

23.5.1 Function Enable Register (ZMII0_FER)

ZMII0_FER selects the various interface conversion functions provided by the ZMII bridge. ZMII0_FER also deselects (functionally isolates) an unused EMAC. The interface conversion functions are mutually exclusive, that is, all EMACs must convert to the same interface (SMII, RMII, or MII) in the bridge, or be deselected (an EMAC must be deselected for MII). Mixed interface conversion is not supported.

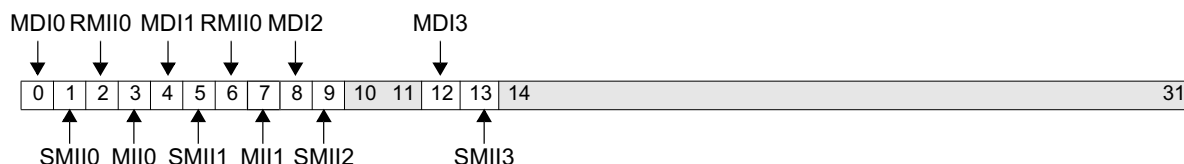


Figure 23-5. Function Enable Register (ZMII0_FER)

0	MDI0	EMAC0 MDI Enable 0 EMAC0 management data and clock are ignored. 1 EMAC0 management data and clock are driven to the PHY.	Must be 0 if MDI1, MDI2, MDI3 = 1.
1	SMII0	EMAC0 SMII Enable 0 EMAC0 SMII PHY interface is disabled. 1 EMAC0 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
2	RMII0	EMAC1 RMII Enable 0 EMAC0 RMII PHY interface is disabled. 1 EMAC0 RMII PHY interface is enabled.	Must be 0 if SMII0, SMII1, SMII2, SMII3, MII0, or MII1 is 1.
3	MII0	EMAC1 MII Enable 0 EMAC0 MII PHY interface is disabled. 1 EMAC0 MII PHY interface is enabled.	Must be 0 if SMII0, SMII1, SMII2, SMII3, RMII0, RMII1, or MII1 is 1.
4	MDI1	EMAC1 MDI Enable 0 EMAC1 management data and clock are ignored. 1 EMAC1 management data and clock are driven to the PHY.	Must be 0 if MDI0, MDI2, MDI3 = 1.
5	SMII1	EMAC1 SMII Enable 0 EMAC1 SMII PHY interface is disabled. 1 EMAC1 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
6	RMII1	EMAC1 RMII Enable 0 EMAC1 RMII PHY interface is disabled. 1 EMAC1 RMII PHY interface is enabled.	Must be 0 if SMII0, SMII1, SMII2, SMII3, MII0, or MII1 is 1.
7	MII1	EMAC1 MII Enable 0 EMAC1 MII PHY interface is disabled. 1 EMAC1 MII PHY interface is enabled.	Must be 0 if SMII0, SMII1, SMII2, SMII3, RMII0, MII0, or RMII1 is 1.
8	MDI2	EMAC2 MDI Enable 0 EMAC2 management data and clock are ignored. 1 EMAC2 management data and clock are driven to the PHY.	Must be 0 if MDI0, MDI1, MDI3 = 1.
9	SMII2	EMAC2 SMII Enable 0 EMAC2 SMII PHY interface is disabled. 1 EMAC2 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
10:11		Reserved	
12	MDI3	EMAC3 MDI Enable 0 EMAC3 management data and clock are ignored. 1 EMAC3 management data and clock are driven to the PHY.	Must be 0 if MDI0, MDI1, MDI2 = 1.
13	SMII3	EMAC3 SMII Enable 0 EMAC3 SMII PHY interface is disabled. 1 EMAC3 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
14:31		Reserved	

Preliminary User's Manual**23.5.2 Speed Select Register (ZMII0_SSR)**

ZMII0_SSR selects the speed at which each EMAC transfer data (10 Mbps or 100 Mbps). When 10 Mbps is selected, the ZMII bridge generates a 2.5 MHz clock. When 100 Mbps is selected, the ZMII bridge generates a 25 MHz clock.

ZMII0_SSR also controls whether collisions are indicated.

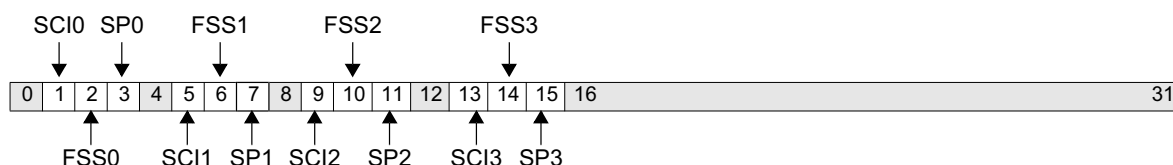


Figure 23-6. Speed Selection Register (ZMII0_SSR)

0		Reserved	
1	SCI0	EMAC0 Suppress Collision Indication 0 EMAC0 collision indication signal is enabled. 1 EMAC0 collision indication signal is suppressed.	
2	FSS0	EMAC0 Force Speed Selection 0 ZMII0_SSR[SP0] does not override IPG status field. 1 ZMII0_SSR[SP0] overrides IPG status field.	SMII only.
3	SP0	EMAC0 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
4		Reserved	
5	SCI1	EMAC1 Suppress Collision Indication 0 EMAC1 collision indication signal is enabled. 1 EMAC1 collision indication signal is suppressed.	
6	FSS1	EMAC1 Force Speed Selection 0 ZMII0_SSR[SP1] does not override IPG status field. 1 ZMII0_SSR[SP1] overrides IPG status field.	SMII only.
7	SP1	EMAC1 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
8		Reserved	
9	SCI2	EMAC2 Suppress Collision Indication 0 EMAC2 collision indication signal is enabled. 1 EMAC2 collision indication signal is suppressed.	
10	FSS2	EMAC2 Force Speed Selection 0 ZMII0_SSR[SP2] does not override IPG status field. 1 ZMII0_SSR[SP2] overrides IPG status field.	SMII only.
11	SP2	EMAC3 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
12		Reserved	
13	SCI3	EMAC3 Suppress Collision Indication 0 EMAC3 collision indication signal is enabled. 1 EMAC3 collision indication signal is suppressed.	

14	FSS3	EMAC3 Force Speed Selection 0 ZMII0_SSR[SP3] does not override IPG status field. 1 ZMII0_SSR[SP3] overrides IPG status field.	SMII only.
15	SP3	EMAC3 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
16:31		Reserved	

Preliminary User's Manual

23.5.3 SMII Status Register (ZMII0_SMIISR)

ZMII0_SMIISR contains the status transferred during the last inter-packet gap in the SMII interface, at the moment it is read. This register is used only if SMII interfaces are enabled. The register bits pertain to EMAC0RxD and EMAC1RxD SMII inter frame status used to convey packet data, RX_ER, and PHY status and are valid only when RX_DV (receive data valid) is set to 0.

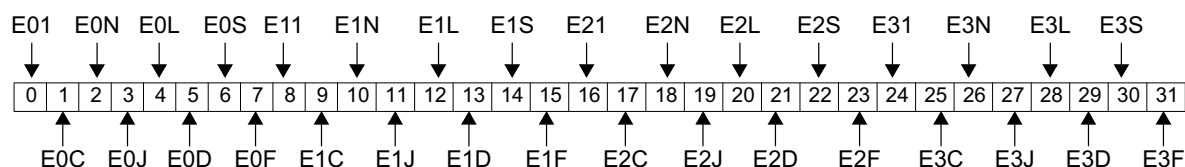


Figure 23-7. SMII Status Register (ZMII0_SMIISR)

0	E01	EMAC0RxD Set to 1	
1	E0C	EMAC0RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
2	E0N	EMAC0RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
3	E0J	EMAC0RxD Jabber 0 OK 1 Error	
4	E0L	EMAC0RxD Link 0 Down 1 Up	
5	E0D	EMAC0RxD Duplex 0 Half 1 Full	
6	E0S	EMAC0RxD Speed 0 10MBit 1 100MBit	
7	E0F	EMAC0RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
8	E11	EMAC1RxD Set to 1	
9	E1C	EMAC1RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
10	E1N	EMAC1RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
11	E1J	EMAC1RxD Jabber 0 OK 1 Error	
12	E1L	EMAC1RxD Link 0 Down 1 Up	
13	E1D	EMAC1RxD Duplex 0 Half 1 Full	
14	E1S	EMAC1RxD Speed 0 10MBit 1 100MBit	

Preliminary User's Manual

15	E1F	EMAC1RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
16	E2I	EMAC2RxD Set to 1	
17	E2C	EMAC2RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
18	E2N	EMAC2RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
19	E2J	EMAC2RxD Jabber 0 OK 1 Error	
20	E2L	EMAC2RxD Link 0 Down 1 Up	
21	E2D	EMAC2RxD Duplex 0 Half 1 Full	
22	E2S	EMAC2RxD Speed 0 10MBit 1 100MBit	
23	E2F	EMAC2RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
24	E3I	EMAC3RxD Set to 1	
25	E3C	EMAC3RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
26	E3N	EMAC3RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
27	E3J	EMAC3RxD Jabber 0 OK 1 Error	
28	E3L	EMAC3RxD Link 0 Down 1 Up	
29	E3D	EMAC3RxD Duplex 0 Half 1 Full	
30	E3S	EMAC3RxD Speed 0 10MBit 1 100MBit	
31	E3F	EMAC3RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.

Preliminary User's Manual

23.6 RGMII Bridge

The RGMII bridge provides the capability for the EMACs to connect to external ethernet PHYs that support the reduced gigabit media independent interface (RGMII) or the reduced ten bit interface (RTBI) or their standard GMII or TBI versions. A GMII port requires twenty four pins of data and control clocks while the RGMII port only needs twelve pins plus a common clock. A TBI port requires twenty pins of data and three clocks while a RTBI port requires ten pins and two clocks.

23.6.1 RGMII Bridge Features

The RGMII/RTBI bridge supports the following features:

Support for one GMII PHY

Support for one TBI PHY

Support for two independent RGMII PHYs

Support for two independent RTBI PHY

Support for one RGMII PHY and one RTBI PHY

Supports 10 Mbps, 100 Mbps or 1Gbps data rates (1.25 Gbaud for RTBI)

Uses single 125Mhz clock reference sourced from an external source that is present during reset

Independent 4-bit transmit and receive paths

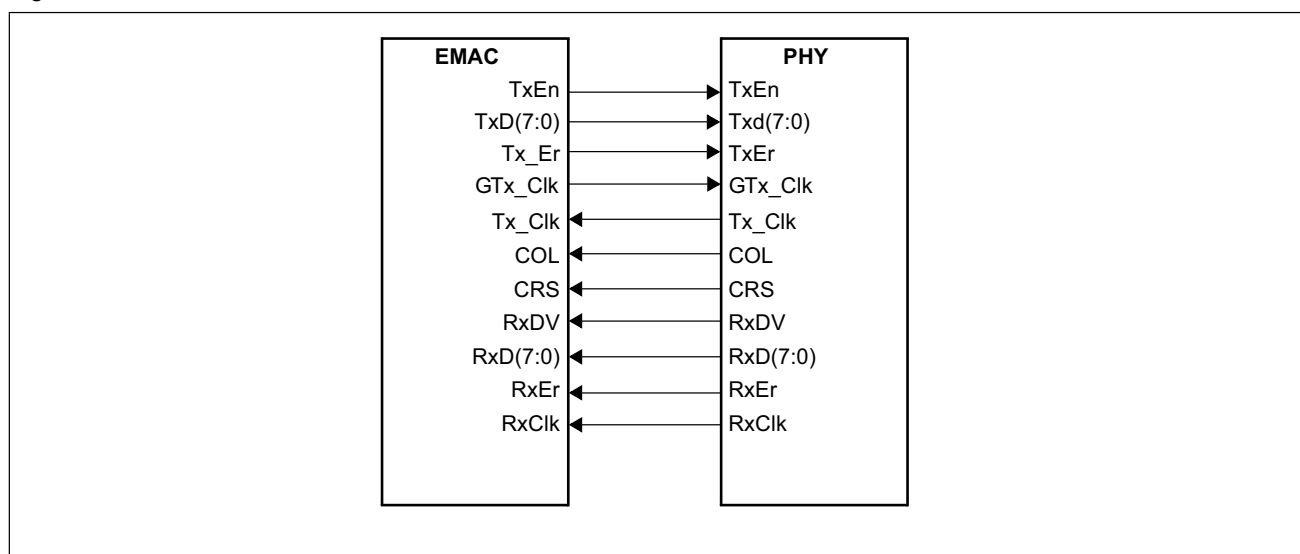
Complies with 2.5v CMOS interface voltages as defined by JEDED EIA/JESD8-5

23.6.2 RGMII/RTBI Bridge Interface

The purpose of RGMII/RTBI bridge is to reduce the pin count of the ethernet gigabit medium independent interface (GMII) by multiplexing the data and control signals. Similarly, the same pins are used to multiplex with the reduced ten bit interface (RTBI) signals.

In the normal GMII mode (as shown in *Figure 23-8*), data between EMAC and PHY is transmitted and received over two eight-bit wide data buses TxD(7:0), RxD(7:0) with six control signals TxEn, TxEr, RxDv, RxEr, COL and CRS. The data and control signals are synchronous to the two clocks TxClk and RxClk. The total number of pins required by GMII would be 24 plus MDIO and MdClk. (MDIO and MdClk are bidirectional serial data and clock signals used by EMAC to configure the internal registers inside the PHYs.

Figure 23-8. GMII 24 Pins + 2



Similarly in normal TBI mode, the data from GPCS (integrated into EMAC) is transmitted and received over two ten-bit data buses PMA_Txd(9:0) and PMA_Rxd(9:0). The data buses are synchronous with PmaTxClk, PmaRxClk0, and PmaRxClk1. Using the RGMII/RTBI module, the databus and control signals between EMAC and PHY are time-multiplexed by using both edges of the clocks as shown in *Table 23-4*.

Table 23-4. RGMII Bridge PHY Interface Signals

RGMII	RTBI	I/O	Description
EMCTxC	EMCTxC	I/O	RGMII/RTBI transmit reference clock is 125Mhz, 25Mhz, or 2.5Mhz+-50ppm depending on speed
EMCTxD(3:0)	EMCTxD(3:0)	O	RGMII transmits bits 3:0 on positive edge of TxClk and bits 7:4 on negative edge of TxClk RTBI transmits bits 3:0 on positive edge of TxClk and bits 8:5 on negative edge of TxClk
EMCTxCTL	EMCTxCTL	O	RGMII transmits TxEN on positive edge of TxClk and a logical derivative of TxEN and TxER on negative edge of TxClk RTBI transmits bit 4 on positive edge of TxClk and bit 9 on negative edge of TxClk
EMCRxC	EMCRxC	O	RGMII/RTBI continuous receive reference clock is 125Mhz, 25Mhz, or 2.5Mhz+-50ppm depending on speed

Preliminary User's Manual*Table 23-4. RGMII Bridge PHY Interface Signals (Continued)*

RGMII	RTBI	I/O	Description
EMCRxD(3:0)	EMCRxD(3:0)	I	RGMII receives bits 3:0 on positive edge of RxC and bits 7:4 on negative edge of RxC RTBI receives bits 3:0 on positive edge of RxC and bits 8:5 on negative edge of RxC
EMCRxCTL	EMCRxCTL	I	RGMII receives RxEN on positive edge of RxC and a logical derivative of RxEN and RxER on negative edge of RxC RTBI receives bit 4 on positive edge of RxC and bit 9 on negative edge of RxC

The data rate achievable at each frequency is as follows:

- TxC = RxC = 125Mhz => (125Mhz x 2) x 4 bits = 1000 Mbits/sec
- TxC = RxC = 25Mhz => (25Mhz) x 4 bits = 100 Mbits/sec (note)
- TxC = RxC = 2.5Mhz => (2.5Mhz) x 4 bits = 10 Mbits/sec (note)

Note: For the ethernet 10/100 modes, the data is duplicated on the negative edge of the clocks; this causes the data bits 0:3 presented on the EMAC/RMII interface to be duplicated on data bits 4:7, thus emulating the MII interface required by the EMAC

23.6.3 Muxing for Ethernet to PHY Bridges

The tables that follow describe the ethernet muxing for ZMII and RGMII bridges. *Table 23-5* describes the ethernet muxing for MII, RMII, SMII and RGMII interfaces while *Table 23-6* describes the ethernet muxing for MII, RMII, SMII and RGMII interfaces.

Note: RGMII/GMII (RTBI/TBI) does not have the MDIO interface. This interface is driven from ZMII bridge. So if using the RGMII/GMII interface the MDIO interface must be accessed through the ZMII bridge.

Table 23-5. Ethernet Muxing for MII, RMII, SMII and RGMII Interfaces

MII	I/O	RMII	I/O	SMII	I/O	RGMII	I/O	Description
EMCMDIO	I/O	EMCMDIO	I/O	EMCMDIO	I/O	EMCMDIO	I/O	MDIO data input/output
EMCMDClk	O	EMCMDCLK	O	EMCMDCLK	O	EMCMDCLK	O	MDIO data clock
EMCRxD0	I	EMC0Rx0	I	EMC0Rx0	I		I	MII receive data 0 RMII0 receive data 0 SMII0 receive data
EMCRxD1	I	EMC0Rx1	I	EMC1Rx0	I		I	MII receive data 1 RMII0 receive data 1 SMII1 receive data
EMCRxD2	I	EMC1Rx0	I	EMC2Rx0	I	GMC0Tx0	O	MII receive data 2 RMII1 receive data 0 SMII2 receive data RGMII0 transmit data 0
EMCRxD3	I	EMC1Rx1	I	EMC3Rx0	I	GMC0Tx1	O	MII receive data 3 RMII1 receive data 1 SMII2 receive data RGMII0 transmit data 1
EMCRxDV	I	EMC1CRSDV	I		I	GMC1Tx0	O	MII receive data valid RMII1 receive data valid RGMII1 transmit data 0
EMCRxClk	I		I		I	GMC1Tx1	O	MII receive medium clock RGMII1 transmit data 1
EMCRxErr	I	EMC0RxErr	I		I	GMC1Tx2	O	MII receive error RMII0 receive error RGMII1 transmit data 2

Preliminary User's Manual

Table 23-5. Ethernet Muxing for MII, RMII, SMII and RGMII Interfaces (Continued)

MII	I/O	RMII	I/O	SMII	I/O	RGMII	I/O	Description
EMCCRS	I	EMC0CRSDV	I		I	GMC1TxD3	O	MII carrier sense RMII0 carrier sense data valid RGMII1 transmit data 3
EMCTxClk	I	EMCRefClk	I	EMC0RefClk	I			MII transmit clock RMII reference clock SMII reference clock
EMCCCD	I	EMC1RxErr	I		I	GMC0TxClk	O	MII collision RMII1 receive error RGMII0 transmit clock
EMCTxErr	O	EMC1TxEn	O		O	GMC0RxClk	I	MII transmit error RMII1 transmit enable RGMII0 receive clock
EMCTxEn	O	EMC0TxEn	O	EMCSync	O			MII transmit enable RMII0 transmit enable SMII sync
EMCTxD0	O	EMC0TxD0	O	EMC0TxD	O			MII transmit data 0 RMII0 transmit data 0 SMII0 transmit data 0
EMCTxD1	O	EMC0TxD1	O	EMC1TxD	O			MII transmit data 1 RMII0 transmit data 1 SMII1 transmit data 1
EMCTxD2	O	EMC1TxD0	O	EMC2TxD	O	GMC0TxD2	O	MII transmit data 2 RMII1 transmit data 0 SMII2 transmit data RGMII0 transmit data 2
EMCTxD3	O	EMC1TxD1	O	EMC3TxD	O	GMC0TxD3	O	MII transmit data 3 RMII1 transmit data 1 SMII3 transmit data RGMII0 transmit data 3
						GMC0RxD0	I	RGMII0 receive data 0
						GMC0RxD1	I	RGMII0 receive data 1
						GMC0RxD2	I	RGMII0 receive data 2
						GMC0RxD3	I	RGMII0 receive data 3
						GMC0RxCtl		RGMII0 receive control
						GMC0TxCtl	O	RGMII0 transmit control
						GMC1RxClk	I	RGMII1 receive clock
						GMC1RxD0	I	RGMII1 receive data 0
						GMC1RxD1	I	RGMII1 receive data 1
						GMC1RxD2	I	RGMII1 receive data 2
						GMC1RxD3	I	RGMII1 receive data 3
						GMC1TxClk	O	RGMII1 transmit clock
						GMC1RefClk	I	RGMII1 reference clock
						GMC1RxCtl	I	RGMII1 receive control
						GMC1TxCtl	O	RGMII1 transmit control

Preliminary User's Manual

Table 23-6. Ethernet Muxing for GMII, TBI, and RTBI Interfaces

RGMII	I/O	GMII	I/O	TBI	I/O	RTBI	I/O	Description
GMC0TxD0	O	GMCTxD0	O	TBITxD0	O	RTBI0TxD0	O	RGMII0 transmit data 0 GMII transmit data 0 TBI transmit data 0 RTBI0 transmit data 0
GMC0TxD1	O	GMCTxD1	O	TBITxD1	O	RTBI0TxD1	O	RGMII0 transmit data 1 GMII transmit data 1 TBI transmit data 1 RTBI0 transmit data 1
GMC1TxD0	O	GMCTxD4	O	TBITxD4	O	RTBI1TxD0	O	RGMII1 transmit data 0 GMII transmit data 4 TBI transmit data 4 RTBI1 transmit data 0
GMC1TxD1	O	GMCTxD5	O	TBITxD5	O	RTBI1TxD1	O	RGMII1 transmit data 1 GMII transmit data 5 TBI transmit data 5 RTBI1 transmit data 1
GMC1TxD2	O	GMCTxD6	O	TBITxD6	O	RTBI1TxD2	O	RGMII1 transmit data 2 GMII transmit data 6 TBI transmit data 6 RTBI1 transmit data 2
GMC1TxD3	O	GMCTxD7	O	TBITxD7	O	RTBI1TxD3	O	RGMII1 transmit data 3 GMII transmit data 7 TBI transmit data 7 RTBI1 transmit data 3
GMC0TxClk	O	GMCGTxClk	O	TBITxClk	O	RTBI0TxClk	O	RGMII0 transmit clock GMII 1000 Mbps transmit clock TBI transmit clock RTBI transmit clock
GMC0RxClk	I	GMCRxClk	I	TBIRxClk0	I	RTBI0RxClk	I	RGMII0 receive clock GMII receive clock TBI receive clock 0 RTBI receive clock
GMC0TxD2	O	GMCTxD2	O	TBITxD2	O	RTBI0TxD2	O	RGMII0 transmit data 2 GMII transmit data 2 TBI transmit data 2 RTBI0 transmit data 2
GMC0TxD3	O	GMCTxD3	O	TBITxD3	O	RTBI0TxD3	O	RGMII0 transmit data 3 GMII transmit data 3 TBI transmit data 3 RTBI0 transmit data 3
GMC0RxD0	I	GMCRxD0	I	TBIRxD0	I	RTBI0RxD0	I	RGMII0 receive data 0 GMII receive data 0 TBI receive data 0 RTBI0 receive data 0
GMC0RxD1	I	GMCRxD1	I	TBIRxD1	I	RTBI0RxD1	I	RGMII0 receive data 1 GMII receive data 1 TBI receive data 1 RTBI0 receive data 1

Table 23-6. Ethernet Muxing for GMII, TBI, and RTBI Interfaces (Continued)

RGMII	I/O	GMII	I/O	TBI	I/O	RTBI	I/O	Description
GMC0RxD2	I	GMCRxD2	I	TBIRxD2	I	RTBI0RxD2	I	RGMII0 receive data 2 GMII receive data 2 TBI receive data 2 RTBI0 receive data 2
GMC0RxD3	I	GMCRxD3	I	TBIRxD3	I	RTBI0RxD3	I	RGMII0 receive data 3 GMII receive data 3 TBI receive data 3 RTBI0 receive data 3
GMC0RxCtl	I	GMCRxDv	I	TBIRxD8	I	RTBI0RxD4	I	RGMII0 receive control GMII receive divisor TBI receive data 8 RTBI0 receive control
GMC0TxCtl	O	GMCTxEn	O	TBITxD8	O	RTBI0TxD4	O	RGMII0 transmit control GMII transmit enable TBI transmit data 8 RTBI0 transmit control
GMC1RxCik	I	GMCCD	I		I	RTBI1RxCik	I	RGMII1 receive clock GMII collision detect RTBI1 receive clock
		GMCTxCik	I	TBIRxCik1	I			GMII 10/100 Mbps transmit clock TBI receive clock 1
GMC1RxD0	I	GMCRxD4	I	TBIRxD4	I	RTBI1RxD0	I	RGMII1 receive data 0 GMII receive data 4 TBI receive data 4 RTBI1 receive data 0
GMC1RxD1	I	GMCRxD5	I	TBIRxD5	I	RTBI1RxD1	I	RGMII1 receive data 1 GMII receive data 5 TBI receive data 5 RTBI1 receive data 1
GMC1RxD2	I	GMCRxD6	I	TBIRxD6	I	RTBI1RxD2	I	RGMII1 receive data 2 GMII receive data 6 TBI receive data 6 RTBI1 receive data 2
GMC1RxD3	I	GMCRxD7	I	TBIRxD7	I	RTBI1RxD3	I	RGMII1 receive data 3 GMII receive data 7 TBI receive data 7 RTBI1 receive data 3
GMC1TxClk	O	GMCCrs	I			RTBI1TxClk	O	RGMII1 transmit clock GMII RTBI1 transmit clock

Preliminary User's Manual

Table 23-6. Ethernet Muxing for GMII, TBI, and RTBI Interfaces (Continued)

RGMII	I/O	GMII	I/O	TBI	I/O	RTBI	I/O	Description
GMCRfClk	I	GMCRfClk	I	TBIRefClk	I	RTBI1RefClk	I	RGMII1 reference clock GMII reference clock TBI reference clock RTBI1 reference clock
GMC1RxCtl	I	GMCRxEr	I	TBIRxD9	I	RTBI1RxD4	I	RGMII1 receive control GMII receive error TBI receive data 9 RTBI1 receive control
GMC1TxCtl	O	GMCTxEr	O	TBITxD9	O	RTBI1TxD4	O	RGMII1 transmit control GMII transmit error TBI transmit data 9 RTBI1 transmit control

23.6.4 EMAC-RGMII Bridge Interfaces

Figure 23-9 shows how the GMII and TBI interfaces are multiplexed by the RGMII/RTBI module which reduces pins by using both edges of clocks (reducing data buses TxD, RxD by half), multiplexing control pins TxEn/TxEr, RxDv/RxEr on two clock edges, and regenerating indicator signals COL, CRS from data stream bit patterns.

