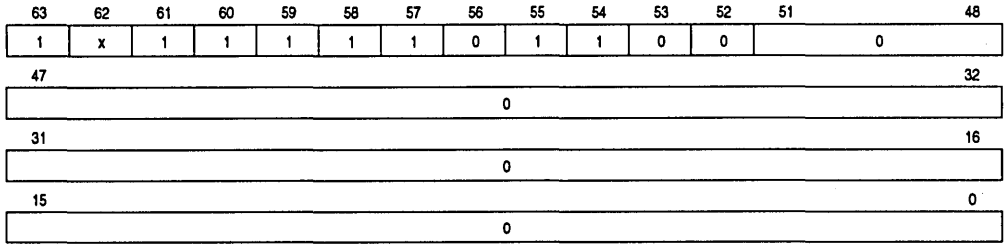


Figure 9-37. RESOURCE REQUEST Command

9.2.4.9 RESOURCE RELEASE COMMAND. The internal RESOURCE REQUEST and RELEASE commands can be used for synchronization between the activities of various channels. The RESOURCE RELEASE command causes this channel to release the internal general-purpose resource. The format of this command is shown in Figure 9-38.

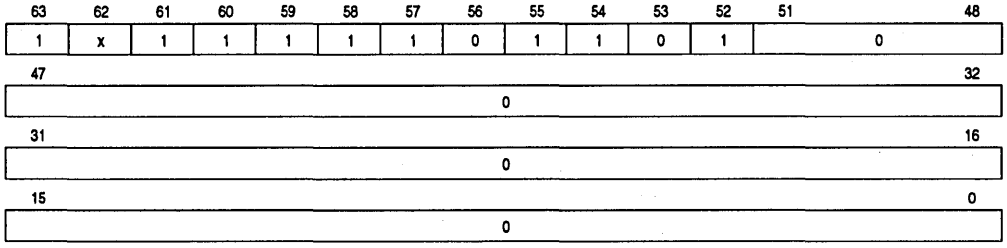


Figure 9-38. RESOURCE RELEASE Command

SECTION 10
IFDDI AS AN FSI

The MC68840 also operates as a stand-alone FSI. During reset, the operational mode used by the MC68840 is determined by the state of the LDADDR/$\overline{TR}$_$\overline{BR}$_$\overline{FWD}$ pin. The exposed FSI is compatible to the MC68839REVD. This section covers all FSI-relevant topics discussed in the IFDDI global specifications and notes functional differences between the MC68840 in IFDDI mode and the MC68840 in FSI mode.

The mode of operation is selected when the IFDDI samples the LDADDR/$\overline{TR}$_$\overline{BR}$_$\overline{FWD}$ pin coming out of reset on the rising edge of $\overline{RESET}$. If the LDADDR/$\overline{TR}$_$\overline{BR}$_$\overline{FWD}$ pin is sampled as V_{CC} on the rising edge of $\overline{RESET}$, the chip will operate as an IFDDI device. If the LDADDR/$\overline{TR}$_$\overline{BR}$_$\overline{FWD}$ pin is connected to GND or an external pulldown, the chip will operate as an FSI (MC68839) device.

10.1 BASIC OPERATION

Refer to **2.2–2.3.**

10.2 FEATURES SUMMARY

Refer to **3.1–3.9** and **3.15.**

10.3 BLOCK DESCRIPTION

Refer to **4.1.**

10.4 FUNCTIONAL OPERATION

Refer to **5.1.**

10.4.1 FSI/MAC Interface Functional Operation

The FSI has transmit and receive interfaces to the MAC. **10.4.1.1 RECEIVE INTERFACE** details the receive interface and **10.4.1.2 TRANSMIT INTERFACE** details the transmit interface.

10.4.1.1 RECEIVE INTERFACE. The receive data MAC interface contains four signal groups: RPATH, RPRITY, RCCTL and RABORT. The data stream is presented on the RPATH7–RPATH0 bus and its parity on the RPRITY line. The RCCTL4–RCCTL0 bus

contains signals that indicate the nature of the data on RPATH. The RABORT signal from the FSI indicates that the FSI is unable to copy the current data.

10.4.1.1.1 RCCTL Bus. The RCCTL bus encodes the following states of data transfer: filler, start-data, data, end-data, frame-status and token-cycle-end. for the exact encoding please refer to table 10-1.

Table 10-1. RCCTL and RPATH Relationship

RCCTL	RPATH	Mnemonic	FSI Function
X0000	0XXX_XXXX	Inter-Frame FILLER	Wait for start data.
00101	DDDD_DDDD	START_DATA	Generates first 4 bytes of the received frame of which the first 3 are 0 (reserved for packet header) and the fourth is copied from RPATH.
00001	DDDD_DDDD	DATA	Add the byte on the RPATH to the received frame.
X0F11	CRRR_DDSS	END_DATA	F is the end-data flush bit. If F=1, the frame reception process is aborted. If the FSI has not yet started to transfer this frame out of internal memory, the frame will be discarded and no indication will be generated. If the beginning of the frame has already been transferred out of memory, a Receive Error Indication will be generated with RRR in bits 47-45. (see 9.2.2.12) If F=0, a Receive Frame Normal Indication is generated with c in bit 56 and DD in bits 47-46. (see 9.2.2.11)
X0000	1XXX_XXXX	Intra-Frame FILLER	Wait for frame status. does not receive the incoming bytes.
XTF10	IIII_IIII	FRAME_STATUS	F is the frame-status flush bit This provides an additional opportunity to discard the frame. If F=1, the FSI acts as described above for END_DATA. If T=1 and the FSI has discarded the frame, a Receive My Frame Indication is generated for this frame, if enabled. (see 9.2.2.15) The SS bits from the END_DATA are placed in bits 47-46. The IIII_IIII field is copied from RPATH to bits 55-48 of the receive indication.
10XXX	XXXX_XXXX	TOKEN_CYCLE_END	The FSI generates a Token Cycle End Indication if enabled. (see 9.2.2.16)

RCCTL should follow the following sequence for each frame:

1. six or more inter-frame FILLERs;
2. one START_DATA;
3. zero or more DATA;
4. one END_DATA;
5. zero or more intra-frame FILLERs;
6. one FRAME_STATUS; and then back to (1).

10.4.1.1.2 RABORT Signal. The RABORT signal is asserted in one of the following cases:

1. The FC of the frame being received doesn't match the receive frame type registers - RFR4 and RFR5 refer to 7.2.2.5.
2. The receive ring which is selected (using RFR) to receive this frame is stopped or not defined.
3. The receive enable bit (in MRR) for the ring selected to receive the frame is reset.
4. The receive data FIFO selected to receive this frame is full, in which case an overrun error is generated.
5. The $\overline{\text{REJECT}}$ input signal was asserted during frame reception.

The FSI asserts RABORT whenever it wishes to abort the frame reception. However, frame reception will be aborted only after END_DATA has been presented on the RCCTL lines.

10.4.1.2 TRANSMIT INTERFACE. The transmit data system interface contains 4 signal groups: TPATH, TXCTL, TABORT, TXRDY. The data to be transmitted is presented on the TPATH7–TPATH0 bus and its parity on the TPRITY line. The TXCTL bus indicates the nature of the data on the TPATH. The TABORT signal is an input signal to the FSI which indicates that frame transmission should be aborted.

10.4.1.2.1 TXCTL Lines. TXCTL1–TXCTL0 encodes one of the following four data transmit states: FILLER, TX_START, TX_END and TX_DATA. For the exact encoding and functionality please refer to Table 10-2.

Table 10-2. TXCTL and TPATH Relationship

TXCTL	TPATH	Mnemonic	When Generated By FSI
00	XXXX_XXXX	FILLER	FSI has nothing to transmit.
01	DDDD_DDDD	TX_START	FSI wants to transmit. TPATH contains the first byte of the frame. (In an FDDI application, it will be the first byte of the packet header.)
10	AXXX_MXXX	TX_END	FSI indicates that it wants to end frame transmission. A=0 indicates a normal end of frame. A=1 indicates that the frame is being aborted and should be treated as a fragment. M=1 indicates that the FSI has at least one more frame ready to send. This could be used to wait more than the normal inter-frame gap of 8 byte clocks before releasing the token.
11	DDDD_DDDD	TX_DATA	TPATH contains a data byte for transmission.

The normal sequence of transfers is as follows:

1. zero or more FILLERs.
2. One TX_START.
3. Many TX_DATAs
4. one TX_END.

10.4.1.2.2 TXRDY Signal. When the FSI is in the TX_DATA state, it will present the next data on the TPATH lines only after TXRDY is asserted. The FSI can however transition to TX_END at anytime.

At frame transmission start, the FSI will present the first byte of the frame with TXCTL set to TX_START and the second byte with TXCTL set to TX_DATA. The second byte will continue to be presented until TXRDY is asserted. Figure 10-1 illustrates the functional relationship between BYTCLK, TXRDY and the frame bytes. PRH is the Packet Request Header, the three bytes of overhead information provided to the MAC. Note that the FSI treats these bytes as data and does not interpret them in any way.

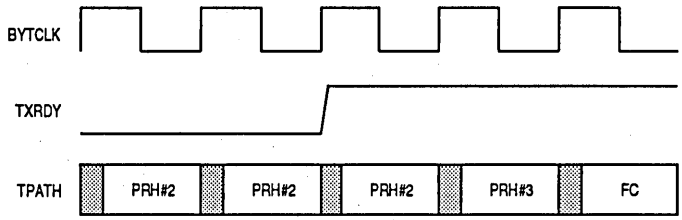


Figure 10-1 TXRDY and Packet Request Header Timing

10

10.4.1.2.3 TABORT Signal. When TABORT is asserted, the FSI will present TX_END on the TXCTL lines, thus terminating the frame transfer cycle.

10.4.1.2.4 Packet Transmission. Every packet to be transmitted by the MAC has an associated control field of three bytes called the packet request header. The MAC uses the packet request header to determine how and when this packet should be sent. The packet request header must be passed to the MAC before the MAC can send the packet data. When the last byte of data in the packet has been passed to the MAC (and before the frame status indicators are sent), the MAC requests the next packet request header by asserting TXRDY. In addition to the packet request header, the FSI must also pass the packet to the MAC. The packet passed to the MAC contains the following fields:

1. FC field
2. DA field
3. SA field
4. INFO field
5. Frame check sequence (FCS) field (unless the MAC has been requested to generate CRC).

10.4.1.2.5 Packet Request Header. The packet request header contains the control bytes presented to the MAC by the FSI. This information is contained in three bytes that control the MAC transmitter process. The third byte is currently ignored and should be passed as all zeros. When the MAC is in the full duplex (FDX) states, only the APPEND_CRC and EXTRA_FS fields have any effect.

First Byte

7	6	5	4	3	2	1	0
FORMAT_TYPE		TOKEN_TYPE		SYNCH_FRAME	IMMED_MODE	SEND_FIRST	BCN_FRAME

Second Byte

7	6	5	4	3	2	1	0
0	SEND_LAST	APPEND_CRC	TOKEN_SEND		EXTRA_FS		

Third Byte

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

FORMAT_TYPE—Type of Packet Request Header

This field provides future MACs with the ability to handle different packet request header formats; it should always be 00.

TOKEN_TYPE—Type of Token Required To Send This Frame

This field determines whether a restricted token, an unrestricted token, any token, or no token is required to send this frame.

00 = Immediate mode. Transmission begins once the MAC enters the Tx_Idle state if the REPEAT_ONLY bit is zero and the RING_OPERATIONAL is one or the IMMED_MODE bit is one. Even though a frame was transmitted without a token, the TOKEN_SEND field could cause a token to be sent. Normally, TOKEN_SEND is 00 (no token sent) when TOKEN_TYPE = 00.

1x = An unrestricted token can be used to send this frame.

x1 = A restricted token can be used to send this frame.

11 = Either type of token can be used to send this frame.

SYNCH_FRAME—Send Synchronous Frame

0 = Send this frame according to the rules for asynchronous frame transmission. The corresponding bit in the FC field is ignored because the MAC does not examine the FC field.

1 = Send this frame according to the rules for synchronous frame transmission.

IMMED_MODE—Ignore RING_OPERATIONAL for This Frame

0 = Use the normal FDDI mode that requires RING_OPERATIONAL to be set before a frame can be sent even if TOKEN_TYPE = 00.

1 = Ignore the value of RING_OPERATIONAL for the determination of whether or not to send this frame. Normally, TOKEN_TYPE = 00 when IMMED_MODE = 1.

SEND_FIRST—Always Send This Frame First

0 = The transmitter treats the frame normally.

1 = The transmitter always ensures that this frame is the first one sent with this token capture. If the token was captured to send an earlier frame when a frame with a SEND_FIRST was given to the MAC, the MAC will release the token after transmitting the earlier frame and wait to capture a later token in order to send this frame.

The SEND_FIRST and SEND_LAST bits can be used to implement the stream concept in the FDDI MA_DATA request service primitive. They can also be used to send a fixed number of packets per token access opportunity (e.g., when both bits are one).

BCN_FRAME—Only Send This Frame in Beacon State

0 = The frame is only sent in the TX_DATA state (state T2).
1 = The frame is only sent in the Tx_Beacon state (state T5) and if the FSI_BEACON bit in the MAC_CNTRL_B register is set. In this state, only the BCN_FRAME, APPEND_CRC, and EXTRA_FS bits in the packet request header have any effect on this frame.

SEND_LAST—Release Token after This Frame Is Sent

0 = The transmitter treats the frame normally.
1 = The transmitter always releases the token after sending this frame, even if this frame was the first one sent with its associated token.

Also see the description of the SEND_FIRST bit.

APPEND_CRC—Generate CRC and Add an FCS Field to the Frame

0 = Since the MAC transmitter will not add an FCS field, the packet passed to the MAC from the FSI should already contain an FCS field as its last four octets. Whether or not an FCS field is added only depends on this bit and is not affected by the FC field transmitted (i.e., the transmitter does not know whether or not this frame is a reserved-for-implementor frame).
1 = The MAC transmitter calculates the CRC and appends it to the end of the frame passed to it by the FSI in the FCS field.

TOKEN_SEND—Type of Token To Send after This Frame

If this frame is the last frame sent with this token, this field indicates the type of token that should be sent (i.e., restricted token, unrestricted token, the token type received, or no token).

00 = No token is released.
01 = An unrestricted token is released.
10 = A restricted token is released.
11 = Whatever token type was originally captured is the type sent (i.e., use R_FLAG). If this is not the last frame sent with this captured token, this bit field has no effect. This bit field can be used in immediate mode to create a token.

EXTRA_FS—Send Extra Frame Status Indicators

This MAC allows an extra two FS indicators to be sent. The MAC will always send the first three FS indicators as R-symbols. There is no way to send less than three or to send S-symbols instead of R-symbols for these first three indicators.

x00	=	TR	RR	II	x = don't care.
001	=	TR	RR	RR	
101	=	TR	RR	SR	
010	=	TR	RR	RS	
110	=	TR	RR	SS	
011	=	TR	RR	RT	
111	=	TR	RR	ST	

10.5 PORT OPERATION

Refer to all of **Section 6.**

10.6 REGISTER DESCRIPTION

Refer to **7.2**; only FSI registers are relevant.

10.7 SIGNAL DESCRIPTION

An IFDDI pinout in FSI mode is shown in Figure 10-2.

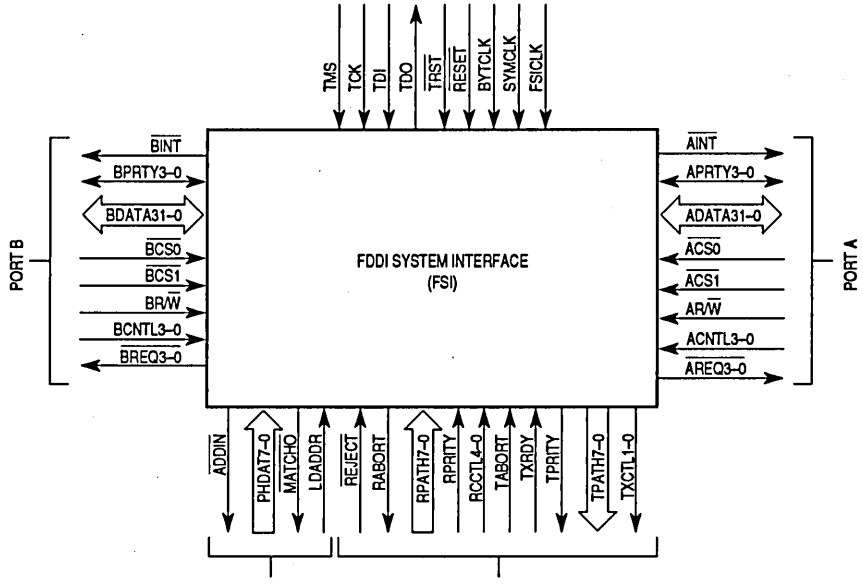


Figure 10-2. IFDDI Pinout in FSI Mode Only

NOTE

The ADDIN pin should be tied high in FSI mode FDDI applications.

10.7.1 Port A Interface

Refer to **8.1.**

10.7.2 Port B Interface

Refer to **8.2.**

10.7.3 FSI/MAC Interface

The FSI/MAC interface signals are discussed in the following paragraphs.

Transmit Data Bus (TPATH7–TPATH0)

This 8-bit, CMOS-level, output data bus is used for byte transfers from the FSI to the MAC. These lines are used at different times to carry either packet data, which could be data to be sent or part of the packet request header, or extra control information.

Transmit Parity (TPRITY)

This CMOS-level output signal indicates the parity of the TPATH7–TPATH0 data bus. The TXCTLx lines are not included in the parity generation.

Transmit Control Signals (TXCTL1–TXCTL0)

These CMOS-level output signals are used by the FSI to indicate the type of data transfer. The encoding for these signals is as follows:

TXCTL1–TXCTL0	Transfer Type
00	Filler
01	Tx_Start
11	Tx_Data
10	Tx_End

Transmit Ready (TXRDY)

When asserted (high), this CMOS-level input signal indicates to the FSI that the MAC is ready to accept additional TX_DATA transfer cycles. When the FSI detects that it is negated (low), it will continue to present the current TX_DATA cycle.

Transmit Abort (TABORT)

When asserted, this CMOS-level input signal indicates to the FSI that it should abort the current transmission.

Receive Data Bus (RPATH7–RPATH0)

This 8-bit, CMOS-level input data bus is used for byte transfers from the MAC to the FSI. These lines are used at different times to carry a pair of data symbols belonging to a received frame, status information describing the frame and why it ended, or packet frame status indicators.

Receive Parity (RPRITY)

This CMOS-level input signal indicates the parity of the RPATH7–RPATH0 data bus. The RCCTLx lines are not included in the parity calculation. The parity may be odd or even. The FSI may be programmed to either check or ignore the parity. The correct parity is stored inside the FSI internal memory together with the associated data.

Receive Control Signals (RCCTL4–RCCTL0)

These CMOS-level input signals, which are controlled by the MAC, are used by the FSI to determine the type of MAC-to-FSI transfer. The RCCTLx encoding definition is listed in Table 10-1.

Receive Abort (RABORT)

This CMOS-level output signal is used to indicate that the FSI will not accept any more data from the MAC. This signal is asserted by the FSI if there is no space in the receive internal data FIFO to store the information, if the data FIFO is disabled for reception, or if the $\overline{\text{REJECT}}$ signal is asserted when doing a CAM or external recognition logic match.

The RABORT signal is generated in any of the following cases:

a. The FC of the frame being received is not specified to be received in either receive data FIFO.
b. The receive data FIFO selected to receive this frame is disabled.
c. The receive data FIFO selected to receive this frame is full, in which case an overrun error is generated.
d. The $\overline{\text{REJECT}}$ input signal was asserted during frame reception.

Reject Input Line ($\overline{\text{REJECT}}$)

This line is a CMOS-level input signal to the FSI from external logic or an external CAM indicating that the FSI should reject the incoming frame. This signal would typically be used when the MAC is in promiscuous mode (MAC control signals are set as: CopyAll = 00, CopyGroup = 1 and CopyIndLLC = 1).

Byte Clock (BYTCLK)

This TTL-level input signal has a frequency of 12.5 MHz and is synchronized with data byte transfers on both the transmit and receive data paths.

Symbol Clock (SYMCLK)

This TTL-level input signal has a frequency of 25 MHz. This clock is the main FSI clock used for all internal synchronous operations in addition to data sampling on the FSI receive path.

10.7.4 CAM Interface

The CAM interface signals are discussed in the following paragraphs.

PHY Receive Data Bus (PHDAT7–PHDAT0)

These eight CMOS-level input signals are connected to the MAC/ELM interface to provide an address matching function through the FSI CAM operating on data currently being received by the MAC. PHDAT3–PHDAT0 should be connected to RCDAT3–RCDAT0 of the ELM circuit, and will contain the second symbol received of a symbol pair. PHDAT7–PHDAT4 should be connected to RCDAT8–RCDAT5 and will contain the first symbol received of a symbol pair. Note, that RCDAT4 and RCDAT9 are not connected to the FSI because they are always zero for data symbols during address or frame data reception.

Load Address (LDADDR)

This CMOS-level input signal is the main CAM control signal from the MAC. It is asserted active for only one BYTCLK cycle just before the first byte of the destination address (DA) or source address (SA) field is presented on the PHDATx bus.

Long or Short Address (ADDR16)

This input was maintained for backward compatibility. Since the MC68838 MAC chip will not copy a frame based on a 16-bit address match, it is no longer a functional signal and should be tied off to ground or V_{CC} or left connected to the MC68838 ADDR16 signal.

Address Match Line ($\overline{\text{MATCHO}}$)

This open-drain output signal indicates to the MAC whether the address presented to the FSI CAM matches an individual or multicast (group) address stored in the CAM. It is asserted low upon match. This signal is examined by the MAC at the end of the received SA or DA fields. When this signal is asserted, the received address belongs to this station's set of long addresses, and the appropriate SA or DA actions should be taken by the MAC. The user must supply a pullup resistor for this signal to operate correctly.

10.7.5 Miscellaneous Signal

BYPASS (BYPASS)

This CMOS level input signal is similar to the DA pin operation when in IFDDI mode. During power-up reset, if this pin is driven low on the rising edge of RESET, the 68840 will operate in bypass mode, and the PSF block is not used.

If this pin is driven high on the rising edge of RESET, the 68840 will operated in non-bypass mode, and the PSF block is used. There is an internal pull-up on this pin.

10.8 COMMANDS AND INDICATIONS

Refer to all of **Section 9** except **9.2.4.3**.

10.9 INITIALIZATION AND PROGRAMMING

Refer to **11.3–11.5.**

10.10 TEST OPERATION

Refer to all of **Section 12** except **12.1.4.** The boundary scan register path from TDI to TDO *in FSI mode* is shown in Table 10-3.

