PM3390

8 PORT GIGABIT EXACT BUS SWITCHING MATRIX

DATASHEET

PRELIMINARY

ISSUE 4: JANUARY 1999

CONTENTS

LIST OF REGISTERS

LIST OF FIGURES

1 FEATURES

System Features

- Single Chip 8 port EXACT™ Bus switch matrix.
- Provides a non-blocking connection between 8 EXACT Bus rings that support up to 16 Gbit/s aggregate system bandwidth.
- Supports expansion for a non-blocking connection between sixteen EXACT Bus rings that provide up to 32 Gbit/s aggregate system bandwidth using four PM3390 devices.
- Simplifies design and manufacturing of high-density non-blocking 10/100/1000 Ethernet Switch products.
- The EXACT Bus is a flexible gigabit full-duplex bus that allows simple chip-to-chip, card-to-card, or box-to-box interconnection.
- The EXACT Bus can operate in a point-to-point or insertion ring configuration.

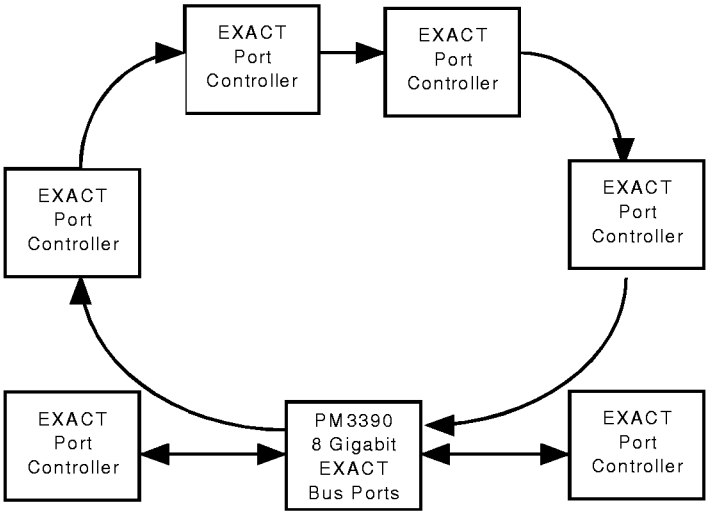


FIGURE 1 EXACT RING CONFIGURATION

- Each PM3390 port can support one EXACT Bus switch port controller such as the PM3370 or PM3371 (Octal 10/100 Ethernet Switch Port Controllers) or one PM3380 (Gigabit Ethernet Switch Port Controller) device in a point-to-point connection. Alternatively, multiple EXACT Bus switch port controllers can be connected in a ring to the PM3390.

- Supports non-blocking configurations up to 128 x 10/100 ports or 16 Gigabit Ethernet ports.
- Utilizes Time and space division switching to ensure a non-blocking datapath from any port to any port.
- Address lookup maps logical ports of an EXACT system to physical ports of the PM3390. Address lookup map entries can be modified dynamically to change data paths through the PM3390.
- Optional cut through operation to reduce latency for small EXACT message transmission.
- Optional 8B/10B encoding on the EXACT bus allows compatiblity with common SERDES devices to provide serial extension using coax or twinax cable.
- Management interface that allows control register access from an external microcontroller or via EXACT Bus messages.
- Pinstraps to easily configure clock frequency and data format for EXACT Bus outputs.
- Provides a standard 5 signal P1149.1 JTAG test port for boundary scan test purposes.
- Low power 0.35u CMOS technology with a single 3.3V power supply.
- 352- pin ball-grid array (BGA) Package.

2 APPLICATIONS

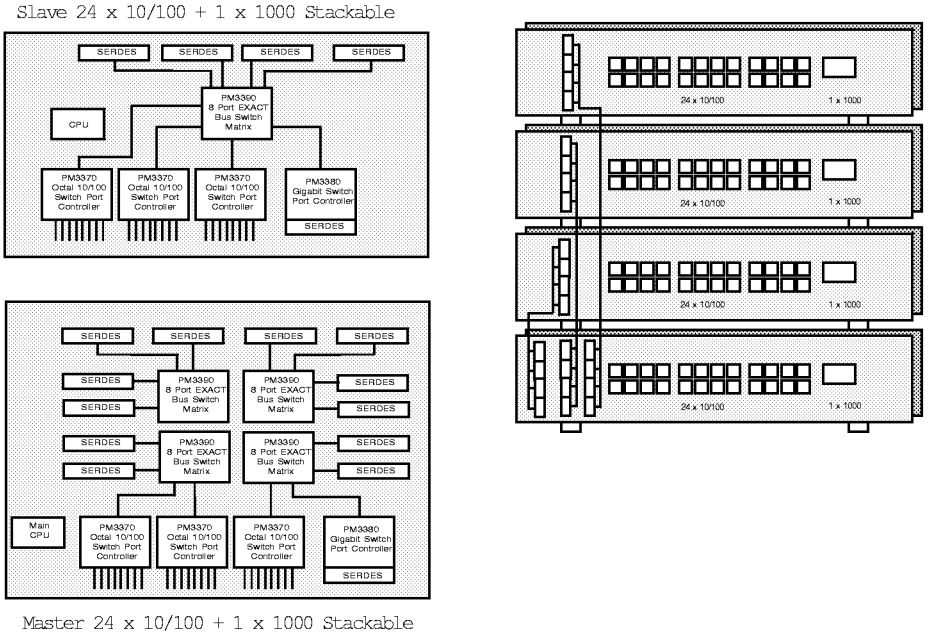
The PM3390 is a non-blocking Ethernet Switching Matrix that contains connections to and between 8 EXACT™ Bus rings. The PM3390 is intended as the backplane connection for Ethernet switching systems with up to 32 Gbit/s aggregate bandwidth requirements. However, the PM3390 is not protocol dependent which allows the device to be used in applications other than Ethernet.

- 10/100/1000 Ethernet Switches
- Layer 3 Ethernet Switches
- Routers

A typical application where the PM3390 can be used is shown in Figure 2. Port controllers such as the PM3380 (1x1000 Mbit/s port controller) and the PM3370 (8x10/100 port controller) use the EXACT Bus for interconnecting large Ethernet Switch Systems.

A block diagram of a non-blocking 96 port 10/100 Mbit/s with four 1 Gbit/s uplinks is shown.

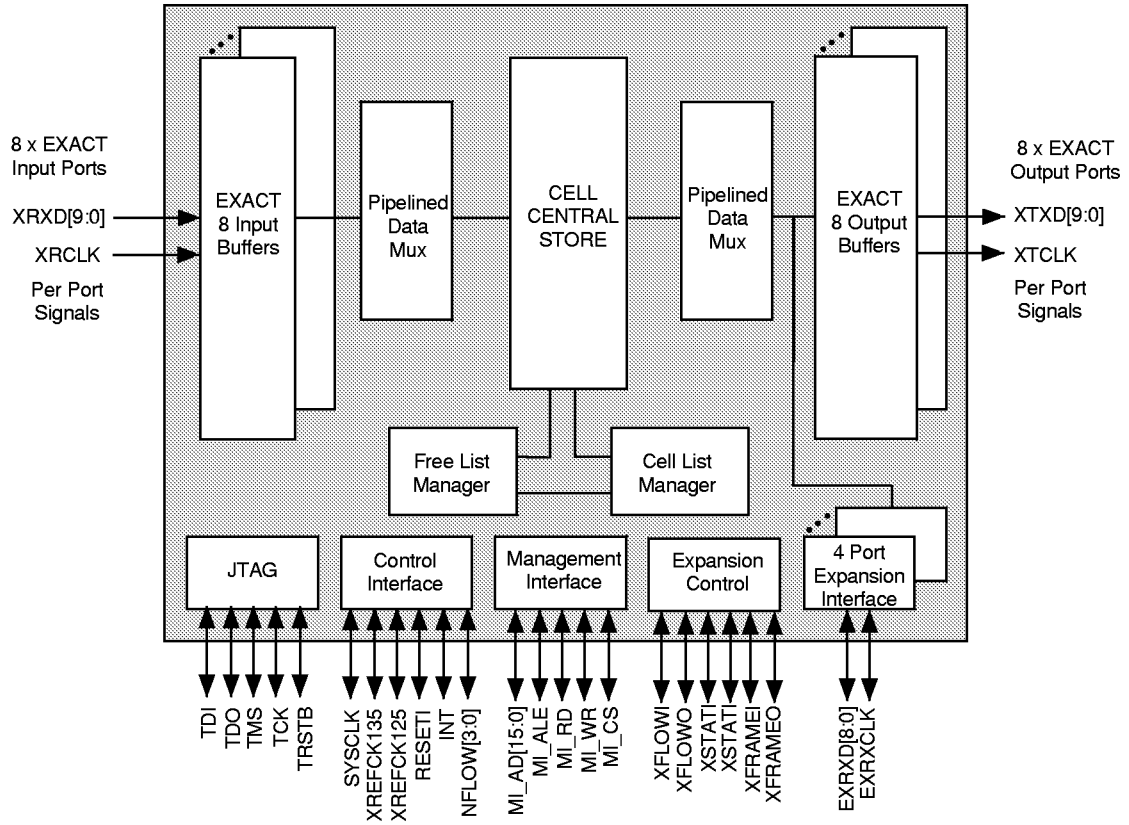
FIGURE 2 96 + 4 NON-BLOCKING ETHERNET SWITCH



3 REFERENCES

- PMC-970215 EXACT Bus Protocol Specification
- PMC-970862 PM3380 Gigabit Ethernet Switch Port Controller Datasheet
- PMC-970861 PM3370 8 x 10/100 Mbit/s Ethernet Switch Port Controller Datasheet
- PMC-980713 EXACT Chipset Control and Management Interface Protocol Specification

4 BLOCK DIAGRAM



5 DESCRIPTION

The PM3390 EXACT Bus Switching Matrix is a monolithic integrated circuit that provides non-blocking connections between 8 x 1000 Mbit/s EXACT Bus rings. The PM3390 uses the EXACT Bus protocol for Physical and Link layer communications within a system. This allows protocol independence for the transport of data through the switch matrix.

The PM3390 is intended as the backplane connection for Ethernet switching systems with up to 32 Gbit/s bandwidth requirements using port controllers that support the EXACT Bus protocol. Devices such as the PM3370 (8 x 10/100 switch port controller) and the PM3380 (1 x 1000 switch port controller) can be used to create 10/100/1000 Ethernet switches.

The PM3390 implements 8 input EXACT Bus interfaces, a central store memory, address resolution and 8 output EXACT Bus interfaces. Each EXACT Bus interface can operate at 125MHz or 135 MHz. 135MHz operation may be used with the PM3380 devices to provide more bandwidth for EXACT control messages while maintaining wire-speed data forwarding.

The PM3390 contains 8 EXACT Bus Input Buffers that each receive the data stream, slice the data into 32-byte cells, code the destination address and sends the cells to the central store memory.

The central store memory used the time sliced data cells and then space slices the cells. This cell-based time and space sliced datapath is used to provide a non-blocking switching matrix with efficient memory utilization.

The PM3390 contains 8 EXACT Bus Output Buffers that retrieve and recombine data from the central store for transmission across the output EXACT Bus.

An expandability interface allows a larger, 16 x16 switching matrix to be built. Four 8 x 8 PM3390 devices may be combined to provide a non-blocking 32 Gbit/s aggregate bandwidth switch core.

The PM3390 can be configured and monitored by a microprocessor through the Management Interface or from a remote agent via the EXACT Bus.

6 PIN DIAGRAM

352 SBGA

	A	B	C	D	E	F	G	H	J	K	L	M	N
1	Vss	vss	NC	NC	rx0d3	rx0d0	tx0d9	tx0d6	tx0d4	tx0d1	NC	NC	vss
2	vss	vdd	vss	rx0d9	rx0d6	rx0d2	NC	NC	NC	NC	NC	tx1d9	tx1d7
3	rx1d1	vss	vdd	tdi	rx0d8	rx0d5	NC	tx0clk	tx0d8	tx0d5	tx0d2	tx1clk	tx1d8
4	rx1d5	rx1d2	tdo	vdd	rx0clk	rx0d7	rx0d4	rx0d1	vdd	tx0d7	tx0d3	tx0d0	vdd
5	rx1d9	rx1d6	rx1d3	rx1d0									
6	rx2d2	rx1clk	rx1d7	rx1d4									
7	rx2d5	rx2d3	rx2d0	rx1d8									
8	rx2d9	rx2d6	rx2d4	rx2d1									
9	rx3d1	rx2clk	rx2d7	vdd									
10	rx3d5	rx3d3	rx3d0	rx2d8									
11	rx3d8	rx3d6	rx3d4	rx3d2									
12	rx4d0	rx3clk	rx3d9	rx3d7									
13	vss	rx4d3	rx4d2	rx4d1									
14	vss	rx4d4	rx4d5	vdd									
15	rx4d6	rx4d7	rx4d8	rx4clk									
16	rx4d9	rx5d0	rx5d2	rx5d4									
17	rx5d1	rx5d3	rx5d6	rx5d9									
18	rx5d5	rx5d7	rx5clk	vdd									
19	rx5d8	rx6d0	rx6d2	rx6d5									
20	rx6d1	rx6d3	rx6d6	rx6d9									
21	rx6d4	rx6d7	rx6clk	rx7d2									
22	rx6d8	rx7d0	rx7d3	rx7d6									
23	rx7d1	rx7d4	rx7d7	vdd	rx7d9	ad14	ad10	ad7	vdd	ad1	mi_rd	test_se	nflow2
24	rx7d5	vss	vdd	rx7d8	NC	ad11	ad8	ad4	ad2	mi_wr	sysclk	test_ mode	nflow1
25	vss	vdd	vss	ad15	ad12	NC	ad5	NC	mi_ale	mi_cs	xrefck 135	reseti	NC
26	vss	vss	rx7clk	ad13	ad9	ad6	ad3	ad0	NC	xrefck 125	vbias5_ 1	nflow3	vss
	A	B	C	D	E	F	G	H	J	K	L	M	N

P	R	T	U	V	W	Y	AA	AB	AC	AD	AE	AF	
vss	tx1d5	tx1d3	NC	tx2d9	tx2d7	NC	NC	tx3clk	tx3d7	tx3d4	vss	vss	1
NC	tx1d4	tx1d1	NC	tx2d8	tx2d5	tx2d3	tx2d0	tx3d8	tx3d5	vss	vdd	vss	2
tx1d6	NC	tx1d0	NC	tx2d6	tx2d4	tx2d1	NC	NC	tx3d2	vdd	vss	tx4clk	3
vbias5_2	tx1d2	tx2clk	NC	vdd	tx2d2	tx3d9	tx3d6	tx3d3	vdd	tx3d1	tx4d9	tx4d6	4
									tx3d0	tx4d8	tx4d5	NC	5
									tx4d7	tx4d4	tx4d2	xstato	6
									tx4d3	tx4d1	xframeo	NC	7
									tx4d0	xstati	xframei	ex3rxck	8
									vdd	xflowo	ex3rxd8	ex3rxd7	9
									xflowi	NC	ex3rxd5	ex3rxd3	10
									ex3rxd6	ex3rxd4	ex3rxd2	ex3rxd1	11
									NC	ex3rxd0	ex2rxck	ex2rxd8	12
									vdd	ex2rxd7	ex2rxd6	vss	13
									ex2rxd4	NC	ex2rxd5	vss	14
									ex1rxck	ex2rxd1	ex2rxd2	ex2rxd3	15
									ex1rxd5	NC	ex1rxd8	ex2rxd0	16
									ex1rxd0	ex1rxd3	ex1rxd6	ex1rxd7	17
									vdd	ex0rxck	ex1rxd2	ex1rxd4	18
									ex0rxd4	ex0rxd7	NC	ex1rxd1	19
									NC	ex0rxd3	ex0rxd6	ex0rxd8	20
									tx5d8	ex0rxd0	ex0rxd2	ex0rxd5	21
									tx5d4	tx5d7	tx5clk	ex0rxd1	22
vdd	tx7d0	tx7d4	tx7d8	vdd	tx6d3	tx6d6	NC	tx5d1	vdd	tx5d3	tx5d6	tx5d9	23
tms	NC	tx7d3	tx7d6	tx7d9	tx6d0	NC	tx6d7	tx6clk	tx5d2	vdd	vss	tx5d5	24
nflow0	trstb	tx7d1	NC	NC	NC	tx6d1	tx6d4	tx6d8	tx5d0	vss	vdd	vss	25
vss	tck	int	tx7d2	tx7d5	tx7d7	tx7clk	tx6d2	tx6d5	tx6d9	NC	vss	vss	26
P	R	T	U	V	W	Y	AA	AB	AC	AD	AE	AF	

7 PIN DESCRIPTION

EXACT Bus Interfaces

Name	Size	Type	Toler-ant	Pin Numbers	Description
RX0D(9:0)	10	I	3.3V	D2, E3, F4, E2, F3, G4, E1, F2, H4, F1	Receive data for EXACT ring 0
RX0CLK	1	I	3.3V	E4	Receive clock for EXACT ring 0
TX0D(9:0)	10	O	3.3V	G1, J3, K4, H1, K3, J1, L4, L3, K1, M4	Transmit data for EXACT ring 0
TX0CLK	1	O	3.3V	H3	Transmit clock for EXACT ring 0
RX1D(9:0)	10	I	3.3V	A5, D7, C6, B5, A4, D6, C5, B4, A3, D5	Receive data for EXACT ring 1
RX1CLK	1	I	3.3V	B6	Receive clock for EXACT ring 1
TX1D(9:0)	10	O	3.3V	M2, N3, N2, P3, R1, R2, T1, R4, T2, T3	Transmit data for EXACT ring 1
TX1CLK	1	O	3.3V	M3	Transmit clock for EXACT ring 1

RX2D(9:0)	10	I	3.3V	A8, D10, C9, B8, A7, C8, B7, A6, D8, C7	Receive data for EXACT ring 2
RX2CLK	1	I	3.3V	B9	Receive clock for EXACT ring 2
TX2D(9:0)	10	O	3.3V	V1, V2, W1, V3, W2, W3, Y2, W4, Y3, AA2	Transmit data for EXACT ring 2
TX2CLK	1	O	3.3V	T4	Transmit clock for EXACT ring 2
RX3D(9:0)	10	I	3.3V	C12, A11, D12, B11, A10, C11, B10, D11, A9, C10	Receive data for EXACT ring 3
RX3CLK	1	I	3.3V	B12	Receive clock for EXACT ring 3
TX3D(9:0)	10	O	3.3V	Y4, AB2, AC1, AA4, AC2, AD1, AB4, AC3, AD4, AC5,	Transmit data for EXACT ring 3
TX3CLK	1	O	3.3V	AB1	Transmit clock for EXACT ring 3
RX4D(9:0)	10	I	3.3V	A16, C15, B15, A15, C14, B14, B13, C13, D13, A12	Receive data for EXACT ring 4
RX4CLK	1	I	3.3V	D15	Receive clock for EXACT ring 4

TX4D(9:0)	10	O	3.3V	AE4, AD5, AC6, AF4, AE5, AD6, AC7, AE6, AD7, AC8	Transmit data for EXACT ring 4
TX4CLK	1	O	3.3V	AF3	Transmit clock for EXACT ring 4
RX5D(9:0)	10	I	3.3V	D17, A19, B18, C17, A18, D16, B17, C16, A17, B16	Receive data for EXACT ring 5
RX5CLK	1	I	3.3V	C18	Receive clock for EXACT ring 5
TX5D(9:0)	10	O	3.3V	AF23, AC21, AD22, AE23, AF24, AC22, AD23, AC24, AB23, AC25	Transmit data for EXACT ring 5
TX5CLK	1	O	3.3V	AE22	Transmit clock for EXACT ring 5
RX6D(9:0)	10	I	3.3V	D20, A22, B21, C20, D19, A21 B20, C19, A20, B19	Receive data for EXACT ring 6
RX6CLK	1	I	3.3V	C21	Receive clock for EXACT ring 6
TX6D(9:0)	10	O	3.3V	AC26, AB25, AA24, Y23, AB26, AA25, W23, AA26, Y25, W24	Transmit data for EXACT ring 6
TX6CLK	1	O	3.3V	AB24	Transmit clock for EXACT ring 6

RX7D(9:0)	10	I	3.3V	E23, D24, C23, D22, A24, B23, C22, D21 A23, B22	Receive data for EXACT ring 7
RX7CLK	1	I	3.3V	C26	Receive clock for EXACT ring 7
TX7D(9:0)	10	O	3.3V	V24, U23, W26, U24, V26, T23, T24, U26, T25, R23	Transmit data for EXACT ring 7
TX7CLK	1	O	3.3V	Y26	Transmit clock for EXACT ring 7

Expandability Interface

Name	Size	Type	Toler-ant	Pin Numbers	Description
XFLOWI	1	I	3.3V	AC10	Input serial flow control stream from master PM3390 to slave PM3390.
XFLOWO	1	O	3.3V	AD9	Output serial flow control stream from master PM3390 to slave PM3390
XSTATI	1	I	3.3V	AD8	Input EXACT ring status from slave PM3390 to master PM3390.
XSTATO	1	O	3.3V	AF6	Output EXACT ring status from slave PM3390 to master PM3390.
XFRAMEI	1	I	3.3V	AE8	Input framing bit for EXPIF status information from slave PM3390 to master PM3390.
XFRAMEO	1	O	3.3V	AE7	Output framing bit for EXPIF status information from slave PM3390 to master PM3390.
EX0RXCLK	1	I	3.3V	AD18	Shared transmit clock from slave PM3390 to master PM3390 for Expansion port 0.

EX0RXD(8:0)	9	I	3.3V	AF20, AD19, AE20, AF21, AC19, AD20, AE21, AF22, AD21	Shared transmit data from slave PM3390 to master PM3390 for Expansion port 0.
EX1RXCLK	1	I	3.3V	AC15	Shared transmit clock from slave PM3390 to master PM3390 for Expansion port 1.
EX1RXD(8:0)	9	I	3.3V	AE16, AF17, AE17, AC16, AF18, AD17, AE18, AF19, AC17	Shared transmit data from slave PM3390 to master PM3390 for Expansion port `.
EX2RXCLK	1	I	3.3V	AE12	Shared transmit clock from slave PM3390 to master PM3390 for Expansion port 2.
EX2RXD(8:0)	9	I	3.3V	AF12, AD13, AE13, AE14, AC14, AF15, AE15, AD15, AF16	Shared transmit data from slave PM3390 to master PM3390 for Expansion port 2.
EX3RXCLK	1	I	3.3V	AF8	Shared transmit clock from slave PM3390 to master PM3390 for Expansion port 3.
EX3RXD(8:0)	9	I	3.3V	AE9, AF9, AC11, AE10, AD11, AF10, AE11, AF11, AD12	Shared transmit data from slave PM3390 to master PM3390 for Expansion port 3.

Microprocessor Interface

Name	Size	Type	Toler-ant	Pin Numbers	Description
MI_AD (15:0)	16	B	5V	D25, F23, D26, E25, F24, G23, E26, G24, H23, F26, G25, H24, G26, J24, K23, H26	Multiplexed address, data
MI_ALE	1	I	5V	J25	Address latch enable
MI_RD	1	I	5V	L23	Read strobe (low-true). De-assertion of MI_RD will cause the PM3390 to tri-state the MI_AD[15:0] bus.
MI_WR	1	I	5V	K24	Write strobe(low-true).
MI_CS	1	I	5V	K25	Chip select (low-true). De-assertion of MI_CS during a read operation will cause the PM3390 to tri-state the MI_AD[15:0] bus.

Miscellaneous

Name	Size	Type	Toler-ant	Pin Numbers	Description
SYSCLK	1	I	3.3V	L24	System clock; must be set to one-half the maximum EXACT Bus TXCLK being used; (i.e. if the 135 MHz clock is being used, SYSCLK is 67.5 MHz; if only the 125 MHz clock is being used, SYSCLK could be either 62.5 MHz or 67.5 MHz).
XREFCK135	1	1	3.3V	L25	EXACT Bus reference clock (135 Mhz)

XREFCK125	1	I	3.3V	K26	EXACT Bus reference clock (125 Mhz)
RESETI	1	I	5V	M25	Master reset in. This signal is active high.
INT	1	O	3.3V	T26	Interrupt out (high true, always driven). May be used by an external device to indicate occurrence of an interrupt condition. See CMIF register 0x02 for additional details.
NFLOW (3:0)	4	O	3.3V	M26, N23, N24, P25	These outputs provide flow-control information for devices supplying data to the PM3390. These pins can be left as no-connects if th PM3390 is used in an EXACT-based Ethernet switch system.

JTAG and Test pins

TEST_MODE	1	I	5V	M24	Tie logic low: used by PMC-Sierra for test purposes.
TEST_SE	1	I	5V	M23	Tie logic low: used by PMC-Sierra for test purposes.
TDI	1	I	5V	D3	JTAG Test Input. When not in use for JTAG testing, this pin must be logic low.
TDO	1	O	5V	C4	JTAG Test Output.
TMS	1	I	5V	P24	JTAG Test Mode Select. When not in use for JTAG testing, this pin must be logic low.
TCK	1	I	5V	R26	JTAG Test Clock. When not in use for JTAG testing, this pin must be logic low.
TRSTB	1	I	5V	R25	JTAG Test Reset. When not in use for JTAG testing, this pin must be **logic high**.

Power and Grounds

Name	Size	Type	Pin Numbers	Description
VDD			B2, B25, C3, C24, D4, D9, D14, D18, D23, J4, J23, N4, P23, V4, V23, AC4, AC9, AC13, AC18, AC23, AD3, AD24, AE2, AE25,	3.3V power
VSS			A1, A2, A13, A14, A25, A26, B1, B3, B24, B26, C2, C25, N1, N26, P1, P26, AD2, AD25, AE1, AE3, AE24, AE26, AF1, AF2, AF13, AF14, AF25, AF26	Ground

VBIAS			L26, P4	+5V Bias pins. The VBIAS inputs are used to implement 5V tolerance on the inputs. VBIAS pins must be connected to a well decoupled +5V rail if 5V tolerant inputs are required. If 5V tolerant inputs are not required, VBIAS pins must be connected to a well-decoupled 3.3V DC supply together with the power pins.

Notes on Pin Description:

1. To avoid damage to the device, during power-up, and power-down the voltage on the VBIAS pin must be kept equal to or greater than the voltage on the pad ring and core power VDD pins.

2. No-Connect and provisional connection rules.
   Pins can be grouped according to the following rules:

   - input only pins
   - output only pins
   - bidirectional pins.

Input only pins

Unused input pins must be pulled either high or low through a resistor. The general rule is to pull the input to its logically deasserted state. If a group of inputs is never to be used then they can be tied together and pulled through a single resistor to their logically deasserted state. Up to 11 input pins can be pulled through a single 4.7K ohm resistor to either VDD or GND; up to 22 input pins can be pulled through a 1K ohm resistor to either VDD or GND.

If an input is to be conditionally used (that is, "provisional usage"), then each input must have a dedicated resistor pulling the input to its logically de-asserted state.

Output only pins

Output pins that are No-Connects can be left floating.

Bi-directional pins

Bi-directional pins that are NO CONNECTS can be left floating.

The following table provides examples for no-connect and provisional connection:

Condition	Connection description
JTAG unused	Tie TDI, TMS, TCK, TEST_MODE, TEST_SE to GND through 4.7K resistor. Tie TRSTB to VDD through 4.7K resistor.
Unused EXACT RX/TX port. Example: RX/TX port 7.	Tie RX7D[9:0] and RX7CLK together, connect to a single 4.7K resistor tied to GND.
Provisional usage of EXACT RX/TX port. Example: RX/TX port 7. in a plug-in module.	Place a 4.7K resistor on each of the RX7D[9:0] and RX7CLK lines to GND. The resistor must be placed as close as possible to the input pin on the SBGA package to minimize its affect on the transmission line characteristics of the EXACT bus.
Microprocessor interface unused	Tie MI_RD, MI_WR, MI_CS, and MI_ALE to VDD through a single 4.7K resistor.

8 FUNCTIONAL DESCRIPTION

8.1 Architecture Overview

The PM3390 is a non-blocking 8 x 8 EXACT Bus switching matrix that can be expanded to a non-blocking 16 x 16 matrix by using four PM3390 devices. The important architectural features of the design are:

- Time and space sliced datapath,
- Output-based data streams,
- Expandability interface.

8.1.1 Time and Space Sliced Datapath

The PM3390 provides connections to eight 1 Gbit/s data streams. If the input and output sections of the PM3390 are considered independently, then to fulfill the purpose of the device, the architecture described can receive up to eight 1 Gbit/s input streams and supply up to eight 1 Gbit/s output streams without the loss on any data.

There are pathological input/output patterns that focus input traffic at a particular output or require multiple output traffic streams to be sourced from a particular input. To provide a non-blocking switching matrix, a data storage mechanism is implemented within the matrix to avoid traffic loss during these cases of focused traffic. A dynamically allocated memory (a large central memory) is implemented as the data stream store, to reduce the memory requirements and to solve these problems. Alternative memory structures such as a static memory storage scheme (individual input/output FIFOs) would require more storage than is economical to implement.

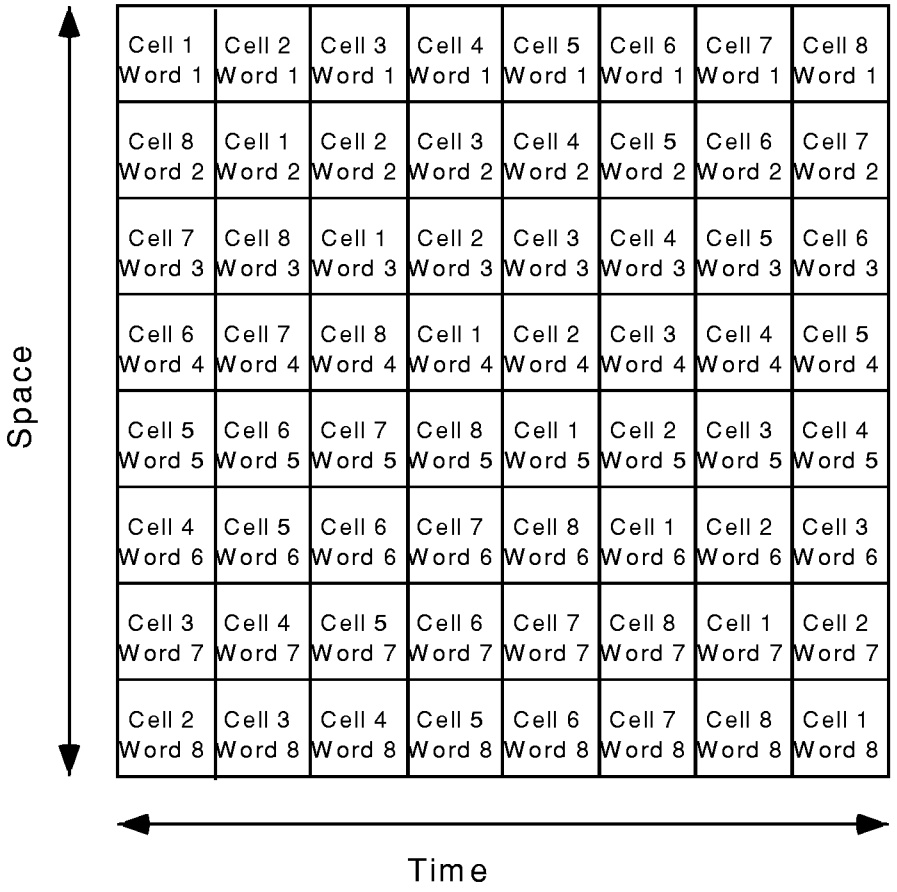
With a central memory resource in the data path, contention through this single resource must be resolved. Each input and each output is guaranteed a path to and from the store without blocking. This is achieved by slicing the input data streams both in time and space and recombining them before the streams exit the device.