On the transmitting path from the EMAC connecting to the PHY, the RGMII/RTBI module shares the same physical pins by multiplexing the inputs - selecting GMII inputs in one mode and TBI inputs in the other mode (mutually exclusive).

On the receiving path from the PHY going to the EMAC, the RGMII/RTBI module places the demuxed data on either the RxD(7:0) bus and associated control signals or on the PMARxd(9:0) bus.

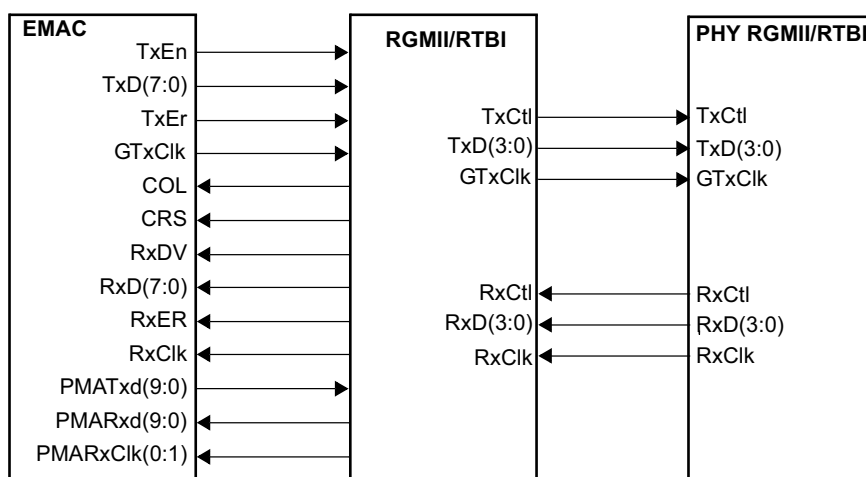


Figure 23-9. EMAC to PHY Using RGMII/RTBI

23.6.5 RGMII Bridge Registers

This section describes the registers in the RGMII bridge, which are listed in *Table 23-7*. The RGMII bridge registers are accessed using an OPB slave interface.

Table 23-7. RGMII Bridge Registers

Register	Description	Address	Access	Page
RGMII0_FER	RGMII Function Enable Register	0x1 40000790	R/W	812
RGMII0_SSR	RGMII Speed Select Register	0x1 40000794	R/W	813

23.6.5.1 RGMII Function Enable Register (RGMII0_FER)

The RGMII function enable register (RGMII0_FER) selects the various interface conversion functions provided by the RGMII bridge. RGMII0_FER also deselects (functionally isolates) an unused EMAC. The reduced and passthrough mode are mutually exclusive, however one EMAC can be set to RTBI mode while the other EMAC to RGMII mode. *Figure 23-10* describes RGMII0_FER register bit descriptions.

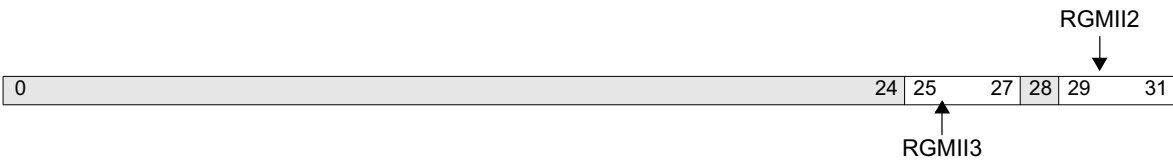


Figure 23-10. RGMII Function Enable Register (RGMII0_FER)

0:24		Reserved
25:27	RGMII3	<p>EMAC3 RGMII Enable</p> <p>0xx EMAC3 RGMII PHY interface is disabled. don't care when disabled</p> <p>100 EMAC3 RTBI PHY interface is enabled. reduced mode when RTBI PHY enabled</p> <p>101 EMAC3 RGMII PHY interface is enabled. reduced mode when RGMII PHY enabled</p> <p>110 EMAC3 TBI PHY interface is enabled. passthru mode when TBI PHY enabled</p> <p>111 EMAC3 GMII PHY interface is enabled. passthru mode when GMII PHY enabled</p> <p>See table note below.</p>
28		Reserved
29:31	RGMII2	<p>EMAC2 RGMII Enable</p> <p>0xx EMAC2 RGMII PHY interface is disabled. don't care when disabled</p> <p>100 EMAC2 RTBI PHY interface is enabled. reduced mode when RTBI PHY enabled</p> <p>101 EMAC2 RGMII PHY interface is enabled. reduced mode when RGMII PHY enabled</p> <p>110 EMAC2 TBI PHY interface is enabled. passthru mode when TBI PHY enabled</p> <p>111 EMAC2 GMII PHY interface is enabled. passthru mode when GMII PHY enabled</p> <p>See table note below.</p>

Note: In reduced mode, the two channels operate independently. In pass-thru mode, only one channel operates. If both channels are enabled and set to pass-thru, or one channel is reduced and the other is pass-thru, these are invalid settings resulting in both channels being disabled.

Preliminary User’s Manual

23.6.5.2 Speed Selection Register (RGMII0_SSR)

The speed selection register RGMII0_SSR selects the speed at which each EMAC transfer data (10 Mbps, 100 Mbps or 1000 Mbps). When 10 Mbps is selected, the RGMII bridge generates a 2.5 MHz clock. When 100 Mbps is selected, the RGMII bridge generates a 25 MHz clock. When 1000 Mbps is selected, the RGMII bridge generates a 125 MHz clock. RGMII0_SSR also controls whether collisions are indicated.

Figure 23-11 describes RGMII0_SSR register bit descriptions.

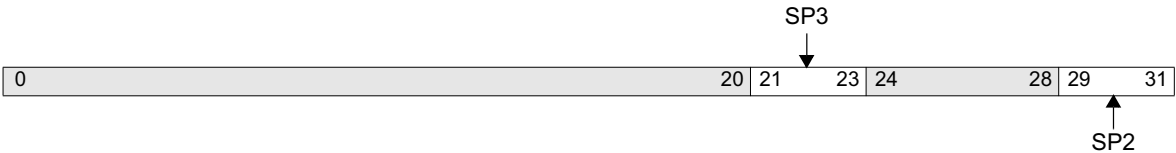


Figure 23-11. Speed Selection Register (RGMII0_SSR)

0:20		Reserved
21:23	SP3	EMAC3 Speed Selection 000 10 Mbps selected. 010 100 Mbps selected. 100 1000 Mbps selected
24:28		Reserved
29:31	SP2	EMAC2 Speed Selection 000 10 Mbps selected. 010 100 Mbps selected. 100 1000 Mbps selected

Preliminary User's Manual

24. Ethernet Media Access Controllers

The PPC440GX Embedded Processor provides four Ethernet media access controllers (EMACs) that are generic implementations of the Ethernet Media Access Control (MAC) protocol complying with ANSI/IEEE Std 802.3 and IEEE 802.3u supplement. All four EMACs support half-duplex (CSMA/CD), full-duplex operation for 10-Mbps and 100-Mbps operations, and operation in the 1000-Mbps range (Gigabit Ethernet) complying with IEEE 802.3z. In the Gigabit Ethernet mode, the real bandwidth is limited only by system latency (attributable to memory access layer (MAL), memory, arbitration, etc.).

All four EMACs are implemented identically, with the exception of register addresses, and while EMAC0 and EMAC1 support only 10-Mbps and 100-Mbps operations, EMAC2 and EMAC3 support 1000-Mbps operations. Because all four EMACs operate identically, the rest of this chapter describes a single EMAC. The EMACs are referred to as EMAC0, EMAC1, EMAC2 and EMAC3. Except in the register summary tables in *EMAC Registers* on page 837, the EMAC registers are prefixed “EMACx_” to denote the identical implementation of registers in each EMAC.

Each EMAC provides one on-chip peripheral bus (OPB based 32-bit bidirectional) slave interfaces and two other extended OPB (EOPB based 128-bit bidirectional) slave interfaces. The first OPB interface provides access to the EMAC configuration and status registers. The PLB/OPB bridge enables the processor core to access these registers.

The two EOPB slave interfaces are used to exchange packet information with the memory access layer (MAL). The MAL is a multi-channel, intermediate hardware layer that resides between packet-based communication cores (such as EMAC) and external memory (such as SDRAM or SRAM). The MAL transfers packet information and status between the EMACs and external memory separately for each of the two EMAC channels (one receive and one transmit). Software (a device driver) maintains a buffer descriptor ring and a set of data buffers in external memory for each channel, and manages the exchange of packet data between the data buffers and the software protocol.

The MAL performs functions such as arbitration between service requests, handling the buffer descriptor memory structure, updating the descriptor status/control fields at the end of packet transfer, and so on. EMAC supports up to 64 words burst length transactions on the MAL interface (which can be programmed to 128 words burst).

In 10/100 Mbps range each EMAC uses a media independent interface (MII) to communicate with the Z media independent interface (ZMII) bridge. The ZMII bridge, described in detail in *EMAC to PHY Interface Bridges* on page 793 communicates with standard physical devices (PHYs). The ZMII bridge supports the media independent interface (MII), the reduced pin count reduced media independent interface (RMII), and serial media independent interface (SMII). Details of PHY communication are in *EMAC-ZMII Bridge Interfaces* on page 796.

In the Gigabit Ethernet mode, the EMAC implements a Gigabit MII (GMII) interface for communication with the PHY device. The EMAC also implements a Gigabit Ethernet PHY coding sublayer (GPCS). The usage of GPCS is optional and it allows the EMAC to be connected to the PMA interface and thus to a Gigabit Ethernet layer directly (see *EMAC to PHY Interface Bridges* on page 793 for more details).

EMAC uses independent receive and transmit FIFOs. Programmable FIFO thresholds minimize overflows and underruns, and can launch integrated IEEE 802.3x pause packets for flow control.

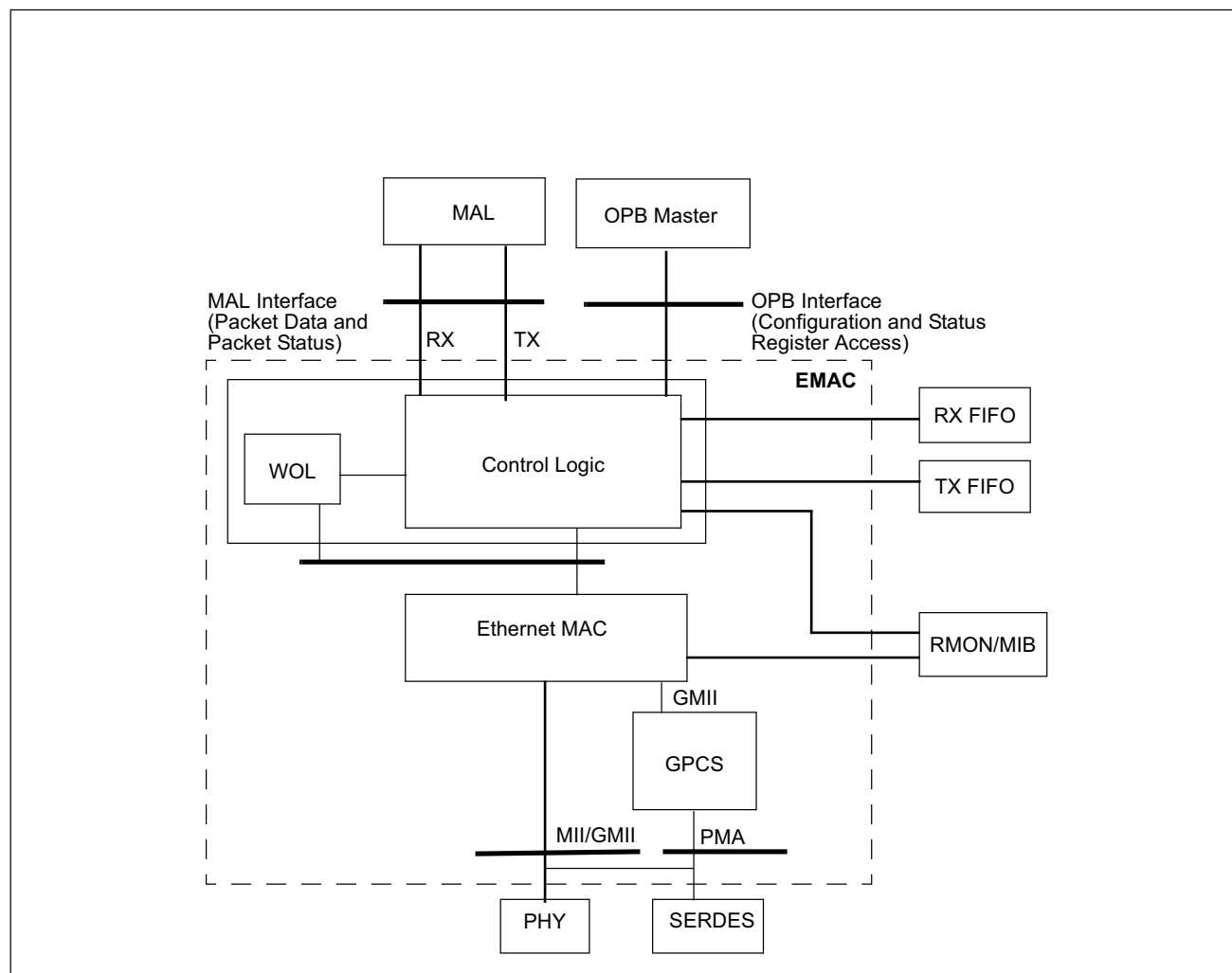
As part of the remote monitoring (RMON) and management information base (MIB) defined in IEEE 802.3z, the EMAC contains registers that count the number of octets transmitted and received.

EMAC supports jumbo packets with no VLAN tag and with VLAN tag, for up to 9018 and 9022 bytes respectively. To enable jumbo packet support, the jumbo enable bit 20 of EMAC mode register 1 must be set as follows: EMACx_MR1[JPSM=1]. In addition to enabling the EMAC to transmit and receive jumbo packets, the MAL driver must support packets received by multiple descriptor/buffers. The MAL implemented in PPC440GX supports a maximum of 4080 bytes.

In order to receive packets greater than the maximum buffer size, a packet greater than 4080 would have to be transmitted or received using multiple descriptors. By using the first and last bits in the descriptor field, software is able to determine the beginning, middle, and end of a packet. For example, if a 9K packet is received and the receive buffer size (RCBSx) is set to the maximum - 0x000000FF, the first descriptor will have its first bit set, and the second will not have either the first or last bit set, and the third descriptor will have the last bit set. The device driver will be required to move these multiple buffers into contiguous memory before passing the packet to the IP stack or use an interface that supports sending a link list of buffers to the IP stack.

Figure 24-1 illustrates an EMAC in a typical Ethernet application.

Figure 24-1. EMAC in a Typical Ethernet Application



Note: Each EMAC is connected to an OPB, to which the OPB master and MAL are also connected. *PPC440GX Block Diagram* on page 52 describes how all four EMACs are connected. EMAC transmit channel operates independently from the receive channel.

Preliminary User's Manual

24.1 EMAC Features

EMAC features:

- Triple speed (10/100/1000 Mbps) CSMA/CD (half-duplex) and full-duplex Ethernet MAC complying with ANSI/IEEE Std. 802.3, 802.3z and IEEE 802.3u supplement.
- Two 128-bit extended OPB (EOPB) interfaces to MAL with one receive and one transmit channel enable high throughput and performance which allows the interfaces to be connected to different MALs in one system via the same or separate MAL-EOPB buses.
- Dual GMII/PMA interfaces (contains a built-in GPCS logic). When configured to the Gigabit Ethernet (GE) mode with the PMA interface, the receive function operates at 62.5 MHz, (which is half of the GE frequency) thus saving power consumption.
- EMAC TX channel has a multiple status mechanism which allows EMAC to send the status/error word that is relevant to the last TX packet only after the next TX packet.
- Support for next page feature in the PCS auto-negotiation process.
- Automatic source address insertion or replacement for transmitted packets is a programmable option.
- Automatic stripping of frame padding bytes and frame check sequence (FCS) is a programmable option.

When padding bytes are stripped, the padding and FCS field are removed. FCS stripping removes only the FCS field.

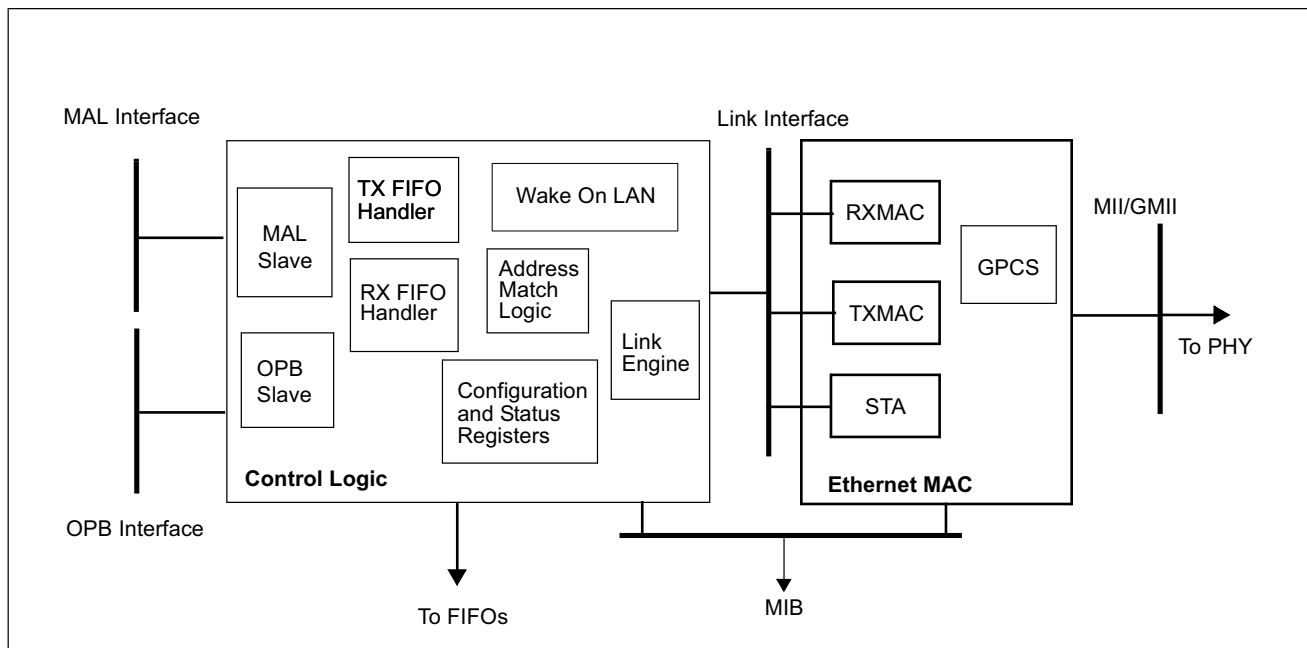
- FCS control for transmit/receive packets.
- Access to registers with support for burst processing.
- MAL for packet moving having one-cycle MAL slave latency.
- Independent, large (2 KB) transmit and up to (16 KB) receive FIFOs with programmable thresholds to minimize overruns and underruns and parity bits indication to the FIFOs and LPRA entries.
- Multiple packet handling in transmit and receive FIFOs.
- Unicast, multicast, broadcast, and promiscuous address filtering capabilities.
- Two 64-bit hash filters for unicast and multicast packets.
- Automatic retransmission of collided packets.
- Rejection of runt packets before providing them to MAL.
- MII for connection to a variety of PHY layer devices (used when 10/100 Mbps mode is chosen).
- Programmable interface for connecting to a Gigabit PHY. A gigabit media independent interface (GMII) or a PMA interface can be used for connection to the attached PHY layer device (used when 1000 Mbps mode is chosen).
- Programmable inter-packet gap to enable tuning for better system performance.
- Compliance with IEEE 802.3x standard packet-based flow control, including self-assembled control pause packet transmitting.
- Support for VLAN tag ID in compliance with IEEE Draft 802.3ac/D1.0 standard.
- VLAN tag insertion or replacement for transmit packets is a programmable option.
- Wake on LAN (WOL) handling.
- Programmable internal and external loop-back capabilities.
- Extensive error/status vector generation for each processed packet.
- Power management using a clock and power management (CPM) unit.

- Internal PCS capabilities with an internal PCS for use in the 1000 Mbps with a PHY PMA interface

24.2 EMAC Operation

The EMAC hardware components and its internal structure are illustrated in *Figure 24-2*. The EMAC is connected to a physical layer device via the MII (for 10/100 Mbps Ethernet), the GMII (for 1000 Mbps Ethernet using an external GPCS) or via the PMA interface (for 1000 MBps Ethernet using the EMAC internal GPCS). The active PHY interface (MII, GMII or PMA) is defined via the EMAC configuration.

Figure 24-2. Internal EMAC Structure



Preliminary User's Manual

The control logic sub-block implements the following functions:

- OPB slave device
- MAL slave device
- FIFO management logic
- Ethernet address and pause packet match logic (address match logic)
- Register file for FIFOs and Ethernet MAC handler management
- Logic for support of WOL technology
- Link engine

The Ethernet MAC sub-block implements the following functions:

- Transmit MAC Handler (TXMAC)
- Receive MAC Handler (RXMAC)
- MII/GMII management function unit (STA)
- Internal GPCS unit

The above functions are described in the following sections.

24.2.1 MAL Slave Logic

The MAL slave (MALS) logic controls MAL transactions. MALS transfers TX and RX data between MAL and the OPB on one side, and the EMAC FIFO handlers on the other side. MALS is a dedicated MAL slave.

24.2.2 OPB Slave Logic

The OPB slave (OPBS) logic controls all OPB transactions between the processor core and the EMAC configuration and status registers.

24.2.3 FIFO Management Logic

The FIFO management logic is used for data interchange handling with the attached receive (RX FIFO) and transmit (TX FIFO) modules.

24.2.4 Ethernet Address Match Logic

Address match logic checks the destination address of received packets against a set of predefined addresses specified by the current address filtering mode. EMAC contains one unicast (individual address) register, two hash tables for filtering individual and group address, and logic for detecting broadcast address (all ones). EMAC supports promiscuous mode and multicast promiscuous mode.

This logic also checks the destination address of the incoming packet against a special multicast address used for control (pause) packet recognition.

All checks for address matching are performed only after the entire destination address field is received (except for promiscuous and multicast promiscuous modes).

24.2.5 Configuration and Status Registers

Configuration and status registers define the EMAC configuration and reflect error/status of recent transmitted or received packets.

24.2.6 Wake On LAN Logic

EMAC supports Wake On LAN (WOL) technology, an industry standard described in the *Wired for Management (WFM)* specification. This technology allows a sleeping or powered-off network node to be awakened with a special packet called a Magic Packet. In the PPC440GX Embedded Processor, with WOL mode enabled, the EMAC discards all incoming packets and does not request data from the MAL for transmission. When a magic packet is detected, an interrupt is generated on UIC1 (bit 29 for EMAC0 or bit 31 for EMAC1) and UIC2 (bit 1 for EMAC2 or bit 3 for EMAC3).

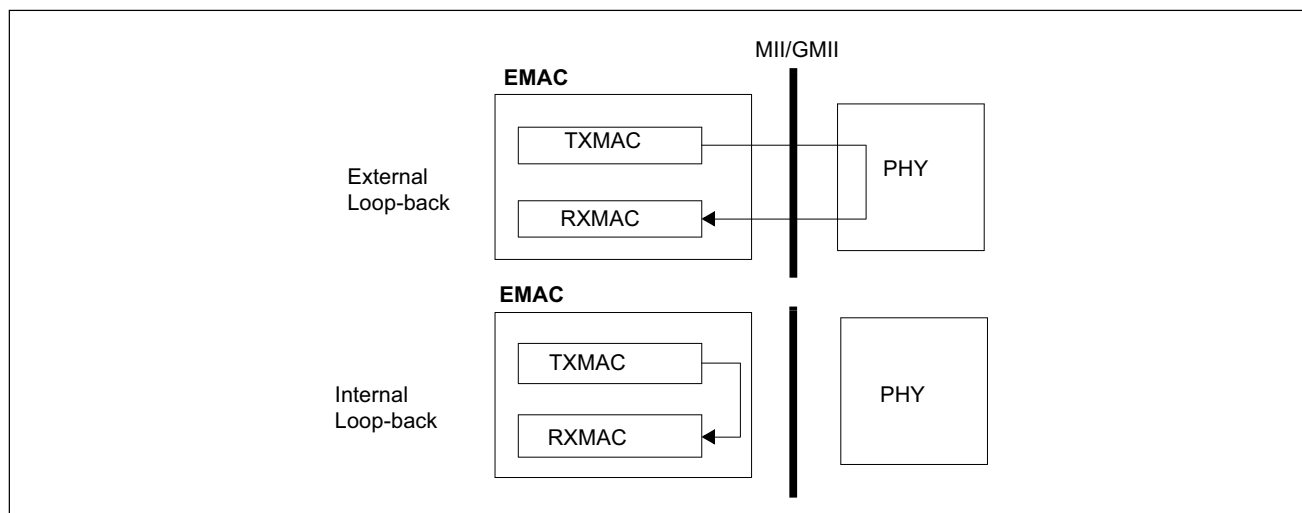
24.2.7 Ethernet MAC

The Ethernet MAC logic supports media independent interface (MII), gigabit media independent interface (GMII) and PMA interface.