Table 10-3. Boundary Scan Register Path in FSI Mode Only

Pin	Name	I/O	Type	Pin	Name	I/O	Type	Pin	Name	I/O	Type
1	PHDAT7	IN	CMOS	2	PHDAT6	IN	CMOS	3	PHDAT5	IN	CMOS
4	PHDAT4	IN	CMOS	5	PHDAT3	IN	CMOS	6	PHDAT2	IN	CMOS
7	PHDAT1	IN	CMOS	8	PHDAT0	IN	CMOS	9	TABORT	IN	CMOS
10	ADDIN	IN	CMOS	11	LDADDR	IN	CMOS	12	TXRDY	IN	CMOS
13	LDA_DIR	DRC	Int	14	"0"			15	TPATH7	OUT	CMOS
16	TPATH6	OUT	CMOS	17	TPATH5	OUT	CMOS	18	TPATH4	OUT	CMOS
19	TPATH3	OUT	CMOS	20	TPATH2	OUT	CMOS	21	TPATH1	OUT	CMOS
22	DIR3	DRC	Int	23	TPATH0	OUT	CMOS	24	DIR2	DRC	Int
25	TPRITY	OUT	CMOS	26	TXCTL1	OUT	CMOS	27	TXCTL0	OUT	CMOS
28	RABORT	OUT	CMOS	29	$\overline{\text{MATCH0}}$	OUT	O_D	30	"0"		
31	FSICLK	IN	TTL	32	RCCTL4	IN	CMOS	33	RCCTL3	IN	CMOS
34	RCCTL2	IN	CMOS	35	RCCTL1	IN	CMOS	36	RCCTL0	IN	CMOS
37	REJECT	IN	CMOS	38	RPATH7	IN	CMOS	39	RPATH6	IN	CMOS
40	RPATH5	IN	CMOS	41	RPATH4	IN	CMOS	42	RPATH3	IN	CMOS
43	RPATH2	IN	CMOS	44	RPATH1	IN	CMOS	45	RPATH0	IN	CMOS
46	RPRITY	IN	CMOS	47	RESET	IN	TTL	48	BYTCLK	IN	TTL
49	SYMCLK	IN	TTL	50	BCNTL3	IN	TTL	51	BCNTL2	IN	TTL
52	BCNTL1	IN	TTL	53	BCNTL0	IN	TTL	54	BRW	IN	TTL
55	ACNTL3	IN	TTL	56	ACNTL2	IN	TTL	57	ACNTL1	IN	TTL
58	ACNTL0	IN	TTL	59	ARW	IN	TTL	60	ADATA31	I/O	TTL
61	ADATA30	I/O	TTL	62	ADATA29	I/O	TTL	63	"0"		
64	ADATA28	I/O	TTL	65	ADATA27	I/O	TTL	66	ADATA26	I/O	TTL
67	ADATA25	I/O	TTL	68	ADATA24	I/O	TTL	69	ADATA23	I/O	TTL
70	ADATA22	I/O	TTL	71	ADATA21	I/O	TTL	72	ADATA20	I/O	TTL
73	ADATA19	I/O	TTL	74	ADATA18	I/O	TTL	75	ADATA17	I/O	TTL
76	ADATA16	I/O	TTL	77	ADATA15	I/O	TTL	78	ADATA14	I/O	TTL
79	ADATA13	I/O	TTL	80	ADATA12	I/O	TTL	81	ADATA11	I/O	TTL
82	"0"			83	ADATA10	I/O	TTL	84	ADATA9	I/O	TTL
85	ADATA8	I/O	TTL	86	ADATA7	I/O	TTL	87	ADATA6	I/O	TTL
88	ADATA5	I/O	TTL	89	ADATA4	I/O	TTL	90	ADATA3	I/O	TTL
91	ADATA2	I/O	TTL	92	ADATA1	I/O	TTL	93	ADATA0	I/O	TTL
94	APRTY3	I/O	TTL	95	APRTY2	I/O	TTL	96	APRTY1	I/O	TTL
97	APRTY0	I/O	TTL	98	AREQ1	OUT	TTL	99	AREQ2	OUT	TTL
100	A_IEN	DRC	Int	101	A_OEN	TSC	Int	102	AREQ0	OUT	TTL
103	AREQ3	OUT	TTL	104	ACS1	IN	TTL	105	ACS0	IN	TTL
106	BCS0	IN	TTL	107	BCS1	IN	TTL	108	BREQ3	OUT	TTL
109	BREQ0	OUT	TTL	110	B_OEN	TSC	Int	111	B_IEN	DRC	Int

Table 10-3. Boundary Scan Register Path in FSI Mode Only (Continued)

Pin	Name	I/O	Type	Pin	Name	I/O	Type	Pin	Name	I/O	Type
112	BREQ2	OUT	TTL	113	BREQ1	OUT	TTL	114	BPRTY0	I/O	TTL
115	BPRTY1	I/O	TTL	116	BPRTY2	I/O	TTL	117	BPRTY3	I/O	TTL
118	BDATA0	I/O	TTL	119	BDATA1	I/O	TTL	120	BDATA2	I/O	TTL
121	BDATA3	I/O	TTL	122	BDATA4	I/O	TTL	123	BDATA5	I/O	TTL
124	BDATA6	I/O	TTL	125	BDATA7	I/O	TTL	126	BDATA8	I/O	TTL
127	BDATA9	I/O	TTL	128	BDATA10	I/O	TTL	129	"0"		
130	BDATA11	I/O	TTL	131	BDATA12	I/O	TTL	132	BDATA13	I/O	TTL
133	BDATA14	I/O	TTL	134	BDATA15	I/O	TTL	135	BDATA16	I/O	TTL
136	BDATA17	I/O	TTL	137	BDATA18	I/O	TTL	138	BDATA19	I/O	TTL
139	BDATA20	I/O	TTL	140	BDATA21	I/O	TTL	141	BDATA22	I/O	TTL
142	BDATA23	I/O	TTL	143	BDATA24	I/O	TTL	144	BDATA25	I/O	TTL
145	BDATA26	I/O	TTL	146	BDATA27	I/O	TTL	147	BDATA28	I/O	TTL
148	"0"			149	BDATA29	I/O	TTL	150	BDATA30	I/O	TTL
151	BDATA31	I/O	TTL	152	AINT	OUT	O_D	153	DIR1	DRC	Int
154	BINT	OUT	O_D	—	—	—	—	—	—	—	—

10.11 ELECTRICAL CHARACTERISTICS

Refer to Table 10-4 and Figure 10-3.

Table 10-4. Clocks and MAC Interface Timing

Num	Characteristics	Min	Max	Unit
T1	BYTCLK Cycle Time	80	—	ns
T2	SYMCLK Cycle Time	40	—	ns
T3	SYMCLK Pulse Width Low	17	23	ns
T4	SYMCLK Pulse Width High	17	23	ns
T5	SYMCLK to BYTCLK High Skew	0	15	ns
T6	SYMCLK to BYTCLK Low Skew	0	15	ns
T7	SYMCLK Rise or Fall Time	—	5	ns
T8	Signal Valid to SYMCLK Falling Edge[1]	0	—	ns
T8A	Signal Valid to SYMCLK Falling Edge[1]	13	—	ns
T9	SYMCLK Falling Edge to Signal Invalid[1]	3	—	ns
T9A	SYMCLK Falling Edge to Signal Invalid[1]	4	—	ns
T10	BYTCLK Rising Edge to Signal Valid[1,2]	—	25	ns

NOTES:
1. Relative to the falling edge of SYMCLK immediately preceding the rising edge of BYTCLK.
2. For every 10-pF loading capacitance more than 50-pF, add 1 ns.
3. Rise and fall times are not tested.

NOTE

The FSI samples its input signals from the MAC with the falling edge of SYMCLK, which precedes the rising edge of BYTCLK. This is consistent with the MAC, which outputs its signals to the FSI on the rising edge of BYTCLK. The FSI outputs and MAC inputs operate in a similar fashion.

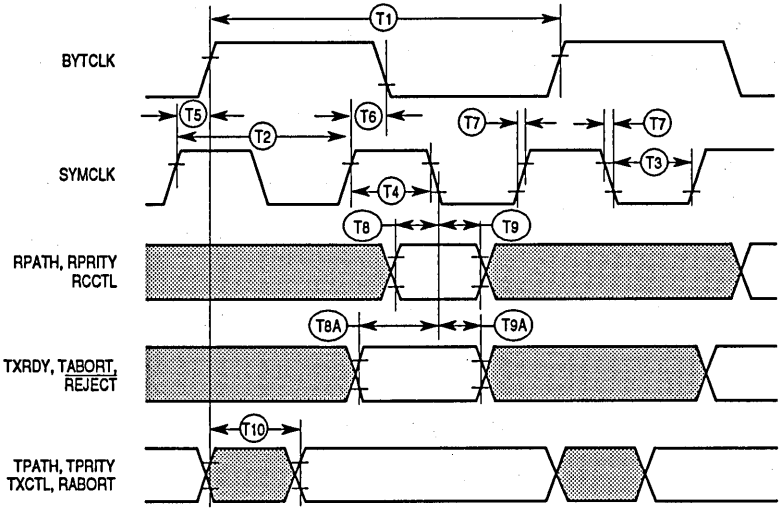


Figure 10-3. FSI Clocks and MAC Interface Timing

10.12 CAM INTERFACE TIMING

Num	Characteristics	Min	Max	Unit
T41	LDADDR Asserted to SYMCLK Falling Edge[1]	0	—	ns
T42	SYMCLK Falling Edge to LDADDR Invalid[1]	3	—	ns
T43	ADDIN Asserted to SYMCLK Falling Edge[1]	0	—	ns
T44	SYMCLK Falling Edge to ADDIN Invalid[1]	3	—	ns
T45	BYTCLK Rising Edge to $\overline{\text{MATCHO}}$ Asserted[2]	—	25	ns
T46	PHDAT Valid to SYMCLK Falling Edge[1]	0	—	ns
T47	SYMCLK Falling Edge to PHDAT Invalid[1]	3	—	ns

NOTES:
1. Relative to the falling edge of SYMCLK when BYTCLK is low.
2. For every 10-pF loading capacitance more than 50 pF, add 1 ns.

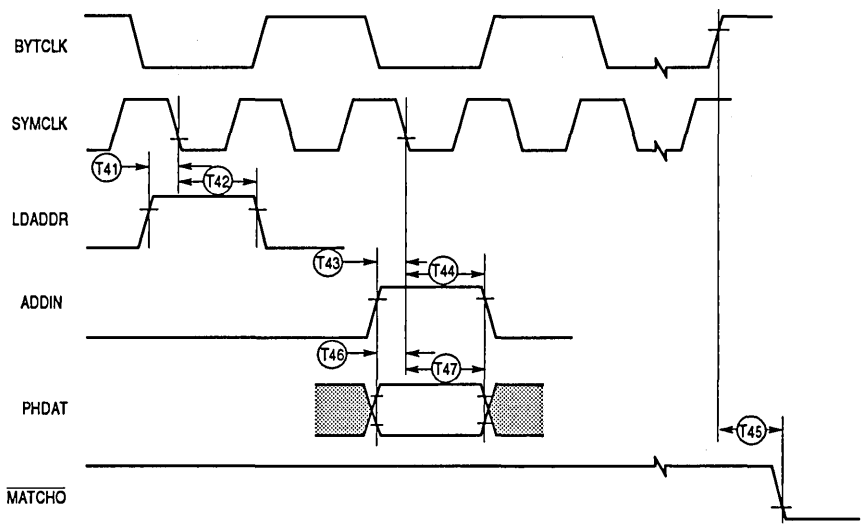


Figure 10-4. FSI-CAM Interface Timing

10.13 FSI $\overline{\text{REJECT}}$ TIMING

When $\overline{\text{REJECT}}$ is asserted, the FSI asserts RABORT to the MAC. In bypass mode (i.e., FSICLK = SYMCLK), RABORT will be asserted on the next rising edge of BYTCLK. In nonbypass mode (i.e., FSICLK >SYMCLK), RABORT will be asserted between 1 and 5 BYTCLK cycles after the assertion of $\overline{\text{REJECT}}$. As a result, the MAC will supply the end data indication to the FSI.

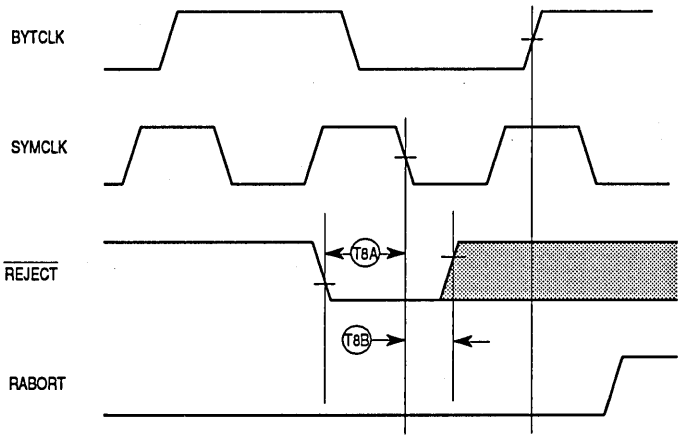


Figure 10-5. FSI $\overline{\text{REJECT}}$/RABORT Timing

SECTION 11
INITIALIZATION AND PROGRAMMING

The initialization sequence for the IFDDI should begin only after the following reset sequence is complete. Upon negation of $\overline{\text{RESET}}$, the command done (CDN) bit should be set in the IMR1 register. Initialization can begin as soon as the following two conditions are satisfied: 1) the IFDDI issues a command done interrupt, which is asserted only after the FSI core performs its reset sequence (approx. 300 FSICLK periods); 2) 100 SYMCLK periods passed when in non-bypass mode, where SYMCLK >= 3 * FSICLK periods. Once these conditions are satisfied, the initialization of the IFDDI should be performed in the following order: ELM, Twisted Pair Operation, MAC, and FSI. IFDDI initialization and programming is discussed in the following subsections; an example of the initialization sequence is given in the next four subsections.

11.1 ELM INITIALIZATION

To begin the initialization process, configure the ELM_CNTRL_A = 0000 (hex) and ELM_CNTRL_B = 0080 (hex) for a single attached station (SAS). For suggested values for the SAS in normal mode, see Table 7-9.

The timers for the physical connection management (PCM) and link confidence test (LCT) operations are written next. The A_MAX, LS_MAX, TB_MIN, T_SCRUB, T_OUT, NS_MAX, and the LC_SHORT registers are written with their initial values (see Table 7-9). The final operations before performing the connection sequence are a read of the ELM_INT, VIOL_SYM_CTR, and LINK_ERR_CTR registers to clear them. In preparation for the connection sequence, the ELM_MASK register should have a minimum of the PCM_ENABLED, PCM_BREAK, and PCM_CODE interrupts enabled out of the 16 available interrupts.

Physical connection management (PCM) is the process by which two stations are connected together on the FDDI ring. During the PCM sequence, two PHY-level devices (or ELMs) exchange synchronization and status information in a lock-step progression. Although the PCM sequence is defined for use in station management (SMT) software, the ELM provides hardware assistance, making only a small amount of pseudo-code necessary to drive the connection sequence. The basic flow of the PCM sequence using the ELM is as follows. First, the pseudo-code writes to the ELM XMIT_VECTOR register the bit or bit sequence to be signaled. Next, the pseudo-code writes to the VECTOR_LENGTH register the number of bits that have been written to the XMIT_VECTOR register. All together, nine bits are signaled between the two stations as per the SMT standard, normally in groups of three bits or less. The nine bits of the PCM sequence represent the following operations: PC_TYPE, accept connection, ESCAPE_BIT, LCT duration, MAC available, PHY loopback, LCT results, MAC loopback

previous to attach, and attach. The LCT is performed after the bit representing PHY loopback is signaled. If the PCM sequence and the LCT are successful, the join sequence is performed by writing a one to the PC_JOIN field in the ELM_CNTRL_B register. The user should note that all SMT vendors porting to the Motorola FDDI chip set provide the necessary pseudo-code with the remaining SMT functionality provided in their software.

11.2 TWISTED-PAIR INITIALIZATION

To initialize the IFDDI for use in systems designed for twisted-pair instead of fiber, three blocks must be configured: streaming cipher scrambler/descrambler, FOTOFF control, and signal detect filtering.

To initialize for twisted-pair operation, the FOTOFF_ASSERT and FOTOFF_DEASSERT timers should be written for the desired operation of FOTOFF. The three options available to the user are: send scrambled quiet only, send scrambled quiet for a minimum of 50uS followed by true quiet, send true quiet only (for fiber systems only). The duration of scrambled quiet and true quiet are controlled by the time in the FOTOFF_ASSERT and FOTOFF_DEASSERT timers. The IFDDI determines which of the above three modes to use by the state of the FOTOFF_CNTRL field in the CIPHER_CNTRL register. Next, the desired configuration of the signal detect operation should be set. There are three signal detect filtering options available to the user: a turn-on filter, a turn-off filter, and an NRZ filter. The operation of signal detect is controlled by the SDOFEN, SDONEN and SDNRZEN bits in the CIPHER_CNTRL register. Finally, the streaming cipher scrambling and descrambling operations should be enabled by setting the CIPHER_EN bit to one in the CIPHER_CNTRL register.

11.3 MAC INITIALIZATION

To begin the initialization process, a read should be performed on the MAC_INT_A, MAC_INT_B, MAC_INT_C, FRAME_CT, and LOST_CT registers to reset the registers to zero. Next, the station addresses (long and short) should be written to the MLA_A, MLA_B, MLA_C, and MSA registers. The FDDI protocol allows for one 48-bit address and one 16-bit address per station. Next, the T_REQ register is written with the station's desired token rotation time. The TVX_VALUE/T_MAX register is set according to the TRT timer time-out to be used when the ring is not operational. Note that the absolute value of T_MAX must be greater than the absolute value of T_REQ and that the MAC does not check these values (see Table 11-1). Finally, the MAC is turned on by setting the MAC_ON bit in the MAC_CNTRL_A register. If an external CAM is being used, the EXT_DA_MATCH bit can be set in MAC_CNTRL_B to provide for extended timing requirements needed for address matching.

Table 11-1. Recommended MAC Register Values

Register	Typical Value (ms)	Register Value
T_REQ	4.0140	FF3C
TVX_VALUE/T_MAX	2.621/167.77	80E0

11.4 FSI INITIALIZATION

After reset, all internal control registers contain zeros; therefore, 1) port A and port B are disabled, 2) transmission from the MAC interface is disabled, 3) reception from the MAC interface is disabled, and 4) all ring state registers have the EX bit reset (i.e., no rings are defined).

The bits in status register 1 (SR1) have the values and meanings listed in Table 11-2, and the bits in status register 2 (SR2) have the values and meanings listed in Table 11-3.

Table 11-2. Status Register 1 Settings

Name	Setting	Meaning
RNR5–RNR0	000000	Rings Are Not Yet Defined
RCC3–RCC0	0000	No Commands Have Been Completed
RXC5–RXC4	00	No Receive Is Complete
CRF	1	Control Register Is Free
CDN	1	Command Done
RER5–RER0	000000	No Ring Error
ROV5–ROV4	00	No Receive Overrun
POEA	0	No Port A Operation Error
POEB	0	No Port B Operation Error
HER	0	No Host Error
IOE	0	No Internal Error
CIN	0	No CAMEL Interrupt Has Occurred
STE	0	No SMT Timer Interrupt Has Occurred

Table 11-3. Status Register 2 Settings

Name	Setting	Meaning
INT7–INT0	00000000	No User- Defined Interrupt Has Occurred.
DRE (7–6, 3–0)	00, 0000	No Error Has Occurred on a Destination Ring .
DXC (7–6, 3–0)	00, 0000	No Frames Transferred to Destination Rings.
DNR (7–6, 3–0)	11, 1111	Destinations Rings are N ot Yet Defined.

Also, the bits of the port status register (PSR) indicate that both ports are in the idle state and that interrupt mask register 1 (IMR1) is all zeros, disabling all interrupts.

Initialization of the FSI should start by programming the internal control registers through accesses to the FSI control register (FCR):

1. Port control registers (PCRs) should be programmed to define the configuration of the external bus connected to each port to enable port DMA operations. Direct access by the host processor to either port's registers is always enabled. The PCR for port A should always be defined. This PCR defines the parity control for the entire FSI, including port B, the internal FSI memory, and the MAC interface. When port B is not used as a separate data transfer port (i.e., it is used for the address associated with port A data or in 64-bit mode for 32 bits of the 64-bit data), it should remain disabled. If port B registers are used, the host parity enable control bit in the PCR of port B should be defined.
2. Port memory page registers (PMPs) should be programmed according to the external memory page used by the system (if any). Note that each port may be connected to separate memory systems and, therefore, the port memory page register value may be different for each port.
3. The ring parameter register (RPR) should be defined for the ring(s) that are used by the system. Other ring parameter registers may remain undefined and not used.
4. Command parameter registers (CPRs) need to be defined if the relevant command register is used for issuing commands directly to the FSI.
5. Parameter extension registers (PERs) need to be defined for any ring using local memory or for a destination ring.
6. FIFO watermark registers (FWRs) should be defined for all the transmit and receive rings used by the system and for each command register that is used for transmit commands.
7. The limit registers (LMTs) must be defined for all rings using local memory.
8. Receive frame type registers (RFRs) should be defined for each of the receive rings if both rings are to be used. Note that, if each control bit relative to one or several frame types is zero in both receive frame type registers, such frames will not be received, and the MAC interface will generate the RABORT signal to the MAC. If the same control bit is set in both receive rings, the frame is directed to receive ring #4.
9. The header length register (HLR) must be defined if operating in split header mode.
10. The maximum receive memory space (RMR) should be defined for each receive ring to be used. Note that the total amount of memory used for transmission and reception should not exceed the FSI internal memory size (8K bytes). The FSI will not check the total memory requirements according to the user definitions of the various memory spaces. An internal operation error will occur during operation if the internal memory overruns.
11. The MAC interface transmit control register (MTR) should be defined to enable data transfer to the MAC. Note that the descriptors and data are transferred by the FSI to internal memory even when the transmitter of the MAC interface is not enabled.
12. To enable the receive interface, the rings to be used are defined and set to ready. Next, the MAC interface receive control register (MRR) should be defined and enabled to specify the operation of the MAC interface receiver. Once RE4 or RE5 is set, data may be received and stored inside the internal memory, which could lead to a receive overrun if a ring is not prepared to receive the incoming data. Therefore, the definition of the MAC interface receive control register should be the last operation after all receive ring parameters have been specified and the receive rings to be used have been defined.

13. The CAM can be disabled by writing a one into the CAM disable (CDS) bit in the IFDDI configuration register (ICR).

After performing the previous operations, the next initialization step is to define all the transmit rings used by the application. This step is performed by issuing DEFINE RING commands through one of the command registers. The CDN status bit in the status register 1 (SR1) should be inspected before issuing each subsequent command. The status register 2 (SR2) is disabled until the user enables it. The DEFINE RING command may include the ready flag in order to transfer the ring state directly to ready. If the user is not yet prepared for ring operation, he may choose to perform a ring ready control access at a later time. As soon as one of the rings is defined as ready, the FSI will immediately begin to read descriptor operations.

NOTE

The destination rings or local memory should be defined *before* defining the transmit or receive rings if local memory or DMA is to be used.