The dimensions of a switching matrix define the number of data streams that it can support. If a switching matrix can support n data streams, to take advantage of the architecture, the data store is divided into *n* independent storage arrays. Additionally, each data stream must be divided into *n* equal-sized pieces and each piece written into the *n* storage arrays over the course of *n* time units. By writing the *n* pieces of the *n* data

streams into *n* storage array over the course of *n* time units, each input data stream will have an independent and guaranteed time/space slice of the switching matrix.

In the above figure, the slicing of the data and the data store is shown for n = 8.

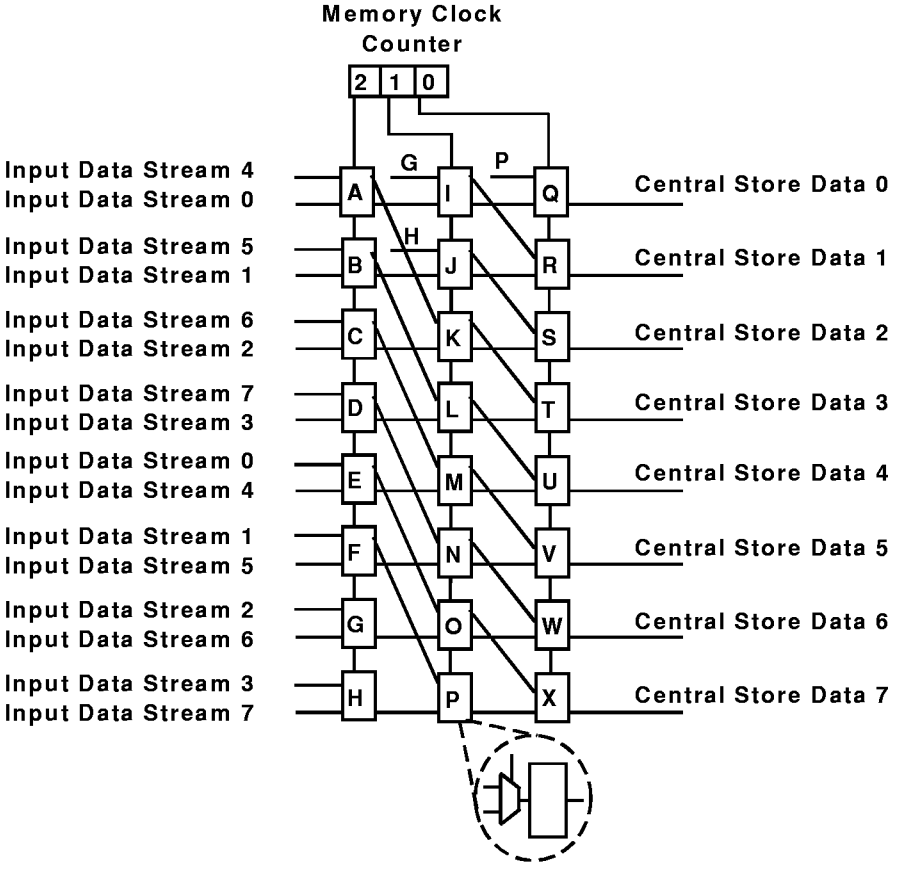
Slicing is accomplished at several points in the data path. The central store



is divided into eight separate banks (one for each input/output stream connected to the device), and the input/output data streams are separated into eight pipelined pieces (one for each memory bank). For these pipelined pieces to be the same size, the incoming stream is also divided into constant-sized pieces. At the input stage in the device, the input stream is divided into 32-byte cells and these cells are used throughout the device.

Once the space (the central store) and the time (pipelined input streams) have been sliced, the only problem left to solve is how to write/read these streams to/from the central store. Both time and space are sliced into a

data stream number of pieces. So, at each time slice, a data element must be read/written into its corresponding space slice. The time to write/read a cell into the central store (hereafter referred to as a cell clock) is divided into *n* separate memory cycles where the above mentioned data/time slice memory transfers take place. Because all input data stream are divided into *n* parts that are each written into a separate bank, all cell transfers require a write/read of all *n* memory banks. These write/reads are coordinated such that each input/output stream is being written to /read from a separate bank of the central store on each memory clock, but that all will have written to /read from each bank at the completion of a cell clock.



The mechanism used to supply the *n* data slices to the *n*-way central store is a $\log_2 n$-stage barrel shifter. This scheme provides each input/output with a continuous, non-blocking path to the central store, which is the goal of the PM3390 space-time sliced architecture.

8.1.2 Output-based Data Streams

Once the input data streams have been placed into the central store, they must be routed to the appropriate output ring. The banks of the central store are composed of cell locations. The used/unused status of these cell locations is maintained in a free list. As these locations are used to store input cells, the locations are linked together based on their destination ring.

Each output ring has an associated linked list that corresponds to the cells that must be transferred onto its EXACT ring. The Cell List Manager (CLM) is responsible for maintaining these lists and supplying addresses to the central store for outbound data. By managing the lists in this fashion, efficient use of the central store is realized.

8.1.3 Expandability Interface

The PM3390 switching matrix is a building block that can be used to create larger port count and higher bandwidth systems. An Expandability Interface (EXPIF) has been added to the PM3390 to allow these larger systems to remain non-blocking and to prevent any electrical problems that could arise from a “bus sharing” expandability solution. The standard expandability option will allow four PM3390 devices to be configured into a 16 x 16 switching matrix. The four PM3390 devices are split into two pairs and each of the pairs communicates via the EXPIF to provide the desired functionality.

Under normal operation the eight input and eight output ports form an 8 port switching matrix. When the PM3390 is configured for expandability, two PM3390 are interconnected and the eight ports are grouped into two sets of four ports. The four transmit ports that are configured in Expansion Mode are connected to the four Expansion Interface receive ports on the second PM3390 device (and vice-versa). Creation of a non-blocking 16x16 switching matrix is achieved by interconnecting the two sets each of two PM3390 devices (so there are a total of four PM3390 devices).

8.1.4 EXACT Bus Physical Layer Coding

The PM3390 input and output interfaces are based on the EXACT Bus Specification. Please refer to document PMC-970215, section 4 (Physical Layer Coding and Flow Control) for a discussion of the physical layer coding that is used on the PM3390. There are two types of coding that can be selected: either 8B10B encoding or a binary-encoded method (referred

to as Clear Channel). The encoding is selectable on a per-port basis during the initialization sequence following reset assertion.

For discussion purposes, assume for the remainder of this paragraph that a given receive port on the PM3390 is configured to accept 8B10B encoded characters. The stream of 8B10B characters received by the PM3390 can be classified by the receive logic into two classes: comma characters (referred to as EXACT bus delimiter characters) and non-comma characters (which are interpreted as EXACT message characters). There is no synchronization performed on the incoming character stream on a message-by-message basis. The on-chip 8B/10B receive logic that is part of the PM3390 receiver uses the detection of the comma character (i.e. K28.5) in the data stream to delimit EXACT messages. A "synchronization" procedure is followed once, upon start-up, which insures that some configurable number of *consecutive* comma characters are seen in the incoming data stream before the PM3390 receive logic will start forwarding incoming EXACT messages.

A similar procedure is followed if the input receive logic had been configured to operate on a binary-encoded character stream (i.e. Clear Channel mode).

The PM3390 device handles EXACT bus delimiter characters as follows:

1. On the EXACT receive ports (XIBs), delimiter characters are used only to determine the beginning and end of EXACT messages. Although the EXACT bus specification states "it is mandatory for an odd number of characters to be inserted between every pair of IDLE or BUSY characters" the PM3390 input receive port will process messages that do not follow this requirement. However, failure to conform to this requirement can lead to an internal overflow condition on the receive ports if the condition is sustained.

2. Internal to the PM3390 device, the delimiter characters are stripped.

3. On the PM3390 EXACT transmit ports the delimiter characters are re-inserted. The insertion logic guarantees that the EXACT bus specification is met: there will be an odd number of characters inserted between every pair of IDLE or BUSY characters.

8.1.5 EXACT Bus Clock Modes

The EXACT bus supports two modes of clock operation: SERDES and non-SERDES.

8.1.5.1 SERDES mode clocking

SERDES is an acronym for Serializer/de-serializer mode. Typically this mode is used to interface to a high-speed Serializer/de-serializer transceiver. Data is encoded using the 8B10B linecode. Receive data (RXD[9:0]) is clocked into the receive port of the PM3390 from the SERDES device on both the rising edge and falling edge of the RXCLK signal. Transmit data (TXD[9:0]) from the PM3390 is valid on the rising edge only of the PM3390 TXCLK signal. So for a 1 Gbit SERDES device, the RXCLK frequency will be 62.5 MHz while the TXCLK frequency will be 125 MHz. Alternatively, the RXCLK can be 67.5 MHz for a TXCLK clock of 135 MHz.

Many SERDES devices include complementary receive clocks. Either clock can be connected to the RXCLK input of the PM3390 device as long as the clock and data relationship meet the AC timing requirements given in the AC section of this data sheet. Either of the complementary receive clocks on the SERDES device can be connected to the RXCLK input because the on-chip 8B/10B receive logic that is part of the EXACT interface logic only uses the detection of the comma character (i.e. K28.5) in the data stream to delimit EXACT messages.

For reference purposes, the PM3575 24x10/100 + 2x1000 Ethernet switch reference design (uses the PM3390, PM3370, and PM3380 devices) connects the RBC0 clock from the HP SERDES device (part number HDMP 1636) to the RXCLK input of the PM3390.

8.1.5.2 non-SERDES mode clocking

Receive data (RXD[9:0]) is clocked into the receive port of the PM3390 from the upstream device on both the rising edge and falling edge of the RXCLK signal. Transmit data (TXD[9:0]) from the PM3390 is valid on both the rising edge and falling edge of the PM3390 TXCLK signal. So for 1 Gbit bit-rate transfer, the RXCLK frequency will be 62.5 MHz while the TXCLK frequency will also be 62.5 MHz. Alternatively, the RXCLK can be 67.5 MHz for a TXCLK clock of 135 MHz.

8.1.5.3 CLOCK and ENCODING modes

The PM3390 supports a total of four clock and data encoding modes:

1. SERDES mode

Data is 8B10B encoded, TXCLK at twice the rate of RXCLK. Both RX and TX busses of a given port are configured to be in the same mode.

2. 8B10B non-SERDES mode

Data is 8B10B encoded, TXCLK at same rate as RXCLK. Both RX and TX busses of a given port are configured to be in the same mode.

3. Clear Channel mode

Data is binary-encoded as per EXACT Bus specification, TXCLK at same rate as RXCLK. Both RX and TX busses of a given port are configured to be in the same mode.

4. Expansion mode

This is a variation of Clear Channel mode and is used only between the Transmit output port of one PM3390 device and the Expansion Receive Data port of a different PM3390 device. On both devices the Expansion mode interface configuration bit must be set. The Data is binary-encoded. TXCLK at same rate as RXCLK.

Only the TX bus of a given port is configured to be in this mode. The RX bus of the port can be independently configured to be in either SERDES, 8B10B non-SERDES, or Clear Channel modes. This mode of operation for a Transmit port is selected by having the Expansion Interface Mode bit set for that port (see CMIF register 0x01 for additional details).

Refer to the configuration section of this document on how to select the clock and data encoding mode on a per-port basis.

8.1.6 EXACT Bus Messages

The PM3390 supports 8 input and 8 output EXACT Bus ports. The EXACT Bus is a gigabit bus that is used to connect ports controllers (PM3380, 1x1000 and PM3370/PM3371, 8x10/100) and the PM3390 switch matrix in an Ethernet switch. Please refer to the EXACT Bus Protocol Specification document for more details.

The EXACT protocol partitions Ethernet frames into multiple 240 byte messages within the port controllers. This document refers to these messages, including system control messages as EXACT Messages. These EXACT messages are then transmitted out on the EXACT Bus ring for distribution to other port controllers as required.

8.1.7 Major Functional Blocks

The PM3390 is partitioned into the following major functional blocks. The operation of each block is described in more detail in subsequent sections.

Block	Function
XIB	EXACT Bus Input Buffering
PDX	Pipelined Data Multiplexer
CCS	Cell Central Store
FLM	Free List Manager
CLM	Cell List Manager
XOB	EXACT Bus Output Buffering
CMIF	Core Management Interface

8.2 EXACT Bus Input Buffer

The EXACT Bus Input Buffer (XIB) receives a stream of 1 Gbit/s EXACT messages from an EXACT ring. The XIB must:

1. Decode the incoming EXACT data in either 8B/10B encoded or in Clear Channel format.

2. Strip the destination address from incoming messages and send it to the Core Management Interface (CMIF) for destination lookup.

3a. For messages not destined to the PM3390, gather incoming EXACT Bus messages into as much of a 32-byte cell as is possible given traffic patterns, and deliver the 32-byte cells to the Pipelined Data Multiplexer to be written into the Cell Central Store.

3b. For messages destined to the PM3390 (EXACT Control messages or EXACT Broadcast messages), deliver the message data to the Core Management Interface (CMIF) for processing.

There is one XIB for each of the eight EXACT rings that the PM3390 supports.

8.2.1 EXACT Bus Interface

Data enters the PM3390 through the EXACT Bus receive port (XIB). The incoming data is encoded according to the EXACT Bus Protocol Specification Document. The XIB must sample the incoming data in either 8B/10B format or in Clear Channel format. The incoming data is received at 62.5 MHz or 67.5 MHz and is sampled on both the rising and falling edges of the receive clock.

Once the data has been sampled, the XIB must strip the destination address from the incoming header and send it to the Core Management Interface (CMIF) where the destination is decoded. If the message is destined for the PM3390, the message data is routed to the CMIF, else the data proceeds through the XIB datapath.

Unlike other EXACT devices, the PM3390 does not modify the EXACT message in any way. By not modifying the data, the PM3390 can be programmed to work in systems that do not implement the EXACT Bus protocol.

The XIB follows a protocol to access the EXACT Bus, this protocol is described in detail in EXACT Bus Specification Document.

8.2.2 EXACT Message to Cell Decomposition

The internal switching architecture of the PM3390 is based on the notion of decomposing EXACT messages into 32-byte cells in the XIB for internal use and then re-assembling the 32-byte cells into complete EXACT messages in the XOB. The internal data path is 32-bits wide; therefore, a cell is composed of eight internal data words.

The XIB stages the data it receives from the EXACT Bus in a FIFO. At each cell clock, if the XIB has a complete cell for transfer it sends the data to the PDX, otherwise it holds this data for the next cell clock where its status will be re-evaluated. A complete cell can be either a full 32-byte data cell, a fraction of a 32-byte cell that represents the last block of a data message, or a number of queue messages (up to three) destined for a given output ring.

8.2.3 EXACT Control and Message Processing

The PM3390 has a bank of control registers and these can be written or read via the EXACT bus protocol using control messages. The XIB strips the destination from the incoming message and route it to the Core Management Interface (CMIF) for destination lookup.

If the destination of the message is to the PM3390 Base Address or to the EXACT Broadcast Address, the CMIF will signal the XIB to route the message data to the CMIF for processing.

8.2.4 EXACT Message Gathering

The internal architecture is based on transferring EXACT messages as 32-byte cells; therefore the transfer quanta to and from the CCS is an entire 32-byte cell. Queue messages are significantly smaller than a cell. So, when a 6-byte queue message is written into the CCS there are wasted memory cycles. For optimal PM3390 performance, some method for increasing efficiency on queue messages is needed. The method implemented in the XIB is an accumulator for EXACT message streams. By accumulating EXACT messages over a number of cell clocks, all messages can be more efficiently stored in the CCS.

The XIB contains an EXACT Message Accumulator (EMA) that can store up to three queue messages for each possible output port. When data

enters the XIB, the contents of the message are stored in the section of the EMA that corresponds to the output port encoded in the DEST[9:0] field (see EXACT Bus Specification Document).

EXACT messages for each output port are accumulated over a programmable period (the default is a period of eight cell clocks). If during the period, the CLM signals that it has nothing queued for the output port (i.e. the output port is not fully utilized), the messages that have been accumulated will be sent to the CCS for storage and later forwarding to the output port.

8.2.5 Destination to Port Mapping

When an EXACT message enters the XIB, the DEST[9:0] field must be decoded into a destination EXACT ring number. The PM3390 contains a Destination Lookup Table that is initialized by the system microcontroller or master port controller. The CPU maps the possible 1024 system destinations to one of eight EXACT rings (or sixteen EXACT rings in the four chip 16 x 16 configuration) to which the PM3390 is connected. This table can be loaded dynamically to support topology changes in the network.

While the cell is being staged in the XIB, the DEST[9:0] field is sent to the lookup table and a four bit value is returned. It contains PORT[2:0] which is the destination port, and DISCARD which is used to determine whether the device will drop the incoming packet because it does not support a connection to the destination EXACT ring at this time. This functionality is needed in the four-chip 16 x 16 configuration when each PM3390 device will receive EXACT messages destined for EXACT rings that it does not drive. The lookup table will power up with all DISCARD bits set which will keep any EXACT traffic from being generated / propagated until the master CPU has initialized the PM3390 to do so. Additionally, the DISCARD signal will be set when the message is of a type that is processed by the CMIF; these types are EXACT Broadcast and EXACT Control messages.

8.2.6 Error Handling and Statistics Gathering

Line coding errors are detected on the EXACT Bus receive port (these can be 8B/10B encoded line coding errors or parity errors in Clear Channel format). If an XIB detects an error of this class and the error is within the header of the message, the PM3390 will drop the message and increment a statistics counter in the control registers section of the device. If an error occurs beyond the header of the message, the PM3390 will insert the

Error Propagation Character (See the EXACT Bus Protocol Specification) at the location of the error and truncate the rest of the message. In both cases, the XIB will increment statistics counters that tally the number of header and non-header errors received on a per EXACT port basis.

When messages are received by an XIB, statistics counters tallying the number of messages and bytes received will be incremented.

8.3 Pipelined Data Multiplexer

The Pipelined Data Multiplexer (PDX) stages both cell data and cell addresses to the Cell Central Store (CCS). The PDX arbitrates access to the CCS by the input and output ring buffers. Arbitration is crucial to provide non-blocking access for each input and output port to the central store. In addition, optimal arbitration directly affects the memory required for the central store. To prevent input overflow or output starvation, the memory architecture allows each input and each output a path to the CCS on each memory clock.

8.4 Cell Central Store

The Cell Central Store (CCS) is composed of eight dual-port memory banks that store all of the EXACT message data streams. The data streams are converted into 32-byte cells in the XIB/XOB. These 32-byte cells are pipelined through the Write/Read PDX to the CCS. The FLM supplies the write addresses to the CCS and the CLM supplies the read addresses to the CCS.

8.5 Free List Manager

The Free List Manager (FLM) is responsible for providing CCS write addresses used by the Write PDX. A bitmap is maintained which represents the used status of each cell location in the CCS. When the destination XOB places the cell onto the EXACT ring, the CLM sends control information to the FLM, which signals to clear the used status information in the bitmap for the address corresponding to the cell.

8.6 Cell List Manager

The Cell List Manager (CLM) manages the EXACT messages that are stored in the CCS. As input data streams enter the chip, the FLM provides write addresses to the CCS. The CLM uses these write addresses and the destination ring and cell type information from the XIB to track the data streams per EXACT output. The CLM orders the data streams in memory for the XOB and provides a read address to the CCS per memory clock so that the XOB can read the data.

8.6.1 Building Input Data Streams

The XIB decomposes EXACT messages into cells and forwards the data through the PDX to be stored in the CCS. The data is stored at an address provided by the FLM. This address, as well as status information about the cell from the XIB, is provided to the CLM. The XIB and FLM stage the address and status information so that they are aligned upon arrival at the CLM. After the CLM links the cells associated with a single message together it sends the cells to the CCS and alerts the destination output port.

8.7 EXACT Bus Output Buffer

The EXACT Bus Output Buffer (XOB) is responsible for routing the data bound for an EXACT Bus output port from the Read PDX. Each XOB receives a data stream from the Read PDX and from the Expandability Interface (EXPIF). Based on control information from the CLM, the XOB determines which data stream will be sent to the EXACT Bus.

8.7.1 EXACT Bus Interface

The XOB prepends a message with a linecode delimiter and then transmits the message as is with no modifications. The data that is transmitted onto the EXACT bus is encoded in 8B-10B linecode or in Clear Channel mode. Transmit data is output to the bus through the EXACT transmit port which contains a 10-bit data bus clocked by a locally provided 125/135 MHz transmit clock. The output frequency and data formats are controllable by pin straps.

The XOB follows a protocol to access the EXACT bus, this protocol is described in detail in EXACT Bus Specification Document.

8.7.2 EXACT Control Register and Broadcast Message Processing

The XOB has a 32-bit data port to the CMIF. When a control register message or an EXACT broadcast is intended for an EXACT ring, the CMIF sends a signal telling the XOB that it has a message pending. The XOB will finish processing the current EXACT message and upon completion send status to the CMIF indicating that it is ready to receive the control or broadcast message. After a successful status from the XOB, the CMIF will forward the message to the XOB. While the XOB is waiting for the CMIF to forward the message, it will process no other EXACT traffic.

8.7.3 Look Ahead Operation

The PM3390 incorporates a look ahead function to increase the efficiency of the EXACT Bus output ports. The look ahead function reduces the number of IDLE messages transmitted by an output port when small EXACT Bus messages are to be sent.

The XOB contains two FIFOs that are used for look ahead operation. When small EXACT messages (either queue messages or the data block fragments at the end of an EXACT message) are sent, the path from

memory is used inefficiently. If the XOB did not use any form of look ahead these small messages would cause the EXACT ring to be temporarily starved. However the path from the CCS which fills these FIFOs operates at twice the bandwidth that the XOB can place data onto the EXACT Bus. Because the additional bandwidth exists, if the CLM has additional cells destined for an output, it will forward these to the XOB and set a look ahead bit.

The two FIFOs in the XOB are sized so that each may store an outbound EXACT message. When the look ahead bit is set, the XOB will allow writes to the FIFO not currently sourcing data to the EXACT Bus. When the XOB encounters the end of a message in the current FIFO, it will switch FIFOs. If the message is a short message (either a queue message or the final cell fragment of a data block), the “dead time” on the EXACT Bus will be removed and the latency for the message following the short message will be decreased. The overall effect of this is to smooth the traffic flow onto the EXACT Bus. Because the XOB will switch FIFOs on EXACT message boundaries, there is no concern about message ordering.

Look ahead operation will reduce latency in sparse traffic focusing conditions. In these traffic situations, the XOB would always read entire cells of data even when the cells were not full of valid data. This occurs on accumulated queue messages as well as in the final data block of EXACT messages. On average these blocks are only half full and the XOB would add one half a cell time of latency to the messages. Additionally, the XOB could not begin transmission until the entire cell had been placed in its output buffers. This would introduce another half cell time of latency. So, the overall latency penalty for not implementing the look ahead operation between the CLM and the XOB would add an additional cell time to the EXACT messages in these traffic situations.

8.8 Cut Through Operation

The PM3390 supports a cut through mode that allows a direct path to be created between input and output EXACT Bus ports. The cut through mode is used to increase output port transmission efficiency when there is a gap between EXACT messages.

The CLM can perform cut through when a message is bound for an output that is idle. If the destination port is busy, the entire message is stored in the CCS before it is forwarded. However, if an incoming EXACT Bus Message is destined for an output that is idle, the CLM initiates cut through after receiving the first 2 – 3 32 byte cells of the message. Once the first 2-3 cells are received, the remainder of the message is streamed through the PM3390 to the output.

Reception of the first 2 cells is required in an EXACT Bus system to ensure that the output is not starved once the message transmission begins. In systems that do not implement the EXACT Bus link protocol, it is recommended that the cut through be set to start after the first 3 32 byte cells of a message.

8.8.1 EXACT Flow Control

The EXACT flow control mechanism is designed to allow a system without a central resource to function fairly. Once a PM3390 device is added to the system, the position of a device on the EXACT ring in relationship to the PM3390 will greatly effect its ability to issue / receive traffic.

The PM3390 does not support "Ring Flow Control" that is described in the EXACT Bus protocol specification (PMC-970215 Section 4.3). The PM3390 receiver logic handles the IDLE, BUSY, FILLN, and FILLP characters (in 8B/10B mode these are K28.5, K28.1, K28.6, and K28.7 characters respectively of either positive or negative disparity) as valid non-EXACT characters. Hence, an EXACT message received with a BUSY character delimiter will be treated no differently than one received with an IDLE character delimiter.

There is no interaction between the receipt of a given EXACT message on a port with a BUSY character delimiter to the delimiter character selected when the given message is sent out one of the eight TX ports of the PM3390. The transmit port of the PM3390 is capable of outputting EXACT messages delimited by either a IDLE or a BUSY characters but the message delimiter selected is based solely on the setting of the FLM XOB

Busy Threshold and the number of Cell Central Store messages that are queued internal to the device (this is a configurable parameter: see Non-EXACT flow control description).

8.8.2 Non-EXACT Flow Control

In non-EXACT systems, it may be necessary to flow control devices that source traffic through a PM3390. There are two methods for implementing flow control for the PM3390. The first method is used to prevent overflow in the CCS, when a threshold is reached, EXACT BUSY characters are transmitted instead of EXACT IDLE characters. This is a global flow control and does not address individual ports. The second method uses status signals that provide flow control information per port in the PM3390. The traffic sourcing devices in non-EXACT systems can use this information to implement flow control. See the systems operation section for additional details on non-EXACT flow control.

8.9 Core Management Interface

The PM3390 provides a Core Management Interface (CMIF) to support low speed external management tasks and for diagnostic purposes. The CMIF supports read and write access of any on-chip register by a remote agent. These register accesses can be initiated either over the EXACT Bus or over a 16-bit asynchronous microprocessor port. The internal bus used for accessing these registers is called the Control Register Interface, or "CRI" bus (this terminology is used elsewhere in this document).

8.9.1 CMIF accesses via the EXACT bus

The EXACT bus input buffer datapath will route EXACT bus messages to the CMIF if the 10-bit Destination Address field in the EXACT bus message matches the 10-bit DeviceAddress field of CMIF register 0x00. If the address match fails, then the received EXACT message is routed as a normal EXACT message.

There is a special case if the 10-bit Destination Address field of the EXACT message is exactly equal to EXACT broadcast address (this address is configurable- see CMIF register 0x03- and defaults to the EXACT Bus broadcast address value of 10'h3FF).

EXACT bus messages that have a Destination Address field that matches the DeviceAddress in CMIF register 0x00 or is equal to the EXACT bus broadcast address will be removed from the normal traffic datapath and routed to the CMIF logic block, where it will go to a 256 byte FIFO. The data in this FIFO is read and processed by a state machine according to the following rules:

- If the EXACT bus message meets the criteria specified for a Control Register read/write, then the register access will be performed
- Else, the received message will be written to the Message Box Receive Buffer. The Message Box feature provides a low bandwidth interface between the EXACT bus and the PM3390's external microprocessor interface.

The EXACT Control message interface allows 32 bit register reads and writes. However, only the first three registers in the PM3390 (CMIF registers 0x00, 0x01, and 0x02) are 32-bit registers: reads and writes to registers outside of these three registers only treat 16-bits as valid. For 16

bit register reads, the upper 16 bits of the EXACT data field are returned zero.

8.9.2 External Microprocessor Interface and CMIF accesses

The External Microprocessor Interface consists of 16 multiplexed address and data lines, an ALE line and active low read and write lines. The interface conforms to the Intel 80186 bus protocol timing specification. Access to all PM3390 CRI registers is provided through the interface. In addition, the interface provides a limited interface to the EXACT bus through the Message Box feature.

To read and write registers using the interface, the 16 bit register address is first placed on the AD lines and the ALE is strobed high. For a read access, the MI_RD line is brought low to enable output of the register data. For a write access, the 16 bit write data is placed on AD lines and the MI_WR line is strobed low. See the External Microprocessor timing diagram in the Functional Timing section.

Most CRI registers are directly accessible through the interface by using the register address. The only exception is reading and writing of the Destination Lookup Table (DLT) registers. The DLT is accessed using the DLT CRI Control Register, DLT CRI Status Register and DLT CRI Write Register.

8.9.3 Avoiding conflict for CRI register accesses

The external microprocessor is given priority use of the CRI bus. If an EXACT control message and the external microprocessor simultaneously attempt a CRI operation, the EXACT control operation will be lost. Remote agents **must** synchronize their accesses to avoid register access conflicts. The internal logic does not arbitrate between the two possible consumers during CRI bus accesses (external CPU versus EXACT bus message receive over the XIB input buffer) and indeterminate results will occur if two consumers are attempting a CRI bus access.

The restriction on **not allowing simultaneous access of internal device registers** extends to the eight EXACT Bus receive ports. For example, if EXACT Control register read messages are received on the same cycle on all eight ports, only the message received on one of the eight ports will be accepted and processed as a register read; the messages from the other seven ports will be discarded by the EXACT bus input buffer (this discard allows traffic to continue to be received on all ports so that there is not

traffic blocking). The selection between receive ports is based strictly by port number, with port 0 having the highest priority and port 7 the lowest.

To summarize:

1. CRI register access conflict must be avoided for proper PM3390 device operation

2. It is up to the system's engineer to externally restrict simultaneous accesses of internal device registers.

8.9.4 EXACT Bus Control and Status Message Processing

The CMIF responds to EXACT register read and write control messages (Ref. PMC-970215). The PM3390 only accepts as valid Control messages a subset of the Control Message type specified in the EXACT bus specification:

- Valid EXACT Control message accepted by the PM3390 if and only if (1) the 4 bit EXACT control message type is 4'b 1000; (2) the REG bit field is a 1.

If the above condition for a valid Control message is not met (for example, if the REG bit field is a 0, which would indicate a memory control read/write) the message will be directed to the Message Box receive FIFO.