24.2.8 EMAC Loopback Modes

EMAC supports the external and internal loop-back modes illustrated in *Figure 24-3*.

Figure 24-3. EMAC Loopback Modes



External loop-back mode uses the PHY device. To EMAC, external loop-back is identical to full-duplex operation. Configuring EMAC to operate in a full-duplex mode enables external loop-back. EMAC does not need an external loopback configuration signal. In external loop-back with the internal GPCS, the GPCS loop-back register bit is set via EMACx_STA register.

In internal loopback mode, data from the EMAC transmit channel is routed to the EMAC receive channel. The loop-back mode functions correctly with or without a connected PHY. In internal loopback mode, EMAC does not activate (monitor) any MII/GMII signals. Transmit channel signals are buffered internally to the receive channel. However, if internal loopback is used without a PHY, the EMAC transmit and receive clocks must be provided by another means (enabling ethernet clock select bit (SDR0_MFR[ECS] when loopback is used without a PHY. See *Miscellaneous Function Register (SDR0_MFR)* on page 300). In internal loopback mode, the EMAC transmit clock and receive clock must be sourced from a single clock. Also in internal loop-back, the internal loop-back enable bit 1 in Mode Register 1 is set.

Preliminary User's Manual

24.3 EMAC Transmit Operation

The transmit part of EMAC handles packet transmission from the MAL device to the MII/GMII or PMA interface. At the end of a transmission process, EMAC provides a status/error word which allows monitoring the transmission operation.

24.3.1 Arbitration Between TX Channels

EMAC's TX channel can be configured to work in either of two ways: single packet mode or multiple packet mode.

In single packet mode, EMACx_MR1[TR] = 0. The channel requests one packet from MAL and resets EMACx_TMR0[GNP=1] when it receives a MAL_TX_STATUS_DONE indication. The channel asks for service again only after EMACx_TMR0[GNP = 1] (set by the device driver).

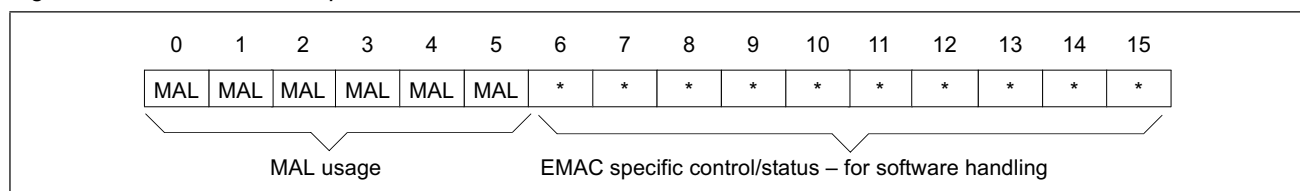
In multiple packet mode, EMACx_MR1[TR] = 1. After the channel finishes transferring a packet and receives a MAL_TX_STATUS_DONE indication, the channel asks MAL for the next packet if there is enough room in the FIFO. The channel continues to request more packets until one of the following events occur:

- The channel receives notification from MAL that the next buffer descriptor is not marked ready for transmission. When this occurs, the channel sets EMACx_TMR0[GNP] = 0, as appropriate, and waits for software to reactivate the channel by setting EMACx_TMR0[GNP] = 1.
- A transmit error or signal quality error (SQE) occurs and the corresponding interrupt is not masked in the EMACx_ISR. After such an error, the channel sets EMACx_TMR0[GNP] = 0, and sets EMACx_ISR[DB] = 1 (the EMACx_ISR field that is set depends on which channel is active) and the corresponding EMACx_ISR error. The channel does not request service again until EMACx_TMR0[GNP] = 1 and EMACx_ISR[DB] = 0.

24.3.1.1 MAL TX Descriptor Control/Status Field

For each transmitted packet, MAL uses the descriptor control/status field of the buffer descriptor to provide an EMAC with control information (write), and to obtain packet status from the EMAC after transmission is complete (read). Software writes the control bits in the buffer descriptor before packet transmission, and reads the status bits from the buffer descriptor after packet transmission has completed. See *MAL Operation* on page 756 for more information on the buffer descriptor structure.

Figure 24-4. MAL TX Descriptor Control/Status Field



Bits	Bit Name	Bit Description	Mode
0:5	MAL Usage	See <i>TX Buffer Descriptor MAL Control</i> on page 764	R
TX Control Information (Write Access)			
6	Generate FCS	0 FCS is not generated by EMAC. 1 EMAC calculates and adds the FCS field to the packet to be transmitted.	W
7	Generate padding	0 Padding is not generated by EMAC. 1 EMAC adds the padding field to the packet to be transmitted (only when Generate FCS is also set).	W

Preliminary User's Manual

Bits	Bit Name	Bit Description	Mode
8	Insert source address	0 EMAC will not insert source address. 1 EMAC inserts the source address field into the packet to be transmitted using the content of the Individual Address High (EMACx_IAHR) and Individual Address Low (EMACx_IALR) Registers.	W
9	Replace source address	0 EMAC will not replace source address. 1 EMAC replaces the source address field in the packet to be transmitted using the content of the Individual Address High (EMACx_IAHR) and Individual Address Low (EMACx_IALR) Registers.	W
10	Insert VLAN Tag	0 EMAC will not insert a VLAN tag. 1 EMAC inserts the VLAN Tag field into the packet to be transmitted using the content of the VLAN TPID register (EMACx_VTPID).	W
11	Replace VLAN Tag	0 EMAC will not replace the VLAN tag. 1 EMAC replaces the VLAN Tag field in the packet to be transmitted using the content of the VLAN TPID register (EMACx_VTPID).	W
TX Status Information (Read Access)			
6	Bad FCS on transmitted frame	0 FCS was correct in the transmitted packet. 1 Indicates that a bad FCS was found in the transmitted packet (this bit is also set when parity error bit 5 is set).	R
7	Reserved	Always zero. (However this bit is used by TCP/IP hardware to report TX error. See <i>TCP/IP Acceleration Hardware</i> on page 879.	R
8	Loss of carrier sense	0 No loss of carrier. 1 During the transmission of a frame, the PHY_CRS input was de-asserted after it previously was asserted, or it was not asserted at all. Applicable only in half duplex mode.	R
9	Excessive deferral	0 No excessive deferral. 1 Indicates that the current frame has been deferred for an excessive period of time. Applicable only in half duplex mode. The value of this period in bit times is calculated in the following ways: For 10/100 Mbps operation it is: 2 x (maxFrameSize x 8) bit times. For 1000 Mbps operation it is: 2 x (burstlimit+maxFrameSize x 8+headerSize) bit times.	R
10	Excessive collisions	0 Less than 16 collisions. 1 Indicates that the current frame transmission had ended with a collision on the 16th consecutive attempt. Applicable only in half-duplex mode.	R
11	Late collision	0 No late collision. 1 Frame collided outside of the collision window. Applicable only in half-duplex mode.	R
12	Multiple collision	0 More than 1 but less than 16 collisions did not occur. 1 Transmitted frame collided more than once but less than 16 times. Applicable only in half-duplex mode.	R
13	Single collision	0 Single collision did not occur. 1 Activates if transmitted frame collided once. Applicable only in half-duplex mode.	R
14	Underrun	0 Underrun did not occur. 1 Frame transmission was aborted because of underrun; data from the Transmit FIFO was not valid in time to allow continuous data transmission on the MII/GMII interface.	R
15	SQE	0 Signal Quality Error did not occur. 1 Signal Quality Error test failed during packet transmission. Applicable only in half -duplex mode during 10 Mbps operation.	R

Preliminary User's Manual

24.3.1.2 Early Packet Termination during Transmit

EMAC can initiate early packet termination during transmit, terminating packet transmission before MAL finishes transferring all packet data from memory to the EMAC transmit FIFO. Early packet termination is typically used when error conditions force the EMAC to abort a transmission.

EMAC performs early termination on the MAL interface if any of the following conditions occur:

- Underrun in the transmit FIFO.
- Excessive collisions.
- Excessive deferral.
- Late collision.

24.3.1.3 Empty Packets

EMAC treats empty packets as if a normal packet had been written, but does not write data to the transmit FIFO. A status word of all 0s is returned after an empty packet. EMAC expects that for quadword-aligned packets, MAL activates the related word transfer indication during the last data transfer, rather than providing an empty packet indication.

24.3.1.4 Automatic Retransmission of Colliding Packets

EMAC automatically retransmits packets that collide on the MII/GMII interface. The transmit FIFO always preserves the first 64 (in 10/100 Mbps) or 512 (in Gigabit Ethernet media) bytes of a packet until it receives an indication that the collision window has passed. Otherwise, if a collision was detected within the collision window, the packet is retransmitted without a new request from MAL.

24.3.1.5 Inter-Packet Gap (IPG) Tuning

EMAC supports user-programmable IPG length using the EMACx_IPGVR register, which contains one-third of the IPG value. Changing the contents of EMACx_IPGVR enables the user to adjust the fairness or aggressiveness of EMAC on the medium. Programming a lower number (but not less than four) causes EMAC becomes more aggressive on the media. This can result in EMAC capturing the network by forcing less aggressive nodes to defer. Programming a larger number of bit times causes EMAC to become less aggressive on the network; it might defer more often than normal. EMAC performance might decrease as the IPG period is increased from the default value, but the resulting behavior can improve media performance by reducing the occurrence of collisions.

24.3.1.6 Full-Duplex Operation

Full-duplex operation allows simultaneous transmit and receive activity on the MII/GMII interface. Software can set EMACx_MR1[FDE] = 1 to enable full-duplex mode. During full-duplex operation, the following changes occur in EMAC functionality.

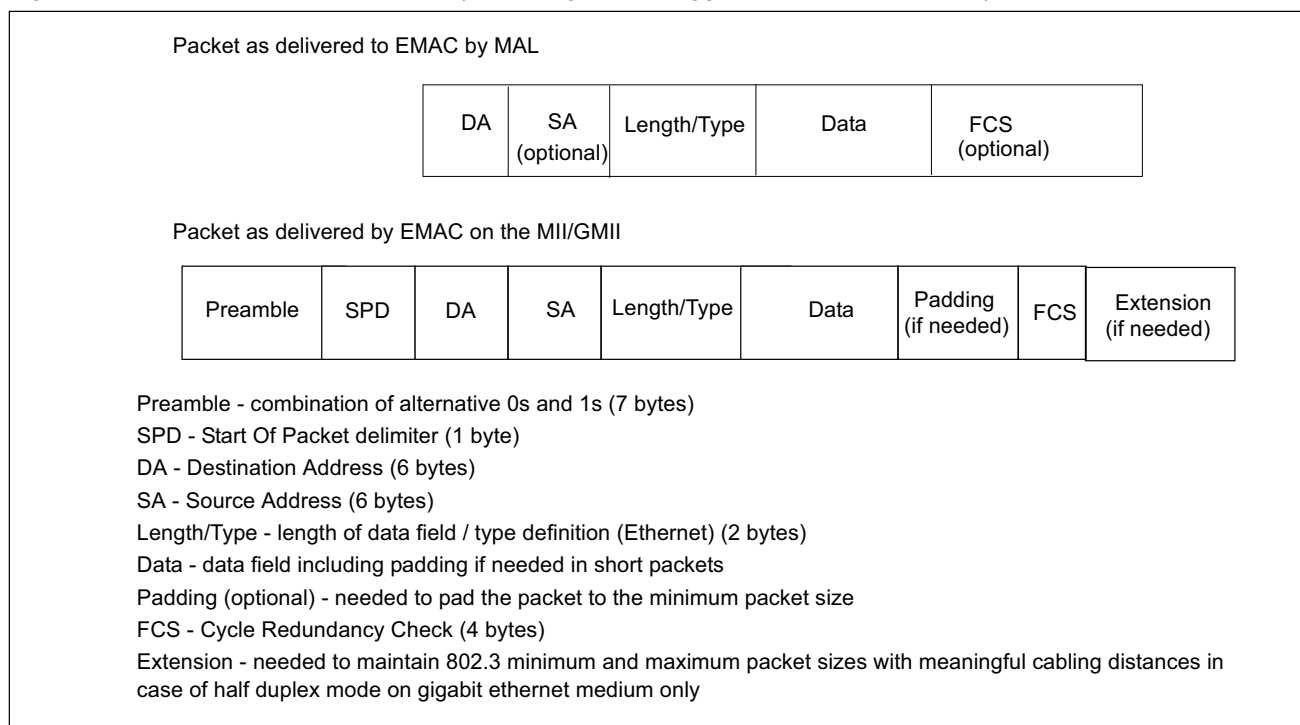
- Transmission is not deferred while receive is active.
- The IPG counter, which controls transmit deferral during the IPG between back-to-back transmits, is started when transmit activity for the first packet ends, instead of when transmit and carrier activity end.
- SQE test is not performed.
- Collision indication is ignored.

24.3.1.7 Packet Content Configuration Options

EMAC can modify the content of the packet coming from MAL before issuing it to the MII/GMII interface.

Figure 24-5 illustrates possible changes in the transmit packet format.

Figure 24-5. Transmit Packet Structure (Excluding VLAN Tagged and Control Packets)



The following options, unless mutually exclusive, can be set for each packet, and can be provided as a part of command write transactions from MAL (see *MAL TX Descriptor Control/Status Field* on page 821).

- Automatic data padding for short transmit packets

EMAC pads the transmit packet with extra bytes between the data and the FCS field to reach a total length of 64 bytes (including FCS). This feature is supported only when the packet coming from the transmit FIFO does not contain the FCS. Automatic padding enables software to avoid sending padding as a part of the packet data field, and, therefore, reduces the amount of data transferred on the system bus during the short packet transmission.

- Source address insertion

EMAC adds the source address (SA) field to the transmitted packet. EMAC uses the contents of the Individual Address High (EMACx_IAHR) and Individual Address Low (EMACx_IALR) registers for the source address value. *This option is mutually exclusive with source address replacement.*

- Source address replacement

EMAC replaces the source address received from the transmit FIFO with the contents of EMACx_LSAH and EMACx_LSAL. *This option is mutually exclusive with source address insertion.*

- Add four FCS bytes

EMAC calculates the FCS for the transmitted packet. The FCS is appended to data coming from the Transmit FIFO or padding bytes (if added).

Preliminary User's Manual

- VLAN Tag insertion

EMAC adds the content of the VLAN Tag field to the transmitted packet (see *VLAN Support* on page 833). EMAC uses the content of the VLAN TPID (EMACx_VTPID) and VLAN TCI (EMACx_VTCI) registers for the VLAN Tag value. This feature is supported only when the packet from the transmit FIFO does not contain an FCS. *This option is mutually exclusive with VLAN tag replacement.*

- VLAN Tag replacement.

EMAC replaces the content of the VLAN Tag field in the transmitted packet. EMAC uses the contents of EMACx_VTPID and EMACx_VTCI for the VLAN Tag value. This feature is supported only when the packet from the transmit FIFO does not contain an FCS. *This option is mutually exclusive with VLAN tag insertion.*

Table 24-1 summarizes the possible options for adding FCS and SA to the transmitted packet.

Table 24-1. FCS/SA Enable - Possible Configurations

Configuration Options			EMAC Action		
Generate FCS	Insert SA	Replace SA	Add FCS	Add SA	Replace SA
0	Don't care	Don't care	N	N	N
1	0	0	Y	N	N
1	0	1	Y	N	Y
1	1	0	Y	Y	N

Table 24-2 summarizes the possible options for adding FCS and padding to the transmitted packet.

Table 24-2. FCS/Pad Enable - Possible Configurations

Configuration Options		EMAC Action	
Generate FCS	Generate Pad	Add FCS	Add Pad
0	Don't care	N	N
1	0	Y	N
1	1	Y	Y

Table 24-3 summarizes the possible options for adding FCS and VLAN Tag to the transmitted packet.

Table 24-3. FCS/VLAN Tag Enable - Possible Configurations

Configuration Options			EMAC Action		
Generate FCS	Insert VLAN Tag	Replace VLAN Tag	Add FCS	Insert VLAN Tag	Replace VLAN Tag
0	Don't care	Don't care	N	N	N
1	0	0	Y	N	N
1	0	1	Y	N	Y
1	1	0	Y	Y	N

24.4 EMAC Receive Operation

The receive part of EMAC is responsible for receiving packets coming from the physical layer (PHY) device (via the MII/GMII) and forwarding them to the receive channel of the attached MAL. At the end of the reception process, EMAC provides a status/error word that enables software to monitor the receive operation.

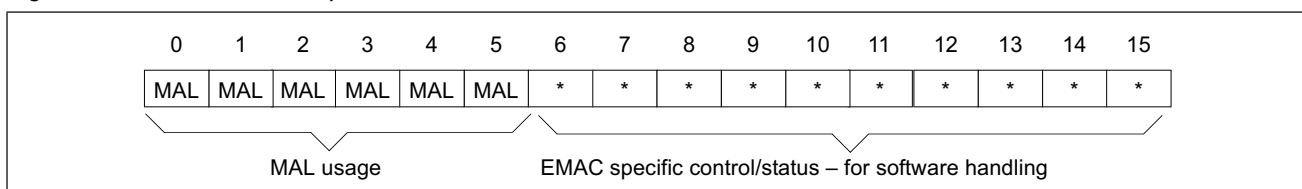
24.4.1 EMAC – MAL RX Packet Transfer Flow

EMAC initiates request for service from MAL when the number of occupied entries in the receive FIFO reaches the low water mark specified in EMACx_RWMR[RLWM].

24.4.2 MAL RX Descriptor Status

For each packet that is received, MAL obtains status from EMAC after reception is complete, and writes this information into the buffer descriptor status/control field. Software uses this information to monitor the status of received packets. See *MAL Operation* on page 756 for more information on the buffer descriptor structure.

Figure 24-6. MAL RX Descriptor Control/Status Field



Bits	Bit Name	Bit Description	Mode
0:5	MAL Usage	See <i>RX Buffer Descriptor MAL Control options</i> on page 765.	R
RX Status Information (Read Access)			
6	Overrun Error	0 No overrun error. 1 EMAC detected an overrun error. An overrun error occurs if the flow of received data to the RX FIFO is corrupted because of insufficient empty space.	R
7	Pause Packet	0 Received packet is not a control pause packet. 1 Received packet is a control pause packet.	R
8	Bad Packet	0 No packet errors. 1 Early termination caused by packet error.	R
9	Runt Packet	0 Duration of PHY_RX_DV signal OK. 1 Duration of PHY_RX_DV signal greater than ShortEventMax Time constant and less than collision windows.	R
10	Short Event	0 Duration of PHY_RX_DV signal OK. 1 Duration of PHY_RX_DV signal was less than ShortEventMaxTime constant	R
11	Alignment Error	0 Received packet length OK. 1 Received packet length not an integral number of octets.	R
12	Bad FCS	0 FCS OK. 1 The FCS value does not match the FCS value calculated by EMAC.	R

Preliminary User's Manual

Bits	Bit Name	Bit Description	Mode
13	Packet Too Long	0 Received packet length OK. 1 Received packet length exceeds maximum packet length. 1518 octets for standard packet (is checked only when the length/type field of the transmitted packet contained length value and Jumbo support is disabled) 1522 octets for VLAN tagged packet (is checked only when the length/type field of the transmitted packet contained length value and Jumbo support is disabled) 9018 octets for jumbo packet (with no VLAN support) 9022 octets for VLAN tagged Jumbo packet Note: The data following the maximum packet length is not transferred to MAL	R
14	Out of Range Error	0 Received packet length field value OK. 1 Received packet length field value greater than maximum allowed LLC data size. The maximum allowed logical link control (LLC) data size is greater than 1500 and less than 1536.	R
15	In Range Error	Refer to <i>Table 24-4</i> for a description of conditions for activating this status bit.	R

Table 24-4. In Range Length Error Behavior for Various Packet Lengths

Programmed Length (Bytes)	Actual Length	EMAC Action
Less than 46 (42 if VLAN Tagged packet)	Differs from 46 (42 in case of VLAN Tagged packet)	In range length error is activated
Less than 46 (42 if VLAN Tagged packet)	46 (42 in case of VLAN Tagged packet)	In range length error is not activated
Greater than or equal to 46 (42 if VLAN Tagged packet) and less than or equal to 1500	Equals the length field value	In range length error is not activated
Greater than or equal to 46 (42 if VLAN Tagged packet) and less than or equal to 1500	Differs from the length field value	In range length error is activated

24.4.3 Early Packet Termination during Receive

Early packet termination occurs when packet reception is aborted by EMAC before the packet data transfer to MAL is completed.

EMAC performs early termination in the following cases.

- An overrun occurs in the receive FIFO
- Packet is too long and the Receive Oversize Packet option is not enabled (EMACx_RMR[ROP] = 0)

24.4.4 Discarding Packets During Receive

Received packets can be discarded if certain error conditions are detected. EMAC behavior depends on whether the packet to be discarded is already being output to MAL. If the packet containing the error is not being provided to MAL when the discard condition is detected, the packet is flushed from the Receive FIFO. In this case, EMAC does not provide status information to MAL. If the packet containing the error is already being output to MAL, EMAC initiates an early packet termination procedure, as described in *Early Packet Termination during Receive* on page 827

Each receive discard condition can be individually controlled using appropriate settings, as described in *Receive Mode Register (EMACx_RMR)* on page 846.

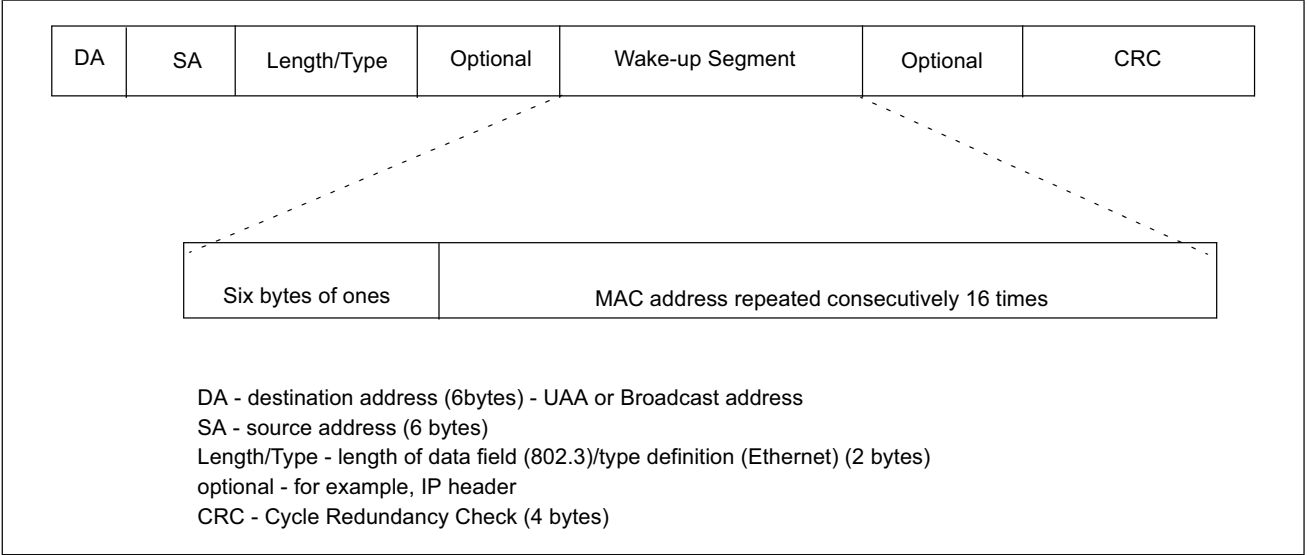
24.4.5 Wakeup On Lan (WOL) Support

WOL logic in EMAC supports WOL technology, an industry standard in the Wired for Management (WFM) Specification. WOL remotely awakens sleeping or powered-off nodes on a network. To wake up a node, a specific packet, called a *magic packet*, is sent to the network node. When so configured, EMAC monitors the incoming bit stream for magic packets.

The magic packet, also called a wake-up packet, contains a unique data field not normally expected in LAN traffic. When a WOL-enabled adapter on a powered-off client decodes this data field, a wake-up signal is generated. In the PPC440GX Embedded Processor, with WOL mode enabled, EMAC discards all incoming packets and does not request data from the MAL for transmission. When a magic packet is detected, an interrupt is generated on UIC1 (bit 29 for EMAC0 or bit 31 for EMAC1) and UIC2 (bit 1 for EMAC2 or bit 3 for EMAC3).

Figure 24-7 shows the wake-up packet format. The key to the wake-up packet is the MAC address of the target client, which is repeated 16 times. This pattern of 16 addresses in the data field is not expected to occur in any packet except the wake-up packet.