11.5 INTERNAL MEMORY ALLOCATION

The FSI provides an internal 8K memory space that is used as a latency buffer for data transfers. The internal memory is also used to contain temporary storage for internal snapshots (20 descriptors) of the external descriptor rings and to contain indications before they are written out to external memory.

The internal memory of the FSI is arranged in blocks of 32 bytes each. The internal memory manager requests and frees the blocks as required so that the internal memory is efficiently used during the data transfer operations.

11.5.1 Transmit Ring

For data transmission, the user allocates the amount of internal memory used by means of the FIFO watermark register (FWR). The FWR should contain a value that is related to the bus latency required to support the working port's bus and to prevent transmit underruns from occurring. Deriving this value is explained in **11.5 Watermark Calculation.** During transmission, the total internal memory used by the active ring or channel is twice the value programmed in the FWR.

11.5.2 Receive Ring

For data reception, the memory allocation is set by the receive memory registers (RMRs). Again, this value is determined based on the bus latency required to prevent internal memory overruns while receiving data. The watermark setting for a receive ring or channel is used to determine how soon the external bus will be accessed for a data transfer, not specifically for bus latency. Since most buses are capable of emptying the internal FIFO very quickly, setting a low watermark means that the external bus will be requested repeatedly following an emptying/filling cycle.

11.6 WATERMARK CALCULATION

The following two examples illustrate the calculation of the maximum internal FSI memory space required for transmission.

Example 1

Three transmit rings are defined, each having 256-byte watermarks. The watermark for transmit commands issued directly to the FSI is 64 bytes. The calculation is as follows:

256 x 3 = 768 bytes	The total data space in a ring inactive state.
256 bytes	Additional data space taken by one of the FIFOs in an active state—i.e., an active FIFO can grow to twice the watermark.
64 bytes	Memory space required by a transmit command issued to the FSI through the command register. The internal command data FIFO will not occupy any internal space until the transmit command is issued through one of the command registers. Once such a command is issued, this FIFO may fill up to its specified watermark when not transmitting and may grow to a maximum of twice its watermark during actual transmission.
160 x 3 = 480 bytes	The total control space in a ring inactive state.
128 bytes	Additional control space taken by a FIFO in an active state.
1696 bytes	Total FSI internal memory required.

The 64 additional bytes required by a transmit command issued directly to the FSI will take the place of 256 additional transmit data FIFO bytes and are therefore not included in the calculations. Since only one transmit FIFO is active at once, if the watermark on the transmit command FIFO exceeded all other transmit FIFO watermarks, then that difference would need to be included. Also, any additional space required by the data during transmission from a buffer descriptor ring will not exceed one additional watermark number of bytes as defined for that buffer descriptor ring.

Example 2

Transmit buffer descriptor ring 0 has a 512-byte watermark. Transmit buffer descriptor ring 1 has a 64-byte watermark. Transmit buffer descriptor ring 2 has a 64-byte watermark. Transmit commands are not issued directly to the FSI through the command register. The calculation is as follows:

512 + 64 + 64	= 640 bytes	The total data space in an inactive state.
Max (512, 64, 64)	= 512 bytes	Additional data space in an active state.
160 x 3	= 480 bytes	The total control space in an inactive state.
	128 bytes	Additional control space in an active state.
	1760 bytes	Total FSI internal memory required.

11.6.1 Transmit Watermark

The calculation of the transmit watermark is usually based on the latency of the system bus. The FSI will transfer the data to its internal memory until there is enough data to start the transmit process. Enough data means that there should not be any underrun errors during the transmission. In other words, the data stored inside the internal memory should be sufficient for the period of time between the FSI bus request and when bus grant is received before additional data reaches the internal FIFO. For example, if the bus latency is 50 μs, the FIFO watermark should be at least (50000 ns / 80 ns) = 625 bytes.

11.6.2 Receive Watermark

The receive watermark should be defined to use the external bus most efficiently. When an external bus is arbitrated by the request/grant mechanism, there is some overhead caused by the arbitration. To reduce this overhead per data transfer, the DMA should transfer as many bytes as the system will allow once it gets the bus. The receive watermark should be defined such that the FSI requests the bus only if it has enough data to transfer. However, all the time during a bus latency, the data is received from the network and saved into the internal memory. Therefore, if the bus burst is 128 bytes and the latency is 50 μs, the internal receive data FIFO may grow up to (128 bytes + 50000 ns / 80 ns) =753 bytes.

Again, the rate of transfer from FSI internal memory to the system must approach the rate of frame reception such that internal FSI memory space allocated for reception is not exceeded before the frame has been fully received. Additionally, the average rate of transfer to the system from the FSI must also exceed the rate of reception on the FSI-MAC interface to allow for buffer and descriptor handling overhead.

SECTION 12
TEST OPERATION

The boundary scan (JTAG) and built-in self-test (BIST) operations are discussed in the following subsections.

12.1 JTAG OVERVIEW

The IFDDI implements standard test access port (TAP) and boundary scan capabilities as per the IEEE P1149.1 standard, also known as JTAG (see Figure 12-1). This facility allows for the isolation of the IFDDI for device testing and for board testing.

The test problem for any product constructed from a collection of components can be divided into three goals:

1. To confirm that the components are correctly interconnected.
2. To confirm that each component performs its required function.
3. To confirm that the components interact correctly and that the product performs its intended function.

The first two goals can be achieved by using a boundary scan register architecture. The third goal must be achieved using either a functional automatic test equipment system or a system-level self-test (in this case, with appropriate loopbacks, etc.).

The boundary scan technique involves the inclusion of a shift register stage adjacent to each component pin. These shift register stages, called boundary scan cells, are interconnected to form a shift register chain around the IFDDI I/O pins. The shift path is provided with a serial input (TDI), a serial output (TDO), and an appropriate clock (TCK) The control of the shift action, the parallel output of the shift register to the output pins, or the parallel output of the shift register to the interior of the chip is achieved by an appropriate sequence of logical values on the test mode select (TMS) signal.

NOTE

According to the JTAG standard, TMS, TDI, and TRST have internal pull up resistors.

The JTAG technique of testing implemented in the IFDDI can detect many of the faults that testers currently address without the need for extensive bed-of-nails access and without the need of expensive test equipment, particularly for surface mount components.

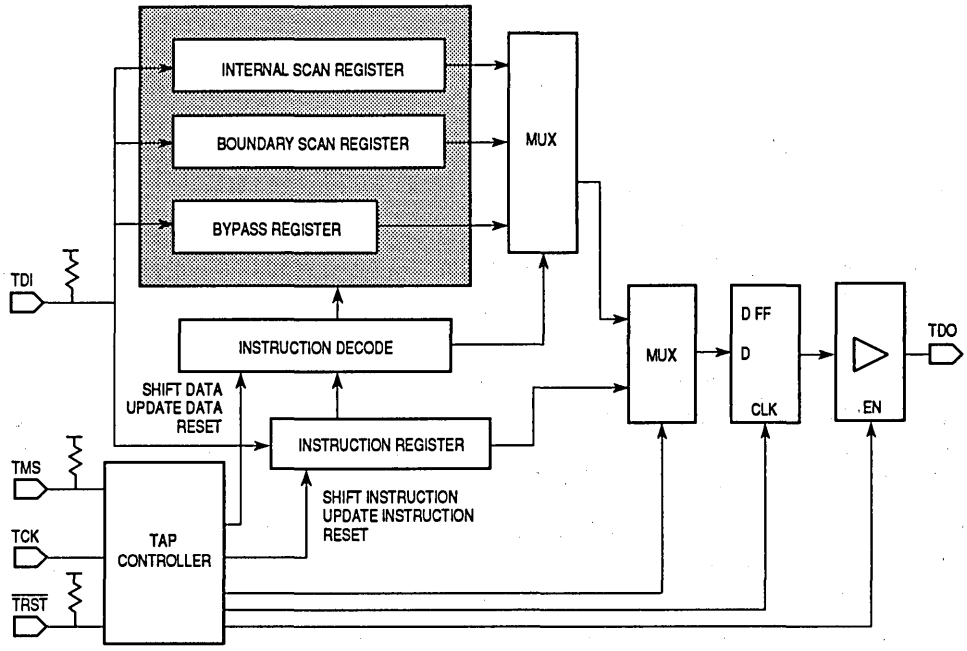


Figure 12-1. JTAG Architecture Model

12.1.1 Functional Blocks

12

The functional blocks of the JTAG architecture are discussed in the following paragraphs.

12.1.1.1 TEST ACCESS PORT (TAP) CONTROLLER. The TAP controller is responsible for interpreting the sequence of logical values on the TMS signal. It is a synchronous state machine that controls the operations of the JTAG logic. The state machine is shown in Figure 12-2; the value shown adjacent to each arc represents the value of the TMS signal sampled on the rising edge of the TCK signal.

12.1.1.2. INSTRUCTION REGISTER. The instruction register is used to select the test to be performed and/or the test data register to be selected. It is loaded serially in the shift-IR state. The loaded data is transferred to the parallel output and interpreted as an instruction when the TAP controller is in the update-IR state. When the TAP controller is in the test logic reset state, the instruction register is set to select the bypass register.

The instruction register is a 3-bit register, I2–I0. I0 is the first bit to be output to TDO when the TAP controller is in the shift-IR state. When the TAP controller is in the capture-IR state, the instruction register is loaded in parallel with the logical value I2–I0 = 001.

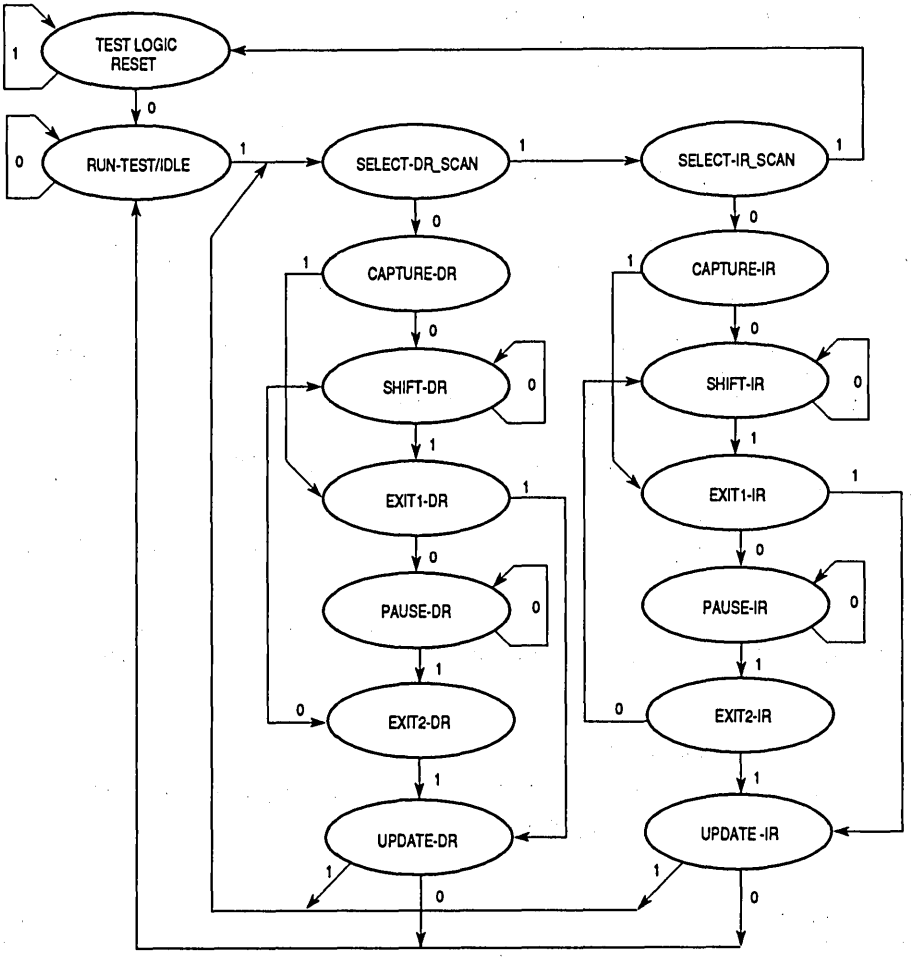


Figure 12-2. TAP Controller State Machine

The instruction code is given to the instruction register with the rightmost bit being the closest to the TDO output:

00X	EXTEST Instruction
010	SAMPLE/PRELOAD Instruction
011	CLAMP Instruction
1X0	INTSCAN Instruction (Private Instruction)
1X1	BYPASS Instruction

12.1.1.3 BOUNDARY SCAN REGISTER. The boundary scan register can be viewed as a parallel-in, parallel-out shift register with some additional functionality— e.g., the ability to propagate data directly from the parallel input to the parallel output without clocking.

The boundary scan register allows testing of circuitry external to the IFDDI. It also permits the signals flowing through the system pins to be sampled and examined without interfering with the operation of the FSI.

The boundary scan register includes all the pins of the IFDDI (except the five pins that belong to the JTAG interface), and eight additional cells that determine the direction and driving state of some of the pins.

12.1.1.4 BYPASS REGISTER. The bypass register provides a short-circuit route for test data in the data register scanning cycle. Use of the bypass register can speed access to test data registers in other components on a board-level test data path.

The bypass register consists of a single shift register stage, which is connected between TDI and TDO when the BYPASS instruction is selected.

When the TAP controller is in the capture-DR state, the bypass register is loaded with zero.

12.1.1.5 INTERNAL SCAN REGISTER. The internal scan register consists of a 139-stage shift register that is connected between TDI and TDO when the INTSCAN private instruction is active.

12.1.2 JTAG Instruction Support

The IFDDI supports the BYPASS, SAMPLE/PRELOAD, CLAMP, and EXTEST public instructions as well as the INTSCAN private instruction:

BYPASS

The operation of the JTAG block has no effect on the operation of the FSI. The BYPASS register is selected to be connected for serial access between TDI and TDO during the appropriate state machine states.

SAMPLE/PRELOAD

The boundary scan register is selected to be connected between TDI and TDO. The FSI continues its normal operation and the signals entering and leaving the FSI are sampled without affecting circuit operation.

CLAMP

The BYPASS register is selected to be connected between TDI and TDO. The cells at output pins are used to apply test stimuli while those at input pins capture test results.

EXTEST

The boundary scan register is selected to be connected between TDI and TDO. The cells at output pins are used to apply test stimuli while those at input pins capture test results. In the capture-DR state, all signals received at the input pins are loaded into the boundary scan register.

INTSCAN

The internal scan register is selected to be connected between TDI and TDO.

12.1.3 Boundary Scan Control

The following signals are internal control signals that are included in the scan chain; their function is noted in the following description.

A_IEN is the direction control of ADATA31–ADATA0 and APRTY3–APRTY0 lines for the EXTEST and CLAMP instructions where:

A_IEN = 1—The I/O buffers of the lines mentioned above are configured as inputs.

A_IEN = 0—The I/O buffers of the lines mentioned above are configured as outputs.

A_OEN is the active control of the ADATA31–ADATA0 and APRTY3–APRTY0 lines for the EXTEST and CLAMP instructions where:

A_OEN = 1—The output buffers of the lines mentioned above are in active state.

A_OEN = 0—The output buffers of the lines mentioned above are in three-state.

B_IEN is the direction control of BDATA31–BDATA0 and BPRTY3–BPRTY0 lines for the EXTEST and CLAMP instructions where:

B_IEN = 1—The I/O buffers of the lines mentioned above are configured as inputs.

B_IEN = 0—The I/O buffers of the lines mentioned above are configured as outputs.

B_OEN is the active control of the BDATA31–BDATA0 and BPRTY3–BPRTY0 lines for the EXTEST and CLAMP instructions where:

B_OEN = 1—The output buffers of the lines mentioned above are in active state.

B_OEN = 0—The output buffers of the lines mentioned above are in three-state.

The next direction lines are used for the two configurations of the chip—IFDDI mode and FSI mode. All the signal name that are in parentheses are FSI mode names.

LDA_DIR is the direction control of the LDADDR/$\overline{TR}$_$\overline{BR}$_$\overline{FWD}$ (NC) line for the EXTEST and CLAMP instructions where:

LDA_DIR = 1—The I/O buffer of the LDADDR/$\overline{TR}$_$\overline{BR}$_$\overline{FWD}$ line is configured as an input (FSI Mode–NC, IFDDI mode–$\overline{TR}$_$\overline{BR}$_$\overline{FWD}$).

LDA_DIR = 0—The I/O buffer of the LDADDR/$\overline{TR}$_$\overline{BR}$_$\overline{FWD}$ line is configured as an output (IFDDI mode–LDADDR).

DIR1 is the direction control of $\overline{\text{CAMINT}}$ (GND), $\overline{\text{LOOPBACK}}$ (ADDIN), and TDATA4–TDATA0 (PHDAT3–PHDAT0, TABORT) for the EXTEST and CLAMP instructions where:

DIR1 = 0—The I/O buffers of the lines mentioned above are configured as outputs (IFDDI mode).

DIR1 = 1—The I/O buffers of the lines mentioned above are configured as inputs (FSI mode).

DIR2 is the direction control of ACNTL4 (MATCH) for the EXTEST and CLAMP instructions where:

DIR2 = 1—The I/O buffer of ACNTL4 is configured as an input (IFDDI mode).

DIR2 = 0—The I/O buffer of MATCH is configured as an output (FSI mode).

DIR3 is the direction control of DA (NC) for the EXTEST and CLAMP instructions where:

DIR3 = 1—The I/O buffer of NC is configured as an input (FSI mode).

DIR3 = 0—The I/O buffer of DA is configured as an output (IFDDI mode).

12.1.4 Boundary Scan Register in IFDDI Mode

The boundary scan register path from TDI to TDO *in IFDDI mode* is shown in Table 12-1:

Table 12-1. Boundary Scan Register Path (TDI–TDO)

Pin	Name	I/O	Type	Pin	Name	I/O	Type	Pin	Name	I/O	Type
1	SD	IN	TTL	2	RSCLK	IN	TTL	3	PRCDAT9	IN	CMOS
4	PRCDAT8	IN	CMOS	5	TDATA0	OUT	CMOS	6	TDATA1	OUT	CMOS
7	TDATA2	OUT	CMOS	8	TDATA3	OUT	CMOS	9	TDATA4	OUT	CMOS
10	LOOPBACK	OUT	CMOS	11	MATCH	IN	TTL	12	ACNTL5	IN	TTL
13	LDA_DIR	DRC	Int	14	LDADDR	I/O	TTL	15	MRCDAT7	OUT	CMOS
16	MRCDAT6	OUT	CMOS	17	MRCDAT5	OUT	CMOS	18	MRCDAT4	OUT	CMOS
19	MRCDAT3	OUT	CMOS	20	MRCDAT2	OUT	CMOS	21	MRCDAT1	OUT	CMOS
22	DIR3	DRC	Int	23	MRCDAT0	OUT	CMOS	24	DIR2	DRC	Int
25	PRCPAR	OUT	CMOS	26	MRCDAT8	OUT	CMOS	27	MRCDAT9	OUT	CMOS
28	FOTOFF	OUT	CMOS	29	ACNTL4	IN	TTL	30	DA	OUT	CMOS
31	FSICLK	IN	TTL	32	RDATA4	IN	TTL	33	RDATA3	IN	TTL
34	RDATA2	IN	TTL	35	RDATA1	IN	TTL	36	RDATA0	IN	TTL
37	REJECT	IN	TTL	38	PRCDAT7	IN	CMOS	39	PRCDAT6	IN	CMOS
40	PRCDAT5	IN	CMOS	41	PRCDAT4	IN	CMOS	42	PRCDAT3	IN	CMOS
43	PRCDAT2	IN	CMOS	44	PRCDAT1	IN	CMOS	45	PRCDAT0	IN	CMOS
46	MRCPAR	IN	CMOS	47	RESET	IN	TTL	48	BYTCLK	IN	TTL
49	SYMCLK	IN	TTL	50	BCNTL3	IN	TTL	51	BCNTL2	IN	TTL
52	BCNTL1	IN	TTL	53	BCNTL0	IN	TTL	54	BRW	IN	TTL

Table 12-1. Boundary Scan Register Path (TDI–TDO) (Continued)

Pin	Name	I/O	Type	Pin	Name	I/O	Type	Pin	Name	I/O	Type
55	ACNTL3	IN	TTL	56	ACNTL2	IN	TTL	57	ACNTL1	IN	TTL
58	ACNTL0	IN	TTL	59	ARW	IN	TTL	60	ADATA31	I/O	TTL
61	ADATA30	I/O	TTL	62	ADATA29	I/O	TTL	63	ACNTL8	IN	TTL
64	ADATA28	I/O	TTL	65	ADATA27	I/O	TTL	66	ADATA26	I/O	TTL
67	ADATA25	I/O	TTL	68	ADATA24	I/O	TTL	69	ADATA23	I/O	TTL
70	ADATA22	I/O	TTL	71	ADATA21	I/O	TTL	72	ADATA20	I/O	TTL
73	ADATA19	I/O	TTL	74	ADATA18	I/O	TTL	75	ADATA17	I/O	TTL
76	ADATA16	I/O	TTL	77	ADATA15	I/O	TTL	78	ADATA14	I/O	TTL
79	ADATA13	I/O	TTL	80	ADATA12	I/O	TTL	81	ADATA11	I/O	TTL
82	ACNTL6	IN	TTL	83	ADATA10	I/O	TTL	84	ADATA9	I/O	TTL
85	ADATA8	I/O	TTL	86	ADATA7	I/O	TTL	87	ADATA6	I/O	TTL
88	ADATA5	I/O	TTL	89	ADATA4	I/O	TTL	90	ADATA3	I/O	TTL
91	ADATA2	I/O	TTL	92	ADATA1	I/O	TTL	93	ADATA0	I/O	TTL
94	APRTY3	I/O	TTL	95	APRTY2	I/O	TTL	96	APRTY1	I/O	TTL
97	APRTY0	I/O	TTL	98	AREQ1	OUT	TTL	99	AREQ2	OUT	TTL
100	A_IEN		Int	101	A_OEN	TSC	Int	102	AREQ0	OUT	TTL
103	AREQ3	OUT	TTL	104	ACS1	IN	TTL	105	ACS0	IN	TTL
106	BCS0	IN	TTL	107	BCS1	IN	TTL	108	BREQ3	OUT	TTL
109	BREQ0	OUT	TTL	110	B_OEN	TSC	Int	111	B_IEN	DRC	Int
112	BREQ2	OUT	TTL	113	BREQ1	OUT	TTL	114	BPRTY0	I/O	TTL
115	BPRTY1	I/O	TTL	116	BPRTY2	I/O	TTL	117	BPRTY3	I/O	TTL
118	BDATA0	I/O	TTL	119	BDATA1	I/O	TTL	120	BDATA2	I/O	TTL
121	BDATA3	I/O	TTL	122	BDATA4	I/O	TTL	123	BDATA5	I/O	TTL
124	BDATA6	I/O	TTL	125	BDATA7	I/O	TTL	126	BDATA8	I/O	TTL
127	BDATA9	I/O	TTL	128	BDATA10	I/O	TTL	129	ACNTL7	IN	TTL
130	BDATA11	I/O	TTL	131	BDATA12	I/O	TTL	132	BDATA13	I/O	TTL
133	BDATA14	I/O	TTL	134	BDATA15	I/O	TTL	135	BDATA16	I/O	TTL
136	BDATA17	I/O	TTL	137	BDATA18	I/O	TTL	138	BDATA19	I/O	TTL
139	BDATA20	I/O	TTL	140	BDATA21	I/O	TTL	141	BDATA22	I/O	TTL
142	BDATA23	I/O	TTL	143	BDATA24	I/O	TTL	144	BDATA25	I/O	TTL
145	BDATA26	I/O	TTL	146	BDATA27	I/O	TTL	147	BDATA28	I/O	TTL
148	CAMINT	OUT	O_D	149	BDATA29	I/O	TTL	150	BDATA30	I/O	TTL
151	BDATA31	I/O	TTL	152	AINT	OUT	O_D	153	DIR1	DRC	Int
154	BINT	OUT	O_D	—	—	—	—	—	—	—	—

12.2 BUILT-IN SELF-TEST OVERVIEW

The test features remain passive during normal chip operation. The BIST feature does not have to be run to provide correct operation of the core. The test is included to provide the user a means of fault-isolation testing for the application.