Read **control** messages are issued to cause the target device (in this case the PM3390) to read the specified data from its internal registers or local memory (and return the results using a single **status** message), and are formatted as follows:

7	6	5	4	3	2	1	0
Delimiter (8)							
HOPCOUNT (8)							
TYPE (4)				CTRL (2)		DEST (2)	
DEST (8)							
READSIZE (6)						SRC (2)	
SRC (8)							
ADDRESS[0] (8)							
ADDRESS[1] (8)							
ADDRESS[2] (8)							

Note that all three of the address bytes are required, as per the EXACT bus specification, even though the least significant byte (ADDRESS[2]) is not used.

Write **control** messages, are issued by external EXACT devices to cause the PM3390 to update the contents of internal registers. Three of these bytes must be used as the address of the register accessed (in the same manner as a read). Write **control** messages are formatted as:

7	6	5	4	3	2	1	0
Delimiter (8)							
HOPCOUNT (8)							
TYPE (4)				**CTRL (2)**		**DEST (2)**	
DEST (8)							
Reserved (6)						**SRC (2)**	
SRC (8)							
ADDRESS[0] (8)							
ADDRESS[1] (8)							
ADDRESS[2] (8)							
DATA[0] (8)							
DATA[1] (8)							
DATA[2] (8)							
DATA[3] (8)							

The **HOPCOUNT** field controls message lifetime: the PM3390 does not process this field. The 4-bit **TYPE** field contains 1000 binary, while the 2-bit **CTRL** field is divided up as follows:

1	0
REG	**R/W**

The EXACT bus specification states: "The **REG** bit indicates, if set, that the least-significant 16 bits of the **ADDRESS** field of the message should be interpreted as the index of an on-chip register; otherwise, all 24 bits of the **ADDRESS** field should be taken as containing a 24-bit local memory address instead". As noted earlier, the PM3390 will only accept Control Register messages so if the REG bit is a 0 the message will be sent to the Message Box receive FIFO. If set, the **R/W** bit signifies that a read operation is being performed (i.e., this is a read **control** message), and a write operation otherwise. The 10-bit **DEST** field identifies the EXACT-ES or EXACT-IS device (and possibly the port) at which the message is aimed, while the **SRC** field gives the index of the entity issuing the **control** message. Note that in the case of a read **control** message the **SRC** field will be used to determine where the data being read will be sent.

Since the PM3390 only accepts a Control Write Register message the number of DATA bytes that are part of the message is fixed to be 4. For accesses to the three CMIF registers that are defined to be 32-bit quantities (CMIF register addresses 0x00-0x02), the 32-bit value write

data is: write_data[31:0] = {DATA3, DATA2, DATA1,DATA0}. For all other register accesses, DATA[3] and DATA[2] are not used and the 16-bit write data is: write_data[15:0] = {DATA1,DATA0}.

From the above discussion, it can be seen that EXACT Control Register messages as received by the PM3390 will be 8 bytes in length for a read and 12 bytes in length for a write (with the Delimiter character not included in this length computation).

EXACT Control Message Register Reads

The CMIF creates an EXACT Status message in response to receiving a Control Register Read message.

	7	6	5	4	3	2	1	0
BYTE 0	HOPCOUNT (8)							
BYTE 1	4'b1001				2'b11		DEST (2)	
BYTE 2	DEST (8)							
BYTE 3	Reserved (6)						SRC (2)	
BYTE 4	SRC (8)							
BYTE 5	ADDRESS[0] (8)							
BYTE 6	ADDRESS[1] (8)							
BYTE 7	ADDRESS[2] (8)							
BYTE 8	REGISTER VALUE [7:0]							
BYTE 9	REGISTER VALUE[15:8]							
BYTE 10	REGISTER VALUE [23:16] (all zeros for 16 bit registers)							
BYTE 11	REGISTER VALUE [31:24] (all zeros for 16 bit registers)							

Table 1. Status Response message

The HOPCOUNT byte is always set to 255 for Status Response messages. The 10 bit DEST field contains the address of the device which originated the register read control message. The 10 bit SRC field contains the PM3390 DeviceAddress.

This Control Read status EXACT message is returned on the Transmit port corresponding to the port it was received on (the only exception is for broadcast messages).

EXACT bus Control Read messages can be generated at a higher rate than the PM3390 can provide Status response messages. The CMIF has a 256 byte input FIFO to hold messages that have been received via any of the eight EXACT bus receive ports. The approximate rate at which

Control Read messages can be processed by the PM3390 before the CMIF input FIFO fills is 25% of the available EXACT bus bandwidth (25% utilization could be the 8-byte Control register Read message followed by 24-bytes of either IDLES or normal traffic). If the CMIF input FIFO should fill, any messages that would normally be written to it would be discarded and there will be no external visibility that this condition occurred. In all cases, the EXACT bus receive ports will continue processing EXACT messages.

8.9.4.2 EXACT Control Message Register Writes

EXACT bus Control Write messages can be generated at a higher rate than the PM3390 can process the message. The approximate rate at which Control Write messages can be processed by the PM3390 before the CMIF input FIFO fills is 25% of the available EXACT bus bandwidth (25% utilization could be the 12-byte Control register Write message followed by 20-bytes of either IDLES or normal traffic). If the CMIF input FIFO should fill, any messages that would normally be written to it would be discarded and there will be no external visibility that this condition occurred. In all cases, the EXACT bus receive ports will continue processing EXACT messages.

8.9.4.3 EXACT Broadcast Message Processing

EXACT messages that have a 10-bit EXACT bus Destination Address that is equal to the configurable 10-bit BroadcastPortAddress (CRI register 0x0003) are interpreted as a broadcast message. The format of the EXACT Control message is independent of whether it is a broadcast or not and must adhere to the message format outlined above.

A Control Register Read or Write message may also be a broadcast message. In this case, the control message is executed and then written to the Transmit FIFO to be processed like any other broadcast message.

Broadcast messages are sent to all of the eight possible transmit ports that have the corresponding enable bit set in CMIF register 0x05.

8.9.5 Broadcast Hold-off Timer

The CMIF contains a programmable timer that begins to decrement when a broadcast message transmission begins. While the timer value is greater than zero, reception of broadcasts on the XIB rings is disabled.

This feature provides a mechanism for preventing broadcast feedback loops on the network.

To program the desired timeout period, the BCTIMEOUT register is written with the desired value. The timeout time period is equal to (SYSCLK period * BCTIMEOUT).

8.9.6 Message Box Feature

As mentioned earlier, EXACT bus traffic that has a destination address of the PM3390 DeviceAddress value and which does not meet the criteria specified for the PM3390 EXACT bus Control Register message will be written to the Message Box Receive buffer. The Message Box feature provides a low bandwidth interface between the EXACT bus and the PM3390's external microprocessor interface. If the Message Box receive buffer should fill, a message that would normally be written to that buffer would be dropped. This condition is externally visible to the user via CMIF register 0x11 (Message Box status register). Under no condition is the EXACT bus receive port blocked.

9 SYSTEM OPERATION

This section provides a description of the PM3390 system operation.

9.1 Configuration and Initialization

9.1.1 Configuration

Several control registers within the device must be configured at reset for the device to operate properly. These are configured via pin-straps on the EXPIF data ports. On reset, the internal drivers of these ports will be tri-stated allowing the values from the pins to propagate to the control registers. There are 36 data pins that can be used to program various registers. One must be used to configure the device for 16 x 16 versus 8 x 8 mode. Another eight pins will be used to select the output frequency for each of the XOBs. The other 35 can be used to program the device index and any other mode registers. See the EXACT Bus Protocol Specification for more system details on device discovery and initialization.

Pins used for pinstrap	Internal Signal	Description
EX1RX[0],EX0RX[8:0]	PM3390 Base Address	EXACT Base Address for the PM3390 device; PM3390 will respond to EXACT messages to this address.
EX1RX[2:1]	EXACT Ring 0 Clockmode	Selects the output mode for the PM3390 TX port. Choices are 00-SERDES, 01- 8B/10B Encoded Mode; 10- Clear Channel Mode; 11- Reserved: do not use.
EX1RX[4:3]	EXACT Ring 1 Clockmode	Selects the output mode for the PM3390 TX port. Choices are 00-SERDES, 01- 8B/10B Encoded Mode; 10- Clear Channel Mode; 11- Reserved: do not use.
EX1RX[6:5]	EXACT Ring 2 Clockmode	Selects the output mode for the PM3390 TX port. Choices are 00-SERDES, 01- 8B/10B Encoded Mode; 10- Clear Channel Mode; 11- Reserved: do not use.
EX1RX[8:7]	EXACT Ring 3 Clockmode	Selects the output mode for the PM3390 TX port. Choices are 00-SERDES, 01- 8B/10B Encoded Mode; 10- Clear Channel Mode; 11- Reserved: do not use.

<table>
<tr><td>EX2RX[1:0]</td><td>EXACT Ring 4 Clockmode</td><td>Selects the output mode for the PM3390 TX port. Choices are 00-SERDES, 01- 8B/10B Encoded Mode; 10- Clear Channel Mode; 11- Reserved: do not use.</td></tr>
<tr><td>EX2RX[3:2]</td><td>EXACT Ring 5 Clockmode</td><td>Selects the output mode for the PM3390 TX port. Choices are 00-SERDES, 01- 8B/10B Encoded Mode; 10- Clear Channel Mode; 11- Reserved: do not use.</td></tr>
<tr><td>EX2RX[5:4]</td><td>EXACT Ring 6 Clockmode</td><td>Selects the output mode for the PM3390 TX port. Choices are 00-SERDES, 01- 8B/10B Encoded Mode; 10- Clear Channel Mode; 11- Reserved: do not use.</td></tr>
<tr><td>EX2RX[7:6]</td><td>EXACT Ring 7 Clockmode</td><td>Selects the output mode for the PM3390 TX port. Choices are 00-SERDES, 01- 8B/10B Encoded Mode; 10- Clear Channel Mode; 11- Reserved: do not use.</td></tr>
<tr><td>EX3RX[6:0], EX2RX[8]</td><td>EXACT TX Clk Frequency</td><td>Selects the output clock multiplexer setting used by a given TX port.
0- select XREFCK125 input as clock reference,
1- select XREFCK135 as the clock reference.
<table>
<tr><th>Pin strap</th><th>TX port clock select</th></tr>
<tr><td>EX2RX[8]</td><td>TX port 0</td></tr>
<tr><td>EX3RX[0]</td><td>TX port 1</td></tr>
<tr><td>EX3RX[1]</td><td>TX port 2</td></tr>
<tr><td>EX3RX[2]</td><td>TX port 3</td></tr>
<tr><td>EX3RX[3]</td><td>TX port 4</td></tr>
<tr><td>EX3RX[4]</td><td>TX port 5</td></tr>
<tr><td>EX3RX[5]</td><td>TX port 6</td></tr>
</table></td></tr>
</table>

		EX3RX[6]	TX port 7
EX3RX[7]	Expansion Mode Select	Enables the PM3390 Expansion Interface (EXPIF). 1-on, 0-off. See also CMIF register 0x01, Device Control Register, Expansion Interface Mode bits.	
EX3RX[8]	Device Subtype	This bit **must be initialized to a logic 0** for proper device operation.	

9.1.2 Initialization

After de-assertion of the RESETI input, the following will occur:

1. Logic internal to the device goes through an initialization sequence that takes 1024 SYSCLK cycles to complete. One result of the initialization is that all 1024 DLT table entries will have the value of 4'b1000: that is, the DISCARD bit will be set on each entry of the table.

2. Each of the eight EXACT bus input receivers must synchronize. The process followed for synchronization is described in the register description for the SYNCH_CNT_MIN register (XIB register 0x09).

9.2 Destination Lookup Table

The Destination Lookup Table (DLT) is used to map the logical ports of an EXACT system to the physical ports of the PM3390. The DLT is accessed in one of two ways: (1) over the external microprocessor interface by accessing the DLT CRI Control Register, DLT CRI Status Register, and the DLT CRI Write Register; (2) over the EXACT bus by doing Control Register reads/writes to addresses in the range 0x1000 to 0x13FF.

The DLT table supports 1024 entries: this is one entry per allowable EXACT logical port address (or port index). Each entry in the table has the following format:

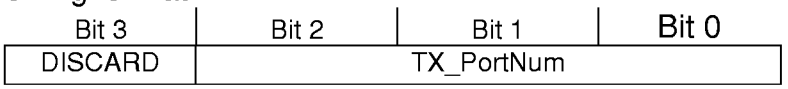


FIGURE 3 Bitfield of a DLT Entry

DISCARD is set to logic 0 for addresses which are mapped within the system. The lower 3 bits of the 4 bit DLT entry indicate the TX port number to which all non-broadcast traffic will be sent. The assignment of Destination Addresses is a higher-level system implementation detail and is outside the scope of this document. The DLT is a logically a simple look-up table with the DEST[9:0] field of the EXACT message used as the address of the look-up table.

The DISCARD bit is set to logic 1 for system addresses that are unmapped. When an EXACT receive port on the PM3390 (XIB) receives a message with an unmapped address, the DLT entry is returned with DISCARD = 1 and the XIB will discard (that is, filter) the entire message.

The DLT lookup is not used for these two exceptional cases of the destination address as has been previously discussed:

1. EXACT messages addressed to the EXACT system broadcast address (which is 0x3FF).
2. The destination address of the message is exactly equal to the DeviceAddress[9:0] in CRI register 0x0000.

All EXACT messages that bypass the DLT lookup are forwarded to the CMIF logic.

9.2.1 DLT Write Operation via the External Microprocessor Interface

Access to the DLT through the external microprocessor interface uses the following three CRI registers:

CRI register address	Register Description
0x000B	DLT CRI Status
0x000C	DLT CRI Control
0x000D	DLT CRI Write

The following sequence is used to perform a DLT write operation using the external microprocessor interface:

1. Prior to performing DLT Writes, read the DLT Status Register to verify that bits 15:14 (Write:Read Done) are both set, indicating that any previous DLT writes and reads have completed.
2. Write the 4 bit value to bits [3:0] of the DLT CRI Write Register.
3. Write the 10 bit table address to bits [9:0] of the DLT CRI Control Register with bit 15 cleared.

The following example in psuedo-code shows how to write a value of 4'h6 to DLT address 10'h239:

1. If (Cpu_read(0x000B) & 0xC000), go to step 2.
2. Cpu_write(0x000D, 0x0006)
3. Cpu_write(0x000C, 0x0239)

Where the operators are:
cpu_read(cri_register_address[15:0]),
cpu_write(cri_register_address[15:0], data[15:0]).

9.2.2 DLT Read Operation via the External Microprocessor Interface

The following sequence is used to perform a DLT read operation using the external microprocessor interface:

1. Prior to performing DLT Writes, read the DLT Status Register to verify that bits 15:14 (Write:Read Done) are both set, indicating that any previous DLT writes and reads have completed.
2. Write the 10 bit EXACT Port Address to bits [9:0] of the DLT CRI Control Register with bit 15 set.

3. Poll the DLT CRI Status Register by reading until bit 14 (Read Done) is set, indicating that the read data in bits [3:0] of the register is valid.

The following example in psuedo-code shows how to read a value from DLT address 10'h1C7:

1. If (Cpu_read(0x000B) & 0xC000), go to step 2.
2. Cpu_write(0x000C, 0x81C7)
3. If (Cpu_read(0x000B) & 0x4000), then DLT read is valid and the table entry was returned on bits [3:0] of the read.

Where the operators are:

cpu_read(cri_register_address[15:0])
cpu_write(cri_register_address[15:0], write_data[15:0]).

9.2.3 DLT access over the EXACT bus

A different mechanism is used for allowing access to the Destination Lookup Table over any of the eight EXACT system ports. The DLT table is directly mapped to the CRI register space, with 0x1000 being the base address of the table and 0x13FF being the last address in the table. Standard EXACT message Control Register reads and writes are issued by the external EXACT-compatible device to setup and maintain the DLT table.

9.2.4 Avoiding simultaneous DLT register access

As previously stated, to ensure reliable access to PM3390 internal registers from the various EXACT bus input ports (0 through 7) simultaneous accesses must be avoided. One simple means of doing this is to allow only one of the possible nine consumers that can read/write the DLT table access to it: for example, only allow register access to the DLT through the external microprocessor interface and not in-band over the EXACT receive ports 0-7. A second example would be to allow access to the DLT solely through EXACT receive port 0 and not through ports 1-7 or the external microprocessor interface.

9.3 Statistics Counters

The XIB and XOB contain counters that are readable either in-band (that is through the EXACT bus interface via Control Register reads) or by an external microprocessor. These counters maintain the number of EXACT messages and bytes that have been sent and received, the number of erred messages received, and the capacity of the Free List Manager. Refer to the normal mode register description for details on these registers.

9.3.1 Non-EXACT Flow Control

There are two methods for implementing flow control in non-EXACT systems that source traffic through a PM3390. The EXACT protocol is a fully reserved protocol that "pulls" packets from the egress to the ingress. Storage within the PM3390 is guaranteed when using the EXACT link protocol. It is possible for systems, which use proprietary protocols to "push" packets through the PM3390, to oversubscribe the storage and may require flow control to be implemented to prevent overflow.

- Global CCS storage threshold to prevent overflow
- PM3390 per port status to provide flow control information for external non-EXACT systems

Global CCS storage threshold to prevent overflow

With the first method, the XOB receives a signal from the Free List Manager (FLM) indicating if the Cell Central Store capacity has reached a programmable threshold. Once this threshold has been reached, the XOB will send out BUSY characters instead of IDLE characters between messages. This mechanism can be used to prevent overflow of the CCS memory in systems that do not issue traffic according to the EXACT Bus Protocol specification.

The global CCS storage threshold based flow control mechanism is very coarse grained and can only be used to prevent CCS overflow. In non-EXACT systems a finer grained flow control mechanism can be used to provide per port flow control information. The device register to set the global CCS storage threshold is FLM register 0x02, XOB Busy Threshold.

PM3390 per port status to provide flow control for non-EXACT systems

Per port flow control can be implemented by a non-EXACT system by using status information from the PM3390. During normal operation, the four NFLOW pins of the PM3390 cycle through an eight clock sequence that can be sampled to provide flow control information to traffic sourcing devices in non-EXACT systems. The following table shows that eight clock sequence:

Clock	Data Sampled On NFLOW[3:0]	Description
1	0000	Synchronization Pattern 0
2	0000	Synchronization Pattern 1
3	0000	Synchronization Pattern 2
4	0000	Synchronization Pattern 3
5	0000	Synchronization Pattern 4
6	FLOW[7:4]	Flow control information for sourcing devices on EXACT Rings [7:4]; this signal is active low: 0-Device BUSY, 1- Device IDLE
7	FLOW[3:0]	Flow control information for sourcing devices on EXACT Rings [3:0]; this signal is active low: 0-Device BUSY, 1- Device IDLE
8	1111	Framing Pattern 1

It is the CLM register 0x14, Flow Control Threshold, that is used to set the threshold reported as FLOW[7:0] and which is output from the PM3390 on the NFLOW[3:0] device pins. The CLM register 0x14 bits [9:0] are "FCDEPTH[9:0]". The FLOW[N] output is logically equal to:

FLOW[N] = !(number of cells queued for output port N > FCDEPTH)

9.4 Expandability Interface

The Expandability Interface (EXPIF) provides the functionality necessary to connect four 8 x 8 PM3390 devices into a 16 x 16 switching matrix. To accomplish this, four PM3390s are divided into two pairs. Each pair drives eight of the sixteen EXACT rings. Within each pair, each PM3390 must act as master for four of the EXACT rings and determine which chip will drive each of these shared EXACT rings; the other device must send status information to the master to allow “drive decisions” to be made.

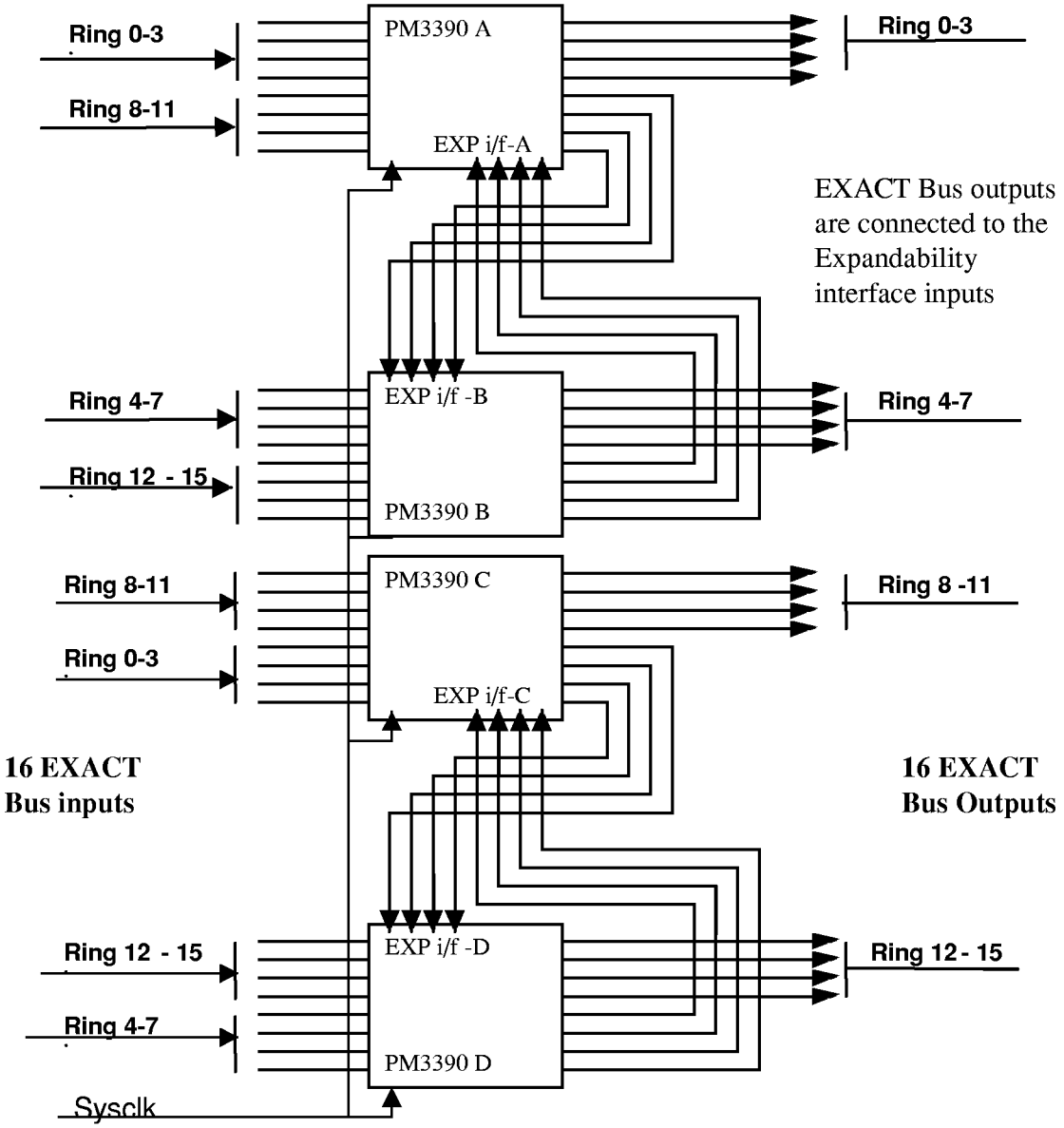
The status / flow control interface between the master and slave PM3390 devices is implemented as a serial protocol. The slave device will send the upper six bits of an eight-bit count used to count the number of cells it currently has stored for each channel. With four channels and six bits per channel, the entire status sequence will take 24 EXACT system clocks. At the start of each cycle the device will drive a framing signal high for one clock to indicate the beginning of the cycle. Additionally, the master device will send a flow control bit for each channel that it shares with the slave device. Over the same 24 clock period, the master device will send six, 4-bit flow control updates. Both devices will act as a master for four channels and a slave for four channels and thus send both flow control and status information to the other device. The devices will use the same framing bit for both the flow control and status information sent to the other device.

Along with the status and flow control information, each chip sends the EXACT Bus outputs for the four shared in Clear Channel format to the other PM3390. The master XOB determines which chips output information will be placed on the EXACT transmit port. The master device also sends flow control information to the slave device based on the amount of space available in it’s FIFO used for shared data.

The EXACT bus clock modes of the 16 ports that map directly to the 16x16 matrix input/output ports can be configured to be any of the three supported clock modes of the PM3390: SERDES, 8B/10B encoded, or Clear Channel. The clock mode for TX ports 7-4 of each PM3390 is internally configured to be of type “Clear Channel” if the corresponding EXPIF_MODE[3:0] bit is set in the CMIF Device Control Register (0x0001). The clock mode for RX ports 4-7 of each PM3390 is independent of the TX clock mode if the corresponding EXPIF_MODE bit is set: the clock mode should be selected consistent with the physical clocking of the given EXACT bus. For example (with reference to diagram below), assuming that port[1] of the 16x16 switch matrix is connected to a SERDES device and maps to PM3390 device A RX port [1] and PM3390

device C RX port[5], then the clock modes for the two RX ports must be set to SERDES.

FIGURE 4: Interconnection of four PM3390s for 16-PORT SWITCH



The pin connection for PM3390 device A and PM3390 device B for the 16-port switch shown above is given in the following table. Please note that the connections for PM3390 device C and PM3390 device D are similar.

Pins on PM3390	Pins on PM3390	Description

device A	device B	
SYSCLK	SYSCLK	SYSCLK for all four PM3390 devices must be in-phase
XFLOWI	XFLOWO	connect
XSTATI	XSTATO	Connect
XFRAMEI	XFRAMEO	Connect
XFLOWO	XFLOWI	Connect
XSTATO	XSTATI	Connect
XFRAMEO	XFRAMEI	Connect
TX4D[8:0]	EX0RXD[8:0]	TX4D[9] is a no-connect
TX4CLK	EX0RXCLK	connect
TX5D[8:0]	EX1RXD[8:0]	TX5D[9] is a no-connect
TX5CLK	EX1RXCLK	connect
TX6D[8:0]	EX2RXD[8:0]	TX6D[9] is a no-connect
TX6CLK	EX2RXCLK	connect
TX7D[8:0]	EX3RXD[8:0]	TX7D[9] is a no-connect
TX7CLK	EX3RXCLK	connect
Connection of the EXACT bus outputs of PM3390 device B to the Expandability Interface on PM3390 device B is done in a similar way.		

Because the XFLOWI/O, XSTATI/O, and XFRAMEI/O signals are clocked on the rising edge of SYSCLK within the PM3390, it is necessary to have the SYSCLK used on each of the four PM3390 devices be phase-aligned (that is, they use the same SYSCLK signal).

9.5 Message Box Feature

The Message Box acts as an interface between the EXACT Bus and the PM3390's external CPU interface. The Message Box contains a 256 byte buffer for storing received EXACT messages and a 256 byte buffer for formatting messages to be sent out on the EXACT bus. The memory locations of the buffers are accessible by the external CPU CRI interface. The locations cannot be read/written using EXACT control messages.

The Message Box feature is intended to be used as a debugging or diagnostic tool as opposed to a "back-door" into the EXACT bus. Only a low rate of message transfer can be supported. Other applications will find the hardware limitations severe for message processing.

Access to the Message Box is done over the external microprocessor interface by accessing these five CRI registers:

CMIF register	Register Description
0x10	Message Box Command
0x11	Message box status
0x12	Message Box Receive Buffer Pointer
0x13	Message Box Receive Counter
0x14	Message Box Error Counter

9.5.1 Message Box Receive Operation

Only messages that are received on one of the eight EXACT input ports can be written to the Message Box Receive buffer. The messages that end up in the Message Box Receive buffer meet all of the following criteria:

1. The destination address is equal to the DeviceAddress[9:0] of this PM3390 device (see CMIF register 0x00)
2. In addition, the message format is NOT an EXACT Control Register Read or Write message.

For example, access to an unimplemented or reserved register via an EXACT Control Register read/writer does not result in the message being sent to the Message Box Receive buffer.

EXACT messages are written into the message box receive buffer as a series of 32 bit words. The first entry in the buffer is a delimiter word called a *status header.* The EXACT message is then written, in the byte-order received, into the buffer. The status header provides status bits indicating the message length and whether receive buffer overrun occurred during reception due to the buffer becoming full. If the message box receive buffer should overrun: (1) the EXACT message being written to the receive buffer will be truncated; (2) the message length field in the status header will correspond to the actual number of valid bytes from this message that are in the receive buffer.

The Message Box Receive Buffer is mapped to CRI register addresses 0x400 to 0x47F: CRI addressing is done with respect to a half-word (i.e. two-byte). A CRI register read to the 128 CRI addresses in the range 0x400 to 0x47F will return 2 bytes of data from the buffer. Bytes from the receive buffer do not have to be read in any order. The total number of message bytes that can be accumulated in the Message Box Read Buffer is 256. The 256 byte limit includes the required status header word, one per message and any required padding for message alignment. In order to continue receiving additional messages in the buffer, the message box must be reset: this can be accomplished by writing a 1 to bit[15] of CMIF register 0x10 (Message Box command register). Each message saved in the Message Box Receive Buffer requires a number of bytes that is a multiple of 4: that is, the status header word will always start on an even CRI address (such as 0x400, 0x402, 0x404, etc).

For example, it takes 12 bytes to save a 5 byte EXACT message (see example at the end of this section). Hence, only 21 of these 5 byte EXACT messages can be accumulated in the Message Box Read Buffer (since 21 *12 = 252): the buffer will fill if the Message Box is not reset before the 22nd EXACT message is received.

The format of the 32-bit *status header* word in the Message Box Receive Buffer is as follows:

Bits	Description of Status Header word in Message Box Receive Buffer
[31]	ValidMessage. 1: indicates that the header is the start of a valid message. 0: the header is empty and there are no following messages.

[30]	Error 0: no error. 1: the message following the header (that is, after the first five bytes) is incomplete due to a line error of a character. This error flag is propagated by the EXACT bus receive buffer (XIB), which truncates the message at the point the error occurs.
[29]	Overrun. 0: no overrun. 1: Receive buffer became full during the message reception and was truncated.
[28:19]	Reserved. Read as zero.
[18:16]	RxPortNumber[2:0]. This is the port number on which the message was received.
[15:8]	The length of the message in bytes.
[7]	Reserved. Read as zero.
[6:0]	Pointer to the first 16 bit word of the **next** message. The pointer is always valid. When there is no following message, the pointer points to itself.

As an example, the following shows the Receive buffer after receiving the following two messages. Assume that the Message Box Receive Buffer has first been reset. The first message is a 4 word EXACT message containing 15 bytes on ring 1. The second message is 5 bytes in length and was received on ring 6. Note: that two CRI reads are necessary to read the 32 bit status header and two EXACT message bytes are read with each of the following CRI reads:

.