Figure 24-7. Wake-Up Packet Format



The destination address can be a specific address, called the universally administered address (UAA), or a broadcast address. If the destination address is a UAA, the wake-up packet is sent only to the client at that address. However, because the client is powered off and is no longer transmitting, some protocols remove the client MAC address from routing tables and internal caches at other nodes. In this case, wake-up packets addressed to a target client are discarded because nodes and routers do not know where to send them. The solution to this problem is to use a broadcast address. A directed broadcast has a valid network address and a broadcast host address. Network routers and nodes forward directed broadcasts to the appropriate network, where it is seen as a MAC-level broadcast and detected by the powered-off client.

Preliminary User's Manual

24.4.5.1 EMAC WOL Support

EMAC enters WOL mode when EMACx_MR0[WKE] = 1.

WOL mode should only be changed while EMACx_MR0[RXI] = 1 and EMACx_MR0[RXE] = 0. After EMACx_MR0[WKE] = 1, EMACx_MR0[RXE] can be set to 1.

A reset (soft or hard) should be issued before programming EMAC to WOL mode. In WOL mode, EMAC does not propagate any received packets to MAL. Also, EMAC transmit channels do not request data from MAL.

24.5 Flow Control

For efficient system performance, each EMAC implements full-duplex flow control by handling specific MAC control packets contained in the pause opcode. EMAC supports flow control as defined in the IEEE 802.3x-1997 standard.

24.5.1 MAC Control Packet

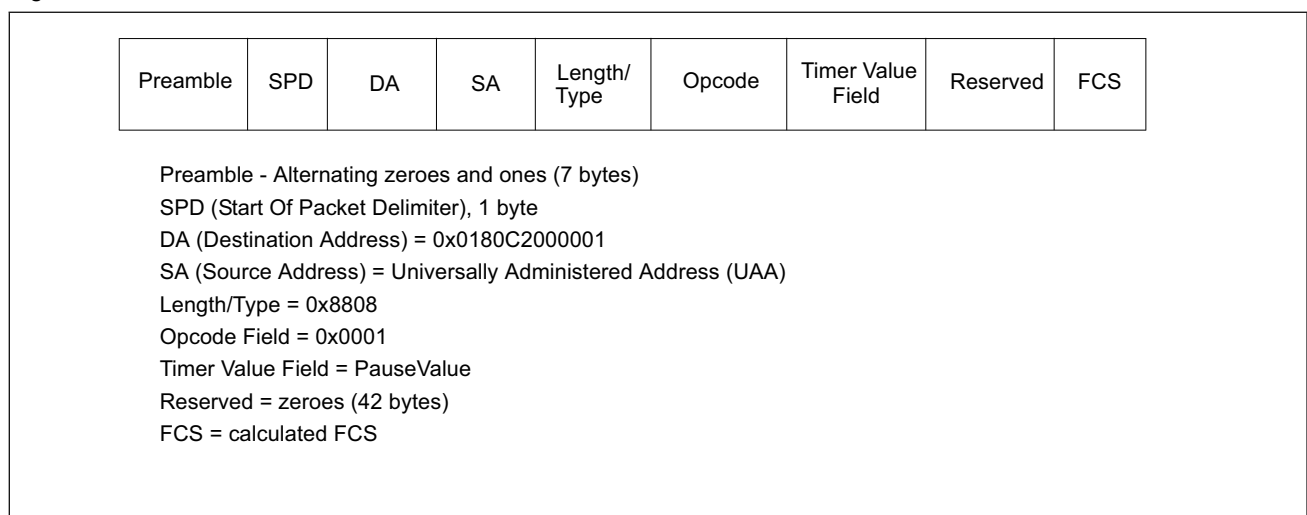
The flow control mechanism enables receive FIFO control logic to automatically notify the node transmitting packets to suspend transmission for a defined period of time. The pause control packet has a fixed length defined as follows:

MinFrameSize – 32 bits (60 bytes)

MAC control packets have a unique value of 0x8808 in the length/type field, and share the same packet format as normal Ethernet packets, except that the data field consists of an opcode field and a parameter field. The opcode field contains an opcode command and the parameter field contains a value associated with the opcode command (0x0001). The only opcode command defined by IEEE 802.3x is the pause opcode; the parameter field for the pause opcode defines the pause time. MAC control packets containing a pause opcode, also called pause packets, can have a destination address equal to a reserved multicast address, or can be the address of the receive station itself. The reserved multicast address is 0x0180C2000001.

Figure 24-8 illustrates the control packet format.

Figure 24-8. Control Packet Format



The timer value field contains the value of the delay interval in resolution of *pause_quanta*, defined in IEEE 802.3x as follows: "MAC Control Parameter[s] (*pause_time*) is a 2-octet, unsigned integer containing the length of time for which the receiving station is requested to inhibit data packet transmission. The field is transmitted most-significant octet first, and least-significant octet second. The *pause_time* is measured in units of *pause_quanta*, equal to 512 bit times. The range of possible *pause_time* is 0 to 65535 *pause_quanta*."

24.5.2 Control Packet Transmission

Two options initiate the pause packet transmission from EMAC. The transmitted pause packet forces the node, with the destination address specified, to temporarily suspend the transmission of packets to EMAC.

- Software initiated

The packet transferred to EMAC by MAL for transmission is a pause packet created by software. EMAC transmits this as a normal packet.

- Automatic flow control initiated

The EMAC integrated flow control mechanism detects the need for and then transmits a control (pause) packet automatically. When building the control packet, EMAC obtains the SA (source address) field from EMACx_IAHR and EMACx_IALR, and the timer value from the Pause Timer Register (EMACx_PTR[TVF]). The contents of the other fields in the packet are shown in *Figure 24-8*.

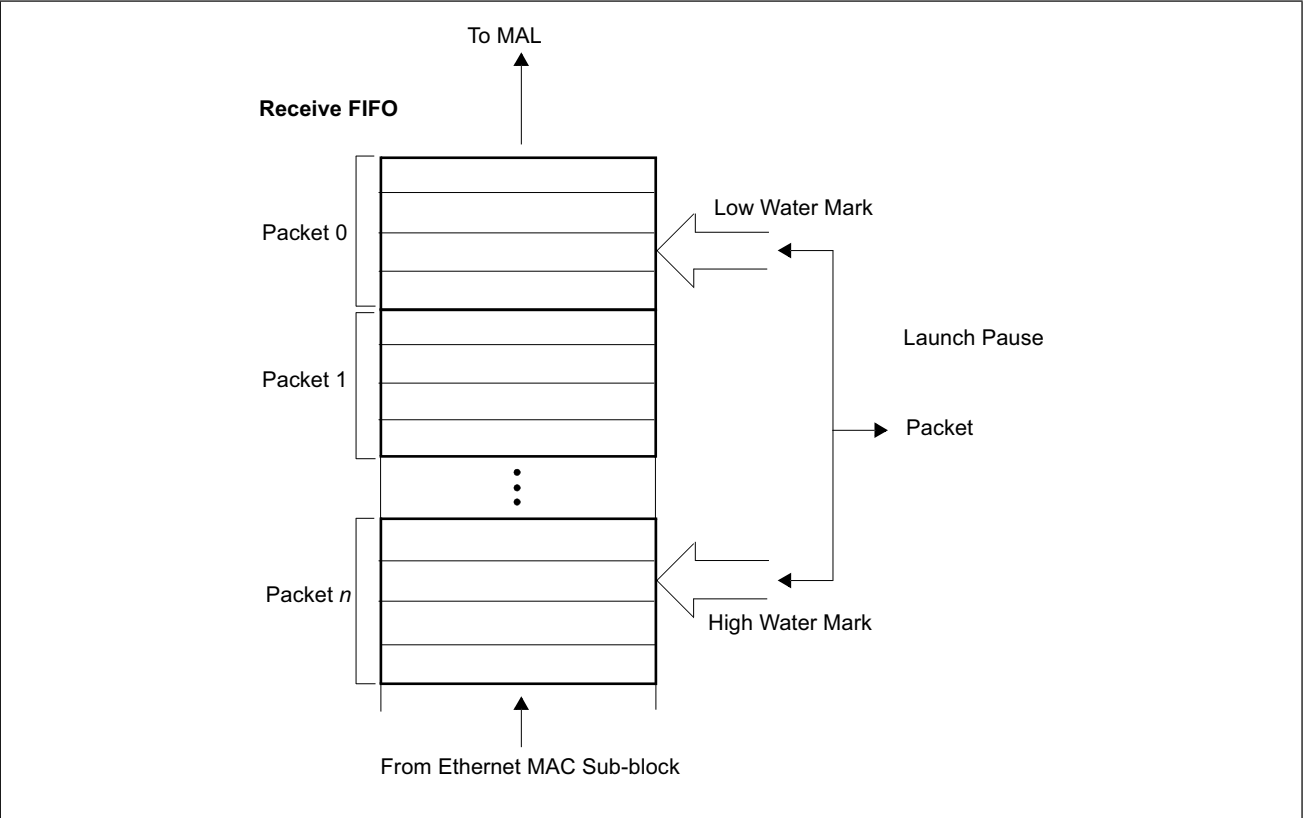
24.5.3 Integrated Flow Control

To enable integrated flow control in full-duplex mode, set EMACx_MR1[EIFC] = 1. When the receive FIFO reaches a predefined threshold level (called a high water mark and specified by EMACx_RWMR[RHWM]), an internal request for control (pause) packet transmission is activated. EMAC sends a control (pause) packet when a new packet enters the receive FIFO and the number of vacant entries in the Receive FIFO is less than the high water mark. When the receive FIFO reaches another predefined threshold level (the low water mark, specified by

Preliminary User's Manual

EMACx_RWMR[RLWM]), a new internal request for a pause packet transmission, with a pause timer value of 0, is activated. EMAC sends a pause packet, with a pause timer value of 0, only once, and only if a pause packet with a non-zero value in the pause timer was transmitted earlier.

Figure 24-9. Integrated Flow Control Mechanism

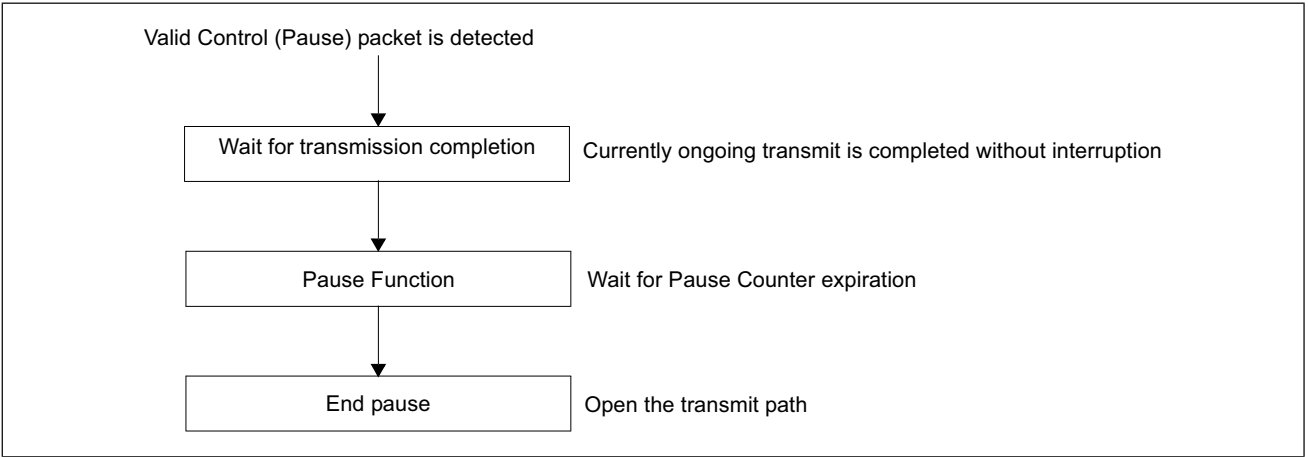


24.5.4 Control Packet Reception

In the receive path, the EMAC can be configured to respond to pause packets, or ignore them, as specified by EMACx_MR1[APP]. When response to pause packets is enabled and EMAC detects a valid MAC control packet with a pause opcode, the EMAC stores the value of the timer value field. The received packet is considered a valid control packet only if no error was detected during the packet reception. If, at the end of packet reception, the packet is considered valid, EMAC launches a pause operation state machine, as specified in the IEEE P802.3x standard. Figure 24-10 illustrates the pause operation state machine.

If a control (pause) packet is received while another packet is transmitted, the ongoing transmission process is completed and the transmitter is paused. If other packets are in the transmit FIFO, their transmission is delayed until the pause timer expires. EMAC normally does not pass the MAC control packets to MAL unless EMACx_RMR[PPP] = 1.

Figure 24-10. Pause Operation State Machine



In the Pause Function state, EMAC decrements its internal pause timer, which was set to the timer value field of the received control packet.

Note 1: The transmission of control (pause) packets is not affected by the reception of a receive control (pause) packet. Received control (pause) packets inhibit only the transmission of regular packets from the Transmit FIFO.

Note 2: Receipt of a new valid control (pause) packet causes the pause timer of EMAC to be reloaded with the contents of the timer value field of the recently received packet, regardless of the current pause timer setting. This indicates new pause operations take precedence over earlier pause operations.

Preliminary User's Manual

24.6 VLAN Support

EMAC can handle VLAN tagged packets, as specified in IEEE Draft P802.3ac/D1.0a standard when EMACx_MR1[VLE] = 1.

A VLAN tagged frame is an extension of the standard MAC packet. The extension for VLAN tag support consists of a 4-octet VLAN tag inserted between the end of the source address and the beginning of the length/type fields of the MAC packet.

The VLAN tag consists of two fields:

- A 2-octet constant Type field value equal to the VLAN Tag Protocol Identifier (0x8100)
- A 2-octet field containing Tag Control Information (TCI)

The MAC client data and FCS fields of the basic MAC packet follow the VLAN tag. The length of the packet is extended by four octets by the VLAN tag (up to 1522 bytes and 9022 bytes for standard and jumbo frame respectively). The FCS is calculated over all fields from the destination address through the end of the MAC client data or pad (if present); that is, all fields except the preamble, SPD, and FCS.

Figure 24-11 illustrates the VLAN tagged frame format.

Figure 24-11. VLAN Tagged Packet Format

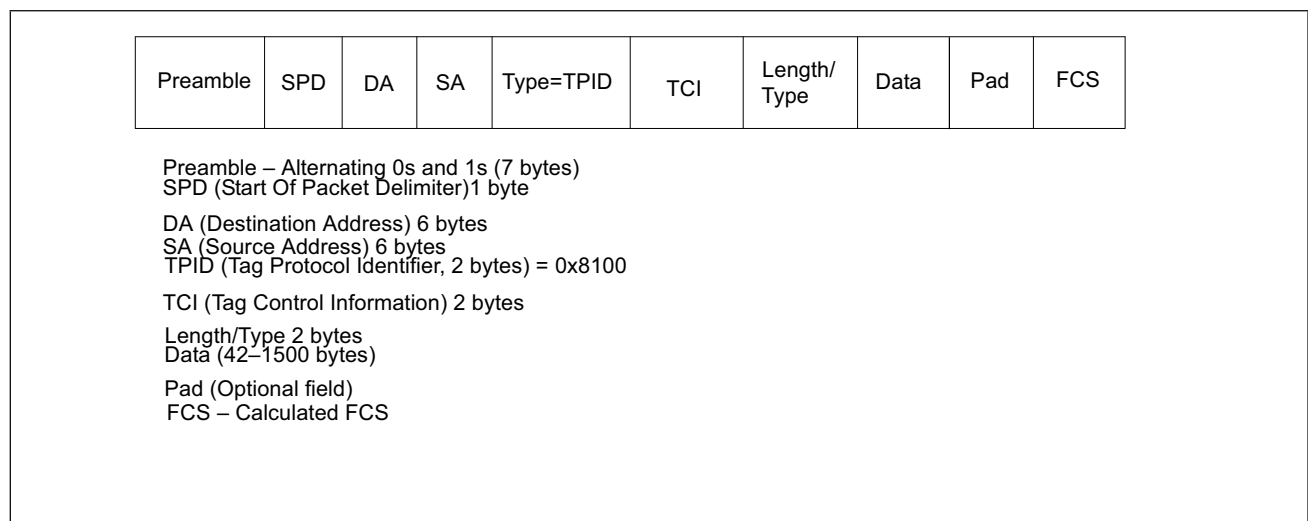
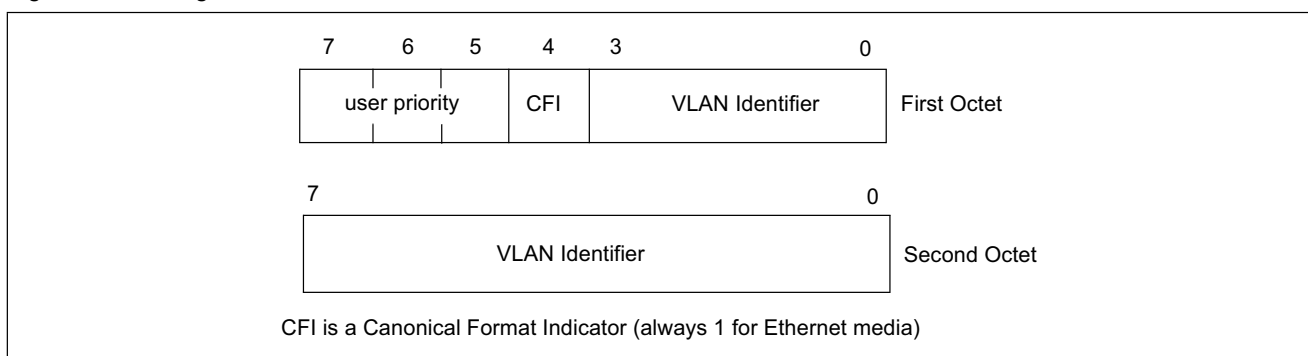


Figure 24-12 illustrates the structure of the TCI field.

Figure 24-12. Tag Control Information Field Structure



24.6.1 VLAN Tagged Packet Transmission

When `EMACx_MR1[VLE] = 1`, the following configuration options are available, depending on the content of the appropriate bits in the MAL control word (See *MAL TX Descriptor Control/Status Field* on page 821.)

- The Generate FCS bit (bit 6) is not set or both Insert VLAN Tag and Replace VLAN Tag bits (bits 10 and 11, respectively) are not set: EMAC transmits the packet without any changes
- Bit 6 is set and bit 10 is also set: EMAC will insert TPID and TCI for the transmitting packet using the content of `EMACx_VTPID` and `EMACx_VTCI`, respectively
- Bit 6 is set and bit 11 is also set: EMAC will replace TPID and TCI for the transmitting packet using the content of `EMACx_VTPID` and `EMACx_VTCI`, respectively

24.6.2 VLAN Tagged Packet Reception

If `EMACx_MR1[VLE] = 1`, the EMAC parses the VLAN Tag unique type/length in the incoming packet during the receive process. If the VLAN Tag is equal to the value stored in `EMACx_VTPID`, EMAC continues the receive process and allows the received packet to contain up to 1522 octets and 9022 when Jumbo packets support is enabled. Otherwise, the receive process is continued unless the length is greater than 1518 bytes 9018 when Jumbo packets support is enabled.

24.6.3 Address Match Mechanism

The address match (or filtering) mechanism is a hardware aid that reduces the average amount of CPU cycles required to determine whether an incoming packet should be accepted.

EMAC uses various address filters for incoming packets by using the following address recognition modes.

- Individual mode (also referred to as physical)
- Multicast mode (also referred to as group)
- Broadcast mode (an all-ones group address)
- Promiscuous mode
- Promiscuous multicast mode
- WOL mode

A flowchart for address recognition of received packets is shown in *Figure 24-13* on page 836. If the least significant bit (LSb) of the first byte of the destination address (DA) is 0, the packet is considered individual. If the first bit received is 1, the packet is considered multicast. When the DA field contains all 1s, the packet is broadcast, a special type of multicast.

24.6.3.1 Non-WOL Mode

When EMAC operates in single individual mode (`EMACx_RMR[IAE] = 1`), the DA of the received packet is compared to the physical address stored in `EMACx_IAHR` and `EMACx_IALR`.

When EMAC operates in multiple individual mode (`EMACx_RMR[MIAE] = 1`), EMAC performs a calculation on the contents of the DA field (logical address filter) to determine whether or not to accept the packet.

When EMAC operates in promiscuous mode (`EMACx_RMR[PME] = 1`), all properly formed packets are received, regardless of the content of the DA field.

When EMAC operates in multicast promiscuous mode (`EMACx_RMR[PMME] = 1`), all multicast packets are received, regardless of the content of the DA field.

Preliminary User's Manual

When EMAC operates in broadcast address mode (EMACx_RMR[BAE] = 1), EMAC performs an address compare on received packets with broadcast addresses.

When EMAC operates in multicast address mode (EMACx_RMR[MAE] = 1), EMAC performs a calculation on the contents of the DA field (logical address filter) as in multiple individual mode, in order to determine whether or not the packet should be accepted.

The logical address filter hardware implements a hash code searching technique commonly used by programmers. The hardware maps the DA of the incoming packet into one of 64 categories corresponding to 64 bits stored in the EMACx_IAHT1–EMACx_IAHT4 or EMACx_GAHT1–EMACx_GAHT4 registers. The hardware accepts or rejects the packet, depending on the state of the corresponding bit in the EMACx_IAHT1–EMACx_IAHT4 or EMACx_GAHT1–EMACx_GAHT4 registers corresponding to the selected category.

Figure 24-14 on page 837 shows the details of the hardware mapping algorithm. The example depicts multiple individual address mode, but with changes can be used for the multicast address mode.

If the most significant bit (MSb) of an incoming address is 0, the address is individual and is passed to the individual address filter. If the MSb of an incoming address is 1, the address is multicast and is passed to the multicast address filter. The individual/multicast address filter is a 64-bit mask composed of the EMACx_IAHT1–EMACx_IAHT4 or EMACx_GAHT1–EMACx_GAHT4 registers (each register contains 16 bits of a 64-bit mask). The incoming address is sent through the FCS circuit. After the 48 address bits have gone through the FCS circuit, the high-order 6 bits of the resulting FCS (32-bit CRC) are used to select a the 64-bit positions in the individual/multicast address filter. If the selected filter bit is 1, the address is accepted.

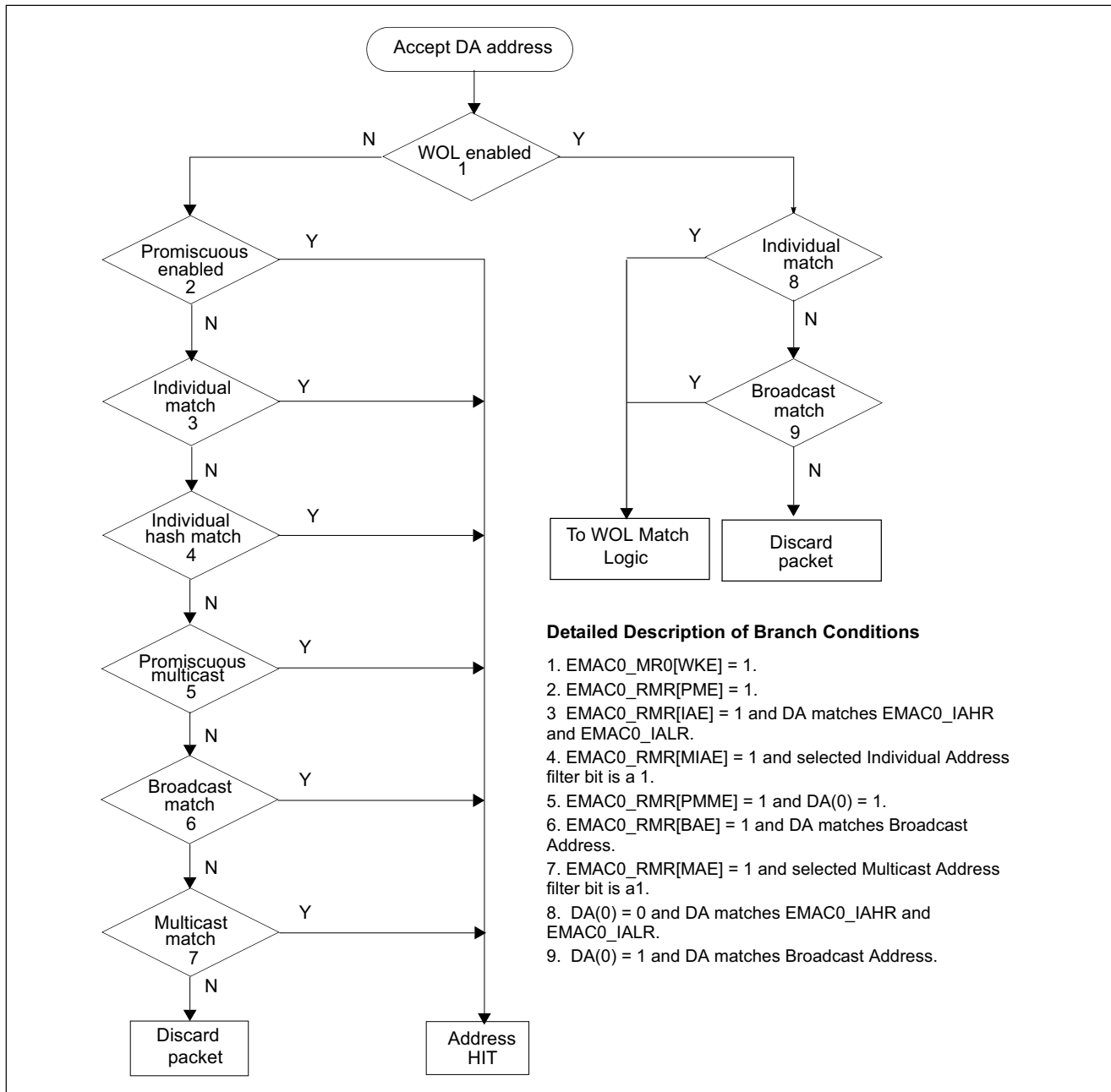
Note: The individual/multicast address filter ensures only that there is a possibility that the incoming packet belongs to this node. To determine if the packet belongs to the node, the incoming individual/multicast address propagated to the main memory is compared by software to the list of logical addresses to be accepted by this node.