12.2.1 ELM BIST Operation

BIST tests the ELM by circulating pseudo-random data throughout the core. The various subcircuits within the core are observed as they respond to the data, and a signature based upon their behavior is generated. This signature may be checked against a correct signature to verify functioning of the core. A fault in the ELM (within the coverage of BIST) causes a different signature to be generated.

BIST is activated by setting the RUN_BIST bit in ELM_CNTRL_A. Upon activation, the data path linear feedback shift register and signature generator are enabled, and the test proceeds. The procedure for running the BIST is as follows:

- Perform a power-up reset.
- Set the EB_LOC_LOOP bit in ELM_CNTRL_A.
- Read the violation symbol counter register.
- Read the link error symbol counter register.
- Set the RUN_ BIST bit in ELM_CNTRL_A.
- Get an interrupt when BIST has finished running.
- Read the BIST signature register; if the test was successful, it will have a value of 5B6B.
- Perform a power-up reset.
- Set ELM registers to desired value for operational mode.

When BIST has completed, the signature is frozen and may be read from the CAMEL. The test concludes when a value of zero is reached in the linear feedback shift register. Using a 16-bit linear feedback shift register clocked by the 80-ns BYTCLK takes approximately 5 ms to circulate about 62820 test patterns through the core. An interrupt to the node processor/system interface after RUN_BIST has been set signifies the completion of the ELM BIST. This interrupt is cleared by clearing the RUN_BIST bit in ELM control register A (*not* by reading the interrupt event register). BIST is aborted if the RUN_BIST bit is cleared before the test completes.

12.2.2 MAC BIST Operation

BIST tests the MAC by circulating pseudo-random data throughout the core. The various subcircuits within the core are observed as they respond to the data, and a signature based upon their behavior is generated. This signature may be checked against a correct signature to verify the functioning of the core. A fault in the MAC (within the coverage of BIST) causes a different signature to be generated.

BIST is activated by setting the RUN_BIST bit in MAC_CNTRL_A. Upon activation, the data path linear feedback shift register and signature generator are enabled, and the test proceeds. The procedure for running the BIST is as follows:

- Perform a power-up reset.
- Write the MAC_CNTRL_B, MSA, MLA_A, MLA_B, MLA_C, T_REQ, TVX_VALUE, TRT_TIMERA, TRT_TIMERB, and the MAC_CNTRL_A registers with the values shown in Table 12-2. The final write to the MAC_CNTRL_A will activate the BIST.
- Wait for the $\overline{\text{CAMINT}}$ interrupt indicating BIST is complete.
- Read the BIST signature.
- Perform a power-up reset.

When BIST has completed, the signature is frozen and may be read from the CAMEL. Using a 16-bit linear feedback shift register clocked by the 80-ns BYTCLK takes approximately 5.24 ms to circulate 65535 test patterns through the core. The actual signature depends on the values of each of the writable registers in Table 12-2. SYMCLK, MRCDATx, and TXCTLx have no effect during BIST.

Table 12-2. BIST Register Values

Register	Address	Test 1 Value	Test 2 Value	Test 3 Value
MAC_CNTRL_B	41	0200	82FF	13FF
MSA	50	7777	8888	137F
MLA_A	51	AAAA	5555	37F1
MLA_B	52	BBBB	4444	7F13
MLA_C	53	CCCC	3333	F137
T_REQ	54	FF00	00FF	FEC8
TVX _VALUE/TRT_TIMERA, TRT_TIMERB	55	AACC	5533	F0FC
MAC_CNTRL_A	40	88C4	F127	A414
Run BYTCLK until BIST_Done is indicated by $\overline{\text{CAMINT}}$ assertion				
BIST Signature Register	6A	9FA2	9324	8445

NOTE: All values are hexadecimal.

SECTION 13
ELECTRICAL CHARACTERISTICS

The AC specifications presented consist of output delays, input setup and hold times, and signal skew times. All signals are specified relative to an appropriate edge of the clock(s) and possibly to one or more other signals.

13.1 MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	–0.3 to +7.0	V
Input Voltage	V_{in}	–0.3 to +7.0	V
Operating Temperature Range MC68339	T_A	0 to 70	°C
Storage Temperature Range	T_{stg}	–55 to +150	°C

This device contains protective circuitry against damage due to high static voltages or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either GND or V_{CC}).

13.2 THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance for PGA	θ_{JA}	20	°C/W
	θ_{JC}	TBD	°C/W
Thermal Resistance for CQFP	θ_{JA}	TBD	°C/W
	θ_{JC}	TBD	°C/W

13

13.3 POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \bullet \theta_{JA}) \quad (1)$$

where:

T_A = Ambient Temperature, °C
θ_{JA} = Package Thermal Resistance, Junction-to-Ambient, °C/W
P_D = $P_{INT} + P_{I/O}$
P_{INT} = $I_{CC} \times V_{CC}$, Watts—Chip Internal Power
$P_{I/O}$ = Power Dissipation on Input and Output Pins—User Determined

For most applications, $P_{I/O} < P_{INT}$ and can be neglected.

The following is an approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected):

$$P_D = K + (T_J + 273°C) \quad (2)$$

Solving Equations (1) and (2) for K gives:

$$K = P_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D^2 \quad (3)$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at thermal equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

The total thermal resistance of a package (θ_{JA}) can be separated into two components, θ_{JC} and θ_{CA}, representing the barrier to heat flow from the semiconductor junction to the package (case) surface (θ_{JC}) and from the case to the outside ambient air (θ_{CA}). These terms are related by the equation:

$$\theta_{JA} = \theta_{JC} + \theta_{CA} \quad (4)$$

θ_{JC} is device related and cannot be influenced by the user. However, θ_{CA} is user dependent and can be minimized by such thermal management techniques as heat sinks, ambient air cooling, and thermal convection. Thus, good thermal management on the part of the user can significantly reduce θ_{CA} so that θ_{JA} approximately equals θ_{JC}. Substitution of θ_{JC} for θ_{JA} in equation (1) results in a lower semiconductor junction temperature.

Values for thermal resistance presented in this document, unless estimated, were derived using the procedure described in Motorola Reliability Report 7843, "Thermal Resistance Measurement Method for MC68XX Microcomponent Devices," and are provided for design purposes only. Thermal measurements are complex and dependent on procedure and setup. User derived values for thermal resistance may differ.

13.4 FSI DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 5.0 V_{dc} ± 10%, GND = 0 V_{dc}, T_A = 0°C to 70°C)

Symbol	Characteristics	Min	Typical	Max	Unit
V_{IH}	Input High Voltage (TTL Pins)	2.0		V_{CC}	V
V_{IL}	Input Low Voltage (TTL Pins)	GND – 0.3		0.8	V
V_{IHC}	Input High Voltage (CMOS Pins)	0.7 • V_{CC}		V_{CC}	V
V_{ILC}	Input Low Voltage (CMOS Pins)	GND – 0.3		0.2 • V_{CC}	V
I_{in}	Input Leakage Current	—		20	µA
C_{in}	Input Capacitance (All Pins)	—		15	pF
I_{tsi}	Three State Leakage Current	—		20	µA
V_{OH}	Output High Voltage (I_{OH} = 400 µA)	V_{CC} – 1.0		—	V
V_{OL}	Output Low Voltage (I_{OL}= 5.3 mA) TTL Pins: ADATAx, APRTYx, BDATAx, BPRTYx (I_{OL} = 3.2 mA) All Other TTL Pins (I_{OL} = 1.9 mA) CMOS Pins	— — —		0.5 0.5 0.3 • V_{CC}	V
P_D	Power Dissipation	@ V_{CC}	1.5 @ 5.5V	@ 5.5V	W
I_{DD}	Operating Current in FSI mode	@ V_{CC}	320 @ 5.5V[2]	@ 5.5V[3]	mA
I_{DD}	Operating Current	@ V_{CC}	320 @ 5.5V[2]	@ 5.5V[3]	mA

NOTES:
1. Input capacitance is periodically sampled rather than 100% tested
2. Typical operating current measured on test board
3. Maximum operating current not to exceed 500mA

13.5 CLOCKS (see Figure 13-1)

Num	Characteristics	Min	Max	Unit
1	BYTCLK Cycle Time	80	—	ns
2	SYMCLK Cycle Time	40	—	ns
3	SYMCLK Pulse Width Low	17	23	ns
4	SYMCLK Pulse Width High	17	23	ns
5	SYMCLK to BYTCLK High Skew	0	15	ns
6	SYMCLK to BYTCLK Low Skew	0	15	ns
7	SYMCLK Rise or Fall Time[3,4]	—	5	ns
11	RSCLK Cycle Time	40	—	ns
12	RSCLK Pulse Width Low	15	25	ns
13	RSCLK Pulse Width High	15	25	ns
14	RSCLK Rise/Fall Time[3]	—	4	ns
15	FSICLK Cycle Time	40	—	ns
16	FSICLK Pulse Width Low	17	23	ns
17	FSICLK Pulse Width High	17	23	ns
18	FSICLK Rise or Fall Time[3,4]	—	5	ns

NOTES:
1. If FSICLK is used in non-bypass mode, the frequency of SYMCLK must be < FSICLK.
2. In bypass mode, FSICLK = SYMCLK where FSICLK can be grounded or shorted to SYMCLK.
3. Note that rise and fall times are not measured
4. This timing is for low frequencies, for 25MHz, it is limited to 3ns.

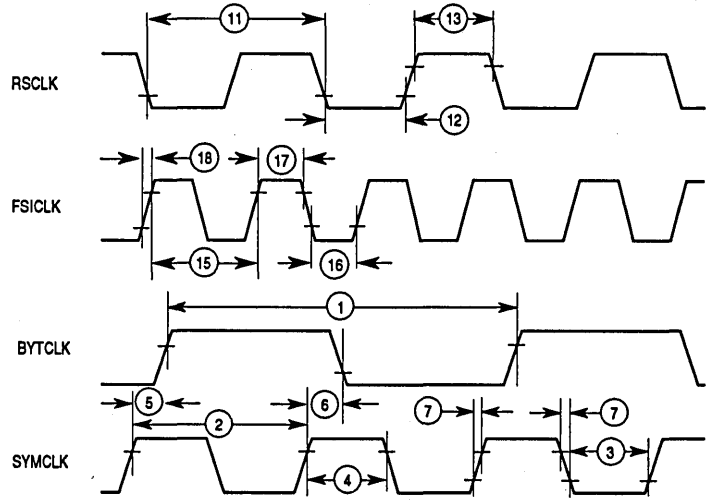


Figure 13-1. FSI Clocks and Interface Timing

13.6 PIPLINE MODE READ AND WRITE (see Figures 13-6 to 13-11)

Pipeline Mode				
Num	**Characteristics**	**Min**	**Max**	**Unit**
50	READY Setup Time to Chip Select Negated	4	—	ns
51	READY Hold Time to Chip Select Negated	0	—	ns
52	Output Enable Asserted to Data_Out Active	3	—	ns
53	Output Enable Asserted to Data_Out Valid	—	12	ns
54	Output Enable Negated to Data_Out Three-State	0	13	ns
55	Chip Select Negated to Data_Out Valid (OE = 1)	—	14	ns
56	Chip Select Negated to Data_Out Invalid	0	—	ns
57	Chip Select Asserted to Last Chip Select Negated of Direct CAMEL Access[1]	—	C	ns
70	CS0 Cycle Time	40	—	ns
71	CS0 Pulse Width Low	18	22	ns
72	CS0 Pulse Width High	18	22	ns
73	CS0 Rise/Fall Time[2]	—	5	ns

NOTE:
1. This timing depends upon which access is used:
   a. Only CAMEL direct access is used: C = 8 x SYMCLK period.
   b. When CAMEL direct access is used with FCR and CMR: C= 10 x SYMCLK period.
      @25 MHz: C = 320 ns / 400 ns.
2. This timing is valid for low frequencies only. At 25MHz the Rise/Fall time is limited to 2ns.

13.7 SYSTEM INTERFACE READ AND WRITE TIMING (see Figures 13-2 to 13-10)

Num	Characteristics	25 MHz Min	25 MHz Max	Unit
	Normal Mode			
21	Chip Select Width Asserted	18	—	ns
22	Chip Select Width Negated	18	—	ns
22A	Chip Select Width Negated for Other Port Address Access (CNTLx = 1000)	10	—	ns
23	Control Valid to Chip Select Asserted (Setup)	12	—	ns
24	Chip Select Asserted to Control Invalid	0	—	ns
25	R/$\overline{W}$ Valid to Chip Select Asserted	10	—	ns
26	Chip Select Asserted to R/$\overline{W}$ Invalid	0	—	ns
27	Data_In Valid to Chip Select Negated	4	—	ns
27A	Data_In Valid to Chip Select Negated with Parity Checking	9	—	ns
28	Chip Select Negated to Data_In Invalid	3	—	ns
30	Chip Select Asserted to Data_Out Valid for Data Accesses	—	15	ns
30A	Chip Select Asserted to Data_Out Valid for Address Accesses	—	20	ns
30B	Chip Select Asserted to Data_Out Valid for Register Accesses	—	24	ns
30C	Chip Select Asserted to Parity_Out Valid for Data Accesses	—	15	ns
30D	Chip Select Asserted to Parity_Out Valid for Address Accesses	—	25	ns
30E	Chip Select Asserted to Parity_Out Valid for Register Accesses	—	29	ns
30F	Chip Select Asserted to CAMEL Direct Access Data_Out[1]	—	A	ns
31	Chip Select Negated to Data_Out Three-State	—	13	ns
31A	Chip Select Negated to Data_Out Invalid	0	—	ns
32	Chip Select Asserted to Request Valid[4]	—	12	ns
32A	Chip Select Asserted to Request Invalid	0	—	ns
33	$\overline{ACSx}$ ($\overline{BCSx}$) Asserted for Data Access (ACNTLx (BCNTLx) = 0010) to $\overline{BCSx}$ ($\overline{ACSx}$) Asserted for Other Port Address Accesses (BCNTLx (ACNTLx) = 1000)	15	—	ns
34	Chip Select Asserted to Data_Out Active[2]	3	—	ns
35	Chip Select Asserted to Data_In Valid for CAMEL Direct Access[3]	—	B	ns
36	Chip Select Negated to Data_In Invalid for CAMEL Direct Access	0	—	ns

NOTES:
1. This timing depends upon which access is used:
   a. Only CAMEL direct access is used: A = 12 ns + 7 x SYMCLK period + T30A or T30D.
   b. When CAMEL direct access is used with FCR or CMR: A = 12 ns + 9 x SYMCLK period + T30A or T30D.
   @25 MHz: A = 292 ns / 372 ns + T30A or T30D.
2. New timing for FSI REVC also.
3. B = SYMCLK period.
4. In case of Port Operation Error, the request timing may be 5ns longer

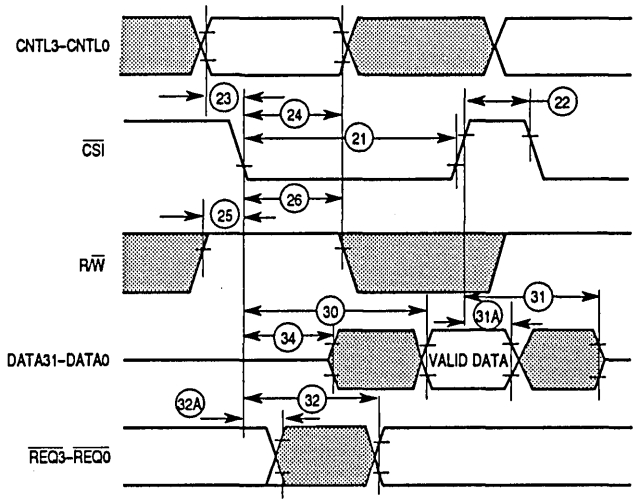


Figure 13-2. FSI Read Timing

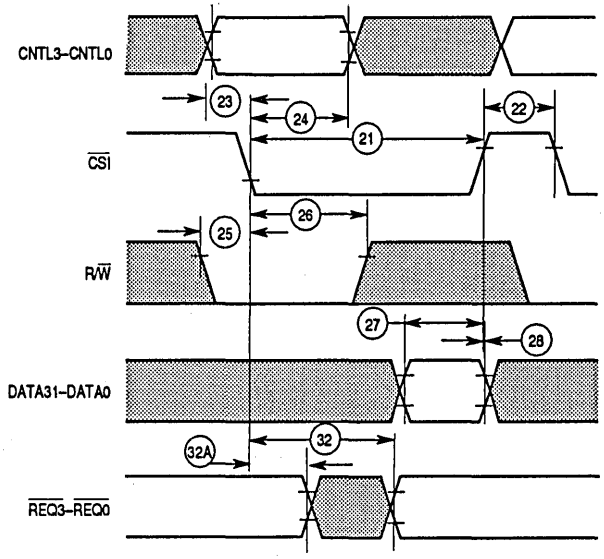


Figure 13-3. FSI Write Timing

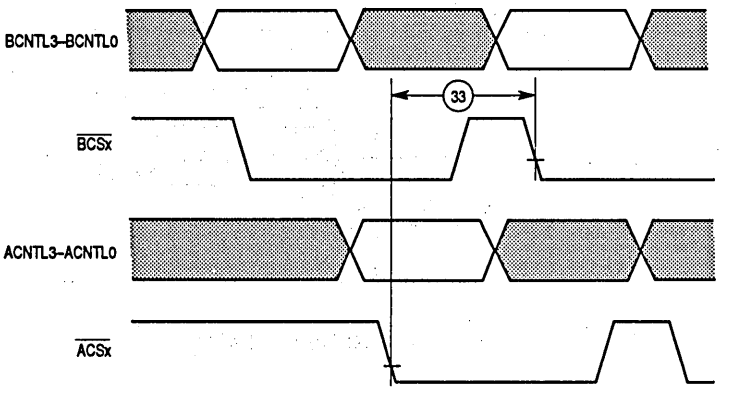


Figure 13-4. FSI Nonmultiplexed Two-Port Timing

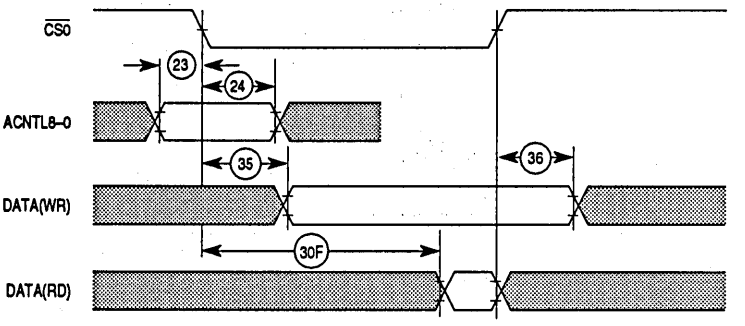


Figure 13-5. CAMEL Direct Access (Normal Mode)

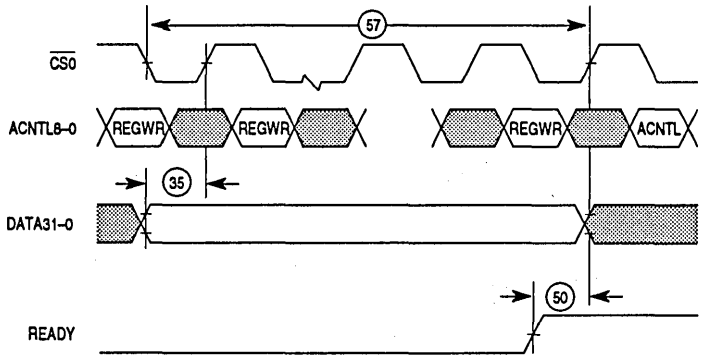
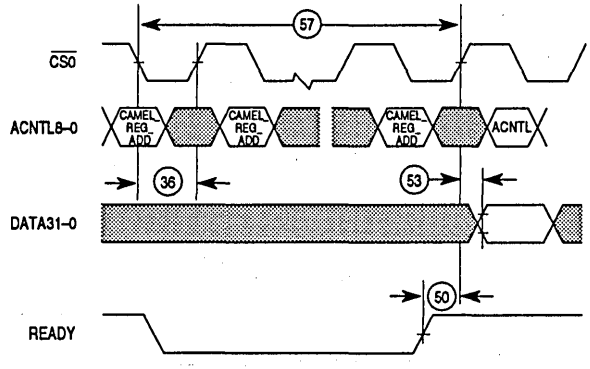


Figure 13-6. CAMEL Direct Access Write Cycle (Pipeline Mode)



NOTES:

1. The CAMEL Direct Access Read Cycle is T57 in length.
   The following requirements must be fufilled:
   a. Ready is low T50 before the rising edge of CS0 at the start of the cycle
   b. Ready is high T50 before the rising edge of CS0 at the end of the cycle

Figure 13-7. CAMEL Direct Access Read Cycle (Pipeline Mode)

13

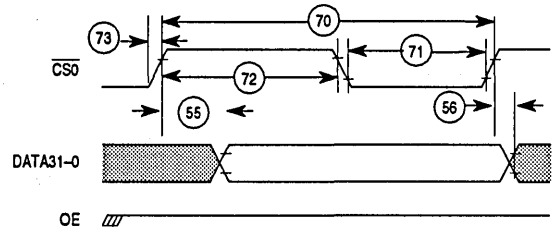


Figure 13-8. Pipeline Mode—FDDI Read

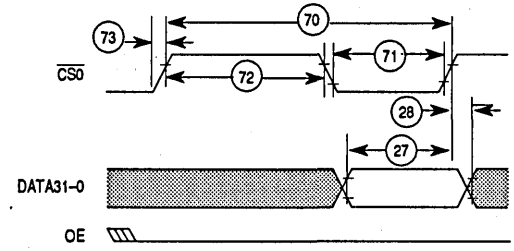


Figure 13-9. Pipeline Mode—FDDI Write

NOTE

OE is low prior to the rising edge of CS0

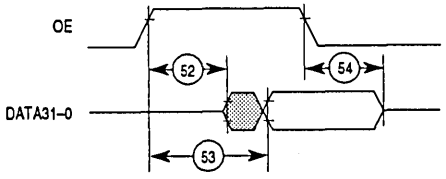


Figure 13-10. Enable Timing

NOTE

Figure 13-10 assumes CS0 is high and OE is clocking the data. If OE and CS0 rise simultaneously, the data will be valid at T55.