CRI address	CRI data [15:8]	CRI data [7:0]	Description
0x400	0x0F	0x14	Status Hdr, Msg #1 low halfword
0x401	0x80	0x01	Status Hdr, Msg #1 high halfword
0x402	EXACT Byte 1	EXACT Byte 0	Msg #1, bytes 1:0
0x403	EXACT Byte 3	EXACT Byte 2	Msg #1, bytes 3:2
0x404	EXACT Byte 5	EXACT Byte 4	Msg #1, bytes 5:4
0x405	EXACT Byte 7	EXACT Byte 6	Msg #1, bytes 7:6

0x406	EXACT Byte 9	EXACT Byte 8	Msg #1, bytes 9:8
0x407	EXACT Byte 11	EXACT Byte 10	Msg #1, bytes 11:10
0x408	EXACT Byte 13	EXACT Byte 12	Msg #1, bytes 13:12
0x409	Don't care (pad)	EXACT Byte 14	{don't-care, Msg #1, byte 14}
0x40A	0x05	0x20	Status Hdr, Msg #2 low halfword
0x40B	0x80	0x07	Status Hdr, Msg #2 high halfword
0x40C	EXACT Byte 1	EXACT Byte 0	Msg #2, bytes 1:0
0x40D	EXACT Byte 3	EXACT Byte 2	Msg #2, bytes 3:2
0x40E	Don't care (pad)	EXACT Byte 4	{don't-care, Msg #2, byte 4}
0x40F	Don't care (pad)	Don't care (pad)	pad
0x410	0x00	0x10	Next status header: low
0x411	0x00	0x00	Status header: high. next message not valid

9.5.2 Transmit Message Operation

EXACT messages are formatted in the 256 byte Transmit buffer by an external microprocessor. Each message must be preceded by a 32 bit Transmit Buffer Header similar to the Receive Buffer message header. Multiple messages are formatted using pointers in the message headers to link the messages in the desired transmit order. The Transmit Buffer Header has the following format:

Bits	Description
[31]	Reserved: write to 0
[30:29]	Transmit command (the value in bit [30] selects whether to transmit the message according to additional decode of bits in this header word or using the CMIF register 0x10).

	2'b00, 2'b01: send message using CMIF register 0x10 (Transmit Message Box Command): can be unicast or broadcast. 2'b10: unicast, routed out EXACT port given by bit-value in [26:24] 2'b11: broadcast
[28:27]	Reserved: write to 0
[26:24]	If transmit command is 2'b10, then this 3-bit value gives the EXACT port on which the EXACT message will be transmit. Otherwise, this bit field is a don't care.
[23:16]	Reserved: write to 0
[15:8]	The length of the transmit message in bytes. Minimum value of this field is 0x05 (the length does not include any EXACT bus delimiter characters).
7	Reserved: write to 0
[6:0]	Pointer to the 16 bit word containing the lower 16 bits of the 32 bit header for the next message. The pointer address must be even because the CMIF reads data from the buffer in 32 bit words composed of an odd (upper 16 bits) and even (lower 16 bits) word. If this is the header of the last message to be transmitted, set the pointer to point to itself.

The Message Box Transmit Command Register (CMIF register 0x10) is used to initiate the transmission of messages formatted in the Transmit Mailbox Buffer. It is written as a normal 16 bit CRI register by the external microprocessor or an EXACT control message. Transmission is initiated by writing a number greater than zero to the MSGCOUNT field of this register or by writing a 1 to the message box loopback mode. Loopback mode will cause the message box transmit state machine to continuously follow the transmit buffer pointer structure, thereby allowing continuous transmission: this is a useful diagnostic feature. Bit [13] and bits [10:8] select the Broadcast mode and destination XOB ring for messages with a transmit header command of "conditional unicast".

The following is an example of three messages, lengths 8 bytes, 15 bytes and 17 bytes, formatted in the Transmit buffer. The second message is formatted to use the Broadcast mode bit and XOB bits from the message header to select XOB Ring 3. The pointer in the header of the last message points to the first message to create a loop.

CRI address	CRI data [15:8]	CRI data [7:0]	Description
0x0480	0x08	0x06	TX Hdr, Msg #1 low halfword
0x0481	0x00	0x00	TX Hdr, Msg #1 high halfword
0x0482	EXACT Byte 1	EXACT Byte 0	Msg #1, bytes 1:0
0x0483	EXACT Byte 3	EXACT Byte 2	Msg #1, bytes 3:2
0x484	EXACT Byte 5	EXACT Byte 4	Msg #1, bytes 5:4
0x485	EXACT Byte 7	EXACT Byte 6	Msg #1, bytes 7:6
0x486	0x0F	0x10	TX Hdr, Msg #2 low halfword
0x487	0x40	0x03	TX Hdr, Msg #2 high halfword
0x488	EXACT Byte 1	EXACT Byte 0	Msg #2, bytes 1:0
0x489	EXACT Byte 3	EXACT Byte 2	Msg #2, bytes 3:2
0x48A	EXACT Byte 5	EXACT Byte 4	Msg #2, bytes 5:4
0x48B	EXACT Byte 7	EXACT Byte 6	Msg #2, bytes 7:6
0x48C	EXACT Byte 9	EXACT Byte 8	Msg #2, bytes 9:8
0x48D	EXACT Byte 11	EXACT Byte 10	Msg #2, bytes 11:10
0x48E	EXACT Byte 13	EXACT Byte 12	Msg #2, bytes 13:12
0x48F	Don't care (pad)	EXACT Byte 14	{don't care, Msg #2, byte 14}
0x490	0x11	0x00	TX Hdr, Msg #3 low halfword
0x491	0x00	0x00	TX Hdr, Msg #3 high halfword
0x492	EXACT Byte 1	EXACT Byte 0	Msg #3, bytes 1:0

0x493	EXACT Byte 3	EXACT Byte 2	Msg #3, bytes 3:2
0x494	EXACT Byte 5	EXACT Byte 4	Msg #3, bytes 5:4
0x495	EXACT Byte 7	EXACT Byte 6	Msg #3, bytes 7:6
0x496	EXACT Byte 9	EXACT Byte 8	Msg #3, bytes 9:8
0x497	EXACT Byte 11	EXACT Byte 10	Msg #3, bytes 11:10
0x498	EXACT Byte 13	EXACT Byte 12	Msg #3, bytes 13:12
0x499	EXACT Byte 15	EXACT Byte 14	Msg #3, bytes 15:14
0x49A	Don't care (pad)	EXACT Byte 16	{pad, Msg #3, byte 16}
0x49B	Don't care (pad)	Don't care (pad)	{pad pad}

9.6 JTAG Test Port

The PM3390 JTAG Test Access Port (TAP) allows access to the TAP controller and the 4 TAP registers: instruction, bypass, device identification and boundary scan. Using the TAP, device input logic levels can be read, device outputs can be forced, the device can be identified and the device scan path can be bypassed. For more details on the JTAG port, please refer to the Operations section.

Table 2 – Instruction Register

Length - 3 bits

Instructions	Selected Register	Instruction Code IR[2:0]
EXTEST	Boundary Scan	000
IDCODE	Identification	001
SAMPLE	Boundary Scan	010
BYPASS	Bypass	011
BYPASS	Bypass	100
STCTEST	Boundary Scan	101
BYPASS	Bypass	110
BYPASS	Bypass	111

9.6.1 Identification Register

Length - 32 bits

Version number - 1H

Part Number - 3390H

Manufacturer's identification code - 0CDH

Device identification - 133900CDH

9.6.2 Boundary Scan Register

The boundary scan register is made up of 365 boundary scan cells, divided into input observation (in_cell), output (out_cell), and bi-directional (io_cell) cells. These cells are detailed in the following section. The first 32 cells form the ID code register, and carry the code 133900CDH. The cells are arranged as follows:

Table 3 – Boundary Scan Chain

Pin/ Enable	Register Bit	Cell Type	Device I.D.
TDI	-	JTAG data in	-
TMS	-	JTAG mode	-
TCK	-	JTAG clock	-
TRSTB	-	JTAG reset	-
RX0CLK	364	IN_CELL	L
RX0D[9]	363	IN_CELL	L
RX0D[8]	362	IN_CELL	L
RX0D[7]	361	IN_CELL	L
RX0D[6]	360	IN_CELL	L
RX0D[5]	359	IN_CELL	L
RX0D[4]	358	IN_CELL	H
RX0D[3]	357	IN_CELL	H
RX0D[2]	356	IN_CELL	L
RX0D[1]	355	IN_CELL	L
RX0D[0]	354	IN_CELL	H
TX0CLK	353	OUT_CELL	H
OEB_TX0CLK	352	OUT_CELL	H
TX0D[9]	351	OUT_CELL	L
OEB_TX0D[9]	350	OUT_CELL	L
TX0D[8]	349	OUT_CELL	H
OEB_TX0D[8]	348	OUT_CELL	L

TX0D[7]	347	OUT_CELL	L
OEB_TX0D[7]	346	OUT_CELL	L
TX0D[6]	345	OUT_CELL	L
OEB_TX0D[6]	344	OUT_CELL	L
TX0D[5]	343	OUT_CELL	L
OEB_TX0D[5]	342	OUT_CELL	L
TX0D[4]	341	OUT_CELL	L
OEB_TX0D[4]	340	OUT_CELL	H
TX0D[3]	339	OUT_CELL	H
OEB_TX0D[3]	338	OUT_CELL	L
TX0D[2]	337	OUT_CELL	L
OEB_TX0D[2]	336	OUT_CELL	H
TX0D[1]	335	OUT_CELL	H
OEB_TX0D[1]	334	OUT_CELL	L
TX0D[0]	333	OUT_CELL	H
OEB_TX0D[0]	332	OUT_CELL	-
TX1CLK	331	OUT_CELL	-
OEB_TX1CLK	330	OUT_CELL	-
TX1D[9]	329	OUT_CELL	-
OEB_TX1D[9]	328	OUT_CELL	-
TX1D[8]	327	OUT_CELL	-
OEB_TX1D[8]	326	OUT_CELL	-
TX1D[7]	325	OUT_CELL	-
OEB_TX1D[7]	324	OUT_CELL	-
TX1D[6]	323	OUT_CELL	-
OEB_TX1D[6]	322	OUT_CELL	-
TX1D[5]	321	OUT_CELL	-
OEB_TX1D[5]	320	OUT_CELL	-
TX1D[4]	319	OUT_CELL	-
OEB_TX1D[4]	318	OUT_CELL	-

TX1D[3]	317	OUT_CELL	-
OEB_TX1D[3]	316	OUT_CELL	-
TX1D[2]	315	OUT_CELL	-
OEB_TX1D[2]	314	OUT_CELL	-
TX1D[1]	313	OUT_CELL	-
OEB_TX1D[1]	312	OUT_CELL	-
TX1D[0]	311	OUT_CELL	-
OEB_TX1D[0]	310	OUT_CELL	-
TX2CLK	309	OUT_CELL	-
OEB_TX2CLK	308	OUT_CELL	-
TX2D[9]	307	OUT_CELL	-
OEB_TX2D[9]	306	OUT_CELL	-
TX2D[8]	305	OUT_CELL	-
OEB_TX2D[8]	304	OUT_CELL	-
TX2D[7]	303	OUT_CELL	-
OEB_TX2D[7]	302	OUT_CELL	-
TX2D[6]	301	OUT_CELL	-
OEB_TX2D[6]	300	OUT_CELL	-
TX2D[5]	299	OUT_CELL	-
OEB_TX2D[5]	298	OUT_CELL	-
TX2D[4]	297	OUT_CELL	-
OEB_TX2D[4]	296	OUT_CELL	-
TX2D[3]	295	OUT_CELL	-
OEB_TX2D[3]	294	OUT_CELL	-
TX2D[2]	293	OUT_CELL	-
OEB_TX2D[2]	292	OUT_CELL	-
TX2D[1]	291	OUT_CELL	-
OEB_TX2D[1]	290	OUT_CELL	-
TX2D[0]	289	OUT_CELL	-
OEB_TX2D[0]	288	OUT_CELL	-

TX3CLK	287	OUT_CELL	-
OEB_TX3CLK	286	OUT_CELL	-
TX3D[9]	285	OUT_CELL	-
OEB_TX3D[9]	284	OUT_CELL	-
TX3D[8]	283	OUT_CELL	-
OEB_TX3D[8]	282	OUT_CELL	-
TX3D[7]	281	OUT_CELL	-
OEB_TX3D[7]	280	OUT_CELL	-
TX3D[6]	279	OUT_CELL	-
OEB_TX3D[6]	278	OUT_CELL	-
TX3D[5]	277	OUT_CELL	-
OEB_TX3D[5]	276	OUT_CELL	-
TX3D[4]	275	OUT_CELL	-
OEB_TX3D[4]	274	OUT_CELL	-
TX3D[3]	273	OUT_CELL	-
OEB_TX3D[3]	272	OUT_CELL	-
TX3D[2]	271	OUT_CELL	-
OEB_TX3D[2]	270	OUT_CELL	-
TX3D[1]	269	OUT_CELL	-
OEB_TX3D[1]	268	OUT_CELL	-
TX3D[0]	267	OUT_CELL	-
OEB_TX3D[0]	266	OUT_CELL	-
TX4CLK	265	OUT_CELL	-
OEB_TX4CLK	264	OUT_CELL	-
TX4D[9]	263	OUT_CELL	-
OEB_TX4D[9]	262	OUT_CELL	-
TX4D[8]	261	OUT_CELL	-
OEB_TX4D[8]	260	OUT_CELL	-
TX4D[7]	259	OUT_CELL	-
OEB_TX4D[7]	258	OUT_CELL	-

TX4D[6]	257	OUT_CELL	-
OEB_TX4D[6]	256	OUT_CELL	-
TX4D[5]	255	OUT_CELL	-
OEB_TX4D[5]	254	OUT_CELL	-
TX4D[4]	253	OUT_CELL	-
OEB_TX4D[4]	252	OUT_CELL	-
TX4D[3]	251	OUT_CELL	-
OEB_TX4D[3]	250	OUT_CELL	-
TX4D[2]	249	OUT_CELL	-
OEB_TX4D[2]	248	OUT_CELL	-
TX4D[1]	247	OUT_CELL	-
OEB_TX4D[1]	246	OUT_CELL	-
TX4D[0]	245	OUT_CELL	-
OEB_TX4D[0]	244	OUT_CELL	-
XSTATO	243	OUT_CELL	-
OEB_XSTATO	242	OUT_CELL	-
XFRAMEO	241	OUT_CELL	-
OEB_XFRAMEO	240	OUT_CELL	-
XSTATI	239	IN_CELL	-
XFRAMEI	238	IN_CELL	-
XFLOWO	237	OUT_CELL	-
OEB_XFLOWO	236	OUT_CELL	-
XFLOWI	235	IN_CELL	-
EX3RXCK	234	IN_CELL	-
EX3RX[8]	233	IN_CELL	-
EX3RX[7]	232	IN_CELL	-
EX3RX[6]	231	IN_CELL	-
EX3RX[5]	230	IN_CELL	-
EX3RX[4]	229	IN_CELL	-
EX3RX[3]	228	IN_CELL	-

EX3RX[2]	227	IN_CELL	-
EX3RX[1]	226	IN_CELL	-
EX3RX[0]	225	IN_CELL	-
EX2RXCK	224	IN_CELL	-
EX2RX[8]	223	IN_CELL	-
EX2RX[7]	222	IN_CELL	-
EX2RX[6]	221	IN_CELL	-
EX2RX[5]	220	IN_CELL	-
EX2RX[4]	219	IN_CELL	-
EX2RX[3]	218	IN_CELL	-
EX2RX[2]	217	IN_CELL	-
EX2RX[1]	216	IN_CELL	-
EX2RX[0]	215	IN_CELL	-
EX1RXCK	214	IN_CELL	-
EX1RX[8]	213	IN_CELL	-
EX1RX[7]	212	IN_CELL	-
EX1RX[6]	211	IN_CELL	-
EX1RX[5]	210	IN_CELL	-
EX1RX[4]	209	IN_CELL	-
EX1RX[3]	208	IN_CELL	-
EX1RX[2]	207	IN_CELL	-
EX1RX[1]	206	IN_CELL	-
EX1RX[0]	205	IN_CELL	-
EX0RXCK	204	IN_CELL	-
EX0RX[8]	203	IN_CELL	-
EX0RX[7]	202	IN_CELL	-
EX0RX[6]	201	IN_CELL	-
EX0RX[5]	200	IN_CELL	-
EX0RX[4]	199	IN_CELL	-
EX0RX[3]	198	IN_CELL	-

EX0RX[2]	197	IN_CELL	-
EX0RX[1]	196	IN_CELL	-
EX0RX[0]	195	IN_CELL	-
TX5CLK	194	OUT_CELL	-
OEB_TX5CLK	193	OUT_CELL	-
TX5D[9]	192	OUT_CELL	-
OEB_TX5D[9]	191	OUT_CELL	-
TX5D[8]	190	OUT_CELL	-
OEB_TX5D[8]	189	OUT_CELL	-
TX5D[7]	188	OUT_CELL	-
OEB_TX5D[7]	187	OUT_CELL	-
TX5D[6]	186	OUT_CELL	-
OEB_TX5D[6]	185	OUT_CELL	-
TX5D[5]	184	OUT_CELL	-
OEB_TX5D[5]	183	OUT_CELL	-
TX5D[4]	182	OUT_CELL	-
OEB_TX5D[4]	181	OUT_CELL	-
TX5D[3]	180	OUT_CELL	-
OEB_TX5D[3]	179	OUT_CELL	-
TX5D[2]	178	OUT_CELL	-
OEB_TX5D[2]	177	OUT_CELL	-
TX5D[1]	176	OUT_CELL	-
OEB_TX5D[1]	175	OUT_CELL	-
TX5D[0]	174	OUT_CELL	-
OEB_TX5D[0]	173	OUT_CELL	-
TX6CLK	172	OUT_CELL	-
OEB_TX6CLK	171	OUT_CELL	-
TX6D[9]	170	OUT_CELL	-
OEB_TX6D[9]	169	OUT_CELL	-
TX6D[8]	168	OUT_CELL	-

OEB_TX6D[8]	167	OUT_CELL	-
TX6D[7]	166	OUT_CELL	-
OEB_TX6D[7]	165	OUT_CELL	-
TX6D[6]	164	OUT_CELL	-
OEB_TX6D[6]	163	OUT_CELL	-
TX6D[5]	162	OUT_CELL	-
OEB_TX6D[5]	161	OUT_CELL	-
TX6D[4]	160	OUT_CELL	-
OEB_TX6D[4]	159	OUT_CELL	-
TX6D[3]	158	OUT_CELL	-
OEB_TX6D[3]	157	OUT_CELL	-
TX6D[2]	156	OUT_CELL	-
OEB_TX6D[2]	155	OUT_CELL	-
TX6D[1]	154	OUT_CELL	-
OEB_TX6D[1]	153	OUT_CELL	-
TX6D[0]	152	OUT_CELL	-
OEB_TX6D[0]	151	OUT_CELL	-
TX7CLK	150	OUT_CELL	-
OEB_TX7CLK	149	OUT_CELL	-
TX7D[9]	148	OUT_CELL	-
OEB_TX7D[9]	147	OUT_CELL	-
TX7D[8]	146	OUT_CELL	-
OEB_TX7D[8]	145	OUT_CELL	-
TX7D[7]	144	OUT_CELL	-
OEB_TX7D[7]	143	OUT_CELL	-
TX7D[6]	142	OUT_CELL	-
OEB_TX7D[6]	141	OUT_CELL	-
TX7D[5]	140	OUT_CELL	-
OEB_TX7D[5]	139	OUT_CELL	-
TX7D[4]	138	OUT_CELL	-

OEB_TX7D[4]	137	OUT_CELL	-
TX7D[3]	136	OUT_CELL	-
OEB_TX7D[3]	135	OUT_CELL	-
TX7D[2]	134	OUT_CELL	-
OEB_TX7D[2]	133	OUT_CELL	-
TX7D[1]	132	OUT_CELL	-
OEB_TX7D[1]	131	OUT_CELL	-
TX7D[0]	130	OUT_CELL	-
OEB_TX7D[0]	129	OUT_CELL	-
INT	128	OUT_CELL	-
OEB_INT	127	OUT_CELL	-
DEBUG[0]	126	OUT_CELL	-
OEB_DEBUG[0]	125	OUT_CELL	-
DEBUG[1]	124	OUT_CELL	-
OEB_DEBUG[1]	123	OUT_CELL	-
DEBUG[2]	122	OUT_CELL	-
OEB_DEBUG[2]	121	OUT_CELL	-
DEBUG[3]	120	OUT_CELL	-
OEB_DEBUG[3]	119	OUT_CELL	-
RESETI	118	IN_CELL	-
TEST_MODE	117	IN_CELL	-
TEST_SE	116	IN_CELL	-
XREFCK135	115	IN_CELL	-
XREFCK125	114	IN_CELL	-
SYSCLK	113	IN_CELL	-
MI_CS	112	IN_CELL	-
MI_RD	111	IN_CELL	-
MI_WR	110	IN_CELL	-
MI_ALE	109	IN_CELL	-
MI_AD[0]	108	IO_CELL	-

OEB_MI_AD[0]	107	OUT_CELL	-
MI_AD[1]	106	IO_CELL	-
OEB_MI_AD[1]	105	OUT_CELL	-
MI_AD[2]	104	IO_CELL	-
OEB_MI_AD[2]	103	OUT_CELL	-
MI_AD[3]	102	IO_CELL	-
OEB_MI_AD[3]	101	OUT_CELL	-
MI_AD[4]	100	IO_CELL	-
OEB_MI_AD[4]	99	OUT_CELL	-
MI_AD[5]	98	IO_CELL	-
OEB_MI_AD[5]	97	OUT_CELL	-
MI_AD[6]	96	IO_CELL	-
OEB_MI_AD[6]	95	OUT_CELL	-
MI_AD[7]	94	IO_CELL	-
OEB_MI_AD[7]	93	OUT_CELL	-
MI_AD[8]	92	IO_CELL	-
OEB_MI_AD[8]	91	OUT_CELL	-
MI_AD[9]	90	IO_CELL	-
OEB_MI_AD[9]	89	OUT_CELL	-
MI_AD[10]	88	IO_CELL	-
OEB_MI_AD[10]	87	OUT_CELL	-
MI_AD[11]	86	IO_CELL	-
OEB_MI_AD[11]	85	OUT_CELL	-
MI_AD[12]	84	IO_CELL	-
OEB_MI_AD[12]	83	OUT_CELL	-
MI_AD[13]	82	IO_CELL	-
OEB_MI_AD[13]	81	OUT_CELL	-
MI_AD[14]	80	IO_CELL	-
OEB_MI_AD[14]	79	OUT_CELL	-
MI_AD[15]	78	IO_CELL	-

OEB_MI_AD[15]	77	OUT_CELL	-
RX7CLK	76	IN_CELL	-
RX7D[9]	75	IN_CELL	-
RX7D[8]	74	IN_CELL	-
RX7D[7]	73	IN_CELL	-
RX7D[6]	72	IN_CELL	-
RX7D[5]	71	IN_CELL	-
RX7D[4]	70	IN_CELL	-
RX7D[3]	69	IN_CELL	-
RX7D[2]	68	IN_CELL	-
RX7D[1]	67	IN_CELL	-
RX7D[0]	66	IN_CELL	-
RX6CLK	65	IN_CELL	-
RX6D[9]	64	IN_CELL	-
RX6D[8]	63	IN_CELL	-
RX6D[7]	62	IN_CELL	-
RX6D[6]	61	IN_CELL	-
RX6D[5]	60	IN_CELL	-
RX6D[4]	59	IN_CELL	-
RX6D[3]	58	IN_CELL	-
RX6D[2]	57	IN_CELL	-
RX6D[1]	56	IN_CELL	-
RX6D[0]	55	IN_CELL	-
RX5CLK	54	IN_CELL	-
RX5D[9]	53	IN_CELL	-
RX5D[8]	52	IN_CELL	-
RX5D[7]	51	IN_CELL	-
RX5D[6]	50	IN_CELL	-
RX5D[5]	49	IN_CELL	-
RX5D[4]	48	IN_CELL	-

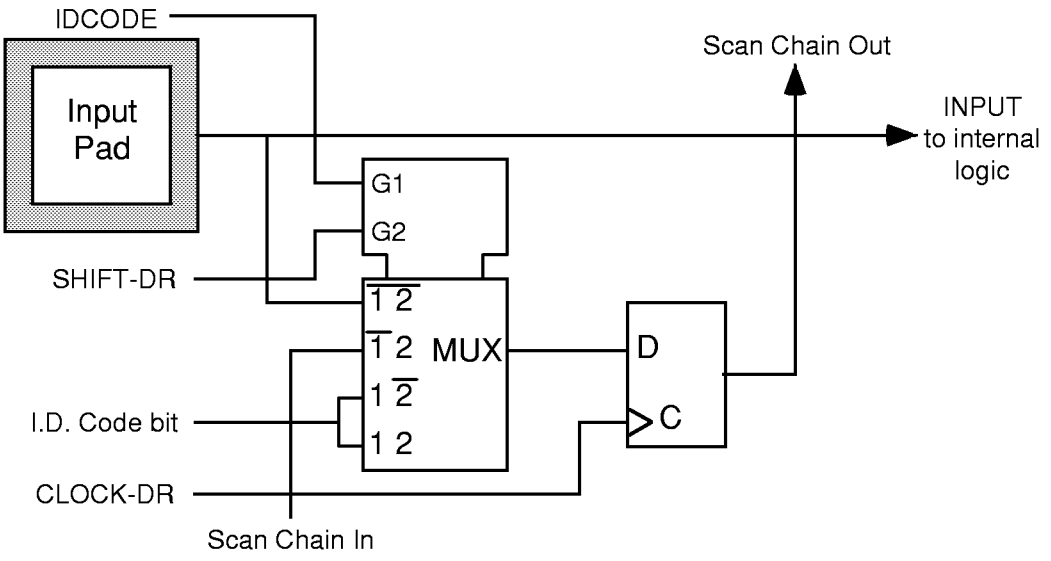
RX5D[3]	47	IN_CELL	-
RX5D[2]	46	IN_CELL	-
RX5D[1]	45	IN_CELL	-
RX5D[0]	44	IN_CELL	-
RX4CLK	43	IN_CELL	-
RX4D[9]	42	IN_CELL	-
RX4D[8]	41	IN_CELL	-
RX4D[7]	40	IN_CELL	-
RX4D[6]	39	IN_CELL	-
RX4D[5]	38	IN_CELL	-
RX4D[4]	37	IN_CELL	-
RX4D[3]	36	IN_CELL	-
RX4D[2]	35	IN_CELL	-
RX4D[1]	34	IN_CELL	-
RX4D[0]	33	IN_CELL	-
RX3CLK	32	IN_CELL	-
RX3D[9]	31	IN_CELL	-
RX3D[8]	30	IN_CELL	-
RX3D[7]	29	IN_CELL	-
RX3D[6]	28	IN_CELL	-
RX3D[5]	27	IN_CELL	-
RX3D[4]	26	IN_CELL	-
RX3D[3]	25	IN_CELL	-
RX3D[2]	24	IN_CELL	-
RX3D[1]	23	IN_CELL	-
RX3D[0]	22	IN_CELL	-
RX2CLK	21	IN_CELL	-
RX2D[9]	20	IN_CELL	-
RX2D[8]	19	IN_CELL	-
RX2D[7]	18	IN_CELL	-

RX2D[6]	17	IN_CELL	-
RX2D[5]	16	IN_CELL	-
RX2D[4]	15	IN_CELL	-
RX2D[3]	14	IN_CELL	-
RX2D[2]	13	IN_CELL	-
RX2D[1]	12	IN_CELL	-
RX2D[0]	11	IN_CELL	-
RX1CLK	10	IN_CELL	-
RX1D[9]	9	IN_CELL	-
RX1D[8]	8	IN_CELL	-
RX1D[7]	7	IN_CELL	-
RX1D[6]	6	IN_CELL	-
RX1D[5]	5	IN_CELL	-
RX1D[4]	4	IN_CELL	-
RX1D[3]	3	IN_CELL	-
RX1D[2]	2	IN_CELL	-
RX1D[1]	1	IN_CELL	-
RX1D[0]	0	IN_CELL	-
TDO	-	JTAG data out	-

Notes:

1. RX0CLK is the first bit of the scan chain (closest to TDI).
2. Enable cell OEB_pinname, tristates pin pinname when set high.

FIGURE 5 Input Observation Cell (IN_CELL)



In this diagram and those that follow, CLOCK-DR is equal to TCK when the current controller state is SHIFT-DR or CAPTURE-DR, and unchanging otherwise. The multiplexor in the center of the diagram selects one of four inputs, depending on the status of select lines G1 and G2. The ID Code bit is as listed in the table above.