For software, the task of mapping an individual/multicast address to one of 64 bit positions requires a program that uses the same CRC algorithm to calculate the hash.

24.6.3.2 WOL Mode

In WOL mode (EMACx_MR0[WKE] = 1), EMAC operates only with the broadcast or individual address in the destination address field.

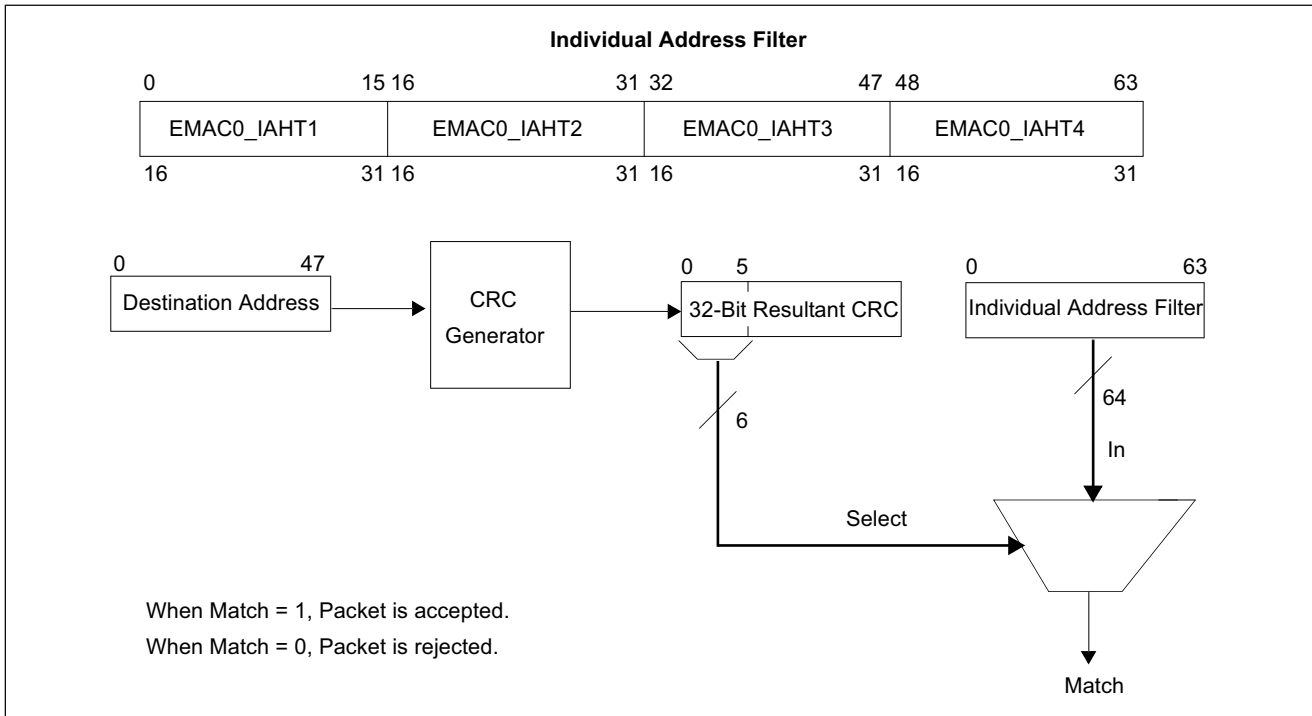
Figure 24-13. Receive Address Recognition Flowchart



Preliminary User's Manual

The example in *Figure 24-14* shows that the received individual address maps into category 24, and that bit 8 in EMACx_IAHT2 is set. The match indication is activated and the packet should be accepted.

Figure 24-14. Ethernet Address Filter Operation



24.7 EMAC Registers

This section describes the EMAC registers, which are listed in *Table 24-5* for EMAC0, *Table 24-6* for EMAC1, *Table 24-7* for EMAC2 and *Table 24-8* for EMAC3. In the register descriptions, the registers are prefixed "EMACx" to denote the identical register implementations in each EMAC. The EMAC registers are accessed using the OPB slave interface. Access to these memory-mapped I/O (MMIO) registers should be word-aligned.

Note: It is required to perform a soft reset of EMAC before initialization as shown in EMACx Mode Register 0 (EMACx_MR0[SRST]).

Table 24-5. EMAC0 Register Summary

Register	Address	Write Access	Power-on Reset Value	Access	Page
EMAC0_MR0	0x1 40000800	See description in <i>Scenario 1</i> on page 876	0xC0000000	R/W	844
EMAC0_MR1	0x1 40000804	Reset	0x00000000	R/W	842
EMAC0_TMR0	0x1 40000808	See description on Page 844	0x00000007	R/W	844
EMAC0_TMR1	0x1 4000080C	See description on Page 845	0x780FC000	R/W	845
EMAC0_RMR	0x1 40000810	Reset	0x00000007	R/W	846
EMAC0_ISR	0x1 40000814	Always	0x00000000	R/W	847
EMAC0_ISER	0x1 40000818	Reset	0x00000000	R/W	849

Note: See *Reset and Initialization* on page 876 for definitions of letters in the Write Access column.

Table 24-5. EMAC0 Register Summary (Continued)

Register	Address	Write Access	Power-on Reset Value	Access	Page
EMAC0_IAHR	0x1 4000081C	Reset, R, T	0x00000000	R/W	851
EMAC0_IALR	0x1 40000820	Reset, R, T	0x00000000	R/W	852
EMAC0_VTPID	0x1 40000824	Reset, R, T	0x00008808	R/W	852
EMAC0_VTCI	0x1 40000828	Reset, R, T	0x00000000	R/W	853
EMAC0_PTR	0x1 4000082C	Reset, T	0x0000FFFF	R/W	853
EMAC0_IAHT1	0x1 40000830	Reset, R	0x00000000	R/W	854
EMAC0_IAHT2	0x1 40000834	Reset, R	0x00000000	R/W	854
EMAC0_IAHT3	0x1 40000838	Reset, R	0x00000000	R/W	854
EMAC0_IAHT4	0x1 4000083C	Reset, R	0x00000000	R/W	854
EMAC0_GAHT1	0x1 40000840	Reset, R	0x00000000	R/W	854
EMAC0_GAHT2	0x1 40000844	Reset, R	0x00000000	R/W	854
EMAC0_GAHT3	0x1 40000848	Reset, R	0x00000000	R/W	854
EMAC0_GAHT4	0x1 4000084C	Reset, R	0x00000000	R/W	854
EMAC0_LSAH	0x1 40000850	Not applicable	0x00000000	R	855
EMAC0_LSAH	0x1 40000854	Not applicable	0x00000000	R	855
EMAC0_IPGVR	0x1 40000858	Reset, T	0x00000004	R/W	855
EMAC0_STACR	0x1 4000085C	See description on Page 856	0x00008000	R/W	856
EMAC0_TRTR	0x1 40000860	See description on Page 857	0x00000000	R/W	857
EMAC0_RWMR	0x1 40000864	Reset	0x04000800	R/W	858
EMAC0_OCTX	0x1 40000868	Not applicable	0x00000000	R	859
EMAC0_OCRX	0x1 4000086C	Not applicable	0x00000000	R	859
EMAC0_IPCR	0x1 40000870	Not applicable	0x00000000	R/W	859

Note: See *Reset and Initialization* on page 876 for definitions of letters in the Write Access column.

Table 24-6. EMAC1 Register Summary

Register	Address	Write Access	Power-on Reset Value	Access	Page
EMAC1_MR0	0x1 40000900	See description in <i>Scenario 1</i> on page 876	0xC0000000	R/W	844
EMAC1_MR1	0x1 40000904	Reset	0x00000000	R/W	842
EMAC1_TMR0	0x1 40000908	See description on Page 844	0x00000007	R/W	844
EMAC1_TMR1	0x1 4000090C	See description on Page 845	0x780FC000	R/W	845
EMAC1_RMR	0x1 40000910	Reset	0x00000000	R/W	846
EMAC1_ISR	0x1 40000914	Always	0x00000000	R/W	847
EMAC1_ISER	0x1 40000918	Reset	0x00000000	R/W	849
EMAC1_IAHR	0x1 4000091C	Reset, R, T	0x00000000	R/W	851
EMAC1_IALR	0x1 40000920	Reset, R, T	0x00000000	R/W	852
EMAC1_VTPID	0x1 40000924	Reset, R, T	0x00008808	R/W	852
EMAC1_VTCI	0x1 40000928	Reset, R, T	0x00000000	R/W	853

Note: See *Reset and Initialization* on page 876 for definitions of letters in the Write Access column.

Preliminary User's Manual

Table 24-6. EMAC1 Register Summary (Continued)

Register	Address	Write Access	Power-on Reset Value	Access	Page
EMAC1_PTR	0x1 4000092C	Reset, T	0x0000FFFF	R/W	853
EMAC1_IAHT1	0x1 40000930	Reset, R	0x00000000	R/W	854
EMAC1_IAHT2	0x1 40000934	Reset, R	0x00000000	R/W	854
EMAC1_IAHT3	0x1 40000938	Reset, R	0x00000000	R/W	854
EMAC1_IAHT4	0x1 4000093C	Reset, R	0x00000000	R/W	854
EMAC1_GAHT1	0x1 40000940	Reset, R	0x00000000	R/W	854
EMAC1_GAHT2	0x1 40000944	Reset, R	0x00000000	R/W	854
EMAC1_GAHT3	0x1 40000948	Reset, R	0x00000000	R/W	854
EMAC1_GAHT4	0x1 4000094C	Reset, R	0x00000000	R/W	854
EMAC1_LSAH	0x1 40000950	Not applicable	0x00000000	R	855
EMAC1_LSAL	0x1 40000954	Not applicable	0x00000000	R	855
EMAC1_IPGVR	0x1 40000958	Reset, T	0x00000004	R/W	855
EMAC1_STACR	0x1 4000095C	See description on Page 856	0x00008000	R/W	856
EMAC1_TRTR	0x1 40000960	See description on Page 857	0x00000000	R/W	857
EMAC1_RWMR	0x1 40000964	Reset	0x04001000	R/W	858
EMAC1_OCTX	0x1 40000968	Not applicable	0x00000000	R	859
EMAC1_OCRX	0x1 4000096C	Not applicable	0x00000000	R	859
EMAC1_IPCR	0x1 40000970	Not applicable	0x00000000	R/W	859

Note: See *Reset and Initialization* on page 876 for definitions of letters in the Write Access column.

Table 24-7. EMAC2 Register Summary

Register	Address	Write Access	Power-on Reset Value	Access	Page
EMAC2_MR0	0x1 40000C00	See description in "Scenario 1" on page -876	0xC0000000	R/W	844
EMAC2_MR1	0x1 40000C04	Reset	0x00000000	R/W	842
EMAC2_TMR0	0x1 40000C08	See description on Page 844	0x00000007	R/W	844
EMAC2_TMR1	0x1 40000C0C	See description on Page 845	0x780FC000	R/W	845
EMAC2_RMR	0x1 40000C10	Reset	0x00000000	R/W	846
EMAC2_ISR	0x1 40000C14	Always	0x00000000	R/W	847
EMAC2_ISER	0x1 40000C18	Reset	0x00000000	R/W	849
EMAC2_IAHR	0x1 40000C1C	Reset, R, T	0x00000000	R/W	851
EMAC2_IALR	0x1 40000C20	Reset, R, T	0x00000000	R/W	852
EMAC2_VTPID	0x1 40000C24	Reset, R, T	0x00008808	R/W	852
EMAC2_VTCI	0x1 40000C28	Reset, R, T	0x00000000	R/W	853
EMAC2_PTR	0x1 40000C2C	Reset, T	0x0000FFFF	R/W	853
EMAC2_IAHT1	0x1 40000C30	Reset, R	0x00000000	R/W	854
EMAC2_IAHT2	0x1 40000C34	Reset, R	0x00000000	R/W	854
EMAC2_IAHT3	0x1 40000C38	Reset, R	0x00000000	R/W	854

Note: See *Reset and Initialization* on page 876 for definitions of letters in the Write Access column.

Table 24-7. EMAC2 Register Summary (Continued)

Register	Address	Write Access	Power-on Reset Value	Access	Page
EMAC2_IAHT4	0x1 40000C3C	Reset, R	0x00000000	R/W	854
EMAC2_GAHT1	0x1 40000C40	Reset, R	0x00000000	R/W	854
EMAC2_GAHT2	0x1 40000C44	Reset, R	0x00000000	R/W	854
EMAC2_GAHT3	0x1 40000C48	Reset, R	0x00000000	R/W	854
EMAC2_GAHT4	0x1 40000C4C	Reset, R	0x00000000	R/W	854
EMAC2_LSAH	0x1 40000C50	Not applicable	0x00000000	R	855
EMAC2_LSAH	0x1 40000C54	Not applicable	0x00000000	R	855
EMAC2_IPGVR	0x1 40000C58	Reset, T	0x00000004	R/W	855
EMAC2_STACR	0x1 40000C5C	See description on Page 856	0x00008000	R/W	856
EMAC2_TRTR	0x1 40000C60	See description on Page 857	0x00000000	R/W	857
EMAC2_RWMR	0x1 40000C64	Reset	0x04001000	R/W	858
EMAC2_OCTX	0x1 40000C68	Not applicable	0x00000000	R	859
EMAC2_OCRX	0x1 40000C6C	Not applicable	0x00000000	R	859
EMAC2_IPCR	0x1 40000C70	Not applicable	0x00000000	R/W	859
Note: See <i>Reset and Initialization</i> on page 876 for definitions of letters in the Write Access column.					

Table 24-8. EMAC3 Register Summary

Register	Address	Write Access	Power-on Reset Value	Access	Page
EMAC3_MR0	0x1 40000E00	See description in "Scenario 1" on page -876	0xC0000000	R/W	844
EMAC3_MR1	0x1 40000E04	Reset	0x00000000	R/W	842
EMAC3_TMR0	0x1 40000E08	See description on Page 844	0x00000007	R/W	844
EMAC3_TMR1	0x1 40000E0C	See description on Page 845	0x780FC000	R/W	845
EMAC3_RMR	0x1 40000E10	Reset	0x00000000	R/W	846
EMAC3_ISR	0x1 40000E14	Always	0x00000000	R/W	847
EMAC3_ISER	0x1 40000E18	Reset	0x00000000	R/W	849
EMAC3_IAHR	0x1 40000E1C	Reset, R, T	0x00000000	R/W	851
EMAC3_IALR	0x1 40000E20	Reset, R, T	0x00000000	R/W	852
EMAC3_VTPID	0x1 40000E24	Reset, R, T	0x00008808	R/W	852
EMAC3_VTCI	0x1 40000E28	Reset, R, T	0x00000000	R/W	853
EMAC3_PTR	0x1 40000E2C	Reset, T	0x0000FFFF	R/W	853
EMAC3_IAHT1	0x1 40000E30	Reset, R	0x00000000	R/W	854
EMAC3_IAHT2	0x1 40000E34	Reset, R	0x00000000	R/W	854
EMAC3_IAHT3	0x1 40000E38	Reset, R	0x00000000	R/W	854
EMAC3_IAHT4	0x1 40000E3C	Reset, R	0x00000000	R/W	854
EMAC3_GAHT1	0x1 40000E40	Reset, R	0x00000000	R/W	854
EMAC3_GAHT2	0x1 40000E44	Reset, R	0x00000000	R/W	854
EMAC3_GAHT3	0x1 40000E48	Reset, R	0x00000000	R/W	854
Note: See <i>Reset and Initialization</i> on page 876 for definitions of letters in the Write Access column.					

Preliminary User's Manual

Table 24-8. EMAC3 Register Summary (Continued)

Register	Address	Write Access	Power-on Reset Value	Access	Page
EMAC3_GAHT4	0x1 40000E4C	Reset, R	0x00000000	R/W	854
EMAC3_LSAH	0x1 40000E50	Not applicable	0x00000000	R	855
EMAC3_LSAL	0x1 40000E54	Not applicable	0x00000000	R	855
EMAC3_IPGVR	0x1 40000E58	Reset, T	0x00000004	R/W	855
EMAC3_STACR	0x1 40000E5C	See description on Page 856	0x00008000	R/W	856
EMAC3_TRTR	0x1 40000E60	See description on Page 857	0x00000000	R/W	857
EMAC3_RWMR	0x1 40000E64	Reset	0x04001000	R/W	858
EMAC3_OCTX	0x1 40000E68	Not applicable	0x00000000	R	859
EMAC3_OCRX	0x1 40000E6C	Not applicable	0x00000000	R	859
EMAC3_IPCR	0x1 40000E70	Not applicable	0x00000000	R	859

Note: See *Reset and Initialization* on page 876 for definitions of letters in the Write Access column.

24.7.1 Mode Register 0 (EMACx_MR0)

EMACx_MR0 defines the operating modes of the EMAC, which can be changed at any time during EMAC operation.

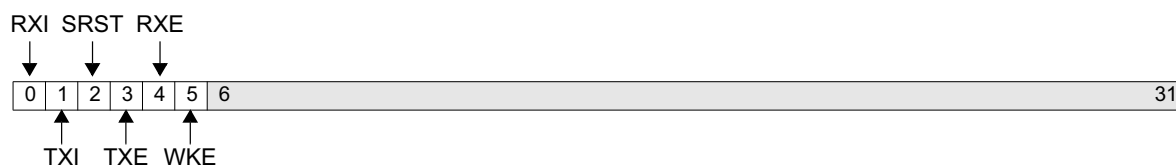


Figure 24-15. Mode Register 0 (EMACx_MR0)

0	RXI	Receive MAC Idle 0 RX MAC processing packet 1 RX MAC idle; RX packet processing complete	Read-only
1	TXI	Transmit MAC Idle 0 TX MAC processing packet 1 TX MAC idle; TX packet processing complete	Read-only
2	SRST	EMAC Software Reset 0 EMAC reset is complete 1 Reset the EMAC	Generates a general reset to EMAC through a software command. After setting this bit, EMAC hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive EMAC to the reset state. The bit is cleared by the hardware when the reset is completed. If EMACx_MR0[SRST] = 1, writing to any EMAC register, and reading any other bit in this register, is not supported.
3	TXE	Transmit MAC Enable 0 TX MAC is disabled 1 TX MAC is enabled	
4	RXE	Receive MAC Enable 0 RX MAC is disabled 1 RX MAC is enabled	

5	WKE	Wake-Up Enable 0 Incoming packets are not examined for wake-up packet 1 Examine incoming packets for wake-up packet	Software can change EMACx_MR0[WKE] only while EMACx_MR0[RXI] = 1 and EMACx_MR0[RXE] = 0.
6:31		Reserved	

24.7.2 Mode Register 1 (EMACx_MR1)

EMACx_MR1 defines the EMAC operating modes, which can be changed only after a reset.

Software can use EMACx_MR1[FDE] and proper programming of the attached PHY to activate external loop-back mode.

When EMACx_MR1[FDE, ILE] = 1, EMAC wraps transmitted packets back to the receive FIFO without accessing the MII.

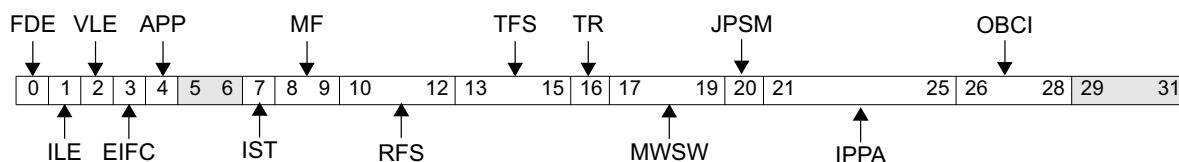


Figure 24-16. Mode Register 1 (EMACx_MR1)

0	FDE	Full-Duplex Enable 0 Disable simultaneous transmit and receive 1 Enable simultaneous transmit and receive	
1	ILE	Internal Loop-back Enable 0 No wrap back 1 Transmitted packets wrapped back to receive FIFO	Full Duplex must also be set (EMACx_MR1[FDE]=1).
2	VLE	VLAN Enable 0 Disable processing of VLAN Tags 1 Enable processing of VLAN Tags	
3	EIFC	Enable Integrated Flow Control 0 Disable integrated flow control mechanism 1 Enable integrated flow control mechanism	Refer to <i>Flow Control</i> on page 829 for more details. Set EMACx_MR1[EIFC] = 0 in half-duplex mode.
4	APP	Allow Pause Packet 0 Disables processing of incoming control (pause) packets 1 Enables processing of incoming control (pause) packets	
5:6		Reserved	Always zero
7	IST	Ignore SQE test 0 Wait for end of SQE test period before activation of valid signal 1 Do not wait for end of SQE test period before activation of valid signal	EMACx_MR1[IST] = 0 only during half-duplex operation on 10 Mbps media.
8:9	MF	Medium Frequency 00 10 Mbps (Ethernet mode) 01 100 Mbps (Fast Ethernet mode) 10 1000 Mbps (Gigabit Ethernet mode) without using the internal GPCS device 11 1000 Mbps (Gigabit Ethernet mode) with the use of internal GPCS device	Defines the possible operational frequency on the MII/GMII interface. Gigabit ethernet mode settings are only valid for EMAC 2 and 3.

Preliminary User's Manual

10:12	RFS	Receive (RX) FIFO Size 000 512 bytes 001 1 KB 010 2 KB 011 4 KB 100 8 KB 101 16 KB 110 Reserved	Note: The highest value for RX FIFO size is 4KB for EMAC0 and EMAC1, and 16KB for EMAC2 and EMAC3.
13:15	TFS	Transmit (TX) FIFO Size 000 512 bytes 001 1 KB 010 2 KB 011 4 KB 100 8 KB 101 16 KB 110 Reserved	Note: The highest value for TX FIFO size is 2KB for EMAC0, EMAC1, EMAC2 and EMAC3.
16	TR	Transmit Request 0 Single packet 1 Multiple packets	Defines the different modes for using transmit channel of EMAC.
17:19	MWSW	Maximum Waiting Status Words 000 A packet is not sent until the status of the previous packet has been received 001 Allows up to one status to be pending when sending the next packet 010 Reserved	Defines the number of status words EMAC can wait for, and still continue transmission. A '0' value will force EMAC to wait for every status until requesting a transmission of new packet. For better performance set EMAC0_MR1[MWSW=001]
20	JPSM	Jumbo Packet Support Mode 0 Jumbo packet support mode disabled 1 Jumbo packet support mode enabled	When enabled, EMAC is capable of handling packets with a length of up to 9018 bytes. This bit can be set only when 1000 Mbps mode is chosen
21:25	IPPA	Internal PCS PHY Address	
26:28	OBCI	OPB Bus Clock Indication 000 50 MHz 001 66 MHz 010 83 MHz 011 100 MHz 100 Above 100 MHz	EMACx_MR1[TFS] and data clock of EMACx_MR1 are used for generation of EMC_MDC clock. Note: When operational frequency differs from this list, then the next greater frequency should be chosen.
29:31		Reserved	

24.7.3 Transmit Mode Register 0 (EMACx_TMR0)

EMACx_TMR0 defines EMAC operating modes during transmit operations (see *EMAC Transmit Operation* on page 821).

EMACx_TMR0[GNP0, GNP1, GNPD] are self-clearing. Writing 0 to these fields has no effect.

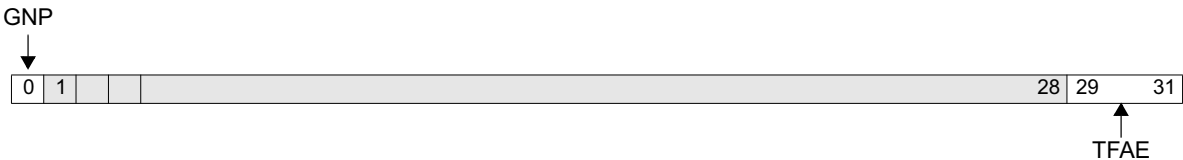


Figure 24-17. Transmit Mode Register 0 (EMACx_TMR0)

0	GNP	Get New Packet 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel	EMACx_TMR0[GNP0] = 0 if EMAC is programmed.
1:28		Reserved	
29:31	TFAE	<p>TX FIFO Almost Empty</p> <p>000 Reserved</p> <p>001 Actual number of entries limit in FIFO is 2 and the limit in value of PHY clock cycles is 32</p> <p>010 Actual number of entries limit in FIFO is 4 and the limit in value of PHY clock cycles is 64</p> <p>011 Actual number of entries limit in FIFO is 8 and the limit in value of PHY clock cycles is 128</p> <p>100 Actual number of entries limit in FIFO is 16 and the limit in value of PHY clock cycles is 256</p> <p>101 Actual number of entries limit in FIFO is 32 and the limit in value of PHY clock cycles is 512</p> <p>110 Actual number of entries limit in FIFO is 64 and the limit in value of PHY clock cycles is 1024</p> <p>111 Actual number of entries limit in FIFO is 128 and the limit in value of PHY clock cycles is 2048</p>	<p>When the number of occupied entries in the TX FIFO is less than or equal to this limit, and while a packet is transmitted, TX_FIFO_ALMOST_EMPTY interrupt is asserted.</p> <p>Note: The value of '111' is applicable only when 'Jumbo Packet' option is enabled. See <i>Transmit Mode Register 1 (EMACx_TMR1)</i> on page 845. Otherwise, writing a value of '111' will disable the TX FIFO almost empty limit option, that is, no interrupt will occur. This is also set as default option.</p>

24.7.4 Transmit Mode Register 1 (EMACx_TMR1)

EMACx_TMR1 shown in *Figure 24-18* defines conditions for activation of MAL service requests during transmit operations (see *EMAC Transmit Operation* on page 821).