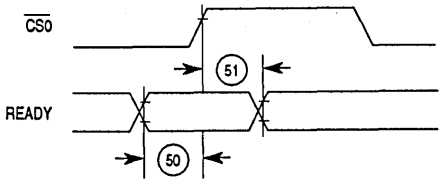


Figure 13-11. Pipeline Mode—READY Input

13.8 EXTERNAL CAM INTERFACE TIMING (see Figure 13-12)

Num	Characteristics	Min	Max	Unit
80	$\overline{\text{MATCH}}$/LDADDR Setup Time[4]	40	—	ns
81	$\overline{\text{MATCH}}$/LDADDR Hold Time[4]	5	—	ns
84	BYTCLK to LDADDR Valid	—	40	ns
85	BYTCLK to LDADDR Invalid	4	—	ns
86	BYTCLK to DA Asserted	—	24	ns
87	BYTCLK to DA Negated	2	—	ns

NOTES:
1. All signals are shown relative to the rising edge of BYTCLK.
2. Only the timing of the DA match indication is given. The SA match indication would need to be provided to the MAC six bytes later with the same relative timing for the proper SA actions to occur.
3. Figure 13-14 shows timing requirements for the CAM interface signal timing. This figure is drawn based on the functional timing required when EXT_DA_MATCH = 0.
4. Use for LDADDR when activated as an input ($\overline{\text{TR\_BR\_FWD}}$).

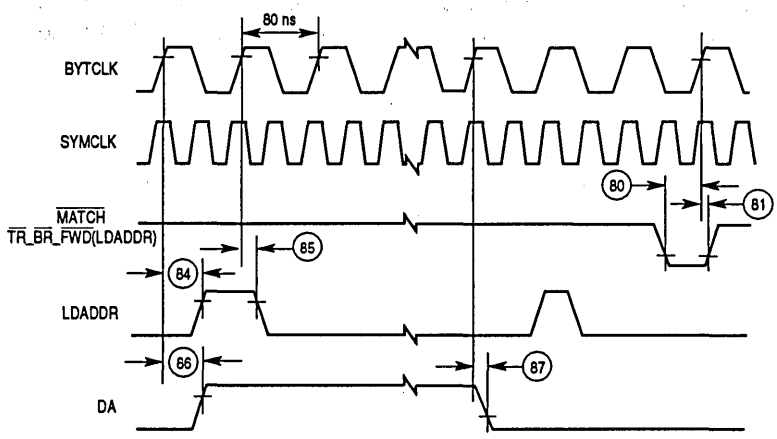


Figure 13-12. CAM Interface Timing

13

13.9 MISCELLANEOUS SIGNALS TIMING (see Figure 13-13)

Num	Characteristics	Min	Max	Unit
8A	$\overline{\text{REJECT}}$ Setup Time Before SYMCLK Falling Edge[1]	13	—	ns
9A	REJECT Hold Time After SYMCLK Falling Edge[1]	4	—	ns
88	$\overline{\text{LOOPBACK}}$, $\overline{\text{FOTOFF}}$[2] invalid	2	—	ns
89	$\overline{\text{LOOPBACK}}$, $\overline{\text{FOTOFF}}$[2] valid	—	25	ns
98	SD Setup Time[2]	5	—	ns
99	SD Hold Time[2]	20	—	ns
102	Time to $\overline{\text{CAMINT}}$ Asserted[3]	—	35	ns
103	Time to $\overline{\text{CAMINT}}$ Tristated[3 4]	—	35	ns

NOTES:
1. Times are relative to the falling edge of SYMCLK.
2. Times are relative to the rising edge of BYTCLK.
3. Times are relative to the rising edge of SYMCLK.
4. This timing depends on the external pull-up value

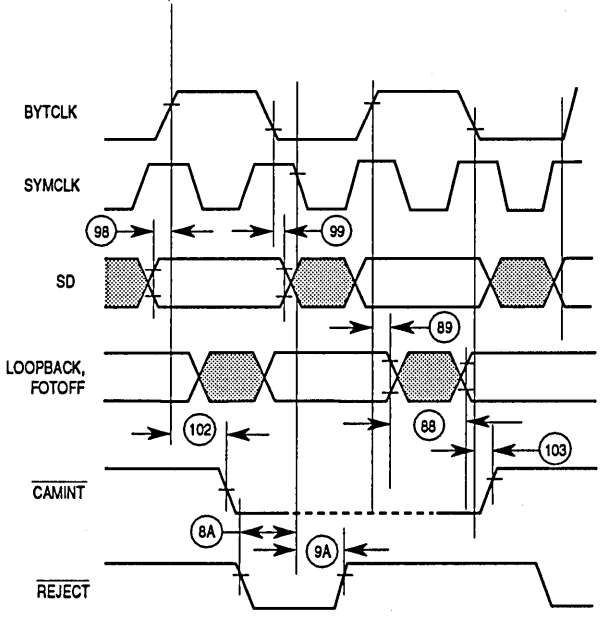
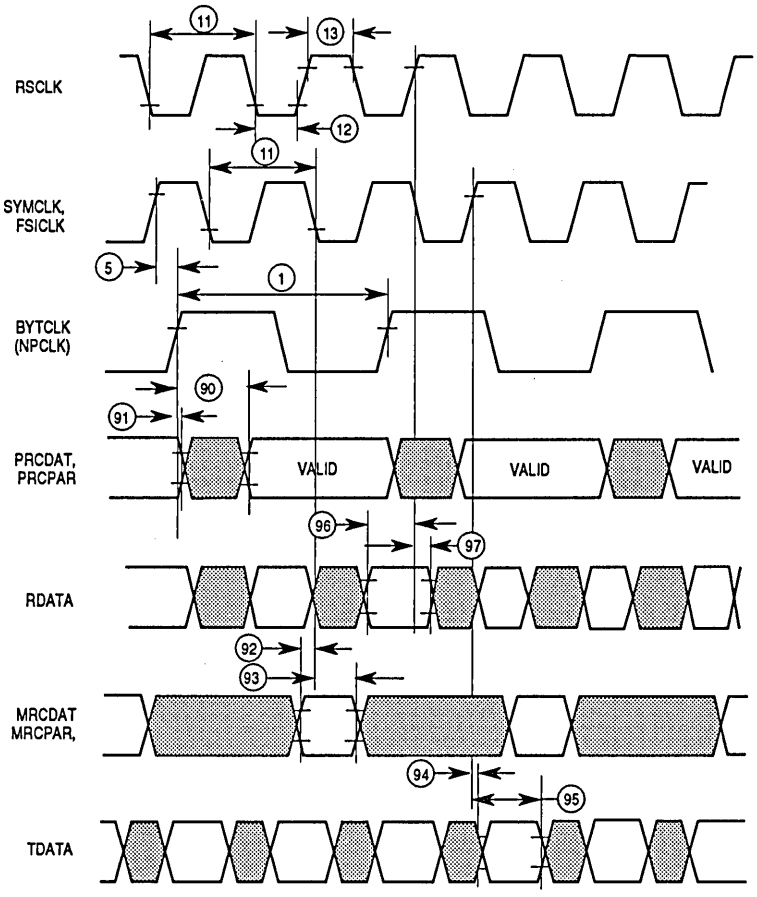


Figure 13-13. Miscellaneous Signals Timing

13.10 ELM CORE DATA I/O PORT TIMING (see Figure 13-14)

Num	Parameter	Min	Max	Unit
90	Time to PRCDATx, PRCPAR Valid[1]	—	30	ns
91	Time to PRCDATx Invalid[1]	2	—	ns
92	MRCDATx, MRCPAR Setup Time[2]	5	—	ns
93	MRCDATx, MRCPAR Hold Time[2]	7	—	ns
94	Time to TDATA Valid[3]	—	18	ns
95	Time to TDATA Invalid[3]	2	—	ns
96	RDATA Setup Time[4]	5	—	ns
97	RDATA Hold Time[4]	8	—	ns

NOTES:
1. Times are relative to the rising edge of BYCLK.
2. Times are relative to the falling edge of SYMCLK.
3. Times are relative to the rising edge of SYMCLK.
4. Times are relative to the rising edge of RSCLK.



NOTE:
1. If FSICLK is used in non-bypass mode, the frequency of SYMCLK must be < FSICLK.
2. In bypass mode, FSICLK = SYMCLK where FSICLK can be grounded or shorted to SYMCLK.

Figure 13-14. ELM Core Data I/O Port Timing

13.11 JTAG TIMING (see Figure 13-15)

Num	Characteristics	Min	Max	Unit
61	TDI Valid to TCK Falling Edge	2	—	ns
62	TMS Valid to TCK Falling Edge	6	—	ns
63	TCK Asserted to TDI Invalid	1	—	ns
64	TCK Asserted to TMS Invalid	0	—	ns
65	TCK Negated to TDO Valid[1]	—	15	ns
66	TCK Negated to Data_Out Valid[1]	—	23	ns

NOTE:
1. For every 10-pF loading capacitance more than 50 pF, add 0.75 ns.

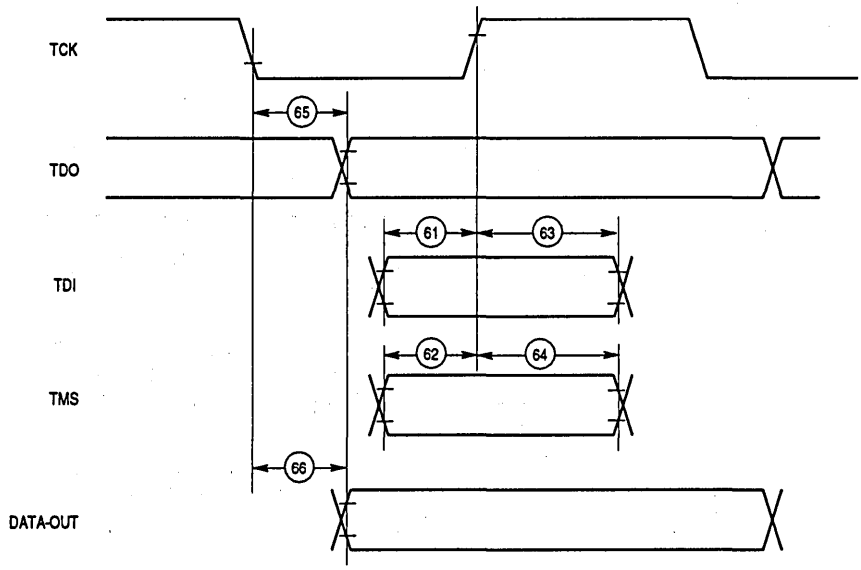


Figure 13-15. JTAG Timing

SECTION 14
ORDERING INFORMATION AND MECHANICAL DATA

This section contains ordering information, pin assignments, and package dimensions for the MC68840 IFDDI.

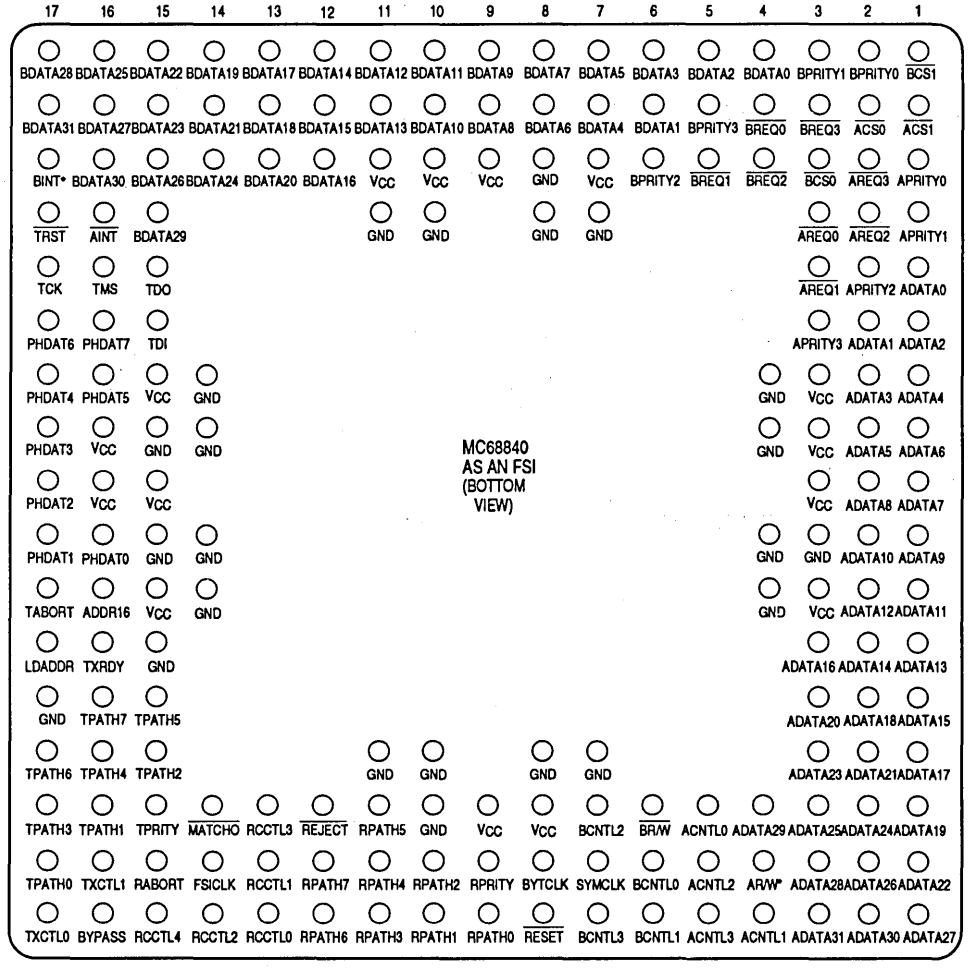
14.1 ORDERING INFORMATION

Package Type	Frequency	Temperature	Order Number
Pin Grid Array (RC Suffix)	25 MHz	0–70°C	MC68840RC
Ceramic Quad Flat Pack (FE Suffix)	25 MHz	0–70°C	MC68840FE

14.2 PIN ASSIGNMENTS

The pin assignments are given for both the PGA and CQFP.

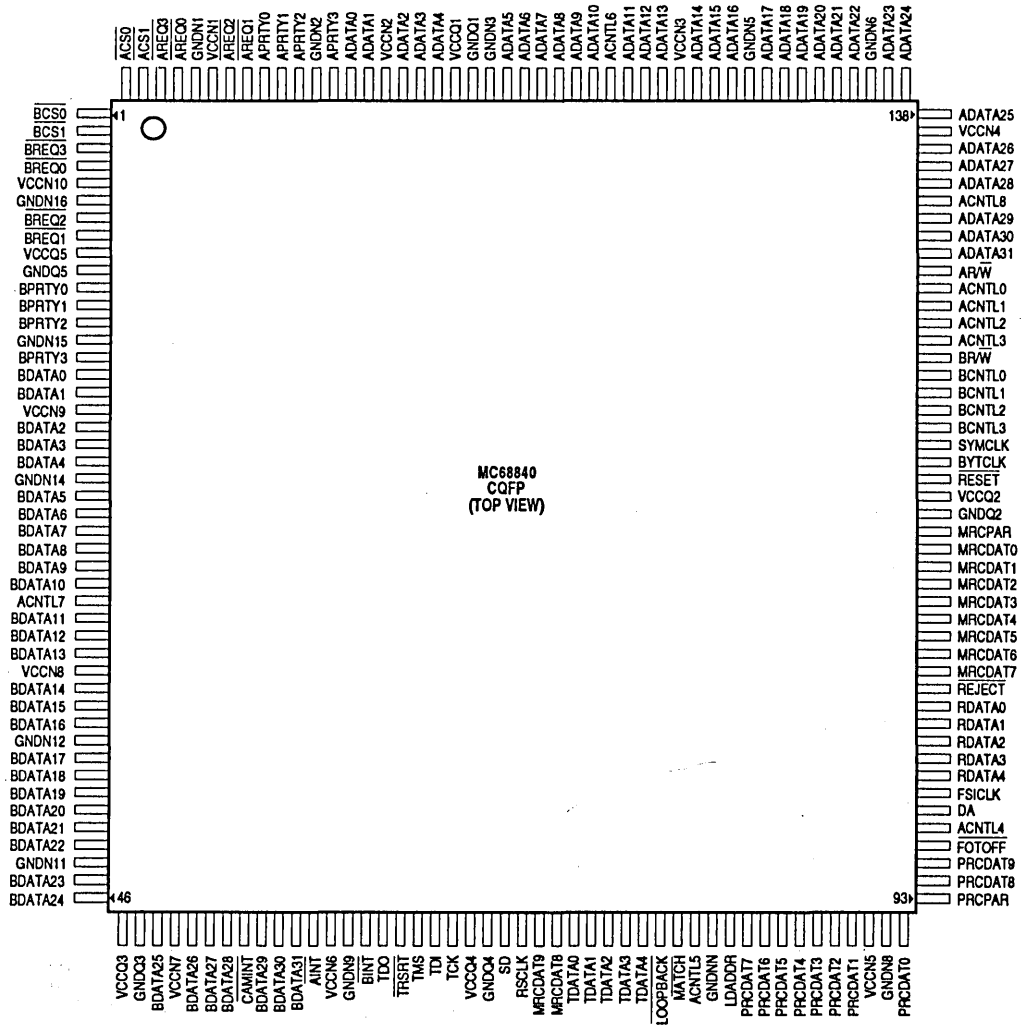
Pin Grid Array (PGA)—IFDDI



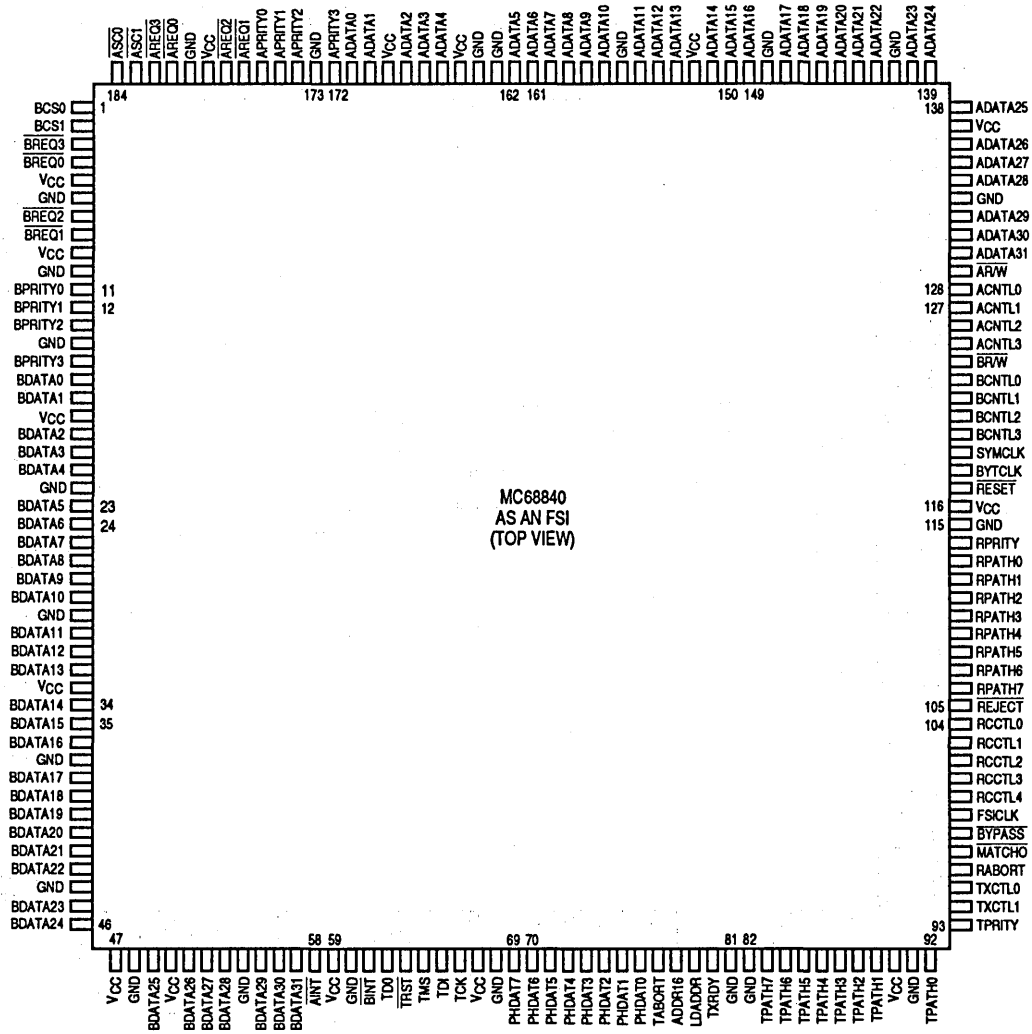
NOTE

Pin J14 is a KEY pin. This pin may be used as a GND pin or broken off.