FIGURE 6 Output Cell (OUT_CELL)

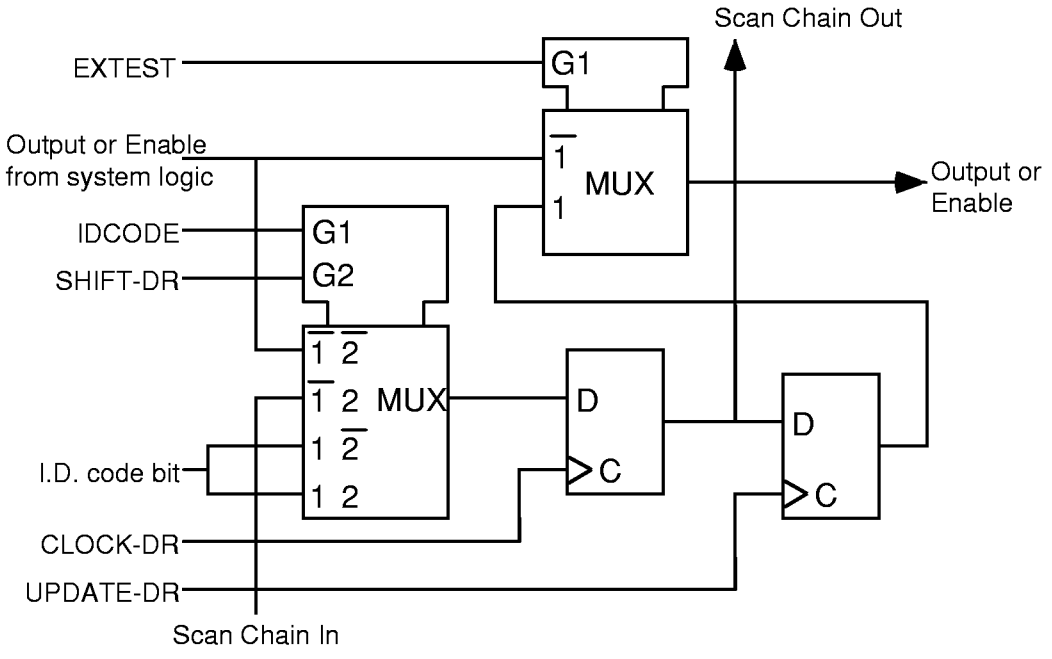


FIGURE 7 Bi-directional Cell (IO_CELL)

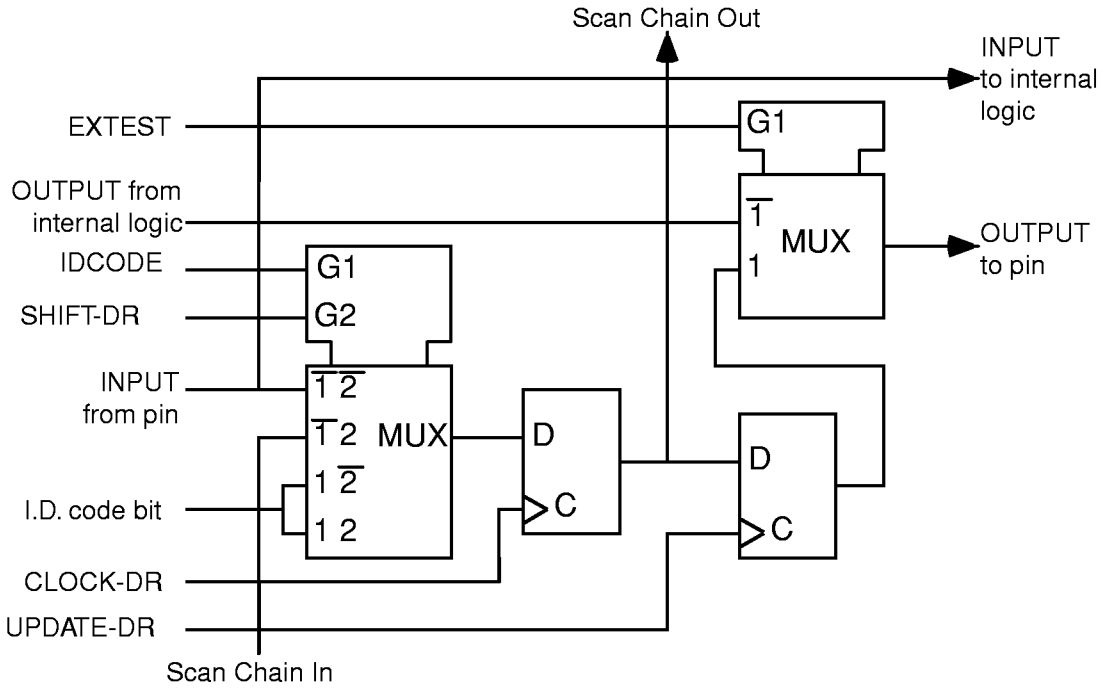
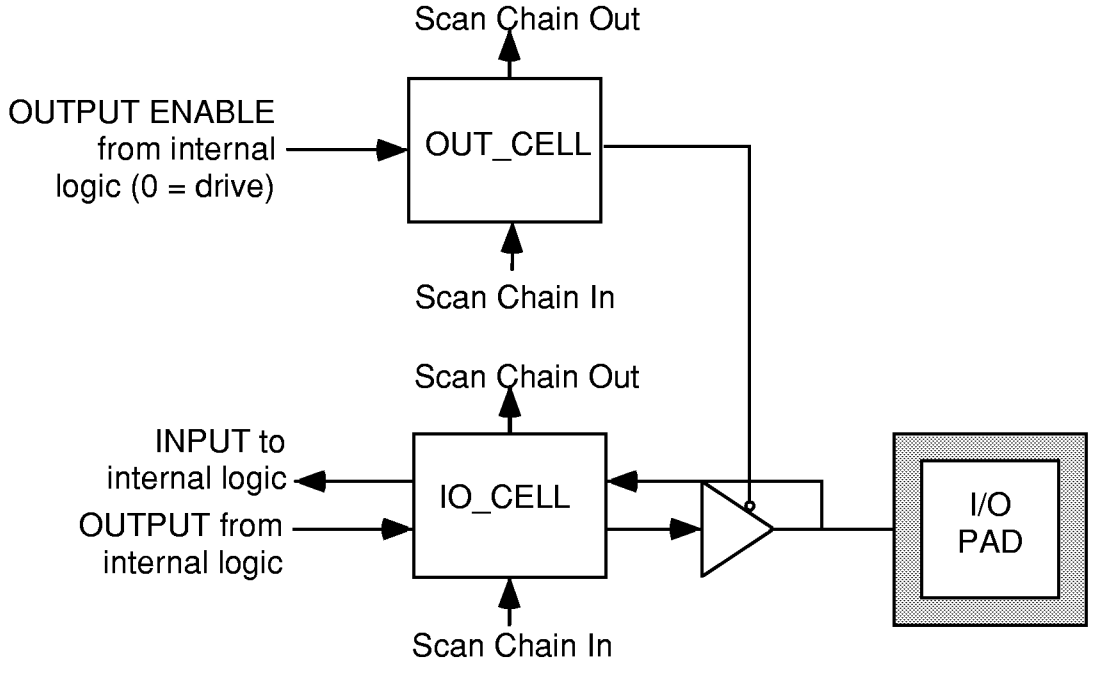


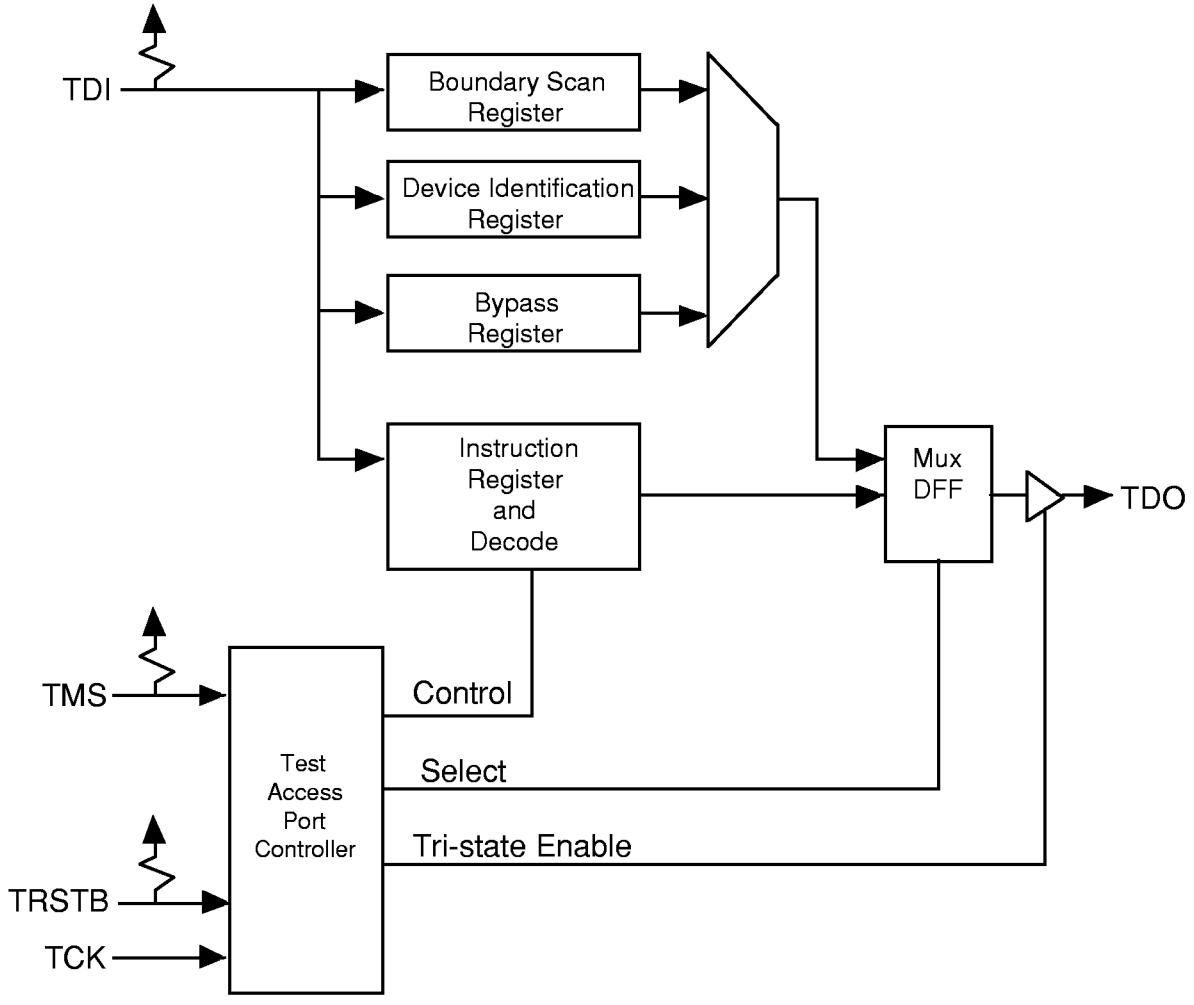
FIGURE 8 Layout of Output Enable and Bi-directional Cells



9.7 JTAG Support

The PM3390 supports the IEEE Boundary Scan Specification as described in the IEEE 1149.1 standards. The Test Access Port (TAP) consists of the five standard pins, TRSTB, TCK, TMS, TDI and TDO used to control the TAP controller and the boundary scan registers. The TRSTB input is the active low reset signal used to reset the TAP controller. TCK is the test clock used to sample data on input, TDI and to output data on output, TDO. The TMS input is used to direct the TAP controller through its states. The basic boundary scan architecture is shown below.

FIGURE 9 Boundary Scan Architecture



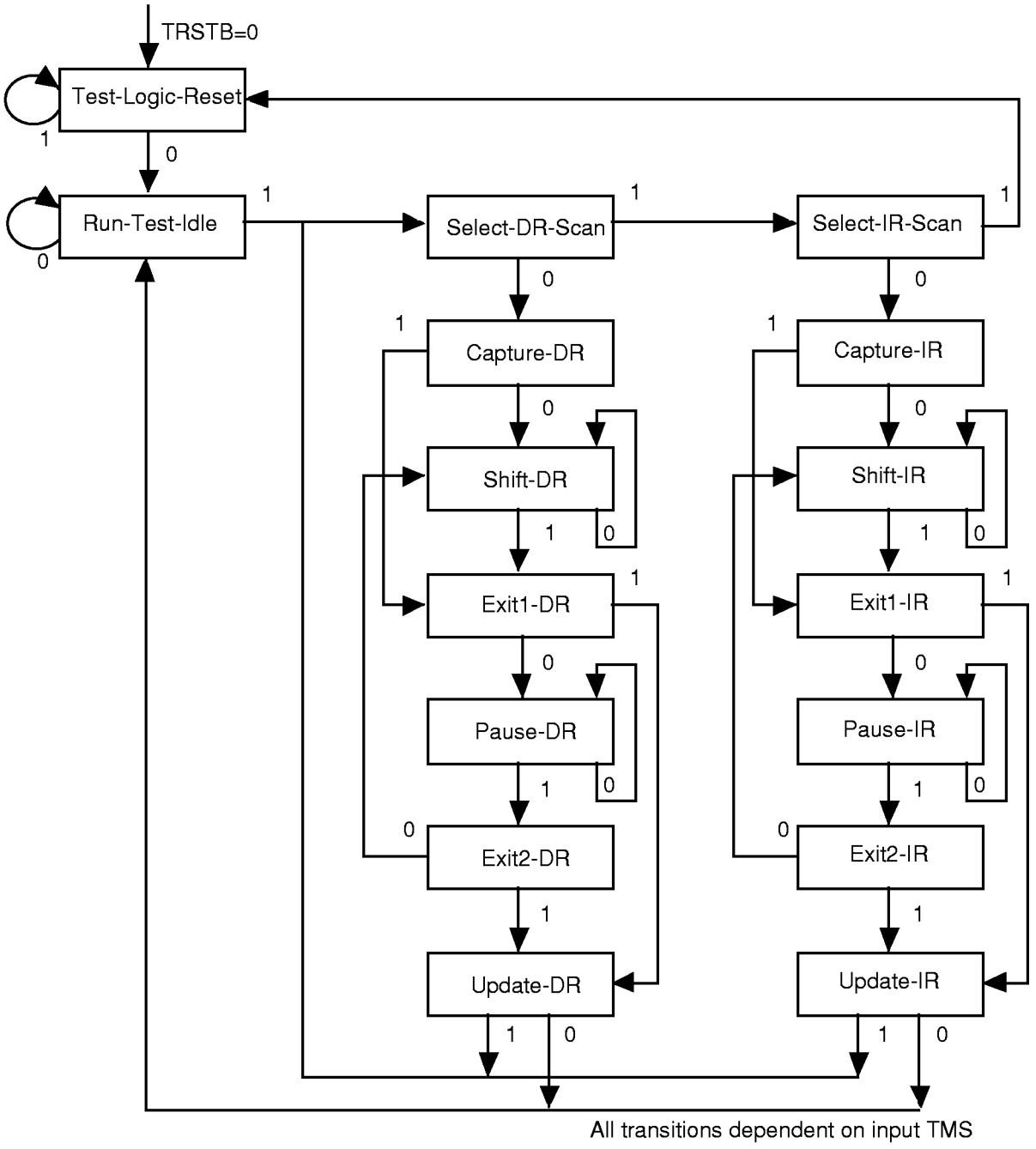
The boundary scan architecture consists of a TAP controller, an instruction register with instruction decode, a bypass register, a device identification register and a boundary scan register. The TAP controller interprets the TMS input and generates control signals to load the instruction and data registers. The instruction register with instruction decode block is used to select the test to be executed and/or the register to be accessed. The bypass register offers a single bit delay from primary input, TDI to primary output, TDO. The device identification register contains the device identification code.

The boundary scan register allows testing of board inter-connectivity. The boundary scan register consists of a shift register placed in series with device inputs and outputs. Using the boundary scan register, all digital inputs can be sampled and shifted out on primary output TDO. In addition, patterns can be shifted in on primary input, TDI and forced onto all digital outputs.

9.8 TAP Controller

The TAP controller is a synchronous finite state machine clocked by the rising edge of primary input, TCK. All state transitions are controlled using primary input, TMS. The finite state machine is described below.

FIGURE 10 TAP Controller Finite State Machine



Test-Logic-Reset

The test logic reset state is used to disable the TAP logic when the device is in normal mode operation. The state is entered asynchronously by asserting input, TRSTB. The state is entered synchronously regardless of the current TAP controller state by forcing input, TMS high for 5 TCK clock cycles. While in this state, the instruction register is set to the IDCODE instruction.

Run-Test-Idle

The run test/idle state is used to execute tests.

Capture-DR

The capture data register state is used to load parallel data into the test data registers selected by the current instruction. If the selected register does not allow parallel loads or no loading is required by the current instruction, the test register maintains its value. Loading occurs on the rising edge of TCK.

Shift-DR

The shift data register state is used to shift the selected test data registers by one stage. Shifting is from MSB to LSB and occurs on the rising edge of TCK.

Update-DR

The update data register state is used to load a test register's parallel output latch. In general, the output latches are used to control the device. For example, for the EXTEST instruction, the boundary scan test register's parallel output latches are used to control the device's outputs. The parallel output latches are updated on the falling edge of TCK.

Capture-IR

The capture instruction register state is used to load the instruction register with a fixed instruction. The load occurs on the rising edge of TCK.

Shift-IR

The shift instruction register state is used to shift both the instruction register and the selected test data registers by one stage. Shifting is from MSB to LSB and occurs on the rising edge of TCK.

Update-IR

The update instruction register state is used to load a new instruction into the instruction register. The new instruction must be scanned in using the Shift-IR state. The load occurs on the falling edge of TCK.

The Pause-DR and Pause-IR states are provided to allow shifting through the test data and/or instruction registers to be momentarily paused.

Boundary Scan Instructions

The following is a description of the standard instructions. Each instruction selects a serial test data register path between input, TDI and output, TDO.

BYPASS

The bypass instruction shifts data from input, TDI to output, TDO with one TCK clock period delay. The instruction is used to bypass the device.

EXTEST

The external test instruction allows testing of the interconnection to other devices. When the current instruction is the EXTEST instruction, the boundary scan register is place between input, TDI and output, TDO. Primary device inputs can be sampled by loading the boundary scan register using the Capture-DR state. The sampled values can then be viewed by shifting the boundary scan register using the Shift-DR state. Primary device outputs can be controlled by loading patterns shifted in through input TDI into the boundary scan register using the Update-DR state.

SAMPLE

The sample instruction samples all the device inputs and outputs. For this instruction, the boundary scan register is placed between TDI and TDO. Primary device inputs and outputs can be sampled by loading the boundary scan register using the Capture-DR state. The sampled values can then be viewed by shifting the boundary scan register using the Shift-DR state.

IDCODE

The identification instruction is used to connect the identification register between TDI and TDO. The device's identification code can then be shifted out using the Shift-DR state.

STCTEST

The single transport chain instruction is used to test out the TAP controller and the boundary scan register during production test. When this instruction is the current instruction, the boundary scan register is connected between TDI and TDO. During the Capture-DR state, the device identification code is loaded into the boundary scan register. The code can then be shifted out output, TDO using the Shift-DR state.

INTEST

The internal test instruction is used to exercise the device's internal core logic. When this instruction is the current instruction, the boundary scan register is connected between TDI and TDO. During the Update-DR state, patterns shifted in on input, TDI are used to drive primary inputs. During the Capture-DR state, primary outputs are sampled and loaded into the boundary scan register.

10 NORMAL MODE REGISTER DESCRIPTION

10.1 Notes on Normal Mode Register Bits:

1. Writing values into Reserved register bits can have side-effects since some registers marked as "Reserved" are actually implemented for internal use. **Unless explicitly noted in the register description for that register, all Reserved bits in the register are to be written to their default value.** Failure to do so may result in having the PM3390 device operate incorrectly. Reading back Reserved bits can produce either a logic one or a logic zero; hence Reserved register bits should be masked off by software when read.

2. All configuration bits that can be written into can also be read back, unless noted in the register description. This allows the external processor controlling the PM3390 to determine the programming state of the block.

3. Writeable normal mode register bits are cleared to logic zero upon reset unless otherwise noted.

4. Writing into read-only normal mode register bit locations does not affect the operation of the PM3390 unless otherwise noted.

4. Unless explicitly noted in the register description, the PM3390 will allow CRI access to the that register through either the external microprocessor interface or in-band by using an EXACT bus Control Register read/write message. For easy cross-reference, the list of registers that are not universally accessible is given in the following table.

CRI address	Register name	Register Access allowed	Register Access excluded
0x0008	CMIF Device ID Mirror	External micro	EXACT bus
0x0009	CMIF Device Control Mirror	External micro	EXACT bus
0x000A	CMIF Device Status Mirror	External micro	EXACT bus
0x000B	DLT CRI Status	External micro	EXACT bus

0x000C	DLT CRI Control	External micro	EXACT bus
0x000D	DLT CRI Write	External micro	EXACT bus
0x0010	Message Box Command	External micro	EXACT bus
0x0011	Message box status	External micro	EXACT bus
0x0012	Message Box Receive Buffer Pointer	External micro	EXACT bus
0x0013	Message Box Receive Counter	External micro	EXACT bus
0x0014	Message Box Error Counter	External micro	EXACT bus
0x0400-0x4FF	Message Box RX/TX Buffer	External micro	EXACT bus
0x1000-0x13FF	Destination Lookup Table	EXACT bus	External micro

10.2 Control Register Interface Address Space

Each functional block of the PM3390 has a separate base address for accessing the control registers within that block. The CMIF is responsible for decoding the address into a block address / control register pair and initiating the access with the appropriate internal block. The base addresses for each of the functional blocks within the PM3390 are listed in the following table.

PM3390 Register Block	Base Address	Notes
CMIF	0x0000	
CMIF MESSAGE BOX RX/TX BUFFER	0x0400	RX Buffer mapped directly to CRI address space: 0x0400-0x047F. TX Buffer maps to 0x0480-0x04FF.
FLM	0x0800	
CLM	0x0C00	
EXACT access of DLT	0x1000	Mapped directly to CRI address space 0x1000-0x13FF
XIB0	0x8000	
XOB0	0x8400	
XIB1	0x8800	
XOB1	0x8C00	
XIB2	0x9000	
XOB2	0x9400	
XIB3	0x9800	
XOB3	0x9C00	
XIB4	0xA000	
XOB4	0xA400	
XIB5	0xA800	
XOB5	0xAC00	
XIB6	0xB000	
XOB6	0xB400	
XIB7	0xB800	
XOB7	0xBC00	

Table 4 PM3390 Control Register Base Addresses

Within each functional block the registers are accessed starting with location 0x0000. The bits in the address[15:0] that are decoded as the base_address

are address [15:10]; that is, the high six bits of the address. In the next section the control registers for each of the functional blocks within the PM3390 are described.

The first three registers in the PM3390 CRI address space are the EXACT Device Registers: CRI addresses of 0x0000, 0x0001, and 0x0002. They provide PM3390 device-specific configuration and status functions and are the only 32 bit registers on the PM3390. The EXACT Control message format allows 32 bit register accesses whereas the external microprocessor interface allows only 16-bit register accesses. Therefore an alternate address is provided to allow an external microprocessor interface to read/write the upper 16 bits of these three registers at locations 0x0008, 0x0009, and 0x000A.

10.2.1 General Purpose Software Registers

There are a set of sixteen registers that can be used on the PM3390 as general purpose registers by external software. These registers are physically located on the PM3390 in the XOB output block: there are two sixteen bit registers per XOB, and eight XOB's in the design. The registers are called "General Purpose Software Registers 0/1" (i.e. GP0 and GP1) and may be found in the XOB register description as 0x05 and 0x06. For easy reference, the 16-bit CRI addresses of the GP0 and GP1 registers is given in the following table:

CRI register address	XOB port	Naming convention
0x8405	0	GP00
0x8406	0	GP01
0x8C05	1	GP10
0x8C06	1	GP11
0x9405	2	GP20
0x9406	2	GP21
0x9C05	3	GP30
0x9C06	3	GP31

0xA405	4	GP40
0xA406	4	GP41
0xAC05	5	GP50
0xAC06	5	GP51
0xB405	6	GP60
0xB406	6	GP61
0xBC05	7	GP70
0xBC06	7	GP71

NOTE: in an Ethernet-based switch system that uses microcode developed by PMC-Sierra these registers may be already be used so please contact the factory before attempting to write to them.

10.3 CMIF Registers

CMIF Register 0x00: Device ID Register (32 bits)[1]

BIT	TYPE	FUNCTION	DEFAULT
31	R	DeviceType bit 7	1
30	R	DeviceType bit 6	0
29	R	DeviceType bit 5	0
28	R	DeviceType bit 4	0
27	R	DeviceType bit 3	0
26	R	DeviceType bit 2	0
25	R	DeviceType bit 1	0
24	R	DeviceType bit 0	0
23	R	DeviceRevNum bit 7	0
22	R	DeviceRevNum bit 6	0
21	R	DeviceRevNum bit 5	0
20	R	DeviceRevNum bit 4	0
19	R	DeviceRevNum bit 3	0
18	R	DeviceRevNum bit 2	0
17	R	DeviceRevNum bit 1	0
16	R	DeviceRevNum bit 0	0
15		Reserved	0
14		Reserved	0
13		Reserved	0
12		Reserved	0
11		Reserved	0
10		Reserved	0
9	R/W	DeviceAddress bit 9	Pinstrap
8	R/W	DeviceAddress bit 8	Pinstrap
7	R/W	DeviceAddress bit 7	Pinstrap
6	R/W	DeviceAddress bit 6	Pinstrap
5	R/W	DeviceAddress bit 5	Pinstrap
4	R/W	DeviceAddress bit 4	Pinstrap
3	R/W	DeviceAddress bit 3	Pinstrap
2	R/W	DeviceAddress bit 2	Pinstrap
1	R/W	DeviceAddress bit 1	Pinstrap
0	R/W	DeviceAddress bit 0	Pinstrap

DeviceType [7:0]:

The DeviceType field is a hardwired value and may not be modified through CRI access. The Device Type number for the PM3390 is decimal 128.

DeviceRevNum [7:0]:

The DeviceRevNum field is a hardwired value and may not be modified through CRI access. It represents the numeric revision number of the PM3390 device.

[1] NOTE: Bits 31:16 of the Device ID Register are mirrored at address 0x08 for access through the external microprocessor interface.

DeviceAddress[9:0]:

During the reset cycle, the PM3390 DeviceAddress field is written with the 10 bit Address pin-strap value (see section titled "Configuration and Initialization for details). The Address field may be read and written through CRI access at any time after reset.

This address field is referred to as the "logical port address" in the EXACT Bus protocol specification. This address is the terminating address for EXACT messages to a given PM3390 device. For multiple PM3390 devices in a system, each PM3390 device must have a unique DeviceAddress field if the system implementation requires that CRI registers in each of the devices be accessed over the EXACT bus. This restriction on CRI register accesses does not apply to external microprocessor accesses, since it is the assertion of the MI_CS pin (chip select) that does the required device selection.

CMIF Register 0x01: Device Control Register (32 bits)[2]

BIT	TYPE	FUNCTION	DEFAULT
31	W	Master Reset	0
30	R/W	Halt	0
29	**R/W**	**Reserved**	**0**
28	**R/W**	**Reserved**	**1**
27	R/W	EXPIF_Mode bit 3	Pinstrap
26	R/W	EXPIF_Mode bit 2	Pinstrap
25	R/W	EXPIF_Mode bit 1	Pinstrap
24	R/W	EXPIF_Mode bit 0	Pinstrap
23	**R/W**	**Reserved**	**0**
22	**R/W**	**Reserved**	**0**
21	**R/W**	**Reserved**	**0**
20	**R/W**	**Reserved**	**0**
19		Reserved	0
18		Reserved	0
17		Reserved	0
16		Reserved	0
15	**R/W**	**Reserved**	**0**
14	**R/W**	**Reserved**	**0**
13	**R/W**	**Reserved**	**0**
12	**R/W**	**Reserved**	**0**
11	**R/W**	**Reserved**	**0**
10	**R/W**	**Reserved**	**0**
9	**R/W**	**Reserved**	**0**
8	**R/W**	**Reserved**	**0**
7		Reserved	0
6		Reserved	0
5		Reserved	0
4		Reserved	0
3		Reserved	0
2		Reserved	0
1		Reserved	0
0		Reserved	0

This register allows configuration of the PM3390 device.

Master Reset

The Master Reset bit initiates a device soft reset and self-clears after 1 SYSCLK when written to 1. The Master Reset bit always returns 0 when read.

[2] NOTE: Bits 31:16 of the Device Control Register are mirrored at address 0x09 for access through the external microprocessor interface.

Halt

The Halt bit prevents normal traffic from being routed through the CCS when set to 1.

EXPIF_Mode[3:0]

Default value depends on setting of EXPIF_ON pin-strap value. If EXPIF_ON is 1, then the default value of EXPIF_Mode[3:0] is 4'hF; if EXPIF_ON is 0, then the default value is 4'h0.

The value in this register must not be changed after reset. Each Expansion_Interface_Mode bit corresponds to an output ring on the device:

EXPIF_Mode bit	XOB ring configured
0	4
1	5
2	6
3	7

CMIF Register 0x02: Device Status Register (32 bits)[3]

BIT	TYPE	FUNCTION	DEFAULT
31	R	Reset Status	0
30		Reserved	0
29		Reserved	0
28		Reserved	0
27		Reserved	0
26		Reserved	0
25		Reserved	0
24		Reserved	0
23		Reserved	0
22		Reserved	0
21		Reserved	0
20		Reserved	0
19		Reserved	0
18		Reserved	0
17		Reserved	0
16		Reserved	0
15		Reserved	0
14		Reserved	0
13		Reserved	0
12	R	Last CMIF Tx Type bit 1	0
11	R	Last CMIF Tx Type bit 0	0
10	R	Last CMIF Tx Port bit 2	0
9	R	Last CMIF Tx Port bit 1	0
8	R	Last CMIF Tx Port bit 0	0
7		Reserved	0
6		Reserved	0
5	R	Interrupt_State	0
4	R	DeviceSubType	0
3	R	INT_Code bit 3	0
2	R	INT _Code bit 2	0
1	R	INT _Code bit 1	0
0	R	INT _Code bit 0	0

Reset_Status:

When set, the Reset_Status bit indicates that the PM3390 has not finished its 1024 SYSCLK initialization period.

Last CMIF TxType [1:0]

Last CMIF TxPort[2:0]

The type and TX port of the last message sent by the CMIF are

[3] NOTE: Bits 31:16 of the Device Status Register are mirrored at address 0x0A for access through the external microprocessor interface.

provided by these status bits. The 3-bit field for Last CMIF TxPort indicates the last TX port to which the CMIF transmitted a message. The 2-bit type field indicates the type of message last sent by the CMIF.

Last CMIF TxType[1:0]	Description
00	No messages sent (default state after reset)
01	EXACT Control register Status response
10	Message Box TX
11	Broadcast

Interrupt_State

The logic value of this bit corresponds to the logic value on the INT pin of the device.

DeviceSubType

: this bit is read as a 0.

INT_Code[3:0

The INT_Code field can be used to decode the type of interrupt that occurred. The "valid" decode for this bit field is the Interrupt_State bit: if that bit is not set (i.e. 1) then this code is invalid.

The INT_Code reflects the interrupt condition that occurred when the INT pin of the device went from a de-asserted to asserted state. The code is latched upon INT assertion and held until this register (CMIF Device Status) is read. The INT pin will also be de-asserted when the register is read.

None of the Interrupt conditions are maskable (that is, there is no interrupt mask bits implemented on the PM3390).

INT_Code[3:0]	Description (valid if Interrupt_State is 1)
4'b1111	CMIF Message Box Receive Buffer non-empty (there is at least one valid message in the Receive Buffer)
4'b1000	Free List Manager logic block discarded a cell

	because there was no free space in the Central Store buffer. The discard occurred at one of the XIB ports.
4'b0000	XIB for port 0 reset by CLM because a sequence error was detected.
4'b0001	XIB for port 1 reset by CLM because a sequence error was detected.
4'b0010	XIB for port 2 reset by CLM because a sequence error was detected.
4'b0011	XIB for port 3 reset by CLM because a sequence error was detected.
4'b0100	XIB for port 4 reset by CLM because a sequence error was detected.
4'b0101	XIB for port 5 reset by CLM because a sequence error was detected.
4'b0110	XIB for port 6 reset by CLM because a sequence error was detected.
4'b0111	XIB for port 7 reset by CLM because a sequence error was detected.

The interrupt codes for XIB port reset or Free List Manager block discard should never be asserted during normal operation of the device.

The interrupt code for the XIB port reset were implemented to aid in product validation by PMC-Sierra; the bit should never be read as being set (i.e. 1).

The interrupt code for Free List Manager discard of a cell should never occur in a system that is constructed using the PM3370 and PM3380 Ethernet port controller chips. If using the "non-EXACT" flow control mechanism -AND- this bit is read as being set (1), then the flow control as implemented by the non-EXACT entity did not work to prevent at least one EXACT message from being discarded.