24.7.4.1 Low-Priority Requests

EMAC requests low priority service from MAL when the number of vacant entries in the transmit FIFO exceeds the decimal transmit low request (TLR) value. EMACx_TMR1[TLR] must be at least 17 when MAL burst length is using the default 64 words.

To avoid a deadlock, the sum of EMACx_TMR1[TLR] and EMACx_TRTR[TRT] must be at least 4 smaller than the transmit FIFO size specified by EMACx_MR1[TFS].

24.7.4.2 Urgent-Priority Requests

EMAC requests urgent priority service from MAL if the following conditions occur:

- EMAC begins transmitting the packet to the media before the entire packet is placed in the TX FIFO
- The number of vacant entries for the currently transmitting packet exceeds the decimal TUR value

Software must coordinate the value of EMACx_TMR1[TUR] with the value of EMACx_MR1[TFS].The value of EMACx_TMR1[TUR] must be smaller than that of EMACx_MR1[TFS] so that the array address encoded in EMACx_TMR1[TUR] can access the full 66-bit wide array.

The binary value of EMACx_TMR1[TUR] must be greater than that of EMACx_TMR1[TLR].

The EMACx_TMR1 contents can be changed only when EMACx_TMR0[GNP0, GNP1, GNPD] = 0.

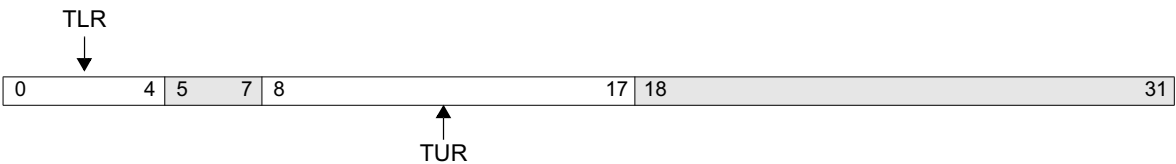


Figure 24-18. Transmit Mode Register 1 (EMACx_TMR1)

0:4	TLR	Transmit Low Request
5:7		Reserved
8:17	TUR	Transmit Urgent Request
18:31		Reserved

24.7.5 Receive Mode Register (EMACx_RMR)

EMACx_RMR defines EMAC operating modes during receive operations.

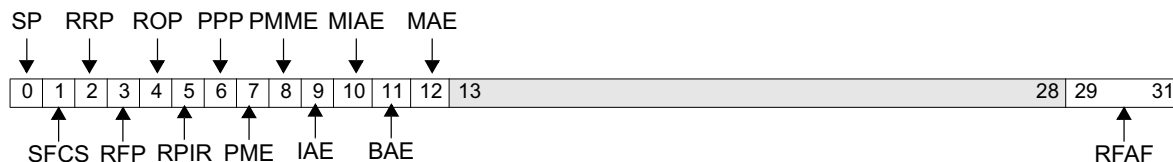


Figure 24-19. Receive Mode Register (EMACx_RMR)

0	SP	Strip Padding 0 Do not strip pad bytes from the received packet. 1 Strip pad/FCS bytes from the received packet.
1	SFCS	Strip FCS 0 Do not strip FCS bytes from the received packet. 1 Strip FCS bytes from the received packet.
2	RRP	Receive Runt Packets 0 Discard packets less than 64 bytes in length. 1 Receive packets less than 64 bytes in length.
3	RFP	Allow Receive Packets with a FCS Error 0 Discard packets containing a FCS error. 1 Receive packets containing a FCS error.
4	ROP	Receive Oversize Packet 0 Discard packets that activate Packet Is Too Long error. 1 Receive packets that activate Packet Is Too Long error.
5	RPIR	Receive Packets with In Range Error 0 Discard packets that activate In Range Error. 1 Receive packets that activate In Range Error.
6	PPP	Propagate Pause Packet 0 Do not propagate incoming pause packet to MAL; remove packet from FIFO. 1 Propagate incoming pause packet to MAL.
7	PME	Promiscuous Mode Enable 0 Do not enable promiscuous mode. 1 Accept all packets.
8	PMME	Promiscuous Multicast Mode Enable 0 Do not accept all multicast packets. 1 Accept all multicast packets.
9	IAE	Individual Address Enable 0 Do not compare address of received packets with content of individual address register. 1 Compare address of received packets with content of individual address register.
10	MIAE	Multiple Individual Address Enable 0 Do not compare address of received packets with hash table of individual addresses. 1 Compare address of received packets with hash table of individual addresses.
11	BAE	Broadcast Address Enable 0 Do not compare address of received packets with broadcast addresses. 1 Compare address of received packets with broadcast addresses.

Preliminary User's Manual

12	MAE	Multicast Address Enable 0 Do not compare address of received packets with multicast addresses. 1 Compare address of received packets with multicast addresses.
13:28		Reserved
29:31	RFAF	RX FIFO Almost Full 000 Reserved 001 Actual number of entries limit in FIFO is 2 and the limit in value of PHY clock cycles is 32 010 Actual number of entries limit in FIFO is 4 and the limit in value of PHY clock cycles is 64 011 Actual number of entries limit in FIFO is 8 and the limit in value of PHY clock cycles is 128 100 Actual number of entries limit in FIFO is 16 and the limit in value of PHY clock cycles is 256 101 Actual number of entries limit in FIFO is 32 and the limit in value of PHY clock cycles is 512 110 Actual number of entries limit in FIFO is 64 and the limit in value of PHY clock cycles is 1024 111 Actual number of entries limit in FIFO is 128 and the limit in value of PHY clock cycles is 2048 When the number of occupied entries in the RX FIFO is greater than or equal to this limit, and while a packet is received, RX_FIFO_ALMOST_FULL interrupt is asserted. Note: The value of '111' is applicable only when 'Jumbo Packet' option is enabled. See <i>Transmit Mode Register 1 (EMACx_TMR1)</i> on page 845. Otherwise, writing a value of '111' will disable the RX FIFO almost full limit option, that is, no interrupt will occur. This is also set as default option.

24.7.6 Interrupt Status Register (EMACx_ISR)

Each EMAC generates a distinct interrupt event indication. The event indication signal is driven out of the EMAC to UIC1 (interrupt 28 for EMAC0 and interrupt 30 for EMAC1) and UIC2 (interrupt 0 for EMAC2 and interrupt 2 for EMAC3). This interrupt is generated from the content of the EMACx_ISR. The content of the EMACx_ISR is first ANDed with the corresponding mask bits in the EMACx_ISR; the resulting bits are then logically ORed to produce the interrupt signal. Thus, if any of the resulting bits is a 1, an interrupt is generated.

Note: EMAC activates its interrupt signal only after an indication that status for the current packet was accepted by MAL (with the exception of "MMA Operation Succeed/MMA Operation Failed," which causes unconditional activation of interrupt, if it is not masked).

The interrupt indication is cleared by writing 1 to the related bit in the EMACx_ISR; writing 0 has no effect.

The event indication signal is cleared when all non-masked event indication bits are cleared.

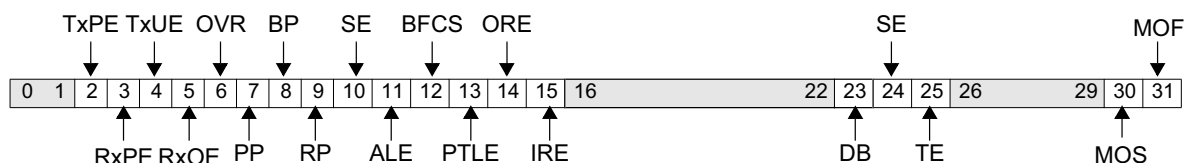


Figure 24-20. Interrupt Status Register (EMACx_ISR)

0:1		Reserved
2	TxPE	TX Parity Error 0 No TX parity error 1 Tx parity error occurs when a transmit data bit is damaged at the transmit FIFO

3	RxPE	RX Parity Error 0 No RX parity error 1 Rx parity error occurs when a receive data bit is damaged at the receive FIFO or internal LPRA	
4	TxUE	TX Underrun Event 0 No underrun event 1 Underrun event while packet is being transferred	TX FIFO almost empty
5	RxOE	RX Overrun Event 0 No overrun event 1 Overrun event while packet is being received	RX FIFO almost full
6	OVR	Overrun 0 No overrun error 1 Overrun error during reception of recent packet	
7	PP	Pause Packet 0 Received packet is not a control pause packet 1 Received packet is a control pause packet	
8	BP	Bad Packet 0 Receive operation OK 1 Early termination was initiated because of a packet error	
9	RP	Runt Packet 0 No Runt packets received 1 Runt packet received	Set when EMACx_RMR[RRP] = 1 and the duration of PHY_RX_DV signal was greater than Short-EventMaxTime constant and less than the collision window.
10	SE	Short Event 0 No short events 1 Duration of PHY_RX_DV signal less than ShortEventMaxTime constant	
11	ALE	Alignment Error 0 No alignment error in received packet 1 Alignment error in received packet	The packet contained an odd number of nibbles (4 bits).
12	BFCS	Bad FCS 0 No FCS error in received packet 1 Packet with an FCS error received	Set if EMACx_RMR[RFP] = 1.
13	PTLE	Packet Too Long Error 0 No oversized packets received 1 Oversized packet received	Set if EMACx_RMR[ROP] = 1 and the received packet length exceeded the maximum allowed value: <ul style="list-style-type: none"> 1518 octets for standard packet (checked only if the length/type field of the transmitted packet contained length value) 1522 octets for VLAN tagged packet (checked only if the length/type field of the transmitted packet contained length value and jumbo support is disabled) 9018 octets for jumbo packet (with no VLAN support) 9022 octets for VLAN tagged Jumbo packet
14	ORE	Out Of Range Error 0 Received packet length field value OK 1 Received packet length field value greater than the maximum allowed LLC data size	Indicates that received packet has a length field value greater than the maximum allowed logical link control (LLC) data size (greater than 1500 and less than 1536).
15	IRE	In Range Error 0 Received packet does not contain an In Range Error 1 Received packet contains an In Range Error	
16:22		Reserved	

Preliminary User's Manual

23	DB	Dead Bit 0 No transmit error or SQE for TX Channel 1 Transmit error or SQE has occurred for TX Channel	If EMACx_ISR[DB] = 1, EMAC does not request service for TX Channel from MAL, even if EMACx_TMR0[GNP] = 1. EMACx_ISR[DB] does not affect EMAC interrupt.
24	SE0	Signal Quality Error 0 No SQEs on TX Channel 1 SQE test failure during transmission of a packet from TX Channel	Applicable only in half-duplex mode during 10 Mbps operations; 0 in all other modes.
25	TE0	Transmit Error 0 TX Channel transmission OK 1 TX Channel transmission aborted	EMAC aborts the transmitted packet if one of the following events takes place: <ul style="list-style-type: none"> • Late collision detection • Excessive collision detection • Excessive deferral • TX FIFO underrun • Loss of carrier sense
26:29		Reserved	Always 0
30	MOS	MMA Operation Succeeded 0 MMA_CONTROL addressed on the OPB 1 PHY transfer valid	The device driver should poll assertion of EMACx_ISR[MOS] or EMACx_ISR[MOF] before issuing a new command or before using data read from the PHY.
31	MOF	MMA Operation Failed 0 MMA_CONTROL addressed on the OPB 1 PHY transfer not valid	The device driver should poll assertion of EMACx_ISR[MOF] or EMACx_ISR[MOS] before issuing a new command or before using data read from the PHY.

24.7.7 Interrupt Status Enable Register (EMACx_ISR)

EMACx_ISR indicates which conditions in the EMACx_ISR can generate an interrupt.

Each masking bit in the EMACx_ISR corresponds to a related bit in the EMACx_ISR. If a mask bit is set to 1, the corresponding status bit, when set, causes an interrupt to be generated. Setting a mask bit to 0 suppresses interrupt generation for the associated condition.

Mask bits for reserved bits in the EMACx_ISR are not implemented, have no effect on write, and return 0 on read.

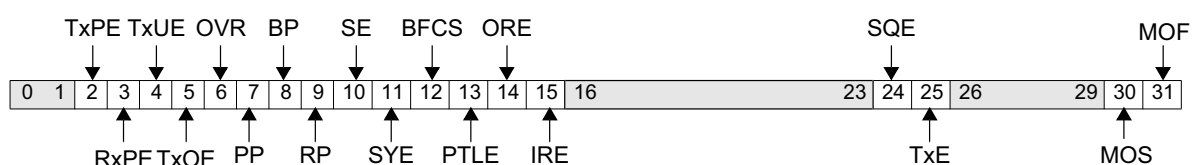


Figure 24-21. Interrupt Status Enable Register (EMACx_ISR)

0:1		Reserved
2	TxPE	TX Parity Error 0 No TX parity error 1 Tx parity error occurs when a transmit data bit is damaged at the transmit FIFO
3	RxPE	RX Parity Error 0 No RX parity error 1 Rx parity error occurs when a receive data bit is damaged at the receive FIFO or internal LPRA
4	TxUE	TX Underrun Event 0 No underrun event 1 Underrun event while packet is being transferred

5	RxOE	RX Overrun Event 0 No overrun event 1 Overrun event while packet is being received	RX FIFO almost full
6	OVR	Overrun 0 Overrun error will not generate an interrupt. 1 Overrun error will generate an interrupt.	
7	PP	Pause Packet 0 Received control pause packet will not generate an interrupt. 1 Received control pause packet will generate an interrupt.	
8	BP	Bad Packet 0 Early termination on received packet will not generate an interrupt. 1 Early termination on received packet will generate an interrupt.	
9	RP	Runt Packet 0 Received runt packet will not generate an interrupt. 1 Received runt packet will generate an interrupt.	
10	SE	Short Event 0 Short event during receive will not generate an interrupt. 1 Short event during receive will generate an interrupt.	
11	ALE	Alignment Error 0 Alignment error in received packet will not generate an interrupt. 1 Alignment error in received packet will generate an interrupt.	
12	BFCS	Bad FCS 0 FCS error in received packet will not generate an interrupt. 1 FCS error in received packet will generate an interrupt.	
13	PTLE	Packet Too Long Error 0 Oversized packets received will not generate an interrupt. 1 Oversized packet received will generate an interrupt.	
14	ORE	Out Of Range Error 0 Out of range error on received packet will not generate an interrupt. 1 Out of range error on received packet will generate an interrupt.	
15	IRE	In Range Error 0 In range error on received packet will not generate an interrupt. 1 In range error on received packet will generate an interrupt.	
16:23		Reserved	
24	SQE	SQE Error 0 SQE error on TX Channel will not generate an interrupt. 1 SQE error on TX Channel will generate an interrupt.	

Preliminary User’s Manual

25	TxE	Transmit Error 0 TX error on TX Channel will not generate an interrupt. 1 TX error on TX Channel will generate an interrupt.
26:29		Reserved
30	MOS	MMA Operation Succeeded 0 Successful MMA Operation with a PHY will not generate an interrupt. 1 Successful MMA Operation with a PHY will generate an interrupt.
31	MOF	MMA Operation Failed 0 Unsuccessful MMA Operation with a PHY will not generate an interrupt. 1 Unsuccessful MMA Operation with a PHY will generate an interrupt.

24.7.8 Individual Address High (EMACx_IAHR)

EMACx_IAHR contains the high-order halfword of the station unique individual address.

During packet reception, if EMAC is programmed in individual address match mode (EMACx_RMR[IAE] = 1), the contents of EMACx_IAHR are concatenated with the content of EMACx_IALR to form a composite address that is compared with the destination address of the received packet. If addresses match, the packet is transferred to MAL.

During packet transmission, EMACx_IAHR is used in source address inclusion/replacement and as the source address field in the self-assembled control (pause) packet.

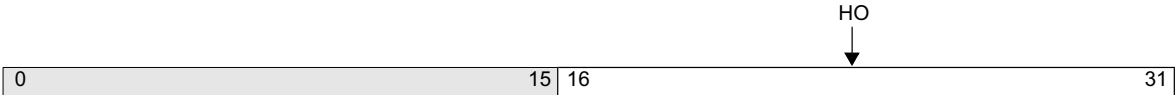


Figure 24-22. Individual Address High Register (EMACx_IAHR)

0:15		Reserved	
16:31	HO	Receive and Transmit High Order High-order halfword of the station unique individual address	This field contains bits 0:15 of the destination address (bit 0 is the most significant bit).

24.7.9 Individual Address Low (EMACx_IALR)

EMACx_IALR contains the low-order word of the station unique individual address.

During packet reception, EMACx_IALR is compared with the corresponding address bits of the received packet.

During packet transmission, EMACx_IALR is used in source address inclusion/replacement and as the source address field in the self-assembled control (pause) packet.

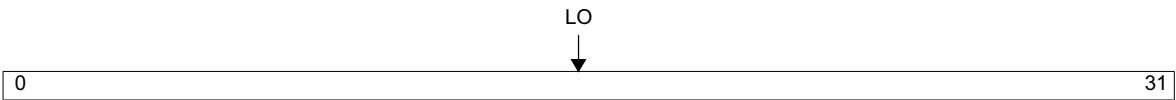


Figure 24-23. Individual Address Low Register (EMACx_IALR)

0:31	LO	Receive and Transmit Low Order Low-order bits of Receive Individual Address or Transmit Source Address
------	----	--

24.7.10 VLAN TPID Register (EMACx_VTPID)

EMACx_VTPID contains the value of the VLAN TPID (Tag Protocol Identifier) field.

During packet reception, packet bytes 13 and 14 are compared to the content of this register to check whether the packet is tagged with a VLAN ID.

During packet transmission, EMAC uses EMACx_VTPID when VLAN Tag replacement or VLAN Tag inclusion mode is chosen.

The value of this register must be a Type field (8100).

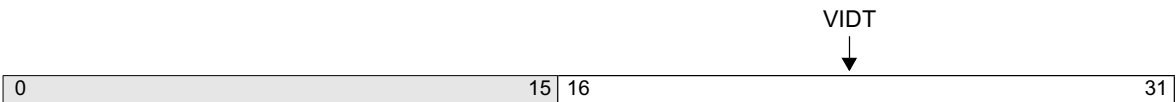


Figure 24-24. VLAN TPID Register (EMACx_VTPID)

0:15		Reserved
16:31	VIDT	VLAN ID tag

24.7.11 VLAN TCI Register (EMACx_VTCI)

EMACx_VTCI contains the value of the VLAN TCI (Tag Control Information) field.

During packet transmission, EMAC uses EMACx_VTCI when VLAN Tag replacement or VLAN Tag inclusion mode is chosen.

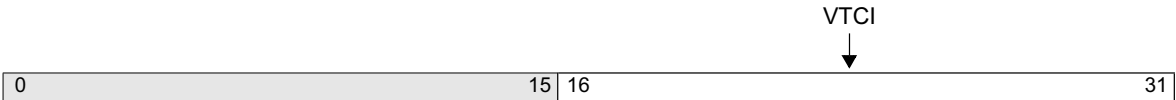


Figure 24-25. VLAN TCI Register (EMACx_VTCI)

0:15		Reserved
16:31	VTCI	VLAN TCI tag

24.7.12 Pause Timer Register (EMACx_PTR)

EMACx_PTR defines the time period for which the pause function is enabled. EMAC uses EMACx_PTR[TVR] as the timer value field of control (pause) packets (see *Control Packet Transmission* on page 830). Each bit corresponds to 512 bit times.

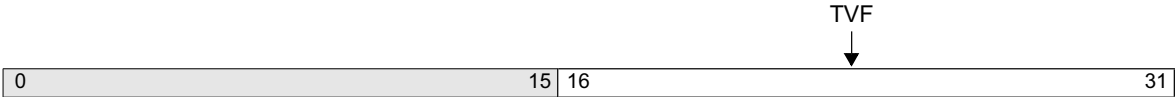


Figure 24-26. Pause Timer Register (EMACx_PTR)

0:15		Reserved
16:31	TVF	Timer Value Field

24.7.13 Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4)

These registers are used in the hash table function of the multiple individual addressing mode.

See *Address Match Mechanism* on page 834 for more information. See *Figure 24-14* on page 837 for bit mapping information.

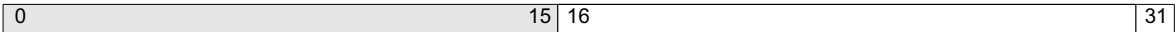


Figure 24-27. Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4)

0:15		Reserved
16:31		Individual Address Hash Number

24.7.14 Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4)

These registers are used in the hash table function of the group addressing mode.

See *Address Match Mechanism* on page 834 for more information. See *Figure 24-14* on page 837 for bit mapping information.

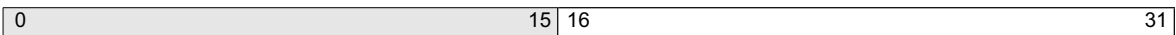


Figure 24-28. Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4)

0:15		Reserved
16:31		Group Address Hash Number

Preliminary User's Manual**24.7.15 Last Source Address High (EMACx_LSAH)**

EMACx_LSAH contains the high-order halfword of the source address of the last “good” received packet. The packet is considered to be “good” if EMAC is programmed to provide this packet to MAL.

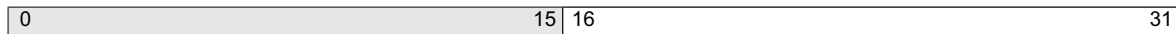


Figure 24-29. Last Source Address High Register (EMACx_LSAH)

0:15		Reserved
16:31		Last Source Address High-Order Halfword

24.7.16 Last Source Address Low (EMACx_LSAL)

EMACx_LSAL contains the low-order word of the source address of the last “good” received packet. The packet is considered “good” if EMAC is programmed to provide this packet to MAL.

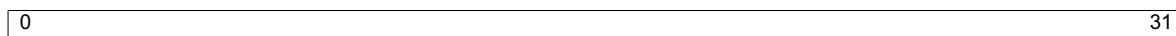


Figure 24-30. Last Source Address Low Register (EMACx_LSAL)

0:31		Last Source Address Low-Order Word
------	--	------------------------------------

24.7.17 Inter-Packet Gap Value Register (EMACx_IPGVR)

EMACx_IPGVR contains the value of one-third of the inter-packet gap (IPG) for the next packet to be transmitted. (“Frame” is synonymous with “packet.”)

The resolution of each bit in the EMACx_IPGVR register is 8-bit times. The minimum value in the register is four, causing the minimum Inter Frame Gap period to be equal to 96-bit times.



Figure 24-31. Inter-Packet Gap Value Register (EMACx_IPGVR)

0:25		Reserved
26:31		Inter-Packet Gap

24.7.18 STA Control Register (EMACx_STACR)

EMACx_STACR controls the MII/GMII Management interface. The software must follow the following steps during access to the EMACx_STACR:

1. Software polls EMACx_STACR[OC], waiting for it to be set by EMAC.

EMAC sets EMACx_STACR[OC] = 0 when the EMACx_STACR is written to.

EMAC then sets EMACx_STACR[OC] = 1 to indicate that the data has been written to the PHY, or the data read from the PHY is valid. The device driver should poll for EMACx_STACR[OC] = 1 before issuing a new command, or before using data read from the PHY.

2. The software can perform read/write access to the EMACx_STACR.
3. EMAC clears EMACx_STACR[OC] (sets EMACx_STACR[OC] = 0) and starts activity on the MII/GMII management interface.
4. Return to step 1.

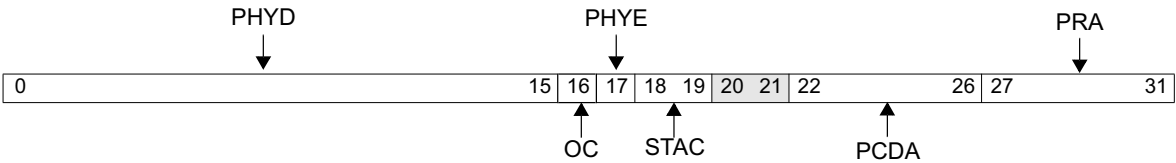


Figure 24-32. STA Control Register (EMACx_STACR)

0:15	PHYD	PHY data	Data to be sent to the PHY if the command is a write, or data is read from the PHY if the command is a read.
16	OC	Operation Complete 0 EMACx_STACR is addressed 1 PHY data transfer complete	
17	PHYE	PHY Error 0 Successful read transaction 1 Read transaction was not successful	EMACx_STACR[PHYE] = 0 when a read is successful.
18:19	STAC	STA Command 00 Reserved 01 Read 10 Write 11 Reserved	EMAC sets EMACx_STACR[STAC] = 0 when the command is completed.
20:21		Reserved	
22:26	PCDA	PHY Command Destination Address	This field contains the address of the PHY intended to receive the command
27:31	PRA	PHY Register Address	This field contains the PHY register address

24.7.19 Transmit Request Threshold Register (EMACx_TRTR)

EMACx_TRTR defines the conditions that cause EMAC to initiate transmission to the Ethernet MAC sub-block, and for requesting service from MAL.