Ceramic Quad Flat Pack (CQFP) - IFDDI



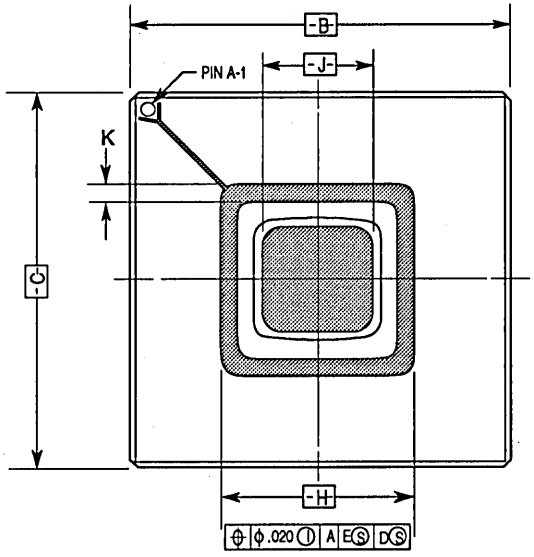
Ceramic Quad Flat Pack (CQFP) - IFDDI as an FSI

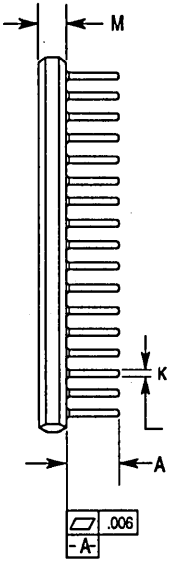


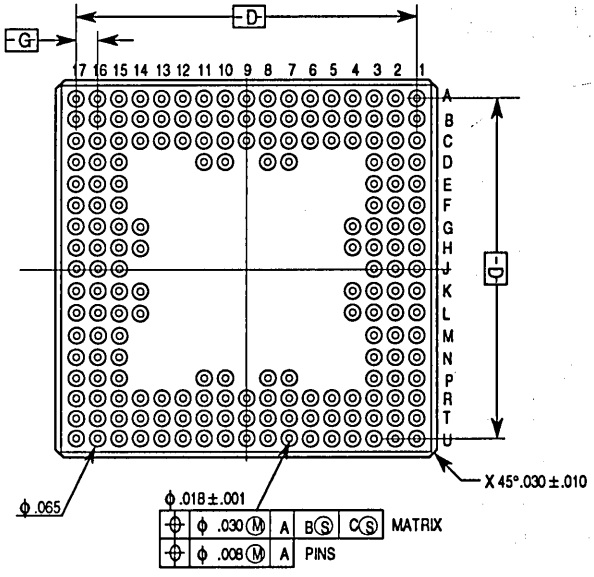
14

14.3 PACKAGE DIMENSIONS

184-PIN PGA
CASE TO BE DETERMINED
(PRELIMINARY)



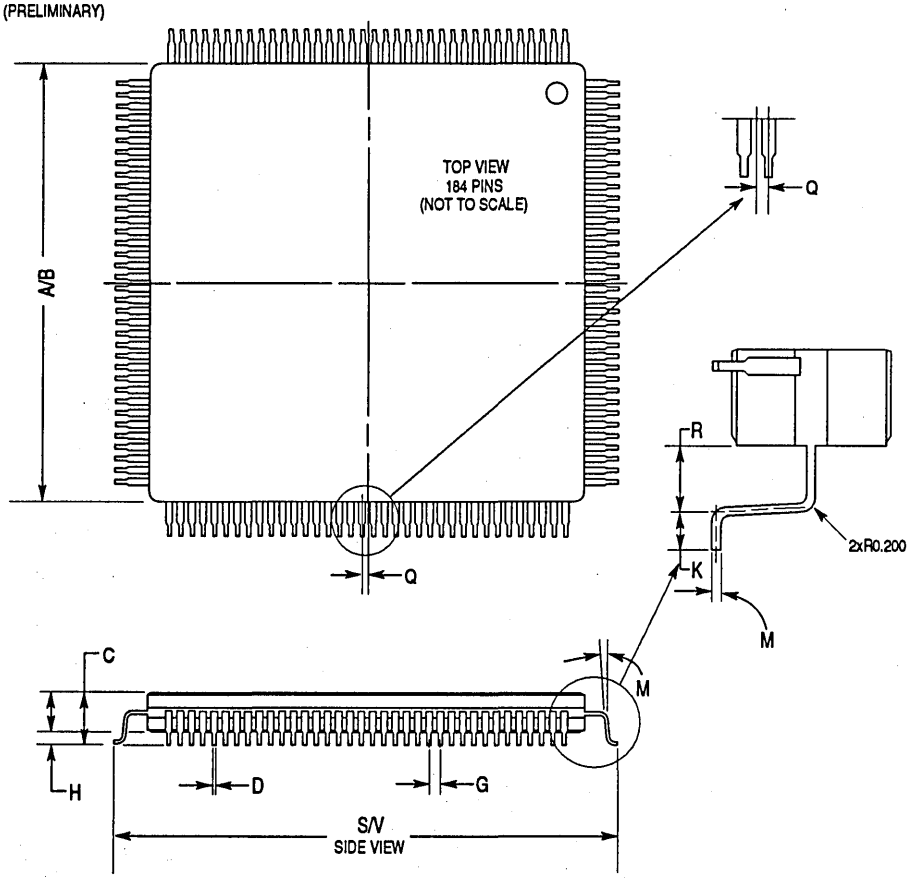




DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A			0.17	0.19
B			1.746	1.774
C			1.746	1.774
D			1.6	1.6
G			.10	.10
H			.95	.97
J			.578	.59
K			0.070	0.070
M			0.09	0.11

14

184 PIN CQFP
CASE TO BE DETERMINED
(PRELIMINARY)



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	30.98	32.85		
B	30.98	32.85		
C	3.27	4.58		
D	0.22	0.41		
G	0.65	—		—
H	0.25	0.88		
J	0.13	0.25		
K	0.65	0.95		
M	0°	8°	0°	8°
Q	0.325	—		—
R	0.50	—		—
S	34.95	35.45		
V	34.95	35.45		

APPENDIX A
SYSTEM CONFIGURATIONS

The FSI provides for a number of different FDDI system configurations. These different configurations allow the user to select the most appropriate configuration for the application. The different configurations are shown in Figures A-1 to A-6. The user chooses among the different configurations based on system latency, functional task division, bus width, and bus format (multiplexed/nonmultiplexed).

The basic configuration shown in Figure A-1 can be used when the FDDI node is either on a separate board from the main system or is on the same bus as the main processor. If it is a separate board, the bus may be a backplane bus, and the latency of the system bus should be short enough to allow the FSI to operate with no local memory. If the bus is the main or a local microprocessor bus, this configuration assumes the system bus can give the FDDI system enough of its bandwidth to handle FDDI traffic. In this case, the bus shown can be the system bus.

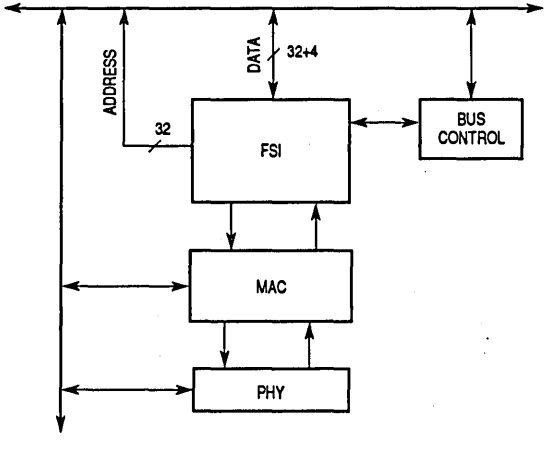


Figure A-1. Basic Configuration

A

The configuration shown in Figure A-2 can be used when the user wants to offload FDDI-specific tasks (such as SMT) and the control of the MAC and PHY chips to a special node processor. These FDDI-specific tasks may include handling SMT and synchronous frames as well as monitoring statistics and network topology. Because these tasks are neither very frequent nor very complex, and because the FSI has the ability to store these frames separately, the node processor may be simple and inexpensive. In this configuration, the data frames are loaded directly from the system memory.

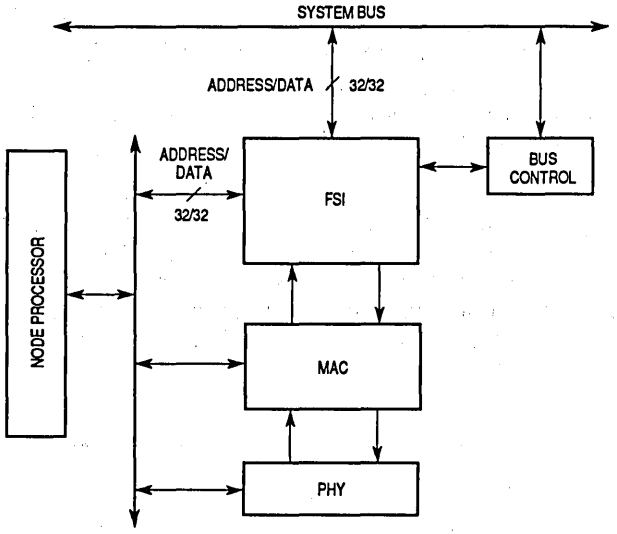


Figure A-2. Basic Configuration with Separate Node Processor

In the configuration shown in Figure A-3, processor bus latency may be insufficient to handle the FDDI network traffic without more buffering than that provided in the FSI. Local memory may be added to handle this additional buffering requirement. The second port may be used for the host to directly access the FSI to give it commands, to read its status, and to handle the transmission of immediate frames. With this configuration, the host processor must handle the control of the MAC and PHY and the operation of SMT.

A

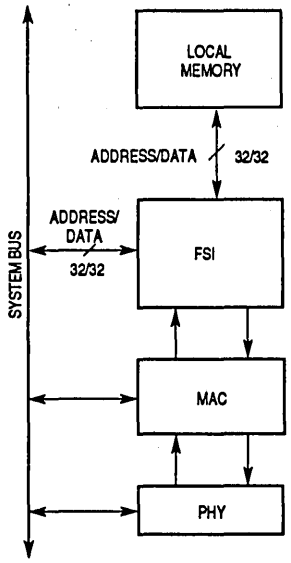


Figure A-3. Basic Configuration with Local Memory

In the configuration shown in Figure A-4, local memory has been added because the latency of the host processor bus cannot handle the FDDI network traffic without more buffering than that provided in the FSI. The node processor may be used to offload FDDI-specific tasks and control the MAC and PHY chips. These FDDI-specific tasks may include the handling of SMT and synchronous frames as well as the monitoring of statistics and network topology.

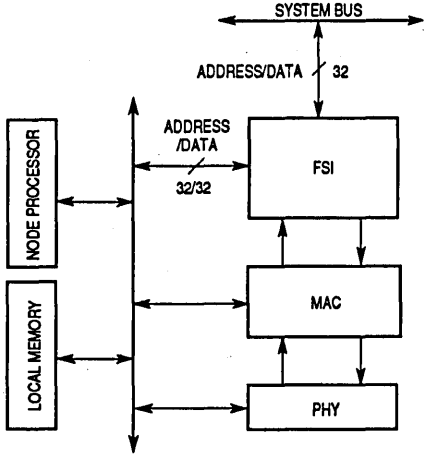


Figure A-4. Local Memory with Separate Node Processor

In the configuration shown in Figure A-5, the FSI is a slave to the host processor. This means that either the system has its own DMA or the host handles the transfer of data over its bus and the FSI's DMA is not used. The FSI connects directly to the host processor bus, which must be fast enough to fill and empty the FIFOs of the FSI. The 64 bits of data enable the FSI to be connected to specialized system backplanes and new-generation, high-performance processors with 64-bit data buses.

This configuration may be used in very high-performance applications and may be sufficient for handling full-duplex communication protocols. Even if the network frequency increases, the FSI could still handle the traffic.

Note that 64-bit data configurations are also possible with the basic configuration shown in Figure A-1.

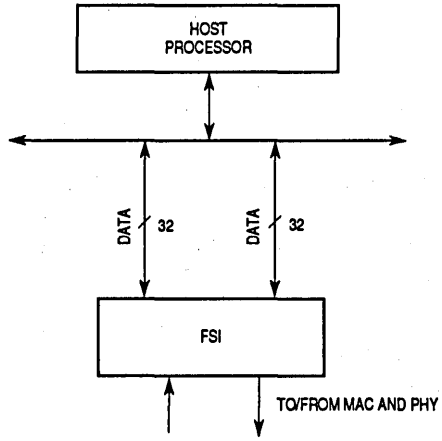


Figure A-5. Slave Configuration with 64-Bit Data

A

In the DMA configuration with 64-bit data, the FSI is connected to a very high-performance microprocessor (see Figure A-6). In this case, there is no system-wide DMA capability, so the DMA of the FSI is used. The port A data register holds the most significant portion of the data, and the port B data register holds the least significant portion of the data. The address is generated in port A, but can be accessed via either port.

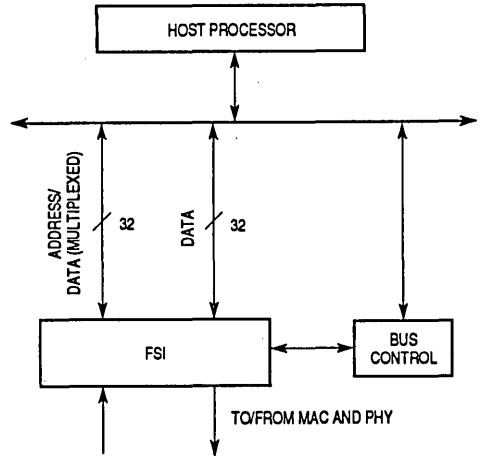


Figure A-6. 64-Bit Data with FSI DMA

A

A

APPENDIX B
DESIGN EXAMPLES

This section presents two examples of the design process using the MC68840 IFDDI device. Both examples include an explanation of the design requirements, a suggested solution, and the initialization sequence for that solution. Also, both examples were chosen to be as close to a real life application as possible and are for the purpose of demonstrating how to adapt a given system to the FDDI network. The first example is simple and the second is slightly more complex.

B.1 DESIGN EXAMPLE 1

In the first example, the design requires an FDDI network interface to a computer system. The system clock is 25MHz of which a maximum 20% can be dedicated to FDDI traffic. The maximum latency of the bus is 50 μs. The system bus is a burst bus. In addition to the system main processor there is an FDDI specific processor that will take care of SMT traffic. The frames are asynchronous only.

B.1.1 Suggested Solution

The suggested solution for this design example is described in the following paragraphs. Excluding overhead, the FDDI bandwidth will be 100 Mbits, because it consumes a maximum system bandwidth of 12.5 Mbytes per second. Since an IFDDI port is 4 bytes wide, the bandwidth will require 12.5/4 or 3.125 MHz bandwidth to deal with FDDI traffic. This is sufficient bandwidth for both traffic and overhead, since the system can allocate 5MHz bandwidth at any time.

The FDDI specific processor will deal with SMT frames using one transmit and one receive channel. The main processor will deal with LLC frames using an additional transmit and an additional receive channel. There will be 4 channels, 2 receive and 2 transmit. Incoming frames will be identified by their frame control (FC) field and directed accordingly. The LLC frames will be directed to ring 4 and the SMT frames will be directed to ring 5. Figure B-1 illustrates this configuration.

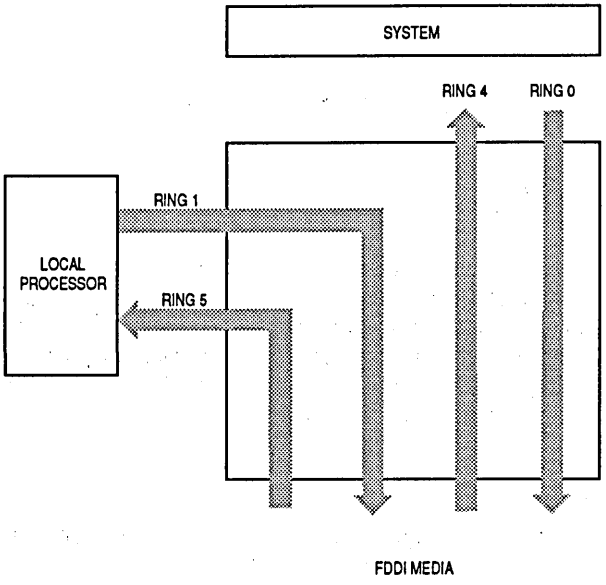


Figure B-1. Ring Set Up

To avoid transmit underrun, the transmit ring FIFO watermark must be calculated. When this station gets the token there should be enough data to transmit until the IFDDI is granted the system bus. With a latency period of 50 µs, the FDDI network can transmit 50 × 12.5 = 625 bytes of memory so the watermark registers should be programmed for 625 + 32 = 20 memory blocks (each memory block is 32 bytes). When the FDDI network is waiting to transmit, both transmit FIFOs will be filled to approximately 625 bytes. When the FDDI transmits from one FIFO then that particular FIFO will grow to twice its watermark. Therefore the space that should be allocated for transmit data is 625 × 2 + 625 = 1875 bytes of internal memory.

The receive FIFO watermark performs another role. When this watermark is reached, the IFDDI requests the system bus. In other words, the larger the receive FIFO watermark, the less the IFDDI interferes with the system bus. The watermark chosen is 1Kbytes. Also note that once the data FIFO reaches the watermark, it may have to store another 625 bytes due to the latency of the system bus. Therefore the space that should be allocated for the receive data is 1K+ 1K+ 625 latency bytes = 2673 bytes. Adding some descriptors as overhead increases this to 3 Kbytes total for both receive FIFOs combined. To assure that the receive internal space does not exceed 4 Kbytes, each FIFO has its RMR set to 2 Kbytes. When this bound is reached, the incoming received frame is aborted and an interrupt is generated to the processor.

To summarize these calculations, there are approximately 2 Kbytes for transmit and 4 Kbytes for receive. Note that the sum of these two memory spaces is less than the 8 Kbytes of internal memory in the IFDDI. It is necessary to connect the IFDDI to external

bus logic. In this case, when the IFDDI operates in pipeline mode, one or two PALs are needed with minimal glue logic. Figure B-2 illustrates this system's block diagram.

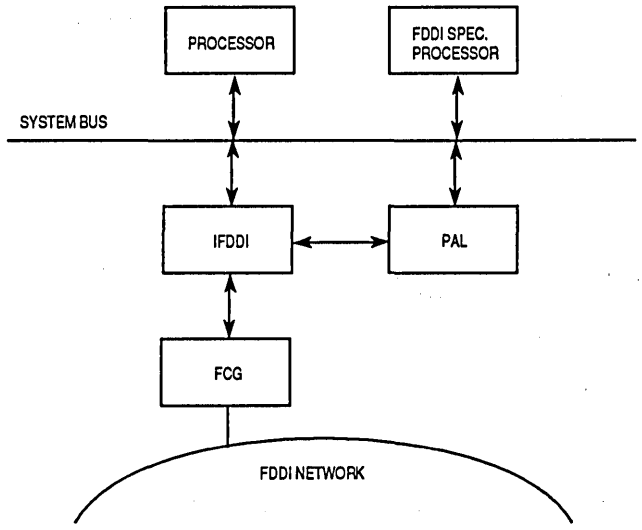


Figure B-2. System Block Diagram

B.1.2 Initialization Sequence

The IFDDI implements 1.5 OSI layers in addition to a system interface. There are essentially 3 stages to initialize the system.

1. Initialization of the ELM - PHY level
2. Initialization of the MAC - MAC level
3. Initialization of the FSI - system interface level.

B.1.2.1 ELM AND MAC INITIALIZATION. The initialization of the MAC and ELM are less sensitive to the system configuration; see **Section 11 Initialization and Programming.**

B.1.2.2 FSI INITIALIZATION. After the MAC and ELM are initialized, the first register to be written within the FSI is the interrupt mask register (IMR) followed by the parameter registers. Most of the parameter registers are accessed indirectly through the FCR. To write the FCR, first check that the control register free (CRF) bit is reset. The code to perform this operation is as follows:

```
define FCR f
/* write PCRA - Port A control register: set up as 32 bit port with no parity, Motorola byte
order (big endian) and port is enabled*/
write (FCR,5E010000);
```

*/*Write Port Memory Page Register, assume 2K bytes/page*/*

write(FCR, 461F0000);

*/*write Ring Parameter Register: rings 0,1,4,5 descriptors and data are on Port A, no parity check. Rings contain up to 64 entries. Command register is treated as ring 6*/*

write (FCR, 30060000);

write (FCR, 31060000);

write (FCR, 34060000);

write (FCR, 35060000);

write (FCR, 36060000);

*/*write FIFO watermark register for rings 0,1,4,5. Rings 4 and 5 have 32 blocks each, rings 0 and 1 have 20 blocks each*/*

write (FCR, 20140000);

write (FCR, 21140000);

write (FCR, 24200000);

write (FCR, 25200000);

*/*write Receive Frame Type Register. Ring 4 deals with LLC, Ring 5 deals with SMT*/*

write (FCR, 0C080000);

write (FCR, 0D020000);

*/*write RMR (Maximum Receive Memory Space Register) for ring 4 and 5*/*

write (FCR, 64400000);

write (FCR, 65400000);

/ write MACIF Transmit Control Register - enable transmission*/*

write (FCR, 08010000);

/ write MACIF Receive Control Register - enable reception for ring 4 and 5*/*

write (FCR, 09030000)

B.1.2.2.1 Transmit Rings. Transmit rings should be prepared as follows. One contiguous space in memory should contain the ring descriptors/indication. Therefore each entry in that ring should describe one frame to be transmitted. Assume that the transmit ring 0 is chosen to be put in the address 12345600. Each frame consists of one buffer and contains 4 Kbytes of memory. Table B-1 list the locations in memory and their structure.

Table B-1. Memory Locations

Memory Address	Data
12345600	B000_1000
12345604	1111_0000
12345608	B000_1000
1234560C	1111_1000
12345610	B000_1000
12345614	1111_2000
12345618	B000_1000
1234561C	1122_3344
12345620	0374_837B
12345624	7BCE_4321

Note that there are four descriptors pointing to one frame each. The next entry in memory has a zero own bit, and is where the IFDDI stops reading the descriptors. The first three frames occupy consecutive places in memory (1111_0000 to 1111_2FFF) while the fourth points to a totally different place in memory (11223344). The same procedure should be performed for transmit ring 1. Figure B-3 illustrates the pointer structure for this ring.

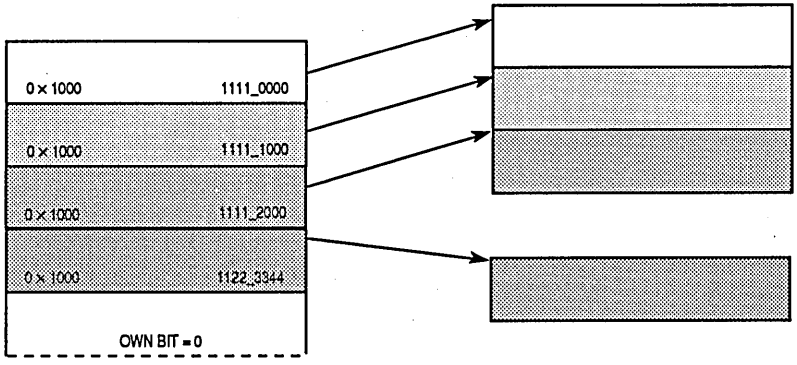


Figure B-3. Descriptor Ring Pointer Structure

B

B.1.2.2.2 Overhead Bytes. At the start of each frame there are always some overhead bytes consisting of the following:

1. Packet Request Header (3 bytes): Program for the following configuration:

— Asynchronous frame.

—Transmitted by unrestricted AND restricted tokens.

—Doesn't have to be transmitted first or last.

—The MAC adds the CRC.

—No extra FS.

This results in the following 3 bytes: 303800. For more information on the Packet Request Header see **10.4.1.2.5 Packet Request Header**.

2. Frame Control Field: 1 byte set to 50.

3. Destination Address (6 bytes): Program the DA as the following

—Bit 47 is zero signifying this is an individual address.

—Bit 46 is zero signifying it is a local address.

This results in the following 6 bytes: 34567890ABCD.

4. Source Address (6 bytes): Program the SA so that the SA matches the MLA_A, MLA_B and MLA_C registers in the MAC. Example of a valid SA: 111122223333.

The frame appearance is illustrated in Figure B-4.