CMIF Register 0x03 Broadcast Port Address

BIT	TYPE	FUNCTION	DEFAULT
15		Reserved	0
14		Reserved	0
13		Reserved	0
12		Reserved	0
11		Reserved	0
10		Reserved	0
9	R/W	Broadcast Port Address bit 9	1
8	R/W	Broadcast Port Address bit 8	1
7	R/W	Broadcast Port Address bit 7	1
6	R/W	Broadcast Port Address bit 6	1
5	R/W	Broadcast Port Address bit 5	1
4	R/W	Broadcast Port Address bit 4	1
3	R/W	Broadcast Port Address bit 3	1
2	R/W	Broadcast Port Address bit 2	1
1	R/W	Broadcast Port Address bit 1	1
0	R/W	Broadcast Port Address bit 0	1

Broadcast Port Address[9:0]:

Messages with a destination address matching the Broadcast Port Address are recognized by the PM3390 as EXACT broadcast messages. The Broadcast Port Address defaults to hex address 0x3FF and may be written through CRI access at any time after reset.

CMIF Register 0x04 Broadcast Hold-off Timeout

BIT	TYPE	FUNCTION	DEFAULT
15	R/W	Timeout Value bit 15	1
14	R/W	Timeout Value bit 14	1
13	R/W	Timeout Value bit 13	1
12	R/W	Timeout Value bit 12	1
11	R/W	Timeout Value bit 11	1
10	R/W	Timeout Value bit 10	1
9	R/W	Timeout Value bit 9	1
8	R/W	Timeout Value bit 8	1
7	R/W	Timeout Value bit 7	1
6	R/W	Timeout Value bit 6	1
5	R/W	Timeout Value bit 5	1
4	R/W	Timeout Value bit 4	1
3	R/W	Timeout Value bit 3	1
2	R/W	Timeout Value bit 2	1
1	R/W	Timeout Value bit 1	1
0	R/W	Timeout Value bit 0	1

Timeout Value[15:0]:

When the CMIF starts transmission of a broadcast message it reloads an internal timer, called the Broadcast Hold-off timer, with TimeoutValue[15:0]. The Broadcast Hold-off timer decrements once each SYSCLK cycle if it is non-zero. Reception of new broadcast messages is prevented until the Broadcast Hold-off timer has decremented to zero.

CMIF Register 0x05 Broadcast Enable

BIT	TYPE	FUNCTION	DEFAULT
15		Reserved	
14		Reserved	
13		Reserved	
12		Reserved	
11		Reserved	
10		Reserved	
9		Reserved	
8		Reserved	
7	R/W	XOB Ring 7 Enable	0
6	R/W	XOB Ring 6 Enable	0
5	R/W	XOB Ring 5 Enable	0
4	R/W	XOB Ring 4 Enable	0
3	R/W	XOB Ring 3 Enable	0
2	R/W	XOB Ring 2 Enable	0
1	R/W	XOB Ring 1 Enable	0
0	R/W	XOB Ring 0 Enable	0

Broadcast Enable[7:0]:

Bits [7:0] enable broadcast transmission for each of the 8 XOB rings when set.

CMIF Register 0x06 EXACT TX Port Output Enable

BIT	TYPE	FUNCTION	DEFAULT
15	R/W	TX7D Tri-state Enable	0
14	R/W	TX6D Tri-state Enable	0
13	R/W	TX5D Tri-state Enable	0
12	R/W	TX4D Tri-state Enable	0
11	R/W	TX3D Tri-state Enable	0
10	R/W	TX2D Tri-state Enable	0
9	R/W	TX1D Tri-state Enable	0
8	R/W	TX0D Tri-state Enable	0
7	R/W	TX7CLK Tri-state Enable	0
6	R/W	TX6CLK Tri-state Enable	0
5	R/W	TX5CLK Tri-state Enable	0
4	R/W	TX4CLK Tri-state Enable	0
3	R/W	TX3CLK Tri-state Enable	0
2	R/W	TX2CLK Tri-state Enable	0
1	R/W	TX1CLK Tri-state Enable	0
0	R/W	TX0CLK Tri-state Enable	0

TXD Tri-State Enables :

Bits [15:8], when set, put the TXD[9:0] port corresponding to that bit into tri-state (i.e. a high-impedance state); when the bit is clear, the TXD port is driven by the PM3390.

Note: during the device reset sequence (which is 1024 SYSCLK cycles following the falling edge of RESETI) all of the TXD ports are tri-stated.

TXCLK Tri-State Enables :

Bits [7:0] put the TXCLK pin corresponding to that bit into tri-state (i.e. a high-impedance state). when the bit is clear, the TXCLK pin is driven by the PM3390

CMIF Register 0x08: Device ID Register Mirror

BIT	TYPE	FUNCTION	DEFAULT
15	R	DeviceType bit 7	1
14	R	DeviceType bit 6	0
13	R	DeviceType bit 5	0
12	R	DeviceType bit 4	0
11	R	DeviceType bit 3	0
10	R	DeviceType bit 2	0
9	R	DeviceType bit 1	0
8	R	DeviceType bit 0	0
7	R	DeviceRevNum bit 7	0
6	R	DeviceRevNum bit 6	0
5	R	DeviceRevNum bit 5	0
4	R	DeviceRevNum bit 4	0
3	R	DeviceRevNum bit 3	0
2	R	DeviceRevNum bit 2	0
1	R	DeviceRevNum bit 1	0
0	R	DeviceRevNum bit 0	0

Access to this CRI register is via the external microprocessor interface only.

DeviceType [7:0]:

The DeviceType field is a hardwired value and may not be modified through CRI access. The Device Type number for the PM3390 is decimal 128.

DeviceRevNum [7:0]:

The DeviceRevNum field is a hardwired value and may not be modified through CRI access. It represents the numeric revision number of the PM3390 device.

CMIF Register 0x09: Device Control Register Mirror

BIT	TYPE	FUNCTION	DEFAULT
15	W	Master Reset	0
14	R/W	Halt	0
13	**R/W**	**Reserved**	**0**
12	**R/W**	**Reserved**	**1**
11	R/W	EXPIF_Mode bit 3	Pinstrap
10	R/W	EXPIF_Mode bit 2	Pinstrap
9	R/W	EXPIF_Mode bit 1	Pinstrap
8	R/W	EXPIF_Mode bit 0	Pinstrap
7	**R/W**	**Reserved**	**0**
6	**R/W**	**Reserved**	**0**
5	**R/W**	**Reserved**	**0**
4	**R/W**	**Reserved**	**0**
3		Reserved	0
2		Reserved	0
1		Reserved	0
0		Reserved	0

Access to this CRI register is via the external microprocessor interface only.

Master Reset

The Master Reset bit initiates a device soft reset and self-clears after 1 SYSCLK when written to 1. The Master Reset bit always returns 0 when read.

Halt

The Halt bit prevents normal traffic from being routed through the CCS when set to 1.

EXPIF_Mode[3:0]

Default value depends on setting of EXPIF_ON pin-strap value. If EXPIF_ON is 1, then the default value of EXPIF_Mode[3:0] is 4'hF; if EXPIF_ON is 0, then the default value is 4'h0.

The value in this register must not be changed after reset. Each Expansion_Interface_Mode bit corresponds to an output ring on the device:

EXPIF_Mode bit	XOB ring configured
0	4
1	5

2	6
3	7

CMIF Register 0x0A: Device Status Register Mirror

BIT	TYPE	FUNCTION	DEFAULT
15	R	Reset Status	0
14		Reserved	0
13		Reserved	0
12		Reserved	0
11		Reserved	0
10		Reserved	0
9		Reserved	0
8		Reserved	0
7		Reserved	0
6		Reserved	0
5		Reserved	0
4		Reserved	0
3		Reserved	0
2		Reserved	0
1		Reserved	0
0		Reserved	0

Access to this CRI register is via the external microprocessor interface only.

Reset_Status:

When set, the Reset_Status bit indicates that the PM3390 has not finished its 1024 SYSCLK initialization period.

CMIF Register 0x0B DLT CRI Status Register

BIT	TYPE	FUNCTION	DEFAULT
15	R	DLT Write Done	1
14	R	DLT Read Done	1
13		Reserved	0
12		Reserved	0
11		Reserved	0
10		Reserved	0
9		Reserved	0
8		Reserved	0
7		Reserved	0
6		Reserved	0
5		Reserved	0
4		Reserved	0
3	R	DLT CRI Read Data bit 3	0
2	R	DLT CRI Read Data bit 2	0
1	R	DLT CRI Read Data bit 1	0
0	R	DLT CRI Read Data bit 0	0

Access to this CRI register is via the external microprocessor interface only.

DLT Write Done:

When set, this bit indicates that the last CRI Write operation to the DLT register has completed.

A write to the DLT CRI Write Register (0x000D) clears this bit; when the DLT write operation has completed, this bit becomes set.

DLT Read Done:

When set, this bit indicates that the last CRI Read operation to the DLT register has completed.

A write to the DLT CRI Control Register (0x000C) with bit[15] a “1” (indicating a Read operation), clears this bit; when the read data in this register is valid, this bit becomes set.

DLT CRI Read Data[3:0]:

Bits [3:0] contain the read data from **the last DLT table access of any type** that was initiated (which could have been over the external microprocessor interface or via an EXACT bus Control Register Read/Write to the DLT table).

In order to guarantee that an access to this register over the external microprocessor interface returns the valid DLT read data:
(1) follow the sequence of steps described in the section “DLT Read

Operation via the External Microprocessor Interface).
(2) guarantee that there is not any intervening EXACT bus Control Register Read/Write to the DLT table while following the above sequence.

For easy reference, the format of a DLT table entry is shown:

Bit 3	Bit 2	Bit 1	Bit 0
DISCARD	TX_PortNum		

CMIF Register 0x0C DLT CRI Control Register

BIT	TYPE	FUNCTION	DEFAULT
15	R/W	DLT Read/Write Select	0
14		Reserved	0
13		Reserved	0
12		Reserved	0
11		Reserved	0
10		Reserved	0
9	R/W	DLT CRI Address bit 9	0
8	R/W	DLT CRI Address bit 8	0
7	R/W	DLT CRI Address bit 7	0
6	R/W	DLT CRI Address bit 6	0
5	R/W	DLT CRI Address bit 5	0
4	R/W	DLT CRI Address bit 4	0
3	R/W	DLT CRI Address bit 3	0
2	R/W	DLT CRI Address bit 2	0
1	R/W	DLT CRI Address bit 1	0
0	R/W	DLT CRI Address bit 0	0

Access to this CRI register is via the external microprocessor interface only.

A write to this register via the external microprocessor interface will always initiate an access to the DLT table.

DLT Read/Write Select:

To initiate a DLT **write** through the external microprocessor interface, this bit is written to the value of 0. To initiate a DLT **read** through the external microprocessor interface, this bit is written to the value of 1.

This bit always returns a 0 when read.

DLT CRI Address[9:0]:

The DLT CRI Address[9:0] is the address used for the access initiated when the external microprocessor writes this CRI register.

CMIF Register 0x0D DLT CRI Write Register

BIT	TYPE	FUNCTION	DEFAULT
15		Reserved	0
14		Reserved	0
13		Reserved	0
12		Reserved	0
11		Reserved	0
10		Reserved	0
9		Reserved	0
8		Reserved	0
7		Reserved	0
6		Reserved	0
5		Reserved	0
4		Reserved	0
3	R/W	DLT Pending Write Data bit 3	0
2	R/W	DLT Pending Write Data bit 2	0
1	R/W	DLT Pending Write Data bit 1	0
0	R/W	DLT Pending Write Data bit 0	0

Access to this CRI register is via the external microprocessor interface only.

For easy reference, the format of a DLT table entry is shown:

Bit 3	Bit 2	Bit 1	Bit 0
DISCARD	TX_PortNum		

DLT CRI Pending Write Data[3:0]:

DLT CRI Pending Write Data[3:0] is the data that is written to the DLT table when the external microprocessor interface **subsequently** writes the DLT CRI Control register (0x000C).

Refer to section titled “DLT Write Operation via the External Microprocessor Interface” for additional details.

CMIF Register 0x10 Message Box Command Register

BIT	TYPE	FUNCTION	DEFAULT
15	W	Message Box Reset	0
14	R/W	Transmit Loop Mode	0
13	R/W	Broadcast Enable	0
12		Reserved	0
11		Reserved	0
10	R/W	TX PortNum bit 2	0
9	R/W	TX PortNum bit 1	0
8	R/W	TX PortNum bit 0	0
7		Reserved	0
6		Reserved	0
5		Reserved	0
4	R/W	Message Count bit 4	0
3	R/W	Message Count bit 3	0
2	R/W	Message Count bit 2	0
1	R/W	Message Count bit 1	0
0	R/W	Message Count bit 0	0

Access to this CRI register is via the external microprocessor interface only.

This register is used to start transmission of messages formatted in the Message Box Transmit Buffer.

Message Box Reset:

Writing a 1 to this bit causes a message box reset. **This bit is self-clearing and will always be read as a 0**.

Transmit Loop Mode:

When set, the Message Box Transmit buffer will continuously transmit. The message box will continue to follow the transmit buffer pointer structure and transmit messages until this bit is cleared by writing it to 0.

Broadcast Enable:

Transmit the messages in the transmit buffer as broadcast messages. The messages will be sent to all TX Ports with broadcast enabled (see register 0x0005). This command parameter is overridden for messages with bit 30 set in the message header.

TX PortNum[2:0]:

Transmit the messages on the TX Port indicated by this value. This command parameter is overridden for messages with bit 30 set in the

message header.

MessageCount[4:0]:

Follow the transmit buffer pointers, starting at transmit buffer location 0, and transmit the number of messages indicated by MessageCount[4:0].

The internal logic state machine will start processing the data in the Message Box TX Buffer when this value is written to a non-zero value. As each message is transmit from the Message Box TX Buffer, the MessageCount is decremented.

CMIF Register 0x11 Message Box Status Register

BIT	TYPE	FUNCTION	DEFAULT
15	R	Transmit Loop Mode Enabled	0
14		Reserved	0
13		Reserved	0
12		Reserved	0
11		Reserved	0
10		Reserved	0
9		Reserved	0
8		Reserved	0
7	R	MBOX Receive Buffer Overrun	0
6		Reserved	0
5		Reserved	0
4		Reserved	0
3		Reserved	0
2		Reserved	0
1		Reserved	0
0		Reserved	0

Access to this CRI register is via the external microprocessor interface only.

Transmit Loop Mode Enabled:

If set, indicates that the Message Box Transmit Loop Mode is enabled.

MBOX Receive Buffer Overrun:

If set, indicates that the Message Box Receive Buffer became full while receiving a message, resulting in lost data (the message in the buffer was truncated when the full condition occurred). This bit, once set, is cleared by resetting the Message Box (see Message Box Command Register).

CMIF Register 0x12 Message Box Receive Buffer Pointer

BIT	TYPE	FUNCTION	DEFAULT
15		Reserved	
14		Reserved	
13		Reserved	
12		Reserved	
11		Reserved	
10		Reserved	
9		Reserved	
8		Reserved	
7		Reserved	
6	R	MBOX Receive Buffer Pointer bit 6	0
5	R	MBOX Receive Buffer Pointer bit 5	0
4	R	MBOX Receive Buffer Pointer bit 4	0
3	R	MBOX Receive Buffer Pointer bit 3	0
2	R	MBOX Receive Buffer Pointer bit 2	0
1	R	MBOX Receive Buffer Pointer bit 1	0
0	R	MBOX Receive Buffer Pointer bit 0	0

Access to this CRI register is via the external microprocessor interface only.

MBOX Receive Buffer Pointer[6:0]:

This is the indicate the location in the Receive Buffer of the last message received by the Message Box. This is the buffer element address for the *status header* of the last message written to the Message Box Receive Buffer (the pointer is to the lower 16 bits of the 32 bit *status header*).

Given that the base address of the Message Box Receive Buffer is at 0x0400, the *status header* word of the last message can be read by the external microprocessor by doing a CRI register read to the address (0x0400 + MBOX_Receive_Buffer_Pointer[6:0]).

This register is reset to 0 when the Message Box is reset.

CMIF Register 0x13 Message Box Receive Counter

BIT	TYPE	FUNCTION	DEFAULT
15	R	MBOX Receive Counter bit 15	0
14	R	MBOX Receive Counter bit 14	0
13	R	MBOX Receive Counter bit 13	0
12	R	MBOX Receive Counter bit 12	0
11	R	MBOX Receive Counter bit 11	0
10	R	MBOX Receive Counter bit 10	0
9	R	MBOX Receive Counter bit 9	0
8	R	MBOX Receive Counter bit 8	0
7	R	MBOX Receive Counter bit 7	0
6	R	MBOX Receive Counter bit 6	0
5	R	MBOX Receive Counter bit 5	0
4	R	MBOX Receive Counter bit 4	0
3	R	MBOX Receive Counter bit 3	0
2	R	MBOX Receive Counter bit 2	0
1	R	MBOX Receive Counter bit 1	0
0	R	MBOX Receive Counter bit 0	0

Access to this CRI register is via the external microprocessor interface only.

MBOX Receive Counter[15:0]:

This counter maintains the total number of messages received to the message box, including messages prematurely terminated by error words. The counter continues to increment after the receive buffer becomes full. This counter will roll-over from 0xFFFF to 0x0000.

This counter is reset to 0 when the Message Box is reset (see register 0x0010 description).

CMIF Register 0x14 Message Box Error Counter

BIT	TYPE	FUNCTION	DEFAULT
15		Reserved	0
14		Reserved	0
13		Reserved	0
12		Reserved	0
11		Reserved	0
10		Reserved	0
9		Reserved	0
8		Reserved	0
7	R	MBOX Error Counter bit 7	0
6	R	MBOX Error Counter bit 6	0
5	R	MBOX Error Counter bit 5	0
4	R	MBOX Error Counter bit 4	0
3	R	MBOX Error Counter bit 3	0
2	R	MBOX Error Counter bit 2	0
1	R	MBOX Error Counter bit 1	0
0	R	MBOX Error Counter bit 0	0

Access to this CRI register is via the external microprocessor interface only.

MBOX Error Counter[7:0]:

The total number of error messages received by the Message Box. This counter will saturate at 0xFF.

This counter is reset to 0 when the Message Box is reset (see register 0x0010 description).

CMIF Register 0x0400-0x047F Message Box Receive Buffer

BIT	TYPE	FUNCTION	DEFAULT
15	R	MBOX RxBufferData bit 15	0
14	R	MBOX RxBufferData bit 14	0
13	R	MBOX RxBufferData bit 13	0
12	R	MBOX RxBufferData bit 12	0
11	R	MBOX RxBufferData bit 11	0
10	R	MBOX RxBufferData bit 10	0
9	R	MBOX RxBufferData bit 9	0
8	R	MBOX RxBufferData bit 8	0
7	R	MBOX RxBufferData bit 7	0
6	R	MBOX RxBufferData bit 6	0
5	R	MBOX RxBufferData bit 5	0
4	R	MBOX RxBufferData bit 4	0
3	R	MBOX RxBufferData bit 3	0
2	R	MBOX RxBufferData bit 2	0
1	R	MBOX RxBufferData bit 1	0
0	R	MBOX RxBufferData bit 0	0

Access to this CRI register is via the external microprocessor interface only.

MBOX RxBufferData[15:0]:

A read by the external microprocessor to the CRI register space noted above will return the two-bytes of data at that address in the Message Box Receive Buffer.

CMIF Register 0x0480-0x04FF Message Box Transmit Buffer

BIT	TYPE	FUNCTION	DEFAULT
15	W	MBOX TxBufferData bit 15	0
14	W	MBOX TxBufferData bit 14	0
13	W	MBOX TxBufferData bit 13	0
12	W	MBOX TxBufferData bit 12	0
11	W	MBOX TxBufferData bit 11	0
10	W	MBOX TxBufferData bit 10	0
9	W	MBOX TxBufferData bit 9	0
8	W	MBOX TxBufferData bit 8	0
7	W	MBOX TxBufferData bit 7	0
6	W	MBOX TxBufferData bit 6	0
5	W	MBOX TxBufferData bit 5	0
4	W	MBOX TxBufferData bit 4	0
3	W	MBOX TxBufferData bit 3	0
2	W	MBOX TxBufferData bit 2	0
1	W	MBOX TxBufferData bit 1	0
0	W	MBOX TxBufferData bit 0	0

Access to this CRI register is via the external microprocessor interface only.

MBOX TxBufferData[15:0]:

A write by the external microprocessor to the CRI register space noted above will write the two bytes of data that is on the microprocessor MI_AD[15:0] bus to the address in the Message Box Transmit Buffer given by the offset from the base address of 0x0480. This CRI register space is write-only; reading from it will return an indeterminate value, but there will be no adverse affects on the PM3390.

As noted in the functional description of the Message Box Transmit Buffer, the addressing is in terms of two-byte elements. The *first* two bytes of data in the 256 byte Message Box Transmit buffer is at CRI address 0x0480; the *last* two bytes of data have a CRI address of 0x04FF.

10.4 EXACT Bus Destination Lookup Table Registers

DLT Registers 0x1000-0x13FF: Destination Lookup Table

BIT	TYPE	FUNCTION	DEFAULT
15		Reserved	0
14		Reserved	0
13		Reserved	0
12		Reserved	0
11		Reserved	0
10		Reserved	0
9		Reserved	0
8		Reserved	0
7		Reserved	0
6		Reserved	0
5		Reserved	0
4		Reserved	0
3	R/W	DISCARD	1
2	R/W	TXPortNum bit 2	0
1	R/W	TXPortNum bit 1	0
0	R/W	TXPortNum bit 0	0

Access to this CRI register is via EXACT Control Register Read or Write messages.

The CRI address space is mapped, at an offset of 0x1000, directly to the DLT address table. For example, to read the DLT entry having a DestinationAddress[9:0] of 0x317, the downstream EXACT device would issue an EXACT Control Register Read having an Address[23:0] of 0x001317 to the PM3390 device. The PM3390 will process the EXACT Control Register Read and return an EXACT Status Register message with the address field of 0x001317 and a data word of {DATA3[7:0], DATA2[7:0], DATA1[7:0], DATA0[7:0]} of the format {28'h0, DISCARD, TXPortNum[2:0]} (that is, only the low 4 bits will be non-zero).

The Destination Lookup Table (DLT) is used to map each incoming EXACT unicast message to a single PM3390 TX port (where a unicast message does not have an address equal to either the DeviceAddress[9:0] of this PM3390 device or the BroadcastAddress[9:0]). Since the mechanism that accesses the DLT over the EXACT bus uses CRI accesses, it follows that only one DLT address can be written or read per EXACT Control Register access.

For easy reference, the format of a DLT table entry is shown:

Bit 3	Bit 2	Bit 1	Bit 0
DISCARD	TX_PortNum		

DISCARD:

DISCARD bit in DLT table entry.

TX_PortNum[2:0]:

TX_PortNum[2:0] bit field in DLT table entry.

10.5 Free List Manager Registers

FLM Register 0x00 XOB High Threshold

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	-	Reserved	0
Bit 14	-	Reserved	0
Bit 13	-	Reserved	0
Bit 12	-	Reserved	0
Bit 11	-	Reserved	0
Bit 10	-	Reserved	0
Bit 9	R/W	Reserved	0
Bit 8	R/W	Reserved	0
Bit 7	R/W	Reserved	**1**
Bit 6	R/W	Reserved	**1**
Bit 5	R/W	Reserved	0
Bit 4	R/W	Reserved	0
Bit 3	R/W	Reserved	0
Bit 2	R/W	Reserved	0
Bit 1	R/W	Reserved	0
Bit 0	R/W	Reserved	0

This register is reserved and should not be modified.

.

FLM Register 0x01 XOB Low Threshold

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	-	Reserved	0
Bit 14	-	Reserved	0
Bit 13	-	Reserved	0
Bit 12	-	Reserved	0
Bit 11	-	Reserved	0
Bit 10	-	Reserved	0
Bit 9	R/W	Reserved	0
Bit 8	R/W	Reserved	0
Bit 7	R/W	Reserved	0
Bit 6	R/W	Reserved	0
Bit 5	R/W	Reserved	0
Bit 4	R/W	Reserved	0
Bit 3	R/W	Reserved	1
Bit 2	R/W	Reserved	0
Bit 1	R/W	Reserved	0
Bit 0	R/W	Reserved	1

This register is reserved and should not be modified.

.

FLM Register 0x02 XOB Busy Threshold

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	-	Reserved	0
Bit 14	-	Reserved	0
Bit 13	-	Reserved	0
Bit 12	-	Reserved	0
Bit 11	-	Reserved	0
Bit 10	-	Reserved	0
Bit 9	R/W	XOB_BUSY_THRESH[9]	0
Bit 8	R/W	XOB_BUSY_THRESH[8]	0
Bit 7	R/W	XOB_BUSY_THRESH[7]	0
Bit 6	R/W	XOB_BUSY_THRESH[6]	0
Bit 5	R/W	XOB_BUSY_THRESH[5]	0
Bit 4	R/W	XOB_BUSY_THRESH[4]	0
Bit 3	R/W	XOB_BUSY_THRESH[3]	**1**
Bit 2	R/W	XOB_BUSY_THRESH[2]	**1**
Bit 1	R/W	XOB_BUSY_THRESH[1]	**1**
Bit 0	R/W	XOB_BUSY_THRESH[0]	**1**

XOB_BUSY_THRESH[9:0]:

The FLM maintains a count of the CCS addresses in use by the CLM. When this count meets or exceeds XOB_BUSY_THRESH[9:0], the XOB will output a code of BUSY as opposed to IDLE on the EXACT Bus to indicate the start of a message and the status of the ring. This applies on a per TX-port basis (0-7), regardless as to the clock mode of that port. The BUSY character and its usage are discussed in the EXACT Bus Protocol Specification (PMC-970215): refer to document section “Physical Layer Coding and Flow Control” for additional information.

See also the functional description earlier in this document of “Global CCS Storage Threshold to prevent overflow”.

FLM Register 0x03 XIB Discard Threshold

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	-	Reserved	0
Bit 14	-	Reserved	0
Bit 13	-	Reserved	0
Bit 12	-	Reserved	0
Bit 11	-	Reserved	0
Bit 10	-	Reserved	0
Bit 9	R/W	XIB _THRESH[9]	**1**
Bit 8	R/W	XIB_THRESH[8]	**0**
Bit 7	R/W	XIB_THRESH[7]	**1**
Bit 6	R/W	XIB_THRESH[6]	**1**
Bit 5	R/W	XIB_THRESH[5]	**1**
Bit 4	R/W	XIB_THRESH[4]	**1**
Bit 3	R/W	XIB_THRESH[3]	**1**
Bit 2	R/W	XIB_THRESH[2]	**0**
Bit 1	R/W	XIB_THRESH[1]	**0**
Bit 0	R/W	XIB_THRESH[0]	**0**

XIB_THRESH[9:0]:

The FLM maintains a count of the CCS addresses in use by the CLM. When this count meets or exceeds XIB_THRESH[9:0], each of the eight XIB's will discard all incoming EXACT Bus messages. The XIB's will resume receiving EXACT Bus messages when the FLM CCS address count decrements below XIB_THRESH[9:0]. The XIB's logically treat the discard due to FLM count exceeding XIB_THRESH in the same way as if the message was received and had a DLT entry of `DISCARD'.

The INT pin on the PM3390 will be asserted if FLM count exceeds XIB_THRESH[9:0]: see CMIF Device Status register description for the interrupt code corresponding to this condition.

FLM Register 0x04 Watermark

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	-	Reserved	0
Bit 14	-	Reserved	0
Bit 13	-	Reserved	0
Bit 12	-	Reserved	0
Bit 11	-	Reserved	0
Bit 10	-	Reserved	0
Bit 9	R	WATERMARK[9]	0
Bit 8	R	WATERMARK[8]	0
Bit 7	R	WATERMARK[7]	0
Bit 6	R	WATERMARK[6]	0
Bit 5	R	WATERMARK[5]	0
Bit 4	R	WATERMARK[4]	0
Bit 3	R	WATERMARK[3]	0
Bit 2	R	WATERMARK[2]	0
Bit 1	R	WATERMARK[1]	0
Bit 0	R	WATERMARK[0]	0

WATERMARK[9:0]

The WATERMARK register contains the highest level of CCS usage since the last read. Note that the WATERMARK bits are cleared on read.

FLM Register 0x05 Discard Count

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	-	Reserved	0
Bit 14	-	Reserved	0
Bit 13	-	Reserved	0
Bit 12	-	Reserved	0
Bit 11	-	Reserved	0
Bit 10	-	Reserved	0
Bit 9	-	Reserved	0
Bit 8	-	Reserved	0
Bit 7	R	DISCARD_COUNT[7]	0
Bit 6	R	DISCARD_COUNT[6]	0
Bit 5	R	DISCARD_COUNT[5]	0
Bit 4	R	DISCARD_COUNT[4]	0
Bit 3	R	DISCARD_COUNT[3]	0
Bit 2	R	DISCARD_COUNT[2]	0
Bit 1	R	DISCARD_COUNT[1]	0
Bit 0	R	DISCARD_COUNT[0]	0

DISCARD_COUNT[7:0]

The DISCARD_COUNT maintains a count of the number of times that the FLM Central Cell Store address count crosses the XIB_THRESH value. Note that the DISCARD_COUNT bits are cleared on read

The DISCARD_COUNT saturates at 0xFF and will not roll-over

FLM Register 0x06 Empty

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	-	Reserved	0
Bit 14	-	Reserved	0
Bit 13	-	Reserved	0
Bit 12	-	Reserved	0
Bit 11	-	Reserved	0
Bit 10	-	Reserved	0
Bit 9	-	Reserved	0
Bit 8	-	Reserved	0
Bit 7	-	Reserved	0
Bit 6	-	Reserved	0
Bit 5	-	Reserved	0
Bit 4	-	Reserved	0
Bit 3	-	Reserved	0
Bit 2	-	Reserved	0
Bit 1	-	Reserved	0
Bit 0	R	EMPTY	0

EMPTY

This bit is latched high if the CCS address space overflows (i.e., if there are no available CCS addresses in the 'free list'). Note that the EMPTY bit is cleared on read.

The assertion of this bit is catastrophic. If the CCS address space overflows, the state of the FLM and CCS becomes corrupted and cannot recover. If the EXACT Bus Protocol specification is followed, this condition will not occur.

The FLM register 0x03, XIB Discard Threshold, was implemented to prevent the EMPTY condition from occurring within the FLM. Cells will be discarded at the receiver instead of resulting in an EMPTY condition.

FLM Register 0x07 Sequential FLM Address Select

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	-	Reserved	0
Bit 14	-	Reserved	0
Bit 13	-	Reserved	0
Bit 12	-	Reserved	0
Bit 11	-	Reserved	0
Bit 10	-	Reserved	0
Bit 9	-	Reserved	0
Bit 8	-	Reserved	0
Bit 7	-	Reserved	0
Bit 6	-	Reserved	0
Bit 5	-	Reserved	0
Bit 4	-	Reserved	0
Bit 3	-	Reserved	0
Bit 2	-	Reserved	0
Bit 1	-	Reserved	0
Bit 0	R/W	Reserved	0

This register is reserved and should not be modified.