EMACx_TRTR[TRT] defines the number of occupied entries in the transmit FIFO that should be written before the transmit FIFO control logic initiates a transmit request to the Ethernet MAC sub-block.

If an entire packet is already located in the transmit FIFO, EMAC initiates a transmit regardless of the programmed value.

The software must coordinate the value of EMACx_TRTR[TRT] with the transmit FIFO size specified in EMACx_MR1[TFS].

To avoid deadlock, the sum of EMACx_TMR1[TLR] and EMACx_TRTR[TRT] must be smaller, by at least 4, than the transmit FIFO specified in EMACx_MR1[TFS].

To avoid an underrun, program this threshold to a high enough value.

In half-duplex mode, in case of collision, to allow packet re-transmission without involving MAL, EMAC preserves the necessary space in the Transmit FIFO unless it gets an indication that the collision window has elapsed.

The EMACx_TRTR can be written only while EMACx_MR0[TXI] = 1.



Figure 24-33. Transmit Request Threshold Register (EMACx_TRTR)

0:7	TRT	Transmit Request Threshold The following number of bytes must be placed in the Transmit FIFO before initiating a transmit request. 0000_0000 64 bytes 0000_0001 128 bytes 0000_0010 192 bytes 0000_0011 256 bytes . . . 1111_1100 16.192 bytes 1111_1101 16.256 bytes 1111_1110 16.320 bytes 1111_1111 16.384 bytes
		Reserved
8:31		

24.7.20 Receive Low/High Water Mark Register (EMACx_RWMR)

The EMACx_RWMR defines the conditions that cause the EMAC to activate a low or urgent priority MAL request, and that manage flow control.

EMAC activates a low priority request if the number of occupied entries in the receive FIFO is greater than or equal to the content of EMACx_RWMR[RLWM] (the receive low water mark is reached). A request for a pause packet with a pause_value of 0 is also issued when the receive low water mark is reached.

Software must coordinate the value of EMACx_RWMR[RLWM] with the value of EMACx_MR1[RFS]. EMACx_RWMR[RLWM] should be smaller than EMACx_MR1[RFS] and larger than the MAL burst length.

Note: In the PPC440GX Embedded Processor, the user can select the MAL burst length for each MAL RX or TX channel by writing to the appropriate fields in SDR0_MALRBL and SDR0_MALTBL registers described in *MAL Receive Burst Length Register (SDR0_MALRBL)* on page 789 and *MAL Transmit Burst Length Register (SDR0_MALTBL)* on page 790.

If the entire packet is already in the receive FIFO, EMAC initiates a low priority request regardless of the programmed value.

EMAC activates an urgent priority request if the number of occupied entries in the Receive FIFO is greater than or equal to EMACx_RWMR[RHWM] (the receive high water mark is reached). A request for a pause packet is also issued when the receive high water mark is reached.

Software must coordinate the value of EMACx_RWMR[RHWM] with the value of EMACx_MR1[RFS]. EMACx_RWMR[RHWM] should be greater than the value of EMACx_RWMR[RLWM] and less than the size of the receive FIFO.

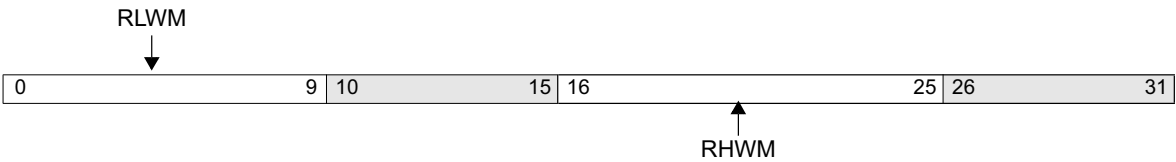


Figure 24-34. Receive Low/High Water Mark Register (EMACx_RWMR)

0:9	RLWM	Receive Low Water Mark
10:15		Reserved
16:25	RHWM	Receive High Water Mark
26:31		Reserved

24.7.21 Transmitted Octects (EMACx_OCTX)

The read-only EMACx_OCTX register contains the number of transmitted octets.



Figure 24-35. Number of Octets Transmitted (EMACx_OCTX)

0:31	OCTX	Number of octets (bytes) transmitted.
------	------	---------------------------------------

24.7.22 Received Octects Register (EMACx_OCRX)

The read-only EMACx_OCRX register contains the number of received octets.



Figure 24-36. Number of Octets Received (EMACx_OCRX)

0:31	OCRX	Number of octets (bytes) received.
------	------	------------------------------------

24.7.23 Internal PCS Configuration Register (EMACx_IPCR)

The read/write EMACx_IPCR register defines various configuration parameters for the internal PCS, Gigabit mode Physical Coding Sublayer (GPCS) unit (see *Gigabit Mode Physical Coding Sublayer (GPCS)* on page 860 for details).

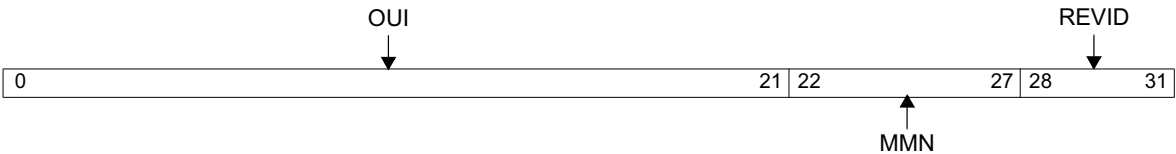


Figure 24-37. Internal PCS Configuration Register (EMACx_IPCR)

0:21	OUI	OUI Value
22:27	MMN	Manufacturer Model Number
28:31	REVID	Revision Number

24.7.24 Gigabit Mode Physical Coding Sublayer (GPCS)

The Gigabit mode Physical Coding Sublayer (GPCS) unit is an implementation of the Physical Coding Sublayer (PCS) which connects a Media Access Control (MAC) with the Physical Medium Attachment (PMA) sublayer, via the PMA service interface. The PCS, MAC and PMA service interface are defined in the IEEE P802.3z Standard, Clauses 36 and 37. PPC440GX implements 2 GPCS (GPCSx) one for each gigabit ethernet GPCS0 for EMAC2 and GPCS1 for EMAC3 respectively. This section only provides information relevant to the functioning of PPC440GX which includes detailed description of all GPCS registers listed in *Table 24-9*.

While *Internal PCS Configuration Register (EMACx_IPCR)* on page 859 defines various configuration parameters for GPCS, *EMACx_STACR* described in *STA Control Register (EMACx_STACR)* on page 856 controls the MII/GMII Management interface. All GPCS 16 bit registers described in this section are accessed via *EMACx_STACR[PHYD]* bits 0:15 which reflect the data to be sent to the PHY if the command is a write, or data is read from the PHY if the command is a read.

Note: All PowerPC registers have bit 0 as the most significant bit while all GPCS 16-bit registers have bit 15 as the most significant bit to be consistent with IEEE specifications.

Table 24-9. GPCS Register Summary

Mnemonic	Register	Address	Access	Page
GPCSx_CR	GPCS Control Register	0x00	R/W	861
GPCSx_SR	GPCS Status Register	0x01	R/W	862
GPCSx_ID0	GPCS ID0 Register	0x02	R	863
GPCSx_ID1	GPCS ID1 Register	0x03	R	864
GPCSx_ANAR	GPCS Auto Negotiation Advertisement Register	0x04	R/W	864
GPCSx_ANLR	GPCS Auto Negotiation Link Partner Base Page Ability Register	0x05	R	866
GPCSx_ANER	GPCS Auto Negotiation Expansion Register	0x06	R	867
GPCSx_ANPTR	GPCS Auto Negotiation Next Page Transmit Register	0x07	R	868
GPCSx_ANLNPR	GPCS Auto Negotiation Link Partner Ability Next Page Register	0x08	R	869
GPCSx_ESR	GPCS Extended Status Register	0x0F	R	870
GPCSx_RAR	GPCS Resolved Ability Register	0x10	R	870
GPCSx_ISR	GPCS Interrupt Status Register	0x11	R/W	871
GPCSx_ISER	GPCS Interrupt Status Enable Register	0x12	R/W	872
GPCSx_CFG	GPCS Configuration Register	0x13	R/W	873

Preliminary User's Manual**24.7.24.1 GPCS Control Register (GPCSx_CR)**

The GPCS control register (GPCSx_CR) controls the activation of some of GPCS operation modes including the activation of the renegotiation function by the SMU. *Figure 24-38* describes the GPCSx_CR register bits.

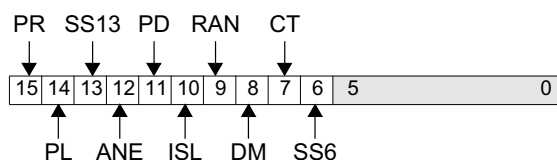


Figure 24-38. GPCS Control Register (GPCSx_CR)

15	PR	PhyReset	This is a soft reset bit. When this bit is set, the GPCS unit logic is reset, and all GPCS registers go to their default value. While the value of this bit is a logic '1', any attempt to write to the GPCS registers will have no effect on the registers. This bit is self-clearing.
14	PL	PhyLoop	When this bit is set, the GPCS unit asserts the PHY_LOOP_EN pin. Typically, the PHY_LOOP_EN pin is connected to the SERDES (such as EWRAP) so that the SERDES can be configured for LoopBack mode. The default value is '0'.
13	SS13	SpeedSelection13	This bit is read as '0'. In combination with bit 6, the SpeedSelection bit in this register, read as '1', it indicates that the GPCS unit works at a PHY speed of 1000 Mbps.
12	ANE	AutoNegEnable	When this bit is set, Auto-Negotiation is enabled. The default value is '0'.
11	PD	PowerDown	When this bit is set, the GPCS unit is placed in the Power Saving mode (the SYS_REG_CLK and PHY_RX_CLK1 can be stopped). When the GPCS unit is in the Power Saving mode, it responds only to the management transactions. The default value is '0'.
10	ISL	Isolate	When this bit is set, the GPCS unit is placed in the Isolation mode. When the GPCS unit is in the Isolation mode, it responds only to the management transactions. This bit has the same influence on GPCS unit power consumption as bit 11, the PowerDown bit in this register. The value is '1' after hard reset and '0' after soft reset.
9	RAN	RestartAutoNegotiation	When this bit is set, Auto-Negotiation is restarted. This bit is self-clearing. If Auto-Negotiation is disabled, then the value of this bit is always a logic '0' and any attempt to write a logic '1' to this bit is ignored. The default value is '0'.
8	DM	DuplexMode	If Auto-Negotiation is disabled (AutoNegEnable bit is cleared), this bit is used in order to configure the GPCS unit for Half or Full Duplex mode. If this bit is set, Full Duplex mode is selected. If this bit is cleared, Half Duplex mode is enabled. Please note that the value of this bit must be set to the same value as the REG_FULL_DUPLEX input. When Auto-Negotiation is enabled, this bit has no effect. The default value is '0'.

7	CT	CollisionTest	When this bit is set, the GPCS unit asserts the COL signal within 512 BT in response to TX_EN assertion, and deasserts the COL within 16 BT in response to TX_EN deassertion. The default value is '0'.
6	SS6	SpeedSelection6	This bit is read as '1'. In combination with bit 13, the SpeedSelection bit in this register, read as '0', it indicates that the GPCS unit works at a PHY speed of 1000 Mbps.
5:0		Reserved	

24.7.24.2 GPCS Status Register (GPCSx_SR)

The GPCS status register (GPCSx_SR), along with the GPCS extended status register (GPCSx_ESR) describe the GPCS current status. *Figure 24-39* describes the GPCSx_SR register bits.

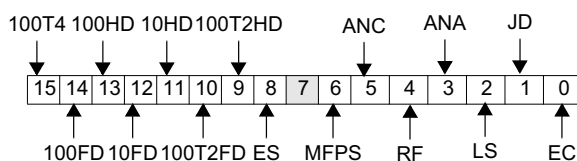


Figure 24-39. GPCS Status Register (GPCSx_SR)

15	100T4	100BASE-T4	Return '0' = PHY cannot perform 100BASE-T4.
14	100FD	100BASE-X Full Duplex	Return '0' = PHY cannot perform full duplex 100BASE-X.
13	100HD	100BASE-X Half Duplex	Return '0' = PHY cannot perform half duplex 100BASE-X.
12	10FD	10 Mbps Full Duplex	When this bit is set, Auto-Negotiation is enabled. The default value is '0'.
11	10HD	10 Mbps Half Duplex	Return '0' = PHY cannot operate at 10 Mbps in Full Duplex mode.
10	100T2FD	100BASE-T2 Full Duplex	Return '0' = PHY cannot operate at 10 Mbps in Half Duplex mode.
9	100T2HD	100BASE-T2 Half Duplex	Return '0' = PHY cannot perform full duplex 100BASE-T2.
8	ES	Extended Status	Return '1' = Extended status information in Register 15.
7		Reserved	Always '0'.
6	MFPS	MF Preamble Suppression	Return '0' = PHY does not use MII management frames.

Preliminary User's Manual

5	ANC	Auto-Negotiation Complete	When read as '1', indicates that the Auto-Negotiation between the GPCS4 unit and its Link Partner has been completed and the contents of the following registers are valid: <ul style="list-style-type: none"> • Auto-Negotiation Advertisement Register • Auto-Negotiation Link Partner Base Page Ability Register • Auto-Negotiation Expansion Register GPCS4 unit returns '0' in this bit if bit 12, the Auto-NegEnable bit in the GPCS Control Register, is '0'. The default value is '0'.
4	RF	Remote Fault	When read as '1', indicates that remote fault condition has been detected. This bit will be set by the Auto-Negotiation block function on receipt of a base page with a non-zero Remote Fault field encoding. It will be cleared each time this register is read via the management interface or by PCS reset. In LoopBack mode, this bit is undefined. The default value is '0'.
3	ANA	Auto-Negotiation Ability	Return '1' = PHY can perform Auto-Negotiation.
2	LS	Link Status	This bit is set to a logic '1' when the <i>xmit</i> flag indicates DATA. When read as '1', indicates that a valid link has been established. The occurrence of a link failure condition clears Link Status bit.
1	JD	Jabber Detect	Always '0'
0	EC	Extended Capability	Return '1' = PHY has extended register capabilities.

24.7.24.3 GPCS ID0 Register (GPCSx_ID0)

The GPCS ID0 register (GPCSx_ID0) together with the GPCS ID1 register (GPCSx_ID1) returns the GPCS organizationally unique identifier (OUI), as it was set by the GPCS sideband signals. *Figure 24-40* describes the GPCSx_ID0 register bits.

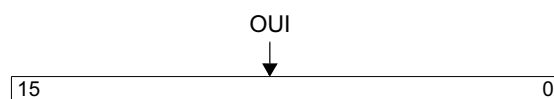


Figure 24-40. GPCS ID0 Register (GPCSx_ID0)

15:0	OUI	OUI(3:18)	The value of these bits adhere to the PHY_OUI(3:18) sideband signals. Bit 15, the OUI(3) bit of this register, corresponds to the PHY_OUI(3) sideband signal, and bit 0, the OUI(18) bit of this register, to the PHY_OUI(18) sideband signal.
------	-----	-----------	--

24.7.24.4 GPCS ID1 Register (GPCSx_ID1)

The GPCS ID1 register (GPCSx_ID1), along with GPCS_ID0 register (GPCSx_ID0), returns the GPCS OUI, and the GPCS manufacturer model number and revision number, as they were set by the GPCS unit sideband signals. Figure 24-41 describes the GPCSx_ID1 register bits.

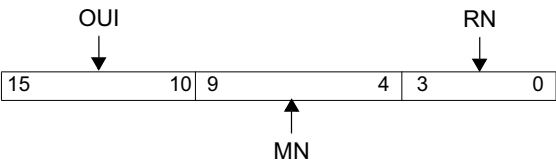


Figure 24-41. GPCS ID1 Register (GPCSx_ID1)

15:10	OUI	OUI(19:24)	The value of these bits adhere to the PHY_OUI(19:24) sideband signals. Bit 15, the OUI(19) bit of this register, corresponds to the PHY_OUI(19) sideband signal, and bit 10, the OUI(24) bit of this register, to the PHY_OUI(24) sideband signal.
9:4	MN	ModeNumber (5:0)	Manufacturer model number, bit (5:0). Bit 9 of this register corresponds to the ModeNumber(5) bit and bit 0 to the ModeNumber(0) bit. The value of these bits will adhere to the PHY_MODEL_NUMB(5:0) sideband signals.
3:0	RN	RevNumber(3:0)	Revision number, bit (3:0). Bit 3 of this register corresponds to the RevNumber(3) bit and bit 0 to the RevNumber(0) bit. The value of these bits will adhere to the sideband PHY_REV_NUMB(3:0) signals.

24.7.24.5 GPCS Auto Negotiation Advertisement Register (GPCSx_ANAR)

The GPCS auto negotiation advertisement register (GPCSx_ANAR) provides the auto-negotiation block function to advertise the EMAC and GPCS unit capabilities to the link partner. Figure 24-42 describes the GPCSx_ANAR register bits.

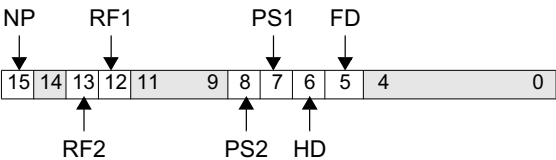


Figure 24-42. GPCS Auto Negotiation Advertisement Register (GPCSx_ANAR)

15	NP	NextPage	This bit shall be set to logic '1' in order to request next page transmission. Subsequent next pages may set the NextPage bit to logic '0' in order to communicate that there is no more next page information to be sent. The default value is '0'.
14		Reserved	Write as '0', ignore on read.

Preliminary User's Manual

13	RF2	Remote Fault 2	<p>This bit, in combination with bit 12, RF1, denotes the remote fault bits. The encoding is as follows:</p> <table><tr><td>RF1</td><td>RF2</td><td></td></tr><tr><td>0</td><td>0</td><td>No error, link OK</td></tr><tr><td>0</td><td>1</td><td>Off-line</td></tr><tr><td>1</td><td>0</td><td>Link _Failure</td></tr><tr><td>1</td><td>1</td><td>Auto-Negotiation Error</td></tr></table> <p>The default value is '00'.</p>	RF1	RF2		0	0	No error, link OK	0	1	Off-line	1	0	Link _Failure	1	1	Auto-Negotiation Error
RF1	RF2																	
0	0	No error, link OK																
0	1	Off-line																
1	0	Link _Failure																
1	1	Auto-Negotiation Error																
12	RF1	Remote Fault 1	<p>This bit, in combination with bit 13, RF2, denotes the remote fault bits. The encoding is as follows:</p> <table><tr><td>RF1</td><td>RF2</td><td></td></tr><tr><td>0</td><td>0</td><td>No error, link OK</td></tr><tr><td>0</td><td>1</td><td>Off-line</td></tr><tr><td>1</td><td>0</td><td>Link _Failure</td></tr><tr><td>1</td><td>1</td><td>Auto-Negotiation Error</td></tr></table> <p>The default value is '00'.</p>	RF1	RF2		0	0	No error, link OK	0	1	Off-line	1	0	Link _Failure	1	1	Auto-Negotiation Error
RF1	RF2																	
0	0	No error, link OK																
0	1	Off-line																
1	0	Link _Failure																
1	1	Auto-Negotiation Error																
11:9		Reserved	Always 0															
8	PS2	Pause 2	<p>This bit, in combination with bit PS1, indicates the pause capability. The encoding is as follows:</p> <table><tr><td>PS1</td><td>PS2</td><td></td></tr><tr><td>0</td><td>0</td><td>No Pause</td></tr><tr><td>0</td><td>1</td><td>Asymmetric Pause towards Link Partner</td></tr><tr><td>1</td><td>0</td><td>Symmetric Pause</td></tr><tr><td>1</td><td>1</td><td>Both Symmetric Pause and Asymmetric Pause towards Link Partner</td></tr></table> <p>The default value is '00'</p>	PS1	PS2		0	0	No Pause	0	1	Asymmetric Pause towards Link Partner	1	0	Symmetric Pause	1	1	Both Symmetric Pause and Asymmetric Pause towards Link Partner
PS1	PS2																	
0	0	No Pause																
0	1	Asymmetric Pause towards Link Partner																
1	0	Symmetric Pause																
1	1	Both Symmetric Pause and Asymmetric Pause towards Link Partner																
7	PS1	Pause 1	<p>This bit, in combination with bit PS2, indicates the pause capability. The encoding is as follows:</p> <table><tr><td>PS1</td><td>PS2</td><td></td></tr><tr><td>0</td><td>0</td><td>No Pause</td></tr><tr><td>0</td><td>1</td><td>Asymmetric Pause towards Link Partner</td></tr><tr><td>1</td><td>0</td><td>Symmetric Pause</td></tr><tr><td>1</td><td>1</td><td>Both Symmetric Pause and Asymmetric Pause towards Link Partner</td></tr></table> <p>The default value is '00'</p>	PS1	PS2		0	0	No Pause	0	1	Asymmetric Pause towards Link Partner	1	0	Symmetric Pause	1	1	Both Symmetric Pause and Asymmetric Pause towards Link Partner
PS1	PS2																	
0	0	No Pause																
0	1	Asymmetric Pause towards Link Partner																
1	0	Symmetric Pause																
1	1	Both Symmetric Pause and Asymmetric Pause towards Link Partner																
6	HD	Half Duplex	<p>When this bit is set, the device is capable of operating in Half Duplex mode. The default value is '0'. Please note that in case Auto-Negotiation is enabled, this bit must be set according to the REG_FULL_DUPLEX input (that is the negated value of REG_FULL_DUPLEX).</p>															
5	FD	Full Duplex	<p>When this bit is set, the device is capable of operating in Full Duplex mode. The default value is '0'. Please note that in case Auto-Negotiation is enabled, this bit must be set according to the REG_FULL_DUPLEX input (that is, the same value as REG_FULL_DUPLEX).</p>															
4:0			Always 0															

24.7.24.6 GPCS Auto Negotiation Link Partner Base Page Ability Register (GPCSx_ANLR)

The GPCS auto negotiation link partner base page ability register (GPCSx_ANLR) provides auto negotiation block function to store the link partner abilities. GPCSx_ANLR is read-only register, and the GPCS uses it to combine with the link partner abilities. *Figure 24-43* describes the GPCSx_ANLR register bits.

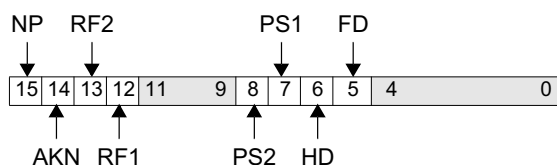


Figure 24-43. GPCS Auto Negotiation Link Partner Base Page Ability Register (GPCSx_ANLR)

15	NP	NextPage	When this bit is set to '1', the next page transmission is requested. The default value is '0'.
14	AKN	Aknowledge	When this bit is set, the Link Partner has successfully received at least three consecutive and matching rx_Config_Reg<D15:D0> values (ignoring the Acknowledge bit value). The default value is '0'.
13	RF2	Remote Fault 2	This bit, in combination with bit 12, RF1, denotes the remote fault bits. The encoding is as follows: RF1 RF2 0 0 No error, link OK 0 1 Off-line 1 0 Link_Failure 1 1 Auto-Negotiation Error The default value is '00'.
12	RF1	Remote Fault 1	This bit, in combination with bit 13, RF2, denotes the remote fault bits. The encoding is as follows: RF1 RF2 0 0 No error, link OK 0 1 Off-line 1 0 Link_Failure 1 1 Auto-Negotiation Error The default value is '00'.
11:9		Reserved	Always 0
8	PS2	Pause 2	This bit, in combination with bit PS1, indicates the pause capability. The encoding is as follows: PS1 PS2 0 0 No Pause 0 1 Asymmetric Pause towards Link Partner 1 0 Symmetric Pause 1 1 Both Symmetric Pause and Asymmetric Pause towards Link Partner The default value is '00'
7	PS1	Pause 1	This bit, in combination with bit PS2, indicates the pause capability. The encoding is as follows: PS1 PS2 0 0 No Pause 0 1 Asymmetric Pause towards Link Partner 1 0 Symmetric Pause 1 1 Both Symmetric Pause and Asymmetric Pause towards Link Partner The default value is '00'

Preliminary User’s Manual

6	HD	Half Duplex	When this bit is set, the device is capable of operating in Half Duplex mode. The default value is '0'.
5	FD	Full Duplex	When this bit is set, the device is capable of operating in Full Duplex mode. The default value is '0'.
4:0			Always 0

24.7.24.7 GPCS Auto Negotiation Expansion Register (GPCSx_ANER)

GPCS auto negotiation expansion register (GPCSx_ANER) is used by the auto-negotiation block function to indicate that a new base page has been written to the auto-negotiation link partner base page ability register. Figure 24-44 describes the GPCSx_ANER register bits.