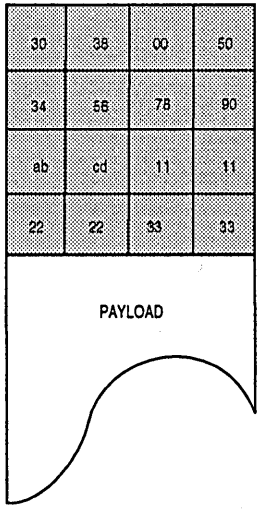


Figure B-4. Frame Appearance

B.1.2.3 Receive Frames and Rings. Since there are no incoming frames, the only preparation necessary is to allocate memory space for the data buffers and one memory space for the ring descriptors. If the receive frames are to be fragmented into 2-Kbyte blocks and the ring 4 receive ring address is 11335500, then memory contain the values as listed in Table B-3.

Table B-2. Memory Location Values

Memory Address	Data
11335500	8000_0400
11335504	2345_0000
11335508	8000_0400
1133550C	2345_7000
11335510	8000_0400
11335514	3245_6743

After preparing the above data structures in memory, the rings should be defined. Defining rings is performed by issuing commands to the IFDDI by writing to the command register (CMR). The CMR can be written only if the CMR done bit in the status register (SR1) is set. Therefore writing to the CMR involves reading the SR1 either by polling, or by setting an interrupt routine specifically for that bit. Each command to the IFDDI contains the ring ID number, the address of the descriptor ring, and the ready bit. It is preferable to define the rings with the ready bit set to put the ring into action immediately. Note that the entire initialization procedure can be inserted into a ring of command descriptors, and the handshaking overhead can therefore be avoided. The code for defining the rings is as follows:

```
DEFINE CER    11
DEFINE CMR    9
/*Command is 64 bits wide, use CER for lower 32 bits of command. Command is
executed immediately following the write to the CMR, so write CER first*/
/* Define Receive rings 4 and 5. Keep read pointer equal to the write pointer*/
write (CER, 12341100);
write (CMR, BE0C1100);
write (CER, 11662100);
write (CMR, BE0D2100);
/* Define transmit rings 0 and 1. Ready bit is set so IFDDI reads from the descriptor
immediately upon ring definition*/
```

write (CER, 43215430);

write (CMR, BE085430);

write (CER, 44445720);

write (CMR, BE095720);

B.2 DESIGN EXAMPLE 2

In the second example, the design requires an FDDI network interfaced to a low performance system bus. The system clock is 5MHz, of which a maximum of 10% can be dedicated to the FDDI traffic. The maximum bus latency is 100 µs. 90% of the IFDDI traffic is due to the transmission of files that are placed in a continuous space in system memory.

B.2.1 Suggested Solution

If the token rotation time (TTRT) is a typical value of 2 ms, then the station is allowed to hold the token for up to 2 ms. During this period, the station will be able to transmit 25 Kbytes (2 ms × 12.5 Mbytes ÷ s). Because the system restricts the bandwidth to 0.5 MHz at any time, it will be able to transfer as much as 8 Kbytes (2 ms × 0.5MHz × 4 Bytes ÷ s) per token and then release it. The major issue in designing low-power computer systems is that their system bus bandwidth cannot support 12.5 Mbytes per sec. sustained traffic. The solution is to accumulate data in a buffer, then when the token arrives, transmit the data in the buffer. This paticular design example assumes that the FDDI network consists of up to 100 stations (FDDI protorocol allows for up to 1000), which implies that each station will transmit on average 1% of the time. The other 99% of the time the station will be accumulating data for transmission. If there are 100 stations on an FDDI network and each station is transmitting asynchronous frames for an average of 2ms per token, then each station can transmit once every 0.2 seconds. Within this period of time, each station can store up to 25 Kbytes in its buffers. The internal memory of the IFDDI is only 8 Kbytes, so there must also be an FDDI dedicated local memory used with the system for additional buffer storage, using ring 0 as a transmit local mode ring. Figure B-5 illustrates the data flow in this system example.

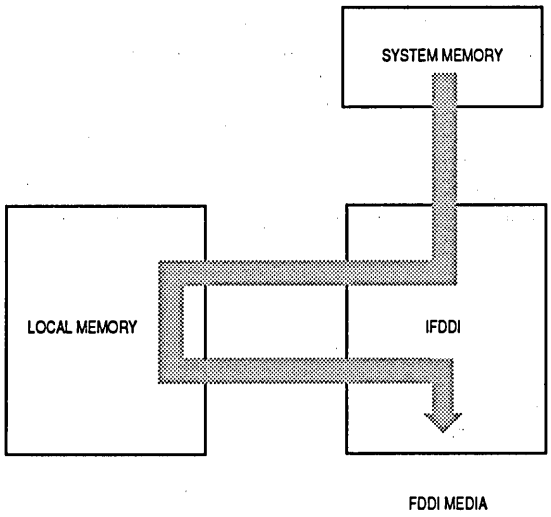


Figure B-5. Data Flow with Local Memory

The main function of the station involving FDDI is file transfer. The idea is to have data segmented into 4-Kbyte frames, with a packet request header (PKT_HEADER), FC, DA, and SA for each frame; then transmit the frames through the local memory. The simplest way to add the header to each frame is to construct the frame from two buffers. The first buffer containing the header fields: PKT_HEADER, FC, DA and SA and the second buffer containing the 4 Kbytes of data. If the data is long, the same first buffer could be used for all of the frames of the file. Figure B-6 illustrates the structure of the four frames with their header information.

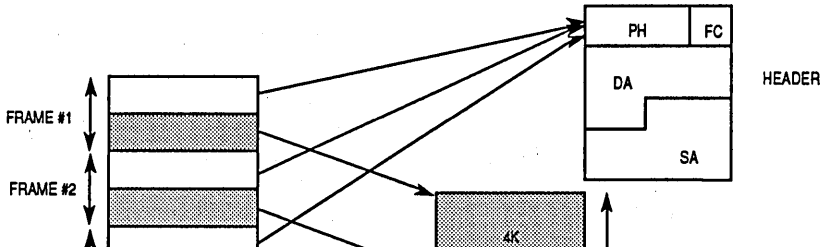


Figure B-6. Frame Structures

Only fixed length buffers are used for reception configurations, since the length of the incoming file is unknown. The most straight forward solution is to have software omit the headers by moving only the data of the file into another space in memory. This costs another read and write cycle for each data write access to system memory, tripling the receive bandwidth of the system. To accommodate this, a processor that is responsible for moving the receive frames without headers into the system memory is placed on the local memory port. This processor's only action is to define the DMA buffers that begin after the header. Figure B-7 illustrates what the local memory should look like.

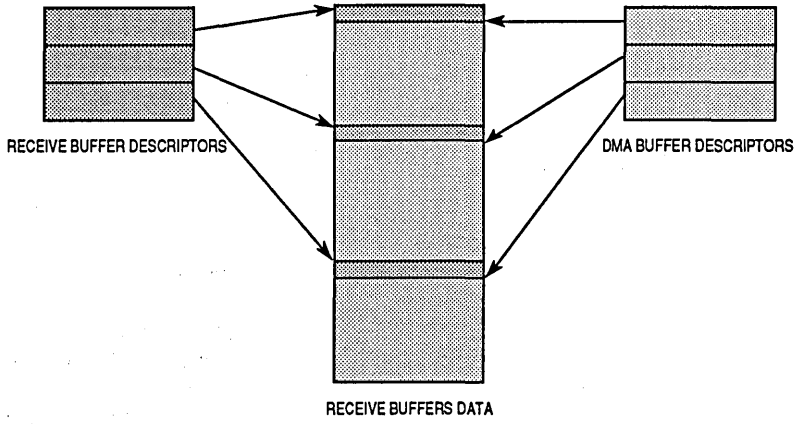


Figure B-7. Local Memory Recieve Buffer Structure

Due to the use of both DMA and normal traffic, two channels are needed. Ring 4 is a normal receive ring and ring 1 is a transmit ring whose commands are DMA commands. Figure B-8 illustrates the LLC ring structure associated with local memory.

Using an additional local processor for processing frames before they are moved to the system, can also help in applications where some of the FDDI frame data is used by higher management levers as a header or a tail. Having the additional processor in local memory also assists in taking care of network issues, particularly those dealing with SMT frames. Therefore all receive traffic is directed first to the local node processor memory then only the LLC frames continue by DMA to the system. This processor uses a different channel for transmitting SMT frames. Figure B-9 illustrates the system diagram.

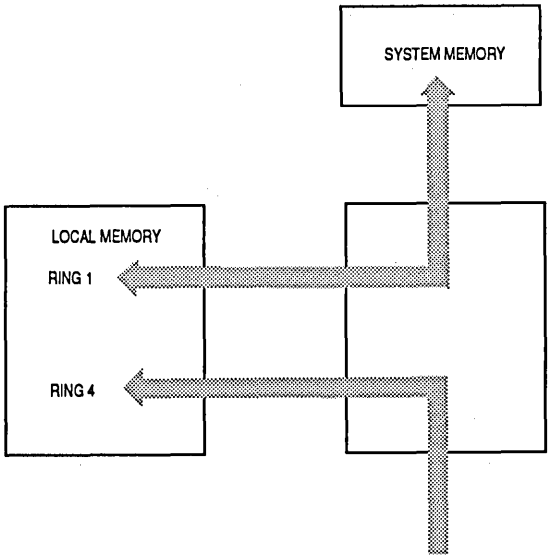


Figure B-8. LLC Associated Data Flow

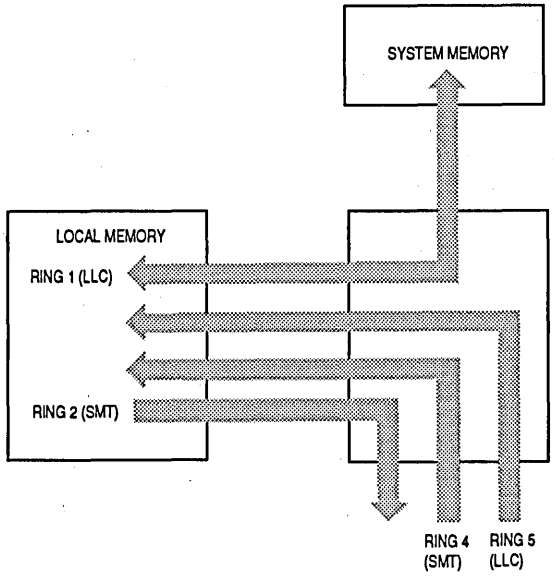


Figure B-9. Local Memory Data Flow

RIng 3 of the IFDDI is unused, so it is allocated as a DMA channel for non-FDDI applications in the system such as DMA from memory to a disc controller or performing a loopback in the system memory. Figure B-10 illustrates the final system diagram.

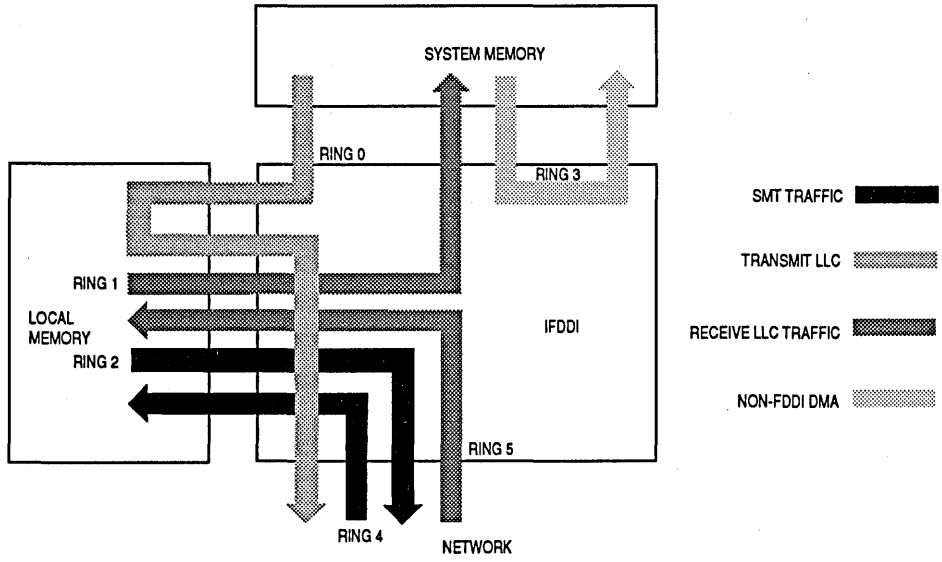


Figure B-10. Complete System Data Flow Diagram

If communication is required between the system processor and the local processor, then the make-indication command (see **Section 9 Commands and Indications**) can be used through both command registers.

B.2.2 Initialization Sequence

The amount of local memory necessary must be calculated. In this example, it was shown that a buffer of 25 Kbytes is sufficient for proper system performance. 32 Kbytes is allocated for the transmit operations. The reception operation is slightly different since the receive traffic doesn't burst as much as the transmit traffic causing reception to be more smooth. This is due to the fact that most of the time this station has no token (only 0.1%), is repeating frames, and occasionally copies frames into its local memory. The only reason for allocating memory is to insure sufficient space for peak bursts of receive frames. 16 Kbytes is allocated for the reception operation. Table B-4 lists the memory map.

Table B-3. Local Memory Map

Memory Address	Ring Number	Ring Size	Ring Usage
0000 - 7FFF	Transmit Ring 0	32 K	IFDDI Chip use only
8000 - AFFF	Receive Ring 4	12 K	Same as Ring 1
B000 - B7FF	Receive Ring 5	2 K	SMT Traffic
B800 - BFFF	Transmit Ring 2	2 K	SMT Traffic
E000 - FFFF	Descriptor Rings	—	—

For this example no code is given, instead the following is a summary of how to initialize each ring.

Ring 0: Transmit Local Memory Mode, ring is on port A.

RPR0—Ring descriptors and data are on Port A; descriptor ring maximum length is 64. RPR0=0x03

PER0—Local memory parameters are as follows: data is on port B, no parity check, with a size of 128 Kbytes. Therefore the PER0 should be set to 0x12.

FWR0—This particular register has no effect in solving the system latency issues, so it is set to 0.5 Kbytes, or 16 memory blocks. FWR0 = 0x10

LMT0—This register limits the number of frames in the local memory. 25 Kbytes of 4K each is no more than 7 Bytes in local memory. The LMT0 = 0x06.

Set Local Memory Command—Contains the local memory start address of 0. The command format is therefore BE180000_00000000.

Define Ring Command—Points to the descriptor ring in the system. The L-bit is set (indicates local memory use), and the I-bit is reset so that the indication for this frame is sent only AFTER the frame is transmitted. The command should then be: BE481238_56781238.

Ring 1: This ring is a DMA channel from local memory into system memory.

RPR1—Ring descriptors and data are on port B; the descriptor ring maximum length is 16, so RPR1=0x31.

PER1—Destination Ring parameters are as follows. Data and descriptors on port A, no parity check, with the destination ring length as 16. The PER1 = 0x03.

Set Destination Ring Command—Contains a pointer to the destination descriptor ring. The command format is therefore BE191110_22221110.

Define Ring Command —Points to the descriptor ring in the local memory. The L-bit is reset, and the I-bit is reset since this is a normal ring. The command is therefore BE09E000_0000E000.

Ring 2: This is a transmit ring for SMT frames.

RPR2—Ring descriptors and the data are on port B; Descriptor Ring maximum length is 16, so RPR2=0x31.

FWR2—Again set to 0.5 Kbytes (16 blocks) since there are no latency problems with the local memory. FWR2=0x10.

Define Ring Command —Points to the descriptor ring in the local memory. The L-bit is reset and the I-bit is reset because it is a normal ring. Therefore the command will be BE0AE400_0000E400.

Ring 3: This is a DMA channel for non-FDDI applications on the system bus.

RPR3—Ring descriptors and data are on port A, and the descriptor ring maximum length is 64, so RPR3=0x03.

PER3—The destination ring parameters are as follows. The data and descriptors are on port A, with no parity check and the destination ring length is 64. PER = 0x03.

Set Destination Ring Command—Contains a pointer to the destination descriptor ring. The command is then BE1B1110_44441110.

Define Ring Command—Points to the descriptor ring in the system. L and I-bits are reset since the ring is normal. The command is then BE0B0000_00030000.

RMR3—2 Kbytes; the RMR4 = 0x40.

Ring 4: This ring is a receive channel to port B for SMT traffic

RPR4—Ring descriptors and data are on port B; Descriptors ring maximum length is 16, so RPR4=0x31

FWR4—Also set to 0.5Kb (16 blocks). FWR4 = 0x10.

RFR4—This ring is receiving SMT asynchronous traffic and is then separating the frames in the local memory. RFR4=0x02.

Define Ring Command—Points to the descriptor ring in the local memory. The L and I-bits are reset since they are normal rings. The command is then BE0CE800_0000E800.

RMR4—2 Kbytes; the RMR4 = 0x40.

Ring 5: This is a receive channel to port B for LLC traffic.

RPR5—Ring descriptors and data are on port B; Descriptors ring maximum length is 16, so RPR5=0x31

FWR5—Also set to 0.5 Kbytes (16 blocks). FWR5 = 0x10.

RFR5—This ring is receiving LLC asynchronous traffic and is then separating the frames in the local memory. RFR5=0x08.

Define Ring Command—Points to the descriptor ring in the local memory. The L and I-bits are reset since they are normal rings. The command is then BE0DEC00_0000EC00.

RMR5—2 Kbytes; the RMR5 = 0x40.

Command Register Ring 6: This is used by the system processor for initialization, monitoring and message transfer to and from the local processor.

CPR6—Command register is assigned to port A. CPR6 = 0x00.

No Define Ring Command is necessary for this ring.

Command Register Ring 7: This is used by the local processor for initialization, monitoring and message transfer to and from the system processor.

CPR7—Command register is assigned to port B. CPR7 = 0x10.

No define ring command is necessary for this ring.

B

APPENDIX C
CAM OPERATION IN NON-FDDI APPLICATIONS

C.1 INTRODUCTION

When the IFDDI operates as a stand alone FSI device, its CAM can be used for non-FDDI applications. Two properties have been added to the IFDDI CAM:

1. the CAM compares words that have a width of: 8, 16, 24, 32, 40, 48 bits;
2. the CAM supports a non-consecutive stream of bytes.

This appendix explains how these properties of the CAM can be used.

C.1.1 CAM Address Modes

The CAM supports two address modes:

1. 48-bit address mode; where only a 48-bit address can be matched.
2. Multiple address mode; In this mode, a 48-bit address and non-48-bit address can be matched. The non-48-bit address shall be 8, 16, 24, 32 or 40 bits wide.

The user can define the CAM operation mode using the COM bit in the ICR. Figure C-1 illustrates the six options of the CAM address width. Note that trailing zeros should be added to all the addresses except a 48-bit address.

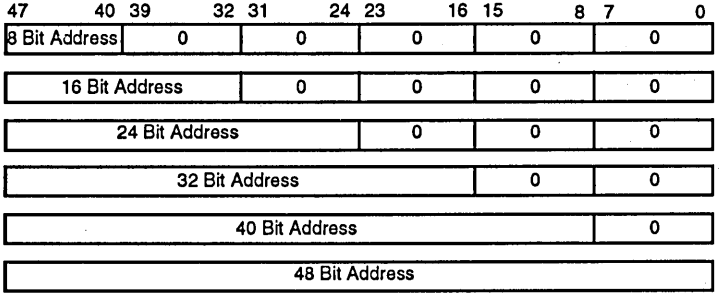


Figure C-1. CAM Address Options

C

C.1.2 ADDIN Input Pin

The ADDIN input pin defines the bytes included in the address comparison, and is used regardless of the CAM operation mode.

C.2 CAM TIMING DIAGRAMS

The following paragraphs describe the timings for CAM operations in 48-bit address mode and non-48-bit address mode.

C.2.1 CAM Operation In 48-bit Address Mode

Figure C-2 illustrates the CAM operation in 48-bit address mode without using ADDIN. Note that LDADDR enables the comparison of the source address and the destination address, and DA enables only the comparison of the destination address.

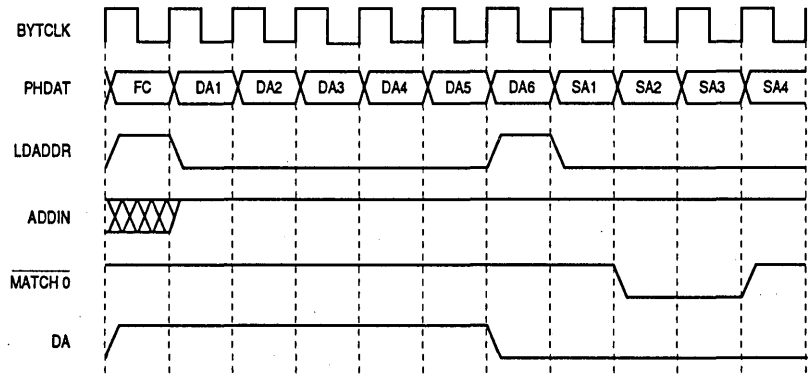


Figure C-2. CAM Operation in 48-bit Address Mode

C.2.2 CAM Operation In Multiple Address Mode

Figure C-3 illustrates the CAM operation in multiple address mode. The LDADDR is asserted one BYTCLK before the first byte of data, and negated on the last byte of data. Note that the pulse width of LDADDR is equal to the number of bytes that the CAM compares. The $\overline{\text{MATCH0}}$ output can be asserted one BYTCLK after the last byte of data.

Figure C-4 illustrates a CAM cycle that compares 40 bits (5 bytes) without using ADDIN; and Figure C-5 illustrates a CAM cycle that compares 32 bits (4 bytes) using ADDIN.