10.6 Cell List Manager Registers

CLM Register 0x00 – 0x07 Output Cell Counts 0-7

BIT	TYPE	FUNCTION	DEFAULT
Bit 15		Reserved	
Bit 14		Reserved	
Bit 13		Reserved	
Bit 12		Reserved	
Bit 11		Reserved	
Bit 10		Reserved	
Bit 9	R	CCNT[9]	0
Bit 8	R	CCNT[8]	0
Bit 7	R	CCNT[7]	0
Bit 6	R	CCNT[6]	0
Bit 5	R	CCNT[5]	0
Bit 4	R	CCNT[4]	0
Bit 3	R	CCNT[3]	0
Bit 2	R	CCNT[2]	0
Bit 1	R	CCNT[1]	0
Bit 0	R	CCNT[0]	0

CCNT[9:0]:

The cell counts represent the number of cells that have been delivered by an input, but not sent to an output, for each output. They include cells in the output list, the cut-through buffers, and the input list generation logic.

Note: CCNT is maintained on a per-port basis, with CLM register 0x00 applicable to port 0 and register 0x07 to port 7.

CLM Register 0x10 Cut-through Depth

BIT	TYPE	FUNCTION	DEFAULT
Bit 15		Reserved	0
Bit 14		Reserved	0
Bit 13		Reserved	0
Bit 12		Reserved	0
Bit 11		Reserved	0
Bit 10		Reserved	0
Bit 9		Reserved	0
Bit 8		Reserved	0
Bit 7	R/W	CWAIT3[7]	0
Bit 6	R/W	CWAIT3[6]	0
Bit 5	R/W	CWAIT3[5]	0
Bit 4	R/W	CWAIT3[4]	0
Bit 3	R/W	CWAIT3[3]	0
Bit 2	R/W	CWAIT3[2]	0
Bit 1	R/W	CWAIT3[1]	0
Bit 0	R/W	CWAIT3[0]	0

CWAIT3[7:0]:

If CWAIT3[N] is set, cut-through data will be sent to port N only after 3 cells of a message have been received. If the bit is clear, cut-through waits for only 2 cells.

In applications other than EXACT Bus-based Ethernet switching, the CWAIT3 bit should be set (refer to Cut Through Operation description for additional details).

CLM Register 0x11 Look-Ahead Disable

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R/W	Reserved	0
Bit 14	R/W	Reserved	0
Bit 13	R/W	Reserved	0
Bit 12	R/W	Reserved	0
Bit 11	R/W	Reserved	0
Bit 10	R/W	Reserved	0
Bit 9	R/W	Reserved	0
Bit 8	R/W	Reserved	0
Bit 7	R/W	Reserved	0
Bit 6	R/W	Reserved	0
Bit 5	R/W	Reserved	0
Bit 4	R/W	Reserved	0
Bit 3	R/W	Reserved	0
Bit 2	R/W	Reserved	0
Bit 1	R/W	Reserved	0
Bit 0	R/W	Reserved	0

This register is reserved and should not be modified.

CLM Register 0x12 Cut-Through Disable

BIT	TYPE	FUNCTION	DEFAULT
Bit 15		Reserved	0
Bit 14		Reserved	0
Bit 13		Reserved	0
Bit 12		Reserved	0
Bit 11		Reserved	0
Bit 10		Reserved	0
Bit 9		Reserved	0
Bit 8		Reserved	0
Bit 7	R/W	Reserved	0
Bit 6	R/W	Reserved	0
Bit 5	R/W	Reserved	0
Bit 4	R/W	Reserved	0
Bit 3	R/W	Reserved	0
Bit 2	R/W	Reserved	0
Bit 1	R/W	Reserved	0
Bit 0	R/W	Reserved	0

This register is reserved and should not be modified.

CLM Register 0x13 Immediate Flush Threshold

BIT	TYPE	FUNCTION	DEFAULT
Bit 15		Reserved	0
Bit 14		Reserved	0
Bit 13		Reserved	0
Bit 12		Reserved	0
Bit 11		Reserved	0
Bit 10		Reserved	0
Bit 9	R/W	Reserved	0
Bit 8	R/W	Reserved	0
Bit 7	R/W	Reserved	0
Bit 6	R/W	Reserved	0
Bit 5	R/W	Reserved	0
Bit 4	R/W	Reserved	0
Bit 3	R/W	Reserved	1
Bit 2	R/W	Reserved	0
Bit 1	R/W	Reserved	0
Bit 0	R/W	Reserved	0

This register is reserved and should not be modified.

CLM Register 0x14: Flow Control Threshold

BIT	TYPE	FUNCTION	DEFAULT
Bit 15		Reserved	0
Bit 14		Reserved	0
Bit 13		Reserved	0
Bit 12		Reserved	0
Bit 11		Reserved	0
Bit 10		Reserved	0
Bit 9	R/W	FCDEPTH[9]	0
Bit 8	R/W	FCDEPTH[8]	0
Bit 7	R/W	FCDEPTH[7]	0
Bit 6	R/W	FCDEPTH[6]	0
Bit 5	R/W	FCDEPTH[5]	1
Bit 4	R/W	FCDEPTH[4]	1
Bit 3	R/W	FCDEPTH[3]	0
Bit 2	R/W	FCDEPTH[2]	0
Bit 1	R/W	FCDEPTH[1]	0
Bit 0	R/W	FCDEPTH[0]	0

FCDEPTH[9:0]:

If this number or more cells are queued for an output, the flow control line is asserted for that output.

If this number or more of cells are queued for output port N, the OFLOW[N] signal is asserted. This signal is output on the NFLOW[3:0] pins of the PM3390 device: see functional description for "per port status to provide flow control or non-Exact systems".

For Exact-based Ethernet switch systems, this register should not be modified.

For other applications, this register can be used to provide status information on the number of cells queued for a given output port. This information could be used by the device at the traffic source (i.e. the device attached to RX port N on the PM3390) to throttle the rate at which traffic is being sent to the RX ports on the PM3390 in order to prevent the Central Cell store capacity from being reached. Once the Central Store capacity is reached, further cells are discarded at the input receiver until the cell storage is re-allocated upon transmit of cells on the output queues.

CLM Register 0x15: Expansion Packet Release Threshold

BIT	TYPE	FUNCTION	DEFAULT
Bit 15		Reserved	
Bit 14		Reserved	
Bit 13		Reserved	
Bit 12		Reserved	
Bit 11		Reserved	
Bit 10		Reserved	
Bit 9	R/W	Reserved	0
Bit 8	R/W	Reserved	0
Bit 7	R/W	Reserved	0
Bit 6	R/W	Reserved	0
Bit 5	R/W	Reserved	0
Bit 4	R/W	Reserved	0
Bit 3	R/W	Reserved	0
Bit 2	R/W	Reserved	1
Bit 1	R/W	Reserved	1
Bit 0	R/W	Reserved	0

This register is reserved and should not be modified.

CLM Register 0x16: Expansion Control Mode

BIT	TYPE	FUNCTION	DEFAULT
Bit 15		Reserved	0
Bit 14		Reserved	0
Bit 13		Reserved	0
Bit 12		Reserved	0
Bit 11		Reserved	0
Bit 10		Reserved	0
Bit 9		Reserved	0
Bit 8		Reserved	0
Bit 7		Reserved	0
Bit 6		Reserved	0
Bit 5		Reserved	0
Bit 4		Reserved	0
Bit 3		Reserved	0
Bit 2	R/W	Reserved	0
Bit 1	R/W	Reserved	0
Bit 0	R/W	Reserved	1

This register is reserved and should not be modified.

10.7 Exact Input Buffer (XIB) Registers

The following notes apply to CRI accesses to the XIB registers. There are eight XIB's on each PM3390 (one per EXACT bus receive port); each XIB implements the same set of registers.

10.7.1.1 XIB counter registers

This includes the following eight CRI registers which, in pairs, implement four logical 32-bit counters:

XIB CRI register pair	Register Name
0x00, 0x01	HeaderErrors
0x02, 0x03	MessageErrors
0x04, 0x05	BytesReceived
0x06, 0x07	MessagesReceived

The LSB counter register must be read first. This saves the MSB counter register contents in a temporary register that is accessed by a read to the MSB counter register. In order to read the MSB counter register, the LSB and then the MSB register pairs must be read in order.

A read to the LSB counter register will synchronously clear the counter register.

Example 1: to read XIB port 0 MessageErrors counter (MSB: 0x03; LSB: 0x02) and XIB port 7 MessagesReceived counter (MSB: 0x07; LSB: 0x06)

1. read 0x8002.
   [Internally on the PM3390, the contents of 0x8003 are saved in the temporary register for XIB port 0; the entire 32-bit counter is cleared on read after saving the MSB].

2. read 0x8003

3. read 0xB806
   [Internally, contents of 0xB807 saved in the temporary register for XIB

port 0; the entire 32-bit counter is cleared on read after saving the MSB].

4. read 0xB807

10.7.1.2 Read of unimplemented XIB registers

A read to an unimplemented register in an XIB will return the TIMEOUT register contents.

XIB Register 0x00: Header Errors LSB

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R	HEADERERRORS[15]	0
Bit 14	R	HEADERERRORS[14]	0
Bit 13	R	HEADERERRORS[13]	0
Bit 12	R	HEADERERRORS[12]	0
Bit 11	R	HEADERERRORS[11]	0
Bit 10	R	HEADERERRORS[10]	0
Bit 9	R	HEADERERRORS[9]	0
Bit 8	R	HEADERERRORS[8]	0
Bit 7	R	HEADERERRORS[7]	0
Bit 6	R	HEADERERRORS[6]	0
Bit 5	R	HEADERERRORS[5]	0
Bit 4	R	HEADERERRORS[4]	0
Bit 3	R	HEADERERRORS[3]	0
Bit 2	R	HEADERERRORS[2]	0
Bit 1	R	HEADERERRORS[1]	0
Bit 0	R	HEADERERRORS[0]	0

HEADERERRORS[15:0]

This is the lower 16 bits of the XIB's Header Errors counter.

XIB Register 0x01: Header Errors MSB

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R	HEADERERRORS[31]	0
Bit 14	R	HEADERERRORS[30]	0
Bit 13	R	HEADERERRORS[29]	0
Bit 12	R	HEADERERRORS[28]	0
Bit 11	R	HEADERERRORS[27]	0
Bit 10	R	HEADERERRORS[26]	0
Bit 9	R	HEADERERRORS[25]	0
Bit 8	R	HEADERERRORS[24]	0
Bit 7	R	HEADERERRORS[23]	0
Bit 6	R	HEADERERRORS[22]	0
Bit 5	R	HEADERERRORS[21]	0
Bit 4	R	HEADERERRORS[20]	0
Bit 3	R	HEADERERRORS[19]	0
Bit 2	R	HEADERERRORS[18]	0
Bit 1	R	HEADERERRORS[17]	0
Bit 0	R	HEADERERRORS[16]	0

HEADERERRORS[31:16]:

This is the upper 16 bits of the XIB's Header Errors counter.

XIB Register 0x02: Message Errors LSB

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R	MESSAGEERRORS[15]	0
Bit 14	R	MESSAGEERRORS[14]	0
Bit 13	R	MESSAGEERRORS[13]	0
Bit 12	R	MESSAGEERRORS[12]	0
Bit 11	R	MESSAGEERRORS[11]	0
Bit 10	R	MESSAGEERRORS[10]	0
Bit 9	R	MESSAGEERRORS[9]	0
Bit 8	R	MESSAGEERRORS[8]	0
Bit 7	R	MESSAGEERRORS[7]	0
Bit 6	R	MESSAGEERRORS[6]	0
Bit 5	R	MESSAGEERRORS[5]	0
Bit 4	R	MESSAGEERRORS[4]	0
Bit 3	R	MESSAGEERRORS[3]	0
Bit 2	R	MESSAGEERRORS[2]	0
Bit 1	R	MESSAGEERRORS[1]	0
Bit 0	R	MESSAGEERRORS[0]	0

MESSAGEERRORS[15:0]:

This is the lower 16 bits of the XIB's Message Errors counter.

XIB Register 0x03: Message Errors MSB

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R	MESSAGEERRORS[31]	0
Bit 14	R	MESSAGEERRORS[30]	0
Bit 13	R	MESSAGEERRORS[29]	0
Bit 12	R	MESSAGEERRORS[28]	0
Bit 11	R	MESSAGEERRORS[27]	0
Bit 10	R	MESSAGEERRORS[26]	0
Bit 9	R	MESSAGEERRORS[25]	0
Bit 8	R	MESSAGEERRORS[24]	0
Bit 7	R	MESSAGEERRORS[23]	0
Bit 6	R	MESSAGEERRORS[22]	0
Bit 5	R	MESSAGEERRORS[21]	0
Bit 4	R	MESSAGEERRORS[20]	0
Bit 3	R	MESSAGEERRORS[19]	0
Bit 2	R	MESSAGEERRORS[18]	0
Bit 1	R	MESSAGEERRORS[17]	0
Bit 0	R	MESSAGEERRORS[16]	0

MESSAGEERRORS[31:16]

This is the upperr 16 bits of the XIB's Message Errors counter.

XIB Register 0x04: Bytes Received LSB

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R	BYTESRECEIVED[15]	0
Bit 14	R	BYTESRECEIVED[14]	0
Bit 13	R	BYTESRECEIVED[13]	0
Bit 12	R	BYTESRECEIVED[12]	0
Bit 11	R	BYTESRECEIVED[11]	0
Bit 10	R	BYTESRECEIVED[10]	0
Bit 9	R	BYTESRECEIVED[9]	0
Bit 8	R	BYTESRECEIVED[8]	0
Bit 7	R	BYTESRECEIVED[7]	0
Bit 6	R	BYTESRECEIVED[6]	0
Bit 5	R	BYTESRECEIVED[5]	0
Bit 4	R	BYTESRECEIVED[4]	0
Bit 3	R	BYTESRECEIVED[3]	0
Bit 2	R	BYTESRECEIVED[2]	0
Bit 1	R	BYTESRECEIVED[1]	0
Bit 0	R	BYTESRECEIVED[0]	0

BYTESRECEIVED[15:0]:

This is the lower 16 bits of the XIB's bytes received counter.

XIB Register 0x05: Bytes Received MSB

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R	Reserved	0
Bit 14	R	Reserved	0
Bit 13	R	Reserved	0
Bit 12	R	Reserved	0
Bit 11	R	Reserved	0
Bit 10	R	Reserved	0
Bit 9	R	Reserved	0
Bit 8	R	Reserved	0
Bit 7	R	Reserved	0
Bit 6	R	Reserved	0
Bit 5	R	Reserved	0
Bit 4	R	Reserved	0
Bit 3	R	Reserved	0
Bit 2	R	Reserved	0
Bit 1	R	Reserved	0
Bit 0	R	Reserved	0

This register is always read as 0.

XIB Register 0x06: Messages Received LSB

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R	MESSAGESRECEIVED[15]	0
Bit 14	R	MESSAGESRECEIVED[14]	0
Bit 13	R	MESSAGESRECEIVED[13]	0
Bit 12	R	MESSAGESRECEIVED[12]	0
Bit 11	R	MESSAGESRECEIVED[11]	0
Bit 10	R	MESSAGESRECEIVED[10]	0
Bit 9	R	MESSAGESRECEIVED[9]	0
Bit 8	R	MESSAGESRECEIVED[8]	0
Bit 7	R	MESSAGESRECEIVED[7]	0
Bit 6	R	MESSAGESRECEIVED[6]	0
Bit 5	R	MESSAGESRECEIVED[5]	0
Bit 4	R	MESSAGESRECEIVED[4]	0
Bit 3	R	MESSAGESRECEIVED[3]	0
Bit 2	R	MESSAGESRECEIVED[2]	0
Bit 1	R	MESSAGESRECEIVED[1]	0
Bit 0	R	MESSAGESRECEIVED[0]	0

MESSAGESRECEIVED[15:0]:

This is the lower 16 bits of the XIB's message received counter.

XIB Register 0x07: Messages Received MSB

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R	MESSAGESRECEIVED[31]	0
Bit 14	R	MESSAGESRECEIVED[30]	0
Bit 13	R	MESSAGESRECEIVED[29]	0
Bit 12	R	MESSAGESRECEIVED[28]	0
Bit 11	R	MESSAGESRECEIVED[27]	0
Bit 10	R	MESSAGESRECEIVED[26]	0
Bit 9	R	MESSAGESRECEIVED[25]	0
Bit 8	R	MESSAGESRECEIVED[24]	0
Bit 7	R	MESSAGESRECEIVED[23]	0
Bit 6	R	MESSAGESRECEIVED[22]	0
Bit 5	R	MESSAGESRECEIVED[21]	0
Bit 4	R	MESSAGESRECEIVED[20]	0
Bit 3	R	MESSAGESRECEIVED[19]	0
Bit 2	R	MESSAGESRECEIVED[18]	0
Bit 1	R	MESSAGESRECEIVED[17]	0
Bit 0	R	MESSAGESRECEIVED[16]	0

MESSAGESRECEIVED[31:16]:

This is the upper 16 bits of the XIB's message received counter.

XIB Register 0x08: Timeout

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R/W	TIMEOUT_ENABLE	**1**
Bit 14		Reserved	0
Bit 13		Reserved	0
Bit 12		Reserved	0
Bit 11		Reserved	0
Bit 10		Reserved	0
Bit 9		Reserved	0
Bit 8		Reserved	0
Bit 7		Reserved	0
Bit 6	R/W	TIMEOUT[6]	0
Bit 5	R/W	TIMEOUT[5]	0
Bit 4	R/W	TIMEOUT[4]	0
Bit 3	R/W	TIMEOUT[3]	**1**
Bit 2	R/W	TIMEOUT[2]	0
Bit 1	R/W	TIMEOUT[1]	0
Bit 0	R/W	TIMEOUT[0]	0

The timeout register must be kept at its default state in order to ensure proper device operation.

TIMEOUT_ENABLE

This enables the Accumulator timeout feature when set.

TIMEOUT[6:0]

This is the Accumulator timeout value.

XIB Register 0x09: Synch Count Minimum

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R/W	SYNCH_CNT_MIN[15]	0
Bit 14	R/W	SYNCH_CNT_MIN[14]	0
Bit 13	R/W	SYNCH_CNT_MIN[13]	0
Bit 12	R/W	SYNCH_CNT_MIN[12]	0
Bit 11	R/W	SYNCH_CNT_MIN[11]	0
Bit 10	R/W	SYNCH_CNT_MIN[10]	0
Bit 9	R/W	SYNCH_CNT_MIN[9]	0
Bit 8	R/W	SYNCH_CNT_MIN[8]	0
Bit 7	R/W	SYNCH_CNT_MIN[7]	0
Bit 6	R/W	SYNCH_CNT_MIN[6]	0
Bit 5	R/W	SYNCH_CNT_MIN[5]	0
Bit 4	R/W	SYNCH_CNT_MIN[4]	1
Bit 3	R/W	SYNCH_CNT_MIN[3]	0
Bit 2	R/W	SYNCH_CNT_MIN[2]	0
Bit 1	R/W	SYNCH_CNT_MIN[1]	0
Bit 0	R/W	SYNCH_CNT_MIN[0]	0

SYNCH_COUNT_MINIMUM[15:0]

This is the minimum number of EXACT message delimiters the XIB must consecutively detect before accepting data for processing.

The process used by each of the eight EXACT receive ports on the PM3390 (XIB logic block) to start processing Exact messages is as follows:

(1) after reset assertion, the internal state bit "SYNCHED" is clear.

(2) the XIB input receive block will start receiving characters after reset has been deasserted. The type of character is decoded into one of two classes: non-erred control characters and other (data or erred characters).

A 16-bit counter is maintained on the number of consecutive non-erred control characters that have been received: this count is the SYNCH_CNT register (see XIB register 0x0A). This counter increments if the internal state bit SYNCHED is not set. Once SYNCH_CNT_MIN number of consecutive non-erred control characters have been received, the XIB the internal *SYNCHED* state bit is set, thereby freezing the value of the SYNCH_CNT register. If the *SYNCHED* state bit is not set and the received character is NOT a non-erred control character, the SYNCH_CNT register is cleared to 0.

Once the *SYNCHED* state bit is set, it will be cleared only upon reset assertion of the PM3390. If the *SYNCHED* state bit is not set and the received character is NOT a non-erred control character, the SYNCH_CNT register is cleared to 0.

The non-erred control character decodes used by the XIB receiver are:

- if the clock mode for the port is Clear Channel, the characters are IDLE, BUSY, and FILL.
- if the clock mode for the port is SERDES or 8B/10B encoded, the characters are IDLE, BUSY, and FILLN, and FILLP.

For additional information on the EXACT bus character encoding/decoding, please refer to the EXACT Bus Protocol Specification.

XIB Register 0x0A: Synchronization Count

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R	SYNCH_CNT[15]	0
Bit 14	R	SYNCH_CNT[14]	0
Bit 13	R	SYNCH_CNT[13]	0
Bit 12	R	SYNCH_CNT[12]	0
Bit 11	R	SYNCH_CNT[11]	0
Bit 10	R	SYNCH_CNT[10]	0
Bit 9	R	SYNCH_CNT[9]	0
Bit 8	R	SYNCH_CNT[8]	0
Bit 7	R	SYNCH_CNT[7]	0
Bit 6	R	SYNCH_CNT[6]	0
Bit 5	R	SYNCH_CNT[5]	0
Bit 4	R	SYNCH_CNT[4]	0
Bit 3	R	SYNCH_CNT[3]	0
Bit 2	R	SYNCH_CNT[2]	0
Bit 1	R	SYNCH_CNT[1]	0
Bit 0	R	SYNCH_CNT[0]	0

SYNCH_CNT[15:0]:

This register maintains the count of consecutive EXACT Bus message delimiters up to the value given in the SYNCH_CNT_MIN[15:0] register. The first time that the SYNCH_CNT counter equals the value of the corresponding SYNCH_CNT_MIN register, the value is latched and held; the SYNCH_CNT register will then hold until the given XIB receive port is reset.

From the above descriptions of the SYNCH_CNT and SYNCH_CNT_MIN registers it can be seen that if SYNCH_CNT_MIN is left at its default value (0x0010), then the *SYNCHED* state bit will be logically true if SYNCH_CNT is equal to 0x0010. The SYNCHED state bit is not by itself directly visible as a CRI register bit.

10.8 EXACT Output Buffer Registers

The following notes apply to CRI accesses to the XOB registers. There are eight XOB's on each PM3390 (one per EXACT bus transmit port); each XOB implements the same set of registers.

10.8.1.1 XOB counter registers

This includes the following four CRI registers which, in pairs, implement two logical 32-bit counters:

XOB CRI register pair	Register Name
0x01, 0x02	TxByteCount
0x03, 0x04	TxMessageCount

The MSB counter register must be read first. This saves the LSB counter register contents in a temporary register which is accessed by a read to the LSB counter register. In order to read the LSB counter register, the MSB and then the LSB register pairs must be read in order.

The XOB counter registers are only cleared on reset. They are not write-able.

Example 1: to read XOB port 0 TxByteCount (MSB: 0x01; LSB: 0x02) to read XIB port 0 and XOB port 7 TxMessageCount counter (MSB: 0x03; LSB: 0x04)

1. read 0x8401.
   [Internally in the PM3390, the contents of register 0x8402 are saved in the temporary register maintained for all XOB ports].
2. read 0x8402
3. read 0xBC06
   [Internally in the PM3390, the contents of register 0xBC07 are saved in the temporary register maintained for all XOB ports].
4. read 0xBC07

10.8.1.2 Read of unimplemented XOB registers

The registers within the XOB register space are decoded as follows:

- the upper 6 bits of the address are decoded as the base address (as described previously in "Control Register Interface address space").
- the low three bits of the address are used to select one of the seven registers implemented per XOB.

So control register address bits [9:3] are not used in decoding a register access to a given XOB. Therefore, "aliasing" occurs within the base address space of an XOB.

A read to XOB register address 0x07, which is unimplemented, returns a 0.

Example of XOB register address aliasing:
The same MessageCount MSB register on XOB port 1 will be read from the following control register addresses because of aliasing: 0x8C03, 0x8C0B, 0x8C1B, 0x8C23, etc.

XOB Register 0x00: XOB Errors

BIT	TYPE	FUNCTION	DEFAULT
Bit 15		Reserved	
Bit 14		Reserved	
Bit 13		Reserved	
Bit 12		Reserved	
Bit 11		Reserved	
Bit 10		Reserved	
Bit 9		Reserved	
Bit 8		Reserved	
Bit 7		Reserved	
Bit 6	R/W	EXACT_ERR_CHAR	0
Bit 5	R/W	OUTPUT_UNDER	0
Bit 4	R/W	EXPIF_OVER (**XOB 0-3 only**)	0
Bit 3	R/W	LKAHD_UNDER	0
Bit 2	R/W	LKAHD_OVER	0
Bit 1	R/W	IN_FIFO_UNDER	0
Bit 0	R/W	IN_FIFO_OVER	0

IN_FIFO_OVER, LKAHD_OVER,EXPIF_OVER:

Any of these bits set means that the associated FIFO has overflowed.

IN_FIFO_UNDER, LKAHD_UNDER, OUTPUT_UNDER:

Any of these bits set means that the associated FIFO has underflowed.

NOTE: Only XOB 0 through XOB 3 have an internal Expansion Input FIFO interface (i.e. EXPIF), therefore the EXPIF_OVER error bit is valid only in the error registers for these four XOBs.

EXACT_ERR_CHAR:

This bit is set when the XOB has received one or more EXACT K30.7 ERROR characters. This ERROR character propagation is carried forward from the XIB receive interface from which the EXACT message was received.

NOTE: all bits in the XOB Errors register (0x00) are **cleared**, on a per-port basis, by writing a "1" to that bit position. For example, to clear the the EXACT_ERR_CHAR bit for XOB port 7, do a CRI register write to address 0xBC00 with data 0x0040.

XOB Register 0x01: Bytes Transmitted MSB

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R	TxBYTECOUNT[31]	0
Bit 14	R	TxBYTECOUNT[30]	0
Bit 13	R	TxBYTECOUNT[29]	0
Bit 12	R	TxBYTECOUNT[28]	0
Bit 11	R	TxBYTECOUNT[27]	0
Bit 10	R	TxBYTECOUNT[26]	0
Bit 9	R	TxBYTECOUNT[25]	0
Bit 8	R	TxBYTECOUNT[25]	0
Bit 7	R	TxBYTECOUNT[23]	0
Bit 6	R	TxBYTECOUNT[22]	0
Bit 5	R	TxBYTECOUNT[21]	0
Bit 4	R	TxBYTECOUNT[20]	0
Bit 3	R	TxBYTECOUNT[19]	0
Bit 2	R	TxBYTECOUNT[18]	0
Bit 1	R	TxBYTECOUNT[17]	0
Bit 0	R	TxBYTECOUNT[16]	0

TxBYTECOUNT[31:16]:

This is the upper 16 bits of the XOB's transmitted byte counter.

XOB Register 0x02: Bytes Transmitted LSB

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R	TxBYTECOUNT[15]	0
Bit 14	R	TxBYTECOUNT[14]	0
Bit 13	R	TxBYTECOUNT[13]	0
Bit 12	R	TxBYTECOUNT[12]	0
Bit 11	R	TxBYTECOUNT[11]	0
Bit 10	R	TxBYTECOUNT[10]	0
Bit 9	R	TxBYTECOUNT[9]	0
Bit 8	R	TxBYTECOUNT[8]	0
Bit 7	R	TxBYTECOUNT[7]	0
Bit 6	R	TxBYTECOUNT[6]	0
Bit 5	R	TxBYTECOUNT[5]	0
Bit 4	R	TxBYTECOUNT[4]	0
Bit 3	R	TxBYTECOUNT[3]	0
Bit 2	R	TxBYTECOUNT[2]	0
Bit 1	R	TxBYTECOUNT[1]	0
Bit 0	R	TxBYTECOUNT[0]	0

TxBYTECOUNT[15:0]:

This is the lower 16 bits of the XOB's transmitted byte counter.

XOB Register 0x03: Messages Transmitted MSB

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R	TxMESSAGECOUNT[31]	0
Bit 14	R	TxMESSAGECOUNT[30]	0
Bit 13	R	TxMESSAGECOUNT[29]	0
Bit 12	R	TxMESSAGECOUNT[28]	0
Bit 11	R	TxMESSAGECOUNT[27]	0
Bit 10	R	TxMESSAGECOUNT[26]	0
Bit 9	R	TxMESSAGECOUNT[25]	0
Bit 8	R	TxMESSAGECOUNT[24]	0
Bit 7	R	TxMESSAGECOUNT[23]	0
Bit 6	R	TxMESSAGECOUNT[22]	0
Bit 5	R	TxMESSAGECOUNT[21]	0
Bit 4	R	TxMESSAGECOUNT[20]	0
Bit 3	R	TxMESSAGECOUNT[19]	0
Bit 2	R	TxMESSAGECOUNT[18]	0
Bit 1	R	TxMESSAGECOUNT[17]	0
Bit 0	R	TxMESSAGECOUNT[16]	0

TxMESSAGECOUNT[31:16]:

This is the upper 16 bits of the XOB's transmitted message counter.

XOB Register 0x04: Messages Transmitted LSB

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R	TxMESSAGECOUNT[15]	0
Bit 14	R	TxMESSAGECOUNT[14]	0
Bit 13	R	TxMESSAGECOUNT[13]	0
Bit 12	R	TxMESSAGECOUNT[12]	0
Bit 11	R	TxMESSAGECOUNT[11]	0
Bit 10	R	TxMESSAGECOUNT[10]	0
Bit 9	R	TxMESSAGECOUNT[9]	0
Bit 8	R	TxMESSAGECOUNT[8]	0
Bit 7	R	TxMESSAGECOUNT[7]	0
Bit 6	R	TxMESSAGECOUNT[6]	0
Bit 5	R	TxMESSAGECOUNT[5]	0
Bit 4	R	TxMESSAGECOUNT[4]	0
Bit 3	R	TxMESSAGECOUNT[3]	0
Bit 2	R	TxMESSAGECOUNT[2]	0
Bit 1	R	TxMESSAGECOUNT[1]	0
Bit 0	R	TxMESSAGECOUNT[0]	0

TxMESSAGECOUNT[31:16]:

This is the lower 16 bits of the XOB's transmitted message counter.

XOB Register 0x05: XOB: General Purpose Software Register 0

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R/W	GP0[15]	0
Bit 14	R/W	GP0[14]	0
Bit 13	R/W	GP0[13]	0
Bit 12	R/W	GP0[12]	0
Bit 11	R/W	GP0[11]	0
Bit 10	R/W	GP0[10]	0
Bit 9	R/W	GP0[9]	0
Bit 8	R/W	GP0[8]	0
Bit 7	R/W	GP0[7]	0
Bit 6	R/W	GP0[6]	0
Bit 5	R/W	GP0[5]	0
Bit 4	R/W	GP0[4]	0
Bit 3	R/W	GP0[3]	0
Bit 2	R/W	GP0[2]	0
Bit 1	R/W	GP0[1]	0
Bit 0	R/W	GP0[0]	0

The GP0 register can be used as a general purpose software register by external firmware.

Note: the PM3390 does not use this register as long as the CMIF register 0x01 (absolute CRI address of 0x0001) bit 22 remains at its default value of 0.

XOB Register 0x06: XOB: General Purpose Software Register 1

BIT	TYPE	FUNCTION	DEFAULT
Bit 15	R/W	GP1[15]	0
Bit 14	R/W	GP1[14]	0
Bit 13	R/W	GP1[13]	0
Bit 12	R/W	GP1[12]	0
Bit 11	R/W	GP1[11]	0
Bit 10	R/W	GP1[10]	0
Bit 9	R/W	GP1[9]	0
Bit 8	R/W	GP1[8]	0
Bit 7	R/W	GP1[7]	0
Bit 6	R/W	GP1[6]	0
Bit 5	R/W	GP1[5]	0
Bit 4	R/W	GP1[4]	0
Bit 3	R/W	GP1[3]	0
Bit 2	R/W	GP1[2]	0
Bit 1	R/W	GP1[1]	0
Bit 0	R/W	GP1[0]	0

The GP1 register can be used as a general purpose software register by external firmware.

Note: the PM3390 does not use this register as long as the CMIF register 0x01 (absolute CRI address of 0x0001) bit 22 remains at its default value of 0.

11 ABSOLUTE MAXIMUM RATINGS

Maximum rating are the worst case limits that the device can withstand without sustaining permanent damage. They are not indicative of normal mode operation conditions.

Case Temperature under Bias	0C to 70C
Storage Temperature	-40C to +125C
Supply Voltage	-0.3V to 3.63V
Bias Voltage (VBIAS5)	(VDD-0.3) to +5.5V
Voltage on Any 5V tolerant Pin	-0.3V to VBIAS5+0.3V
Voltage on Any 3V tolerant Pin	-0.3V to VDD+0.3V
Static Discharge Voltage	+/- 1000V
Latch-Up Current	+/- 100 mA
DC Input Current	+/- 20 mA
Lead Temperature	+300C
Absolute Maximum Junction Temperature	+125C

RECOMMENDED OPERATING CONDITIONS

Parameter	Symbol	Value			Units
		Min	Nom	Max	
Supply Voltage	Vdd	+2.97	+3.30	+3.63	Vdc
BIAS5V Voltage	Vbias5v	+4.75	+5.0	+5.25	Vdc
Operating Ambient Temp.	Ta	0		+70	°C

12 D.C. CHARACTERISTICS

DC characteristics are specified over recommended operating conditions (T_A = 0°C to +70°C, V_{DD} = 3.3 V ±10%, V_{BIAS}= 5.0V ±5%).

Symbol	Parameter	Min	Typ	Max	Units	Conditions
V_{IL}	Input low voltage (TTL only)			0.8	Volts	
V_{IH}	Input high voltage	2.0			Volts	
V_{OL}	Output or bidirect low voltage (TTL only)			0.4	Volts	
V_{OH}	Output or bidirect high voltage (TTL only)	2.4			Volts	
I_{IL}	Input low current	-10		+10	uA	
I_{IH}	Input high current	-10		+10	uA	
C_{IN}	Input capacitance		5		pF	
C_{OUT}	Out capacitance		5		pF	
C_{IO}	Bidirectional capacitance		5		pF	
IDD_{OP}	Operating current			3.00	A	VDD=3.3V, Output unloaded; SYSCLK=67.5MHz; XREFCLK125=125MHz XREFCLK135=135MHz

Notes on D.C. Characteristics:

1. Negative currents flow into the device (sinking), positive currents flow out of the device (sourcing).

13 A.C. TIMING CHARACTERISTICS

AC characteristics are specified over recommended operating conditions (T_A = 0°C to +70°C, V_{DD} = 3.3 V ±10%, V_{BIAS}= 5.0V ±5%).

Notes on Input Timing:

1. When a set-up time is specified between an input and a clock, the set-up time is the time in nanoseconds from the 1.4 Volt point of the input to the 1.4 Volt point of the clock.
2. When a hold time is specified between an input and a clock, the hold time is the time in nanoseconds from the 1.4 Volt point of the clock to the 1.4 Volt point of the input.

Notes on Output Timing:

1. Output propagation delay time is the time in nanoseconds from the 1.4 Volt point of the reference signal to the 1.4 Volt point of the output.
2. Maximum and minimum output propagation delays are specified with a 50 pF load on the outputs, unless otherwise noted.
3. Output tristate delay is the time in nanoseconds from the 1.4 Volt point of the reference signal to ±300mV of the termination voltage on the output. The test load is 50• to 1.4V in parallel with 10 pf to GND.

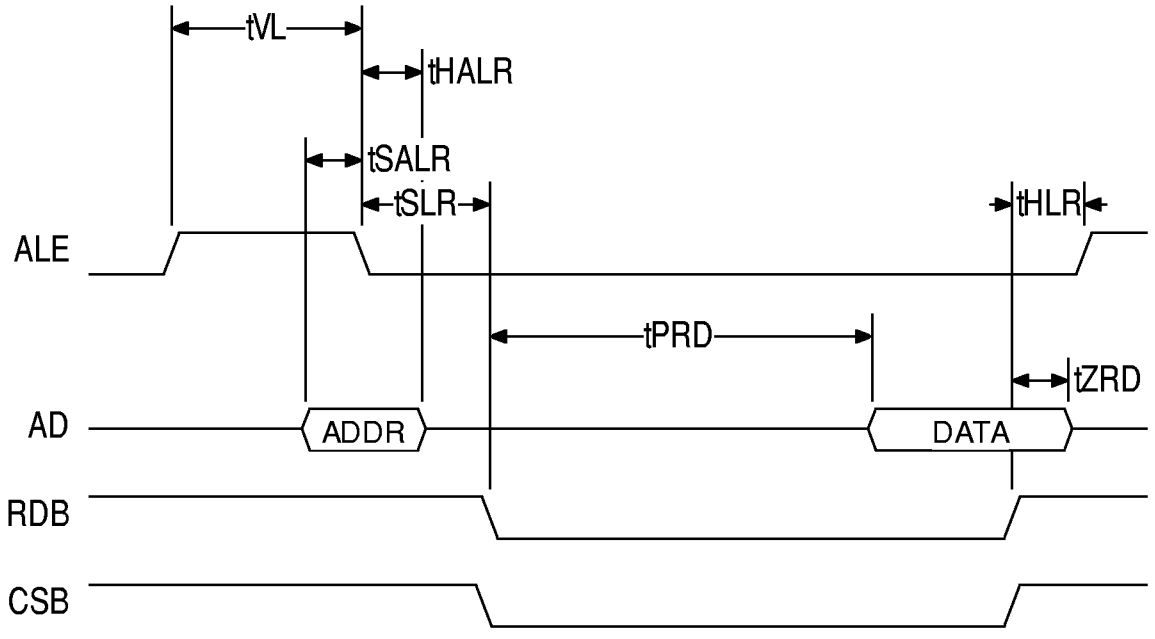
Notes on Typical Values

1. AC parameters shown as **Typical** in the following tables are tested for functionality under typical conditions. No guarantees are implied for maximum or minumum performance.

13.1 MICROPROCESSOR INTERFACE TIMING CHARACTERISTICS

13.1.1 Microprocessor Interface Read Access

Symbol	Parameter	Min	Max	Units
tS_{ALR}	Address to Latch Set-up Time	10		ns
tH_{ALR}	Address to Latch Hold Time	10		ns
tV_{L}	Valid Latch Pulse Width	20		ns
tS_{LR}	Latch to Read Set-up	5		ns
tH_{LR}	Latch to Read Hold	0		ns
tP_{RD}	Valid Read to Valid Data Propagation Delay [4][5]		72	ns
tZ_{RD}	Valid Read Negated to Output Tristate[6]		10	ns



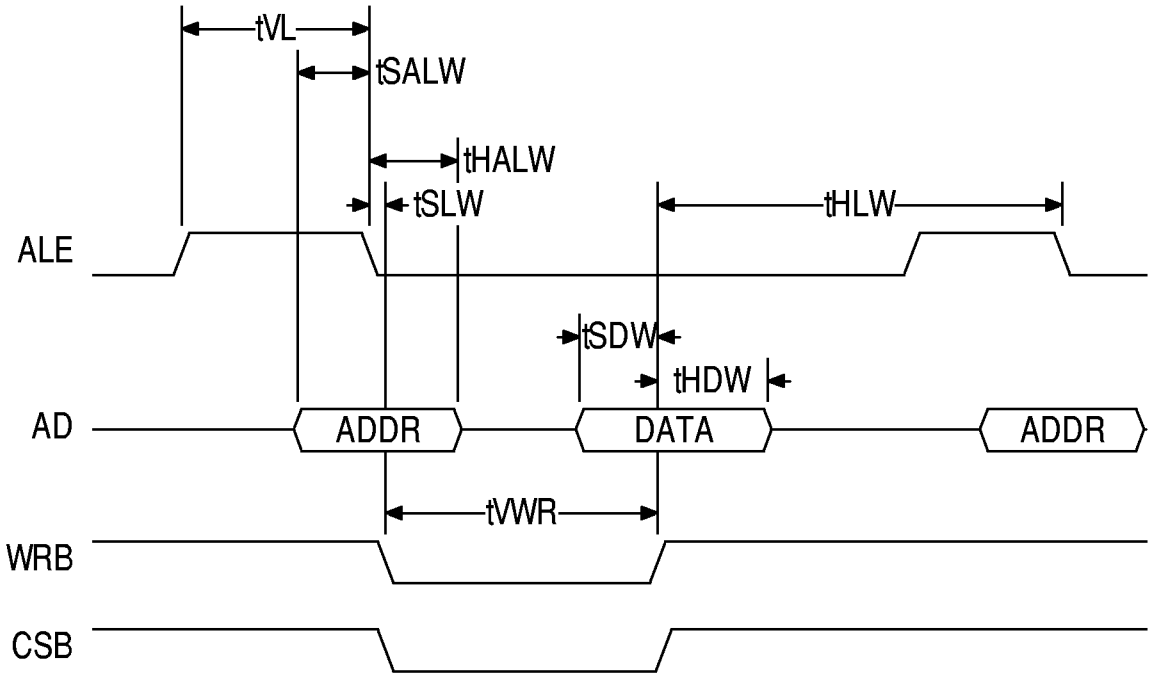
4
5 tPRD = 4 * (T_{SYSCLK}) + 8ns
6 Although not tested in production, the minimum output tri-state time from de-assertion of MI_RD or MI_CS should be considered to be 0 ns.

13.1.2 Microprocessor Interface Write Access

Symbol	Parameter	Min	Max	Units
tS_{DW}	Data to Valid Write Set-up Time	10		ns
tS_{ALW}	Address to Latch Set-up Time	10		ns
tH_{ALW}	Address to Latch Hold Time	10		ns
tV_{L}	Valid Latch Pulse Width	20		ns
tS_{LW}	Latch to Write Set-up	0		ns
tH_{LW}	Latch to Write Hold [7]	64		ns
tH_{DW}	Data to Valid Write Hold Time	5		ns
tV_{WR}	Valid Write Pulse Width	20		ns

[7] $tHLW = 4 * (T_{SYSCLK})$

13.2

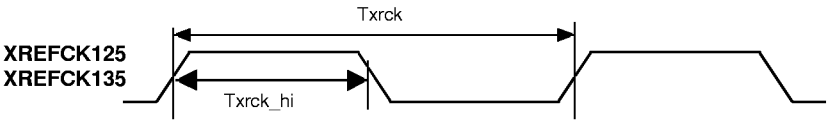

tVL
tSALW
tHALW
tSLW
tHLW
ALE
tSDW
tHDW
AD
ADDR
DATA
ADDR
tVWR
WRB
CSB

EXACT INTERFACE INTERFACE TIMING CHARACTERISTICS

13.2.1 XREFCK input timing

Symbol	Description	Min	Max	Units
F_{xrck}	XREFCK max frequency		135	MHz
T_{xrdc}	XREFCK duty cycle	45	55	%

XREFCK input timing diagram



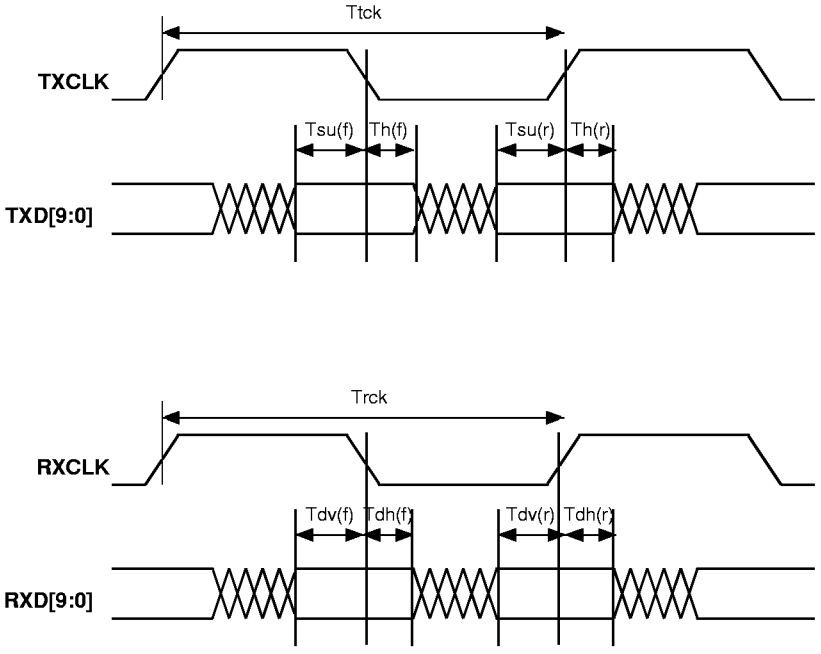
(1) frequency: Fxrck = 1/Txrck
(2) Duty cycle: Txrdc = Txrck_hi / Txrck

13.2.2 Clear Channel clock mode or 8B10B non-SERDES clock mode timing (Load = 30pF)

Symbol	Description	Min	Typ	Max	Units
T_{TCK}	TXCLK Transmit Clock Period		14.8[8]		ns
$T_{SU(R)}$, $T_{SU(F)}$	Transmit Data Setup Time (valid) Before Edge of TXCLK	2.0			ns
$T_{H(R)}$, $T_{H(F)}$	Transmit Data Hold Time (valid) After Edge of TXCLK	2.0			ns
T_{TDC}	Transmit Clock Duty Cycle	40		60	%
T_{RCK}	Receive Clock Period of RXCLK	14.8			ns
$T_{DV(R)}$, $T_{DV(F)}$	Receive Data Valid Before Edge of RXCLK	1.5			ns
$T_{DH(R)}$, $T_{DH(F)}$	Receive Data Hold after Edge of RXCLK	1.5			ns
T_{RDC}	Receive Clock Duty Cycle of RXCLK	35		65	%

[8] By design, TXCLK Transmit Clock Period is 50% of the selected XREFCK period for a given TX port (either XREFCK125 or XREFCK135).

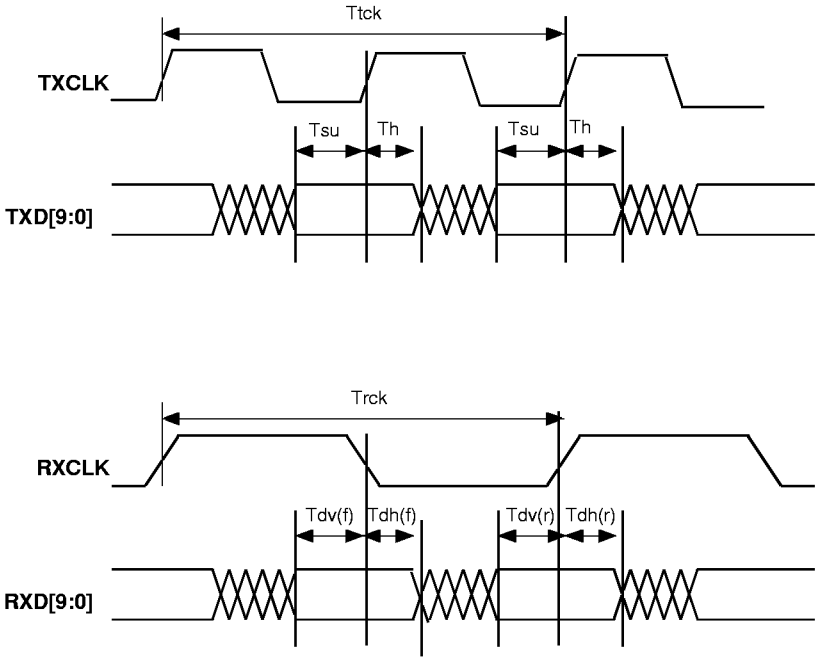
Clear channel clock mode or 8B10B non-SERDES clock mode timing diagram



13.2.3 SERDES mode timing (XREFCK at 125 MHz; Load = 30pF)

Symbol	Description	Min	Typ	Max	Units
T_{TCK}	TXCLK Transmit Clock Period		7.9		ns
T_{SU}	Transmit Data Setup Time (valid) Before TXCLK Rising Edge	2.4			ns
T_{H}	Transmit Data Hold Time (valid) After TXCLK Rising Edge	2.0			ns
T_{TDC}	Transmit Clock Duty Cycle	40		60	%
T_{RCK}	Receive Clock Period of RXCLK	15.8			ns
$T_{DV(R)}$, $T_{DV(R)}$	Receive Data Valid Before Edge of RXCLK	1.5			ns
$T_{DH(R)}$, $T_{DH(R)}$	Receive Data Hold After Edge of RXCLK	1.5			ns
T_{RDC}	Receive Clock Duty Cycle	35		65	%

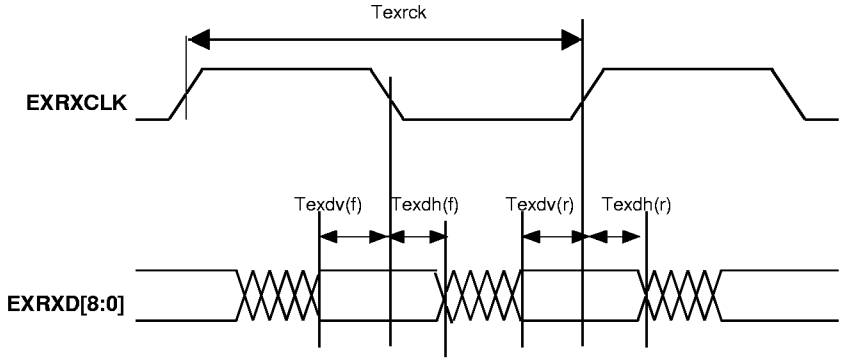
SERDES clock mode timing diagram



13.2.4 Expansion Interface mode timing (Load = 30pF)

Symbol	Description	Min	Typ	Max	Units
T_{EXRCK}	Expansion Port Receive Clock Period (EXRXCLK)		14.8		ns
$T_{EXDV(R)}$, $T_{EXDV(F)}$	Expansion Port Receive Data Valid Before Edge of EXRXCLK	1.5			ns
$T_{EXDH(R)}$, $T_{EXDH(F)}$	Expansion Port Receive Data Hold after Edge of EXRXCLK	1.5			ns
T_{EXRDC}	Expansion Port Receive Clock Duty Cycle	35		65	%

EXACT Expansion Interface timing diagram



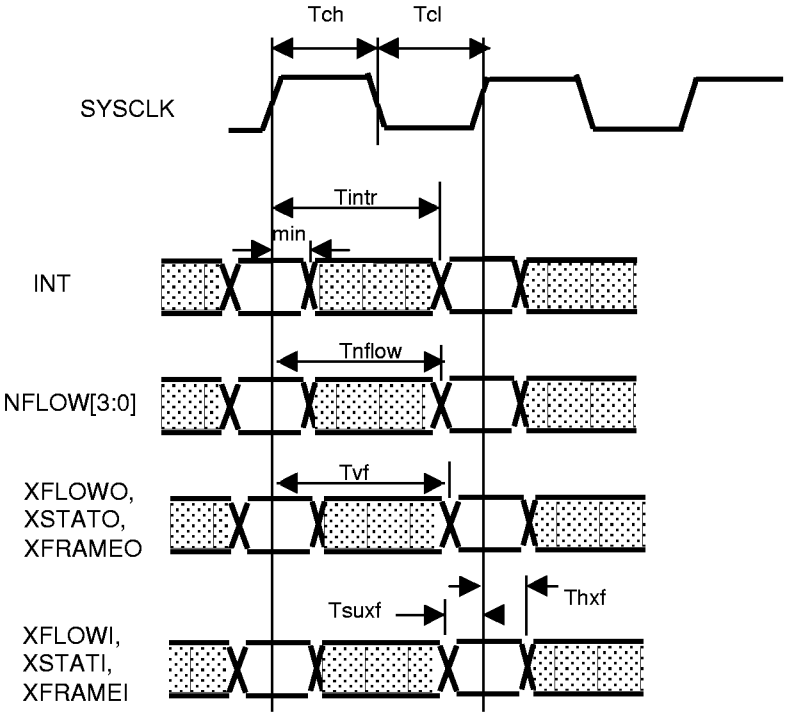
13.3 SYSCLK referenced timing

Symbol	Description	Min	Max	Units
Fck	SYSCLK frequency	5.0	67.5	MHz
Tcdc	SYSCLK duty cycle	45	55	%
Trst	RST assertion time after SYSCLK stable	1.0 [1]		us
Tinit	Initialization time after RST assertion	1050 [1]		SYSCLK cycles
Tintr	INTR valid after SYSCLK rising	2.0	13.0	ns
Tnflow	NFLOW valid after SYSCLK rising	3.0	12.0	ns
Tsuxf	Set-up time on XFLOWI, XSTATI, and XFRAMEI with respect to SYSCLK rising	1.0		ns
Thxf	Hold time on XFLOWI, XSTATI, and XFRAMEI with respect to SYSCLK rising	4.0		ns
Tvxf	XFLOWO, XSTATO, and XFRAMEO valid with respect to SYSCLK rising	4.0	12.0	ns

Notes:

1. Typical. Guaranteed by design.

SYSCLK reference timing diagrams

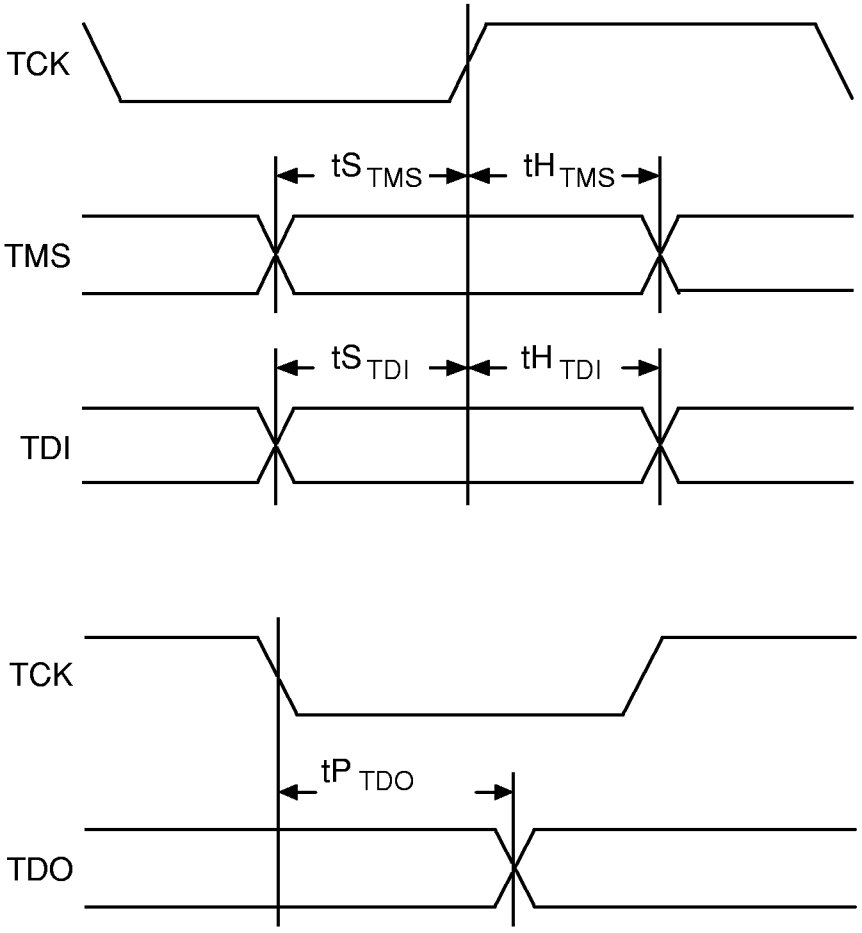


(1) frequency: Fck = 1/(Tch+Tcl)
(2) Duty cycle: Tcdc = Tch/ (Tch+Tcl)

13.4 – JTAG Port Interface

Symbol	Description	Min	Max	Units
	TCK Frequency		1	MHz
	TCK Duty Cycle	40	60	%
tS_{TMS}	TMS Set-up time to TCK	50		ns
tH_{TMS}	TMS Hold time to TCK	50		ns
tS_{TDI}	TDI Set-up time to TCK	50		ns
tH_{TDI}	TDI Hold time to TCK	50		ns
tP_{TDO}	TCK Low to TDO Valid	2	50	ns

FIGURE JTAG Port Interface Timing

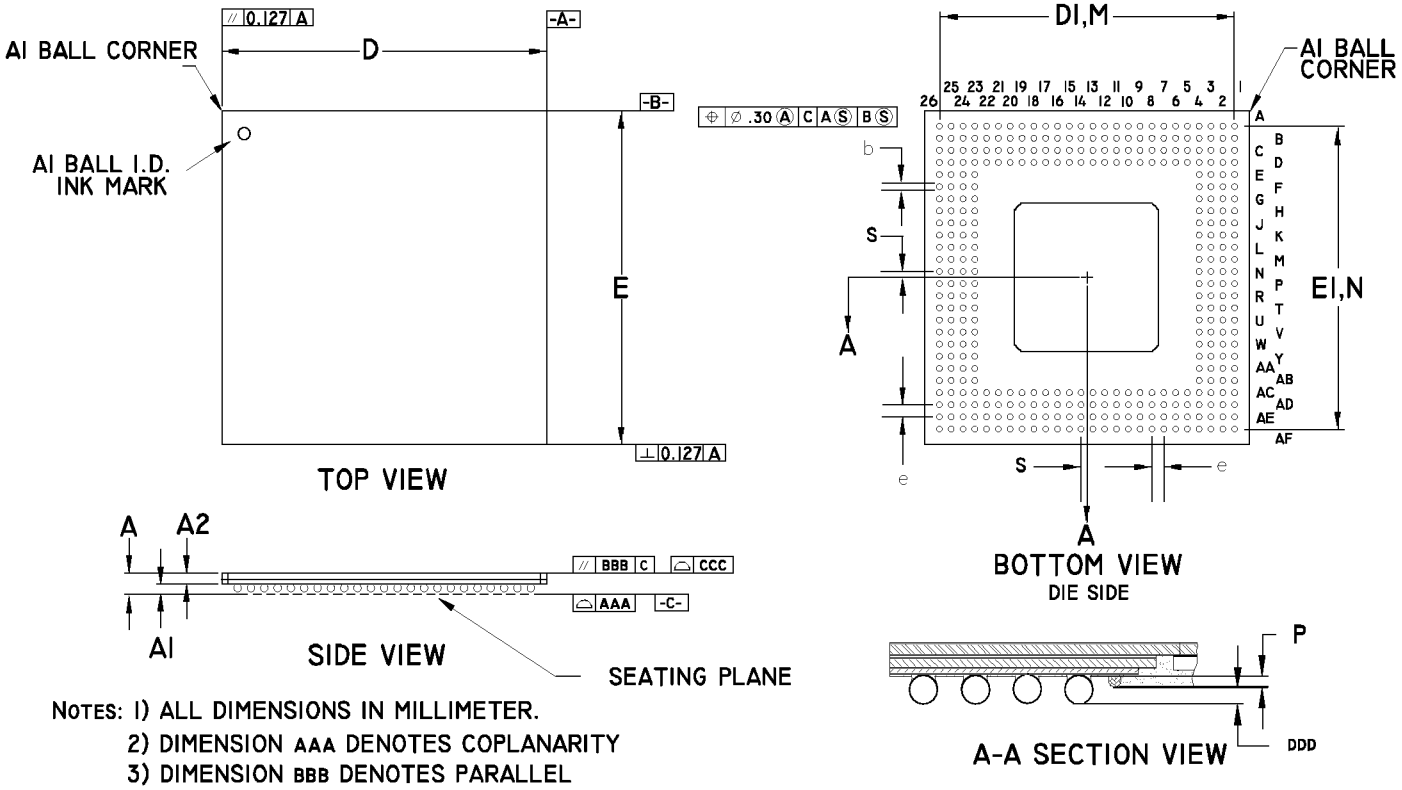


14 THERMAL INFORMATION

Please contact PMC-Sierra for detailed thermal

15 MECHANICAL INFORMATION

FIGURE 11 352 Ball Grid Array (SBGA)



NOTES: 1) ALL DIMENSIONS IN MILLIMETER.
2) DIMENSION AAA DENOTES COPLANARITY
3) DIMENSION BBB DENOTES PARALLEL
4) DIMENSION CCC DENOTES FLATNESS

PACKAGE TYPE: 352 PIN THERMAL BALL GRID ARRAY																
BODY SIZE: 35 x 35 x 1.45 MM																
DIM.	A	A1	A2	D	D1	E	E1	M,N	e	b	AAA	BBB	CCC	DDD	P	S
MIN.	1.41	0.56	0.85	34.90	31.65	34.90	31.65			0.60				0.15	0.20	
NOM.	1.54	0.63	0.91	35.00	31.75	35.00	31.75	26x26	1.27	0.75				0.33	0.30	
MAX.	1.67	0.70	0.97	35.10	31.85	35.10	31.85			0.90	0.15	0.15	0.20	0.50	0.35	0.635

Notes

CONTACTING PMC-SIERRA, INC.

PMC-Sierra, Inc.
105-8555 Baxter Place Burnaby, BC
Canada V5A 4V7

Tel: (604) 415-6000

Fax: (604) 415-6200

Document Information: document@pmc-sierra.com
Corporate Information: info@pmc-sierra.com
Application Information: apps@pmc-sierra.com
Web Site: http://www.pmc-sierra.com

PMC-971034 (p4) ref PMC-970780 (p4) Issue date: January 1999