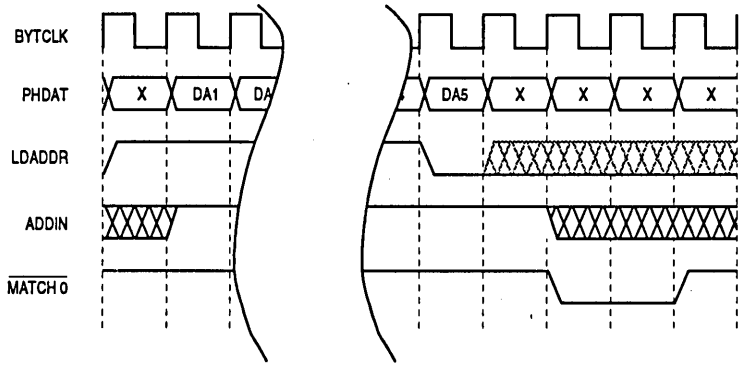


Figure C-3. CAM Operation in Multiple Address Mode

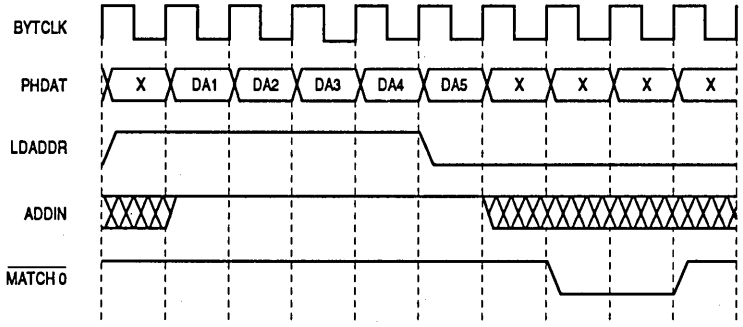


Figure C-4. Five Bytes Comparison in Multiple Address Mode Without Using ADDIN

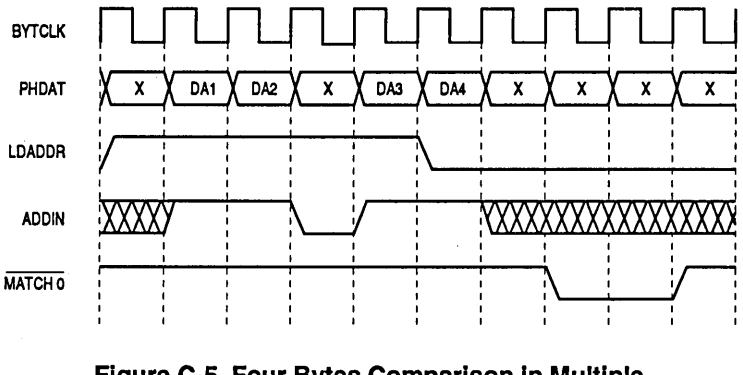


Figure C-5. Four Bytes Comparison in Multiple Address Mode Using ADDIN

C

C.3 UPDATED COMMAND AND INDICATIONS

C.3.1 Set Up CAM Command

The set up CAM command can be issued by the host processor through one of the port's command registers or it can be placed inside one of the transmit buffer descriptor rings. The set up CAM command writes the CAM entry,V-bit and W-bit to the CAM_ID address in the CAM. Figure C-6 illustrates the set up CAM command.

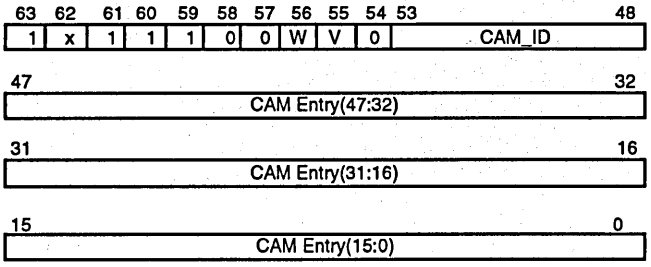


Figure C-6. Set Up CAM Command

W—Address entry width; 1 for a 48-bit address, 0 for a non-48-bit address.

V—Valid; 0 for an invalid entry, 1 for a valid entry.

CAM_ID—CAM entry address.

CAM Entry—The address entry placed inside the CAM at CAM_ID. The CAM can contain two different types of address entries: 48-bit address and non-48-bit address. The non-48-bit addresses that can be enabled are: 8, 16, 24, 32, 40 bits. Figure C-1 describes the six options for the CAM entry field.

C.3.2 Read CAM Entry Indication

This indication is generated in response to a read CAM entry command. Figure C-7 illustrates the read CAM entry indication.

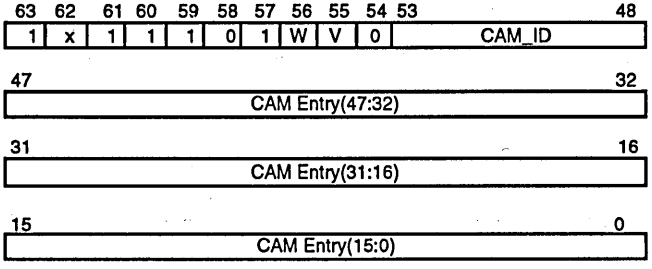


Figure C-7. Read CAM Entry Indication

W—Address entry width; 1 for a 48-bit address, 0 for a non-48-bit address.

C

V—Valid; 0 for an invalid entry, 1 for a valid entry.

CAM_ID—CAM entry address.

CAM Entry—Address entry; the address entry read by this command. Figure C-1 describes the six options of the CAM entry field.

C.3.3 Compare CAM Entry Command

The compare CAM entry command can be issued by the host processor through one of the port's command registers or can be placed inside one of the transmit buffer descriptor rings. This command is used for testing the CAM device and to determine that the CAM has been programmed correctly.

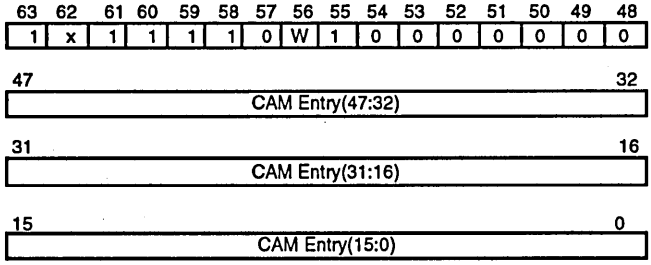


Figure C-8.Compare CAM Entry Command

W—Address entry width; 1 for a 48-bit address, 0 for a non-48-bit address.

CAM Entry—The address entry that this command should search for. Figure C-1 describes the 6 options of the CAM entry field.

C

C

APPENDIX D
PERFORMANCE REQUIREMENTS

This appendix is intended to give a better understanding of the performance requirements for the different configurations available with the FSI.

The FDDI network speed is 100 Mbit/sec or 80 ns/byte. To transmit and receive frames consecutively, the system bus must have a cycle time as calculated in this section. The variable, K, used in the following examples is the coefficient of average bus speed in byte transfer units of 80 ns. K may be viewed as:

K = network speed (12.5 Mbytes per second) / bus speed (bytes per second)

For example, in case of bus width = 32 bit and bus cycle = 160 ns, K will be equal to 0.5—i.e., the bus speed is twice the network speed.

The internal overhead of the FSI, O, before it will start the data transfer is ~1280 ns. This time in byte transfer units (80 ns) is 16 bytes. This overhead is applied when the FSI changes between ring descriptor accesses and data DMA accesses. The total time needed for DMA of X bytes is given in Equation 1:

$$\text{total data DMA time} = \{O + KX\} \text{ or } 16 + KX \quad (1)$$

The examples attempt to show the transmission and reception of back-to-back frames with a minimum interframe spacing of 8 bytes of idle symbols. If one assumes that the CRC, or FCS field, of a data frame does not have to be received, then the network overhead for both transmitted and received frames is 12 bytes. That is, the network (in this case, the MAC) imposes additional symbols that are not a part of the DMA data frame such as the idle symbols, JK, ED, etc. Therefore, the network time for every data frame is the number of frame bytes including FC, DA, SA, and INFO field plus the overhead (see Equation 2):

$$\{\text{network time}\} = X + 12 \quad (2)$$

The information field, I, of each frame with 48-bit addresses is:

$$I = X - (PRH + FC + DA + SA) \text{ or } I = X - 16$$

During the transmission or reception of back-to-back data frames, the general equation for bus utilization is given in Equation 3:

$$U = \{\text{system time}\} / \{\text{network time}\} \quad (3)$$

D

In the following subsections, the user is provided with formulae that describe the transmission and reception of back-to-back frames. The user may determine the bus utilization for any given frame size or amount of data to be transferred. Alternatively, the user may determine the amount of data or frame size to be implemented to achieve any given rate of bus utilization.

D.1 FSI TRANSMIT PERFORMANCE

The FSI has room inside its internal memory for as many as 20 descriptors. Also, the transmit DMA operation has priority higher than descriptor input operation. Therefore, in case of small frames (frame length is smaller than a watermark) and assuming each frame to have one descriptor, the internal activities of the FSI are as follows:

a. Input 20 descriptors (of 20 frames)
b. DMA 20 frames
c. Go to (a)

Note that indication write activities are not taken into account in this analysis.

There are eight bytes in one descriptor. Only one period of internal overhead (16 byte times) is used for 20 descriptors. The total time is in byte transfer units (80 ns). The total time needed for reading 20 descriptors is given in Equation 4:

$$\text{total descriptor read time} = \{16 + 20\ K8\} \tag{4}$$

D.1.1 Case 1—Using Same Port for Data and Descriptors

For applications where both data and descriptors are read from the same port (this is the only configuration when using 64-bit data width), the equation is:

$$\{\text{system time}\} = \{\text{total data DMA time}\} + \{\text{total descriptor read time}\}$$

or

$$F\ (X + N) = F\ (O + KX) + (O + FKD)$$

where:

F is the number of frames, here 20

O is the FSI overhead, 16 bytes

N is the network overhead, 12 bytes

D is the number of bytes in a descriptor, 8

K is the bus coefficient

X is the amount of data in a frame

Computing for 20 data frames, and thereby 20 descriptors, with X data in each frame and a total network time as previously given yields:

$$20\ (16 + KX) + (16 + 20\ (K\ (8))) = 20\ (X + 12)$$

Solving for X yields:

$$X = (24 + K40) / (5\ (1 - K))$$

The external bus utilization should consider all the indications that also need to be written back to the system memory; therefore, following equation 3:

U = {data DMA time} + {indication write time} + {descriptor read time} / {network time}

$U = (KX + K8 + K8) / (X + 12)$

$U = K\ (X + 8 + 8) / (X + 12) = K\ (X + 16) / (X + 12)$

For example:

A. Assuming the average access time to be 160 ns for 32-bit word—i.e., K = 0.5:

   $X = (24 + 20) / 2.5 = 44 / 2.5 = 18$

   $U = 0.5\ (18 + 16) / (18 + 12) = 0.567$ or 56.7%

   Therefore, in this case, the minimum frame length should be 18 bytes or 2 bytes of INFO to transmit 20 data frames back to back.

B. Assuming the average access time to be 160 ns for 64-bit word—i.e., K = 0.25:

   $X = (24 + 10) / 3.75 = 10$

   $U = 0.25\ (10 + 16) / (10 + 12) = 0.295$ or 29.5%

   Therefore, in this case, the minimum frame length should be 10 bytes total to transmit 20 data frames back to back.

D.1.2 Case 2—Using Different Ports for Data and Descriptors

For applications using different ports for data and descriptors, the equation will consider only the DMA overhead; therefore, following Equation 4, it follows that:

$$\{16 + KX\} = X + 12$$

or

$$X = 4 / (1 - K)$$

The external bus utilization for data only is calculated as follows:

$$U = KX / (X + 12)$$

D

For example, assuming the average access time to be 160 ns for 32-bit word—i.e., K = 0.5:

$X = 4 / 0.5 = 8$

$U = 0.5\ (8 / (8 + 12)) = 0.20$ or 20%

Therefore, in this case, the minimum frame length should be 16 bytes or *zero* bytes of INFO.

D.2 FSI RECEIVE OVERHEAD

In the receive case, the receive buffer descriptors are read when the ring has been defined and made ready. Indications for each received frame are written back out to memory as the conditions warrant and as the internal priority scheme of the FSI internal tasks allows. Thus, some receive indications may be written out individually; others may be written out in groups. The total time needed for a single indication (there are 8 bytes in one indication) to be written is:

$$\{20 + K8\}$$

Two different cases that matter for performance analysis of the FSI are described in the following subsections.

D.2.1 Case 1—Using Same Port for Data and indications

For applications where both data and indication are output from the same port (this is the only configuration when using 64-bit data width), the system time equation is:

$$\{\text{system time}\} = \{\text{total DMA time}\} + \{\text{total indication time}\}$$

$$\{\text{total DMA time}\} + \{\text{total indication time}\} = \{\text{network time}\}$$

$$\{20 + KX\} + \{20 + K8\} = X + 12$$

Solving for X yields:

$$X = (28 + K8) / (1-K)$$

The external bus utilization should take into account another 8 bytes of descriptor. Therefore, following equation 3, the general equation for external bus utilization is:

$U = \{\text{DMA time}\} + \{\text{indication time}\} + \{\text{descriptor time}\} / \{\text{network time}\}$

$U = K\ (X + 8 + 8) / (X + 12) = K\ (X + 16) / (X + 12)$

For example:

A. Assuming the average access time to be 160 ns for 32-bit word—i.e., K = 0.5:

$X = (28 + 4) / 0.5 = 64$

$U = 0.5\ (64 + 16) / (64 + 12) = 0.53$ or 53%

Therefore, in this case, the minimum frame length should be 64 bytes or 48 bytes of INFO for continuous reception.

B. Assuming the average access time to be 160 ns for 64-bit word—i.e., K = 0.25:

   X = (28 + 2) / 0.75 = 40

   U = 0.25 (40 + 16) / (40 + 12) = 0.27 or 27%

   Therefore, in this case, the minimum frame length should be 40 bytes or 24 bytes of INFO for continuous reception.

D.2.2 Case 2—Using Different Ports for Data and indications

For applications using different ports for data and indications, the equation will consider only the DMA overhead; therefore, it follows that:

$$\{20 + KX\} = X + 12$$

or

$$X = 8 / (1 - K)$$

The external bus utilization for data only is calculated as follows:

$$U = KX / (X + 12)$$

For example, assuming the average access time to be 160 ns for 32-bit word—i.e.,—K = 0.5:

X = 8 / 0.5= 16

U = 0.5 (16) / (16 + 12) = 0.29 or 29%

Therefore, in this case, the minimum frame length should be 16 bytes or *zero* bytes of INFO for continuous reception.

D.3 LOW-PERFORMANCE SYSTEMS

In a low-performance system, the requirements from the transmitter and receiver are much less. However, this system should be able to transmit the entire frame on the FDDI network, even if the bus bandwidth allocated for the FSI is insufficient. In this case, the use of local memory will alleviate the performance stress on the FDDI system, allowing slower system buses to easily connect to the IFDDI. During reception, local memory offers greater temporary storage for FDDI frames as the frames come in from the MAC interface, essentially increasing the internal data FIFO storage space. During transmission, local memory offers the ability to perform transmission throughout the entire token time by storing frames from the system in local memory during the time the station is not holding the token.

D.4 NODE PROCESSOR

It may also be useful to implement a node processor to handle all the station management (SMT) traffic. This processor does not need to have as much performance as the host processor because the throughput of management frames is low and their frames are short—i.e., the FIFO used to transmit them may be set with a watermark greater than the maximum frame size. It is not a requirement to have a node processor. The chip set handles enough of the real-time portions of SMT so that very little of the host processor is needed to run SMT. The decision of whether or not to have a separate node processor is an architectural preference.

APPENDIX E
ERROR DISCUSSION

The following subsections discuss various errors as they relate to certain registers, commands, and indications.

E.1 DMA INDICATION

ER (Error): This bit is set when the DMA transfer is aborted due to an error specified by the PER bits.

PER (Port Error): This bit field specifies an error during data transfer through the FSI ports.

E.2 DMA ERROR INDICATION (DESTINATION SIDE)

PER (Port Error): This bit field specifies an error during data transfer through the FSI ports.

E.3 INTERNAL ERROR STATUS REGISTER (IER)

IOE (Internal Overrun Error): If response time of the main controller exceeds the maximum allowed limit, this error occurs. It could indicate a problem with the main controller. The receiver is disabled (receive enable (RE4 and RE5) bits are reset in the MACIF receive control register (MRR)), and RABORT is generated.

IUE (Internal Underrun Error): If response time of the main controller exceeds the maximum allowed limit for data transfer, this error occurs. It could indicate a problem with the main controller. The frame being transmitted is aborted with the proper indication, and the transmitter is disabled (transmit enable (TE) bit is reset in the MACIF transmit control register (MTR)).

TPE (Transmit Internal Parity Error): The MAC interface recognizes a parity error on transmit data. An internal parity error detected during data reads from the internal memory always indicates a problem in an internal FSI data path. In this case, the TPE bit is set, the frame currently being transmitted is aborted, and the TE bit in the MTR is reset.

MER (MAC Error): The MAC interface receiver recognizes a protocol handshake error between the MAC and FSI. The MAC interface will generate an RABORT to the MAC and will try to resynchronize with the MAC on the next frame.

MOV (Memory Overrun): The FSI experiences an internal memory overrun that causes the MOV to be set. This error can be caused by an incorrect definition of an internal FIFO's watermark. Check the FSI parameter definitions and reset the FSI.

PBE/PAE (Port B or Port A Internal Error): The port interface recognizes a parity error on received data being transferred to the system bus.

E.4 OVERRUN/UNDERRUN

Overrun occurs if the FSI receiver is unable to move data from its receive buffers to system memory in adequate time to prevent receive buffer overflow. The receiver is accumulating data sent from the MAC in much less time than it takes the FSI to acquire the system bus.

Underrun occurs if the FSI transmitter is unable to move data from system memory to its transmit buffers in adequate time to prevent transmit buffer underflow. The transmitter is sending data to the MAC in much less time than the FSI is receiving data from the system bus. In cases where underrun occurs, the FSI watermarks should be checked and recalculated.

E.5 PARITY ERROR HANDLING

Bits affected by parity errors are listed in Table E-1.

Table E-1. Parity Errors

Register Location	Bit Name	Cause of Error
IER	TPE	Parity Error on Transmission
IER	PAE	Parity Error on RXDATAx during Transfer to System Port A
IER	PBE	Parity Error on RXDATAx during Transfer to System Port B
SR1	HER	Parity Error on Write to Direct Access Register
SR2	DRE	Parity Error on Access to Destination Ring
RSR	PER	Parity Error Detected on Ring Entry Read
TRANSMIT Indication	PER	PER = 010, 110, 111
TRANSMIT Indication	PE	Parity Error on Transmission
DMA Indication	PER	PER = 010, 111

E.5.1 Transmit Data Parity Error

When a parity error is recognized by a port during a transmit data transfer, the FSI:

a. Immediately stops the transfer of the frame to FSI internal memory, and the frame is marked as an error frame.

b. Aborts transmission of the frame, possibly creating a remnant frame, if the frame is currently being transmitted.

c. Writes the error indication for this frame back to the descriptor ring.

Note that the activities on the ring will otherwise continue normally.

E.5.2 Buffer Descriptor Transfer Parity Error

When a parity error is recognized by a port during a descriptor transfer, the FSI:

a. Immediately stops the transfer. The descriptor will not be transferred to the internal ring FIFO. The ring read pointer will point to the beginning of this descriptor as accessed by the READ RING PARAMETER command.
b. Changes the state of this ring to empty, and the parity error (PER) status bit is set in the appropriate ring state register (RSR).
c. Sets the ring not ready (RNR) bit in the FSI status register 1 (SR1), which causes an interrupt if it is enabled.
d. Sets the ring error (RER) bit in the FSI SR1, which causes an interrupt if it is enabled.

To cause the FSI to reread this entry in the case of a parity error being caused by temporary problems on the data bus, the host processor should:

a. Clear the RER and RNR bits in the SR1 by writing the SR1 with these bits set.
b. Transfer the ring to the ready state by performing a ring ready control access.

If the parity error was a severe error, the ring should be redefined to avoid this memory block. To do this, the host processor first issues the STOP command to transfer the ring to the stopped state, and then redefines the ring.

E.5.3 Processor Access to Direct Access Register Parity Error

When a parity error is detected on host processor accesses, the FSI:

a. Ignores the host processor access.
b. Transfers the ring to the ready state by performing a ring ready control access.

If the command extension register (CER) is written with a parity error, the command written to the command register (CMR) is ignored.

E.6 PORT OPERATION ERRORS

These errors are discussed thoroughly in **Section 6 Port Operation.**

E.7 RING STATE REGISTER ERRORS

PER (Parity Error): This bit is set when the FSI detects a parity error while a ring entry is being read. It is reset when:

a. The ring is redefined.
b. Ring ready is signaled by a control access to the FSI control register (FCR).

OER (Operation Error): This bit is set when a POE is received or a FIRST/LAST bit error occurs. It is reset when:

a. The ring is redefined.
b. Ring ready is signaled by a control access to the FCR.

NOTE

For a discussion on FIRST/LAST bit errors, see **E.13.5 FIRST or LAST Bit Errors.**

E.8 FSI STATUS REGISTER 1 ERRORS

IOE (Internal Operation Error): This bit indicates that an internal operation error has been detected. This is a sticky bit and must be cleared by the host. Internal operation errors relate to the IER as discussed previously.

ROV4–5 (Receive Overrun Error): These bits indicate that an overrun condition has occurred for this ring. This is a sticky bit and must be cleared by a host write to the SR1.

RER (Ring Error): This bit is set by:

a. A Parity Error
b. A Port Operation Error
c. A FIRST or LAST Bit Error

E.9 STATUS REGISTER 2 ERRORS

DRE (Destination Ring Error): This bit is set by a parity error or port operation error.

E.10 RECEIVE FRAME NORMAL INDICATION ERRORS

CE (CRC Error): This bit indicates that the cyclical redundancy check on the frame does not agree/pass. There are no guarantees that the frame is good if this bit is set.

E.11 RECEIVE FRAME ERROR INDICATION

This indication is generated when

a. An error, fragment, or secondary NSA frame is received when the FSI is in receive all mode.
b. The receive operation has been aborted by the FSI.
c. The fragment is longer than the receive watermark.

STT (Receive Status of MAC): This bit field indicates the receive status of the MAC.

E.12 TRANSMIT INDICATION ERRORS

PER (Port Error): This bit field specifies an error that occurred during data transfer through the FSI ports.

AB (Abort): This bit indicates that frame transmission has been aborted.

UN (Underrun): This bit indicates that an underrun error caused a frame to be aborted. The following transmit frames will transmit normally. Only the frame delimited by the descriptors when the underrun occurred is discarded.

PE (Parity Error): This bit indicates that transmission of the frame was aborted because of a parity error.

NOTE

If AB = 1, UN = PER = 0 implies that the decision to abort frame transmission is due to the assertion of the TABORT signal.

E.13 FIRST OR LAST BIT ERRORS

The first descriptor of a frame should have the FIRST bit set, and the last descriptor in a frame should have the LAST bit set. Commands and transmit commands using only one descriptor to define an entire frame should have both the FIRST and LAST bits set. With these restrictions, the following error situations may occur:

a. The first descriptor might be detected when the last descriptor of the previous frame is still expected.
b. The FIRST bit may be zero when the first descriptor of a frame or a command is expected.
c. A command entry may have the LAST bit set to zero.

In all of the above cases, the reaction of the FSI is as follows:

a. The transfer is immediately stopped. The entry that caused the error is not transferred to the internal FSI ring FIFO. The ring read pointer will point to the beginning of this entry.
b. The ring state is changed to empty with the operation error (OER) bit set in the appropriate ring state register (RSR).
c. The RNR and RER bits in the SR1 are set, causing an interrupt if they are enabled.

To recover from this error, the host processor should:

a. Clear the RER and RNR bits in the SR1 by writing the SR1 with these bits set.
b. Change the error entry to a valid entry. The host processor can locate the error by reading the ring read pointer by means of the READ RING PARAMETERS command.
c. Transfer the ring to the ready state by performing a ring ready control access.

If the error is severe and the ring should be redefined, the host processor should first issue the STOP command to transfer the ring to a stopped state, and then redefine the ring.

INDEX

— D —

—R—

— S —

— T —

1ATX31415-0 Printed in USA 6/94 COURIER UM 18106 5,000 MPU YGABAA

Literature Distribution Centers:
USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan.
ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.