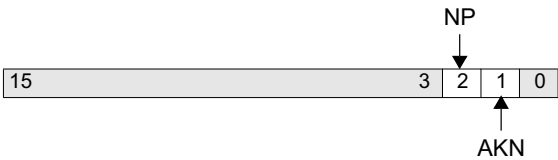


Figure 24-44. GPCS Auto Negotiation Expansion Register (GPCSx_ANER)

15:3		Reserved	Always 0
2	NP	Next Page Able	Indicates that the GPCS unit has the ability to engage in Next Page transactions. Always set to '1'.
1	AKN	Page Received	When this bit is set, a new page has been received and stored in the applicable Auto-Negotiation Link Partner Ability register (Base Page or Next Page). It is reset each time this register is read via the Management interface or by PCS reset. The default value is '0'.
0		Reserved	Always 0

24.7.24.8 GPCS Auto Negotiation Next Page Transmit Register (GPCSx_ANPTR)

GPCS auto negotiation next page transmit register (GPCSx_ANPTR) contains the next page link code word to be transmitted when the next page ability is enabled. *Figure 24-45* describes the GPCSx_ANPTR register bits.

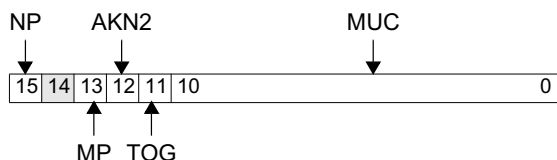


Figure 24-45. GPCS Auto Negotiation Next Page Transmit Register (GPCSx_ANPTR)

15	NP	Next Page Enable 0 Last page 1 Additional Next Page(s) will follow	Used by the Next Page function to indicate whether or not this is the last Next Page to be transmitted. ‘
14		Reserved	Write as 0. Ignore on read
13	MP	Message Page 0 Unformatted Page 1 Message Page (default value)	Used by the Next Page function to differentiate a Message Page from an Unformatted Page.
12	AKN2	Aknowledge 2 0 Cannot comply with message (default value) 1 Will comply with message	Used by the Next Page function to indicate that a device has the ability to comply with the message.
11	TOG	Toggle	Used to ensure synchronization with the Link Partner during Next Page exchange. This bit shall always take the opposite value of the Toggle bit in the previously exchanged Link Code Word. The initial value of the Toggle bit in the first Next Page transmitted is the inverse of bit 11 in the base Link Code Word.
10:0	MUC	Message Unformatted Code	Used by the Next Page function to carry a single predefined Message Code. The default value is “0000000001”

Preliminary User's Manual**24.7.24.9 GPCS Auto Negotiation Next Page Transmit Register (GPCSx_ANLNPR)**

GPCS auto negotiation next page transmit register (GPCSx_ANLNPR) is used to store link partner next pages. Figure 24-46 describes the GPCSx_ANLNPR register bits.

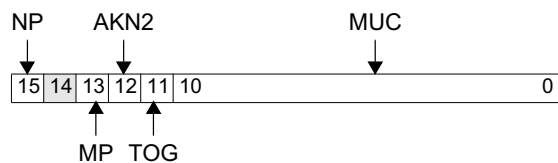


Figure 24-46. GPCS Auto Negotiation Next Page Transmit Register (GPCSx_ANLNPR)

15	NP	Next Page Enable 0 Last page 1 Additional Next Page(s) will follow	Used by the Next Page function to indicate whether or not this is the last Next Page to be transmitted. ‘
14		Reserved	Write as 0. Ignore on read
13	MP	Message Page 0 Unformatted Page 1 Message Page (default value)	Used by the Next Page function to differentiate a Message Page from an Unformatted Page.
12	AKN2	Aknowledge 2 0 Cannot comply with message (default value) 1 Will comply with message	Used by the Next Page function to indicate that a device has the ability to comply with the message.
11	TOG	Toggle	Used to ensure synchronization with the Link Partner during Next Page exchange. This bit shall always take the opposite value of the Toggle bit in the previously exchanged Link Code Word. The initial value of the Toggle bit in the first Next Page transmitted is the inverse of bit 11 in the base Link Code Word.
10:0	MUC	Message Unformatted Code	Used by the Next Page function to carry a single predefined Message Code. The default value is “0000000001”

24.7.24.10 GPCS Extended Status Register (GPCSx_ESR)

The GPCS extended status register (GPCSx_ESR), along with the GPCS status register (GPCSx_ESR), describes the GPCS current status. *Figure 24-47* describes the GPCSx_ESR register bits.

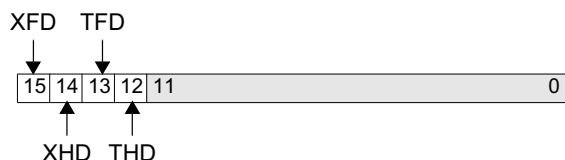


Figure 24-47. GPCS Extended Status Register (GPCSx_ESR)

15	XFD	1000BASE-X Full Duplex	When read as '1', indicates that the PHY has the ability to perform full duplex link transmission and reception. The value of this bit will be adhered to by the PHY_FD sideband signal.
14	XHD	1000BASE-X Half Duplex	When read as '1', indicates that the PHY has the ability to perform half duplex link transmission and reception. The value of this bit will be adhered to by the PHY_HD sideband signal.
13	TFD	1000BASE-T Full Duplex	Always 0
12	THD	1000BASE-T Half Duplex	Always 0
11:0		Reserved	Always 0

24.7.24.11 GPCS Resolved Ability Register (GPCSx_RAR)

The GPCS resolved ability register (GPCSx_RAR) contains the selected operation modes after the auto negotiation block function has resolved the operation modes using the priority resolution function. *Figure 24-48* describes the GPCSx_RAR register bits.

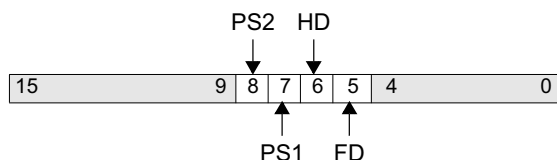


Figure 24-48. GPCS Resolved Ability Register (GPCSx_RAR)

15:9		Reserved	Always 0
8	PS2	Pause 2	This bit reflects the corresponding bit in the Link Partner ability base page register. The default value is '0'.
7	PS1	Pause 1	Pause, bit PS1. This bit reflects the corresponding bit in the Auto-Negotiation Link Partner Base Page Ability register. The default value is '0'.
6	HD	Half Duplex	If this bit is set, the local device will operate in Half Duplex mode.
5	FD	Full Duplex	If this bit is set, the local device will operate in Full Duplex mode.
4:0		Reserved	Always 0

Preliminary User's Manual

GPCS Interrupt Status Register (GPCSx_ISR)

The GPCS interrupt status register (GPCSx_ISR) is a Read/Write_One_for_Reset register (R/WOR). Writing to the interrupt status register can be done only to reset an interrupt cause, and this is done by writing '1' to the interrupt cause. Writing '0' to a bit in the Interrupt Status register will have no effect on the bit. Writing '1' to a bit in the Interrupt Status register which contains a value of '0', will have no effect on the bit.

Note: It is important to disable LP_NO_Response, LP_Zeroes_Response, and AN_Exceed_LTP via the Interrupt Status Enable register, so that these bits will not cause an interrupt. This is due to unique hardware functionality.

Figure 24-49 describes the GPCSx_ISR register bits.

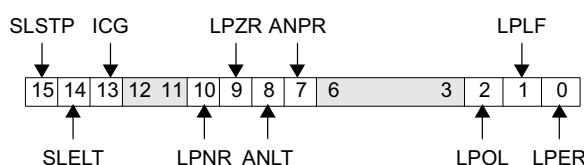


Figure 24-49. GPCS Interrupt Status Register (GPCSx_ISR)

15	SLSTP	Synchronization Loss for Short Time Period. 0 Synchronization loss has not occurred 1 Synchronization loss has occurred	Reset value '0'.
14	SLELT	Synchronization Loss Exceeds Link Timer Period 0 Synchronization loss does not exceed link timer period 1 Synchronization loss exceeds link timer period	Reset value '0'.
13	ICG	Invalid code-group received 0 GPCS unit has not received an invalid code-group 1 GPCS unit has received an invalid code-group	When Auto-Negotiation is enabled, receiving an invalid code-group causes the GPCS unit to renegotiate. Reset value '0'.
12:11		Reserved	Always 0
10	LPNR	Link Partner No Response 0 Link partner responds to GPCS negotiation attempts 1 Link partner does not respond to GPCS negotiation attempts	Reset value '0'.
9	LPZR	Link Partner Zeroes Response 0 Link partner zeroes response duration is shorter than the GPCS unit link timer period. 1 Link partner zeroes response duration is greater than the GPCS unit link timer period.	Reset value '0'.
8	ANLT	Auto Negotiation Exceeds Link Timer Period 0 Auto negotiation does not exceed link timer period 1 Auto negotiation exceeds link timer period	When GPCS0_ISR[ANLT=1] auto-negotiation state machine is non-responsive for a time duration greater than the link timer period. Reset value '0'.
7	ANPR	Auto Negotiation Resolve Priority Error 0 Auto negotiation resolve priority error not detected 1 Auto negotiation resolve priority error detected	Auto-Negotiation Resolve Priority Error recognized by the GPCS unit. GPCS unit cannot resolve abilities with its Link Partner. Reset value '0'.
6:3		Reserved	Always 0
2	LPOL	Link Partner is off-line. 0 Link partner is not off-line 1 Link partner is off-line	Reset value '0'.

1	LPLF	Link Partner Link Failure 0 Link partner link successful 1 Link partner detected link failure	Reset value '0'.
0	LPER	Link Partner Auto Negotiation Error 0 Link partner auto negotiation error has not occurred 1 Link partner auto negotiation error has occurred	When GPCS0_ISR[LPER=1] Link Partner advertises that it cannot resolve abilities with the GPCS unit. Reset value '0'.

24.7.24.12 GPCS Interrupt Status Enable Register (GPCSx_ISR)

Figure 24-50 describes the GPCSx_ISR register bits.

Note: It is important to disable LP_NO_Response, LP_Zeroes_Response, and AN_Exceed_LTP via the Interrupt Status Enable register, so that these bits will not cause an interrupt. This is due to unique hardware functionality.

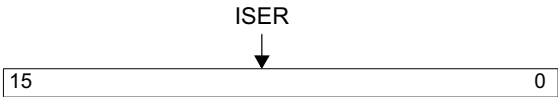


Figure 24-50. GPCS Interrupt Status Enable Register (GPCSx_ISR)

15:0	ISER	ISER	Each bit in the Interrupt Status Enable register corresponds to an associated bit in the Interrupt Status register. Status Enable bits in the Interrupt Status Enable register for reserved Status bits in the Interrupt Status Register are not implemented. These bits have no effect on write and return a logic '0' on read. Reset value '0'.
------	------	------	---

Preliminary User's Manual**24.7.24.13 GPCS Configuration Register (GPCSx_CFG)**

GPCS configuration register (GPCSx_CFG) is used by the SMU to configure the GPCS unit to desirable operation modes as follows:

- Activate a renegotiation function prior to changing the operation mode.
- Operate in different Link Timer periods.

GPCSx_CFG also controls the link timer period to one of four given periods.

Note: The contents of the GPCSx_CFG can be changed only while the GPCS unit is in sleep mode. An attempt to do so at any other time can result in unexpected GPCS unit behavior.

Figure 24-51 describes the GPCSx_CFG register bits.

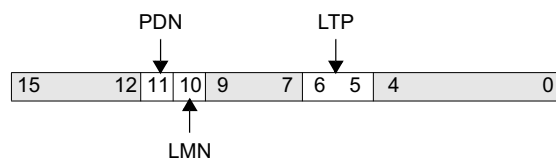


Figure 24-51. GPCS Configuration Register (GPCSx_CFG)

15:12		Reserved	
11	PDN	Power Down Negotiation 0 GPCS does not renegotiate before powering-down or isolating from GMII 1 GPCS renegotiates to announce to the Link Partner that it is going off-line before powering-down or isolating from GMII.	Reset value is '0'.
10	LMN	Loop Mode Negotiation 0 GPCS does not renegotiate before changing to the LoopBack mode. 1 GPCS renegotiates to announce to the Link Partner that it is going off-line before changing to the LoopBack mode	Reset value is '0'.
9:7		Reserved	Read as 0
6:5	LTP	Link Timer Period	The value of these bits determines the period of the GPCS Link Timer. bit 6 bit 5 Link Timer Period 0 0 10 ms (+ 10 ms, -0 ms) 0 1 5 ms (+ 5 ms, -0 ms) 1 0 1 ms (+ 1 ms, -0 ms) 1 1 1 us (+ 1 us, -0 us) The reset value is '00'
4:0		Reserved	Read as 0

24.8 MII/GMII Interface

EMAC implements all MII functionality in accordance with Clause 22 in the IEEE Std. 802.3u (when operating in 10/100 Mbps media) and all GMII interface functionality in accordance with Clause 35 in the IEEE Std. 802.3z (when operating in 1000 Mbps media).

The MII/GMII interface is a reconciliation sublayer interface which allows a variety of PHYs to be attached to the EMAC Ethernet MAC without future upgrade problems.

24.8.1 MII Station Management Interface

The EMAC MII station management unit (STA) implements a specific protocol and a special packet format to exchange management packets with the registers of the attached PHY. EMAC automatically generates MII management packets, which conform to Clause 22 in IEEE Std. 802.3u. EMAC uses the EMACx_STACR for generation of the management packet. illustrates the interface.

24.9 MAL – EMAC Packet Transfer Flow

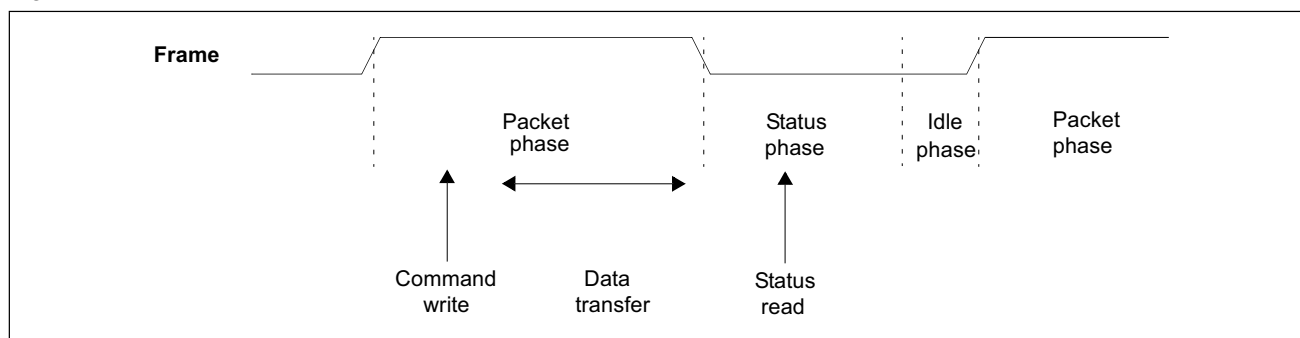
The packet transfer flow consists of three phases. These three phases are used to define the details of the EMAC-MAL protocol.

1. Packet phase - EMAC initiates a packet transfer operation. The packet transfer is started by a command write. During command write MAL provides control information for EMAC on a per-packet basis. Following the command write, MAL begins the data transfer, during which MAL transfers data between the buffers located in the system's memory and EMAC. In transmit, the data is transferred from the system's memory to EMAC, while in receive, the data is transferred from EMAC to the system's memory buffers.
 - EMAC remains in the packet phase until the data transfer has been completed or a ready status can be returned to MAL. The packet phase ends when EMAC deasserts the FRAME signal associated with the related channel (receive/transmit).
 - The packet phase is defined by activity of an appropriate FRAME signal.
2. Status phase - This is the second phase of the packet transfer. Following the de-assertion of the FRAME signal, EMAC switches to the status phase. At this stage, EMAC uses an appropriate signal as a request for service which is interpreted by MAL as a request for status read.
3. Idle phase - EMAC moves into the idle phase following a reset or after status was transferred (end of status phase). During the idle phase, EMAC cannot send any signals to MAL, nor can MAL send any active signals to EMAC. EMAC exits the idle phase by asserting the FRAME signal (and entering the packet phase described above). Idle phase can be skipped when EMAC operates in multiple transfer mode.

Preliminary User's Manual

Figure 24-52 illustrates the different phases in the EMAC-MAL communication.

Figure 24-52. EMAC-MAL Communication Phases



During the packet and status phases EMAC signals a request for service by driving its arbitration level signal to a non-idle level.

24.10 Programming Notes

Certain combinations in device drivers are not allowed when writing to EMAC registers. When creating device drivers, ensure that the following guidelines are used:

- In half-duplex mode (EMACx_MR1[FDE] = 0) while working in Gigabit Ethernet mode 1000 Mbps (EMACx_MR1[MF] = 10), the transmit FIFO size (EMACx_MR1[TFS]) must be greater than 512 bytes.
- When internal loopback is enabled (EMACx_MR1[ILE] = 1), EMAC must be configured in full-duplex mode (EMACx_MR1[FDE] = 1)
- EMACx_MR1[IST] = 0 only when EMACx_MR1[MF] = 0:10 and EMACx_MR1[FDE] = 0
- EMACx_MR1[EIFC] = 0 if EMACx_MR1[FDE] = 0
- EMACx_TMR1[TLR] must be greater than the MAL burst size in entities (6 for MAL)
- EMACx_TMR1[TUR] must be greater or equal to EMACx_TMR1[TLR] and less than the Transmit FIFO size in entries (EMACx_MR1[TFS])
- To avoid deadlock, the sum of EMACx_TMR1[TLR] and the EMACx_TRTR[TRT] must be at least four less than the Transmit FIFO size specified in EMACx_MR1[TFS]
- EMACx_RWMR[RLWM] must be greater than the MAL burst size in entities (six for MAL)
- EMACx_RWMR[RHWM] must be greater than EMACx_RWMR[RLWM]
- EMACx_RWMR[RHWM] must be less than the Receive FIFO size in entities (EMACx_MR1[RFS])
- If EMACx_MR1[MF] = 11 (the internal GPCS is used), the internal GPCS must be reset by the EMACx_STACR.

24.10.1 Reset and Initialization

The EMAC must be initialized after a reset, or before performing configuration changes. The following types of reset operations can be applied to EMAC.

- **Hard Reset.** When RESET input is asserted, EMAC aborts all on-going activities unconditionally, initializes all internal state machines, counters, registers, and flushes transmit and receive FIFOs. To be recognized, the reset signal must be asserted for at least two cycles of the slowest clock domain inside EMAC (indicating that the hard reset must be at least 800 ns).
- **Soft Reset.** Software first should reset the appropriate MAL channels and then begin a soft reset by setting EMACx_MR0[SRST] = 1. In response to the soft reset, EMAC aborts all on-going activities unconditionally, initializes all internal state machines, counters, registers, and flushes transmit and receive FIFOs. After EMAC finishes all activities related to the soft reset processing, EMACx_MR0[SRST] = 0.
- **Smart Reset.** The software initializes smart reset mode by writing 0 to EMACx_MR0[TXE] or EMACx_MR0[RXE], or to both. In this case, the Ethernet MAC sub-block completes on-going activity (receive, transmit, or both) and then goes to the related Idle state (indicated by setting either EMACx_MR0[TXI] = 1 or EMACx_MR0[RXI] = 1, or both). In this case, the control logic sub-block of EMAC is still accessible for OPB and MAL transactions.

Before performing the necessary configuration changes in EMAC, the software must follow one of the following scenarios. Then the EMAC can be properly configured.

24.10.1.1 Scenario 1

- Hard/soft reset was activated.
- During hard/soft reset, EMACx_MR0[TXE] and EMACx_MR0[RXE] are reset.
- Software detects that the EMACx_MR0[SRST] is reset (after soft reset only).
- Software keeps EMACx_TMR0[GNP] = 0.
- The software can change one or more fields in registers marked with a Reset write access mode in *Table 24-5 EMAC0 Register Summary* on page 837 (actually, all EMAC registers are accessible in this scenario).
- The software initializes EMACx_TMR0[GNP] as appropriate.
- The software configures EMACx_MR0[TXE, RXE].

Preliminary User's Manual

24.10.1.2 Scenario 2

- Software sets EMACx_MR1[EIFC] = 0.
- Software sets EMACx_MR0[TXE] = 0.
- The TXMAC component of the Ethernet MAC sub-block completes on-going activity and then sets EMACx_MR0[TXI] = 1 to enter the related Idle state.
- Software detects EMACx_MR0[TXI] = 1.
- Software performs the necessary EMAC configuration, keeping EMACx_MR0[TXE] = 0. The software can access only part of the EMAC registers marked with write access mode T in *Table 24-5 EMAC0 Register Summary* on page 837.
- After all configuration is done, software can set EMACx_MR0[TXE] = 1.

Note: When Scenario 2 occurs, EMAC can still receive packets if EMACx_MR0[RXE] = 1. Scenarios 2 and 3 can occur simultaneously.

24.10.1.3 Scenario 3

- Software sets EMACx_MR0[RXE] = 0.
- The RXMAC component of the Ethernet MAC sub-block completes on-going activity and then sets EMACx_MR0[RXI] = 1 to enter the related Idle state.
- Software detects EMACx_MR0[RXI] = 1.
- Software performs the necessary EMAC configuration, keeping EMACx_MR0[RXE] = 0. The software can access only part of EMAC registers marked with write access mode R in *Table 24-5 EMAC0 Register Summary* on page 837.
- After all configuration is done, software can set EMACx_MR0[RXE] = 1.

Note: When Scenario 3 occurs, EMAC can still transmit packets if EMACx_MR0[TXE] = 1. Scenarios 2 and 3 can occur simultaneously.

Preliminary User's Manual

25. TCP/IP Acceleration Hardware

The PPC440GX provides two TCP/IP acceleration hardware controllers (TAHs) that are inserted between the memory access layer (MAL) and the two gigabit ethernets (EMAC2 and EMAC3). PPC440GX implements four ethernets (EMAC0 and EMAC1 which support 10/100 Mbps and EMAC2 and EMAC3 which support 10/100/1000 Mbps operations). Both TAHs are implemented identically with the exception of register addresses. Because both TAHs operate identically, this chapter uses the word TAH to refer to TAH0 and TAH1. Please note that the hardware acceleration functions apply to EMAC2 and EMAC3 only. Hence the word EMAC in this chapter denotes EMAC2 and EMAC3.

25.1 Overview

TAH provides hardware acceleration functions for EMAC2 and EMAC3 (which support 10/100/1000 Mbps operations) to improve bandwidth and lower PPC440GX processor core utilization. No acceleration functions are available for EMAC0 and EMAC1 which support 10/100 Mbps operations. It provides checksum verification for TCP/UDP/IP headers in the receive path, checksum generation for TCP/UDP/IP headers in the transmit path, and TCP segmentation support in the transmit path. To utilize the acceleration function in TAH, the IP diagram must use the ethernet encapsulation (RFC894) or IEEE802.2/802.3 encapsulation (RFC1042). TAH can handle VLAN-tagged frames and support standard and jumbo packets.

MAL is an intermediate processing layer which manages data transfers between the TAH (or the EMAC if no TAH is present) and memory. MAL performs functions such as arbitration between service requests, handling the buffer descriptor memory structure, and updating the descriptor status/control field at the end of packet transfer. A software device, such as the TCP/IP protocol stack, uses the buffer descriptor to inform the MAL about buffer locations and packet or buffer status. The MAL uses the buffer descriptors to convey packet transfer status from the EMAC back to the protocol stack. For more information see *Memory Access Layer (MAL)* on page 751.

For receive packets, setting the Checksum Verification on Receive (CVR) bit in the accelerate mode register, TAHx_MR, enables hardware checksum verification for all incoming packets for a given EMAC/TAH.

For transmit packets, hardware generated checksums and/or packet segmentation is done on a per-packet basis. This is accomplished by setting the proper bits in the descriptor control/status field of the buffer descriptor. The size that packets should be segmented into (if segmentation is enabled) is controlled by one of six Segment Size Registers (TAHx_SSRx). Bits in the buffer descriptor also determine which SSR is used.

EMAC is a generic implementation of the Ethernet Media Access Control (EMAC) protocol compliant with ANSI/IEEE Std 802.3 and IEEE 802.3u supplement. EMAC0 and EMAC1 support both Half Duplex (CSMA/CD) and Full Duplex operation. When EMAC operates in the 10/100 Mbps range, it communicates with the standard physical (PHY) device using a Media Independent Interface (MII). EMAC2 and EMAC3 support operation in the 1000 Mbps range (Gigabit Ethernet) compliant with IEEE Standard 802.3z. In this mode, the real bandwidth is limited only by system latency (attributable to MAL, memory, arbitration, etc.). A Gigabit MII (GMII) interface is used in the Gigabit Ethernet mode for communication with the PHY device. EMAC2 and EMAC3 also include a Gigabit Ethernet PHY Coding Sublayer (GPCS4). The use of the GPCS4 is optional and it allows EMAC2 and EMAC3 to be connected to the PMA interface and thus to a Gigabit Ethernet layer directly. For more information see *Ethernet Media Access Controllers* on page 815.

TAH interfaces to MAL as an Extended OPB (EOPB) slave on behalf of EMAC, and interfaces to EMAC as an Extended OPB (EOPB) master on behalf of MAL. On each EOPB interfaces, a separate receive and transmit channel is supported. It also provides an on-chip peripheral bus (32-bit bi-directional OPB) slave interface for access to the configuration and status registers. TAH maintains a transmit FIFO to support the hardware checksum function on the transmit side. Various FIFO sizes are supported.

Network bandwidth has increased at a much faster pace than processor and memory performance. Because of these limitations, CPU utilization can easily reach 100% long before the network is saturated. Although 1Gb Ethernet is ten times faster than 100Mb, the overhead of packet processing in software may result in only a 3x or 4x increase in overall throughput. By off-loading some of the packet processing that is traditionally done by software, significant performance increases can be realized. The TCP Acceleration Hardware provides this capability in the PPC440GX Embedded Processor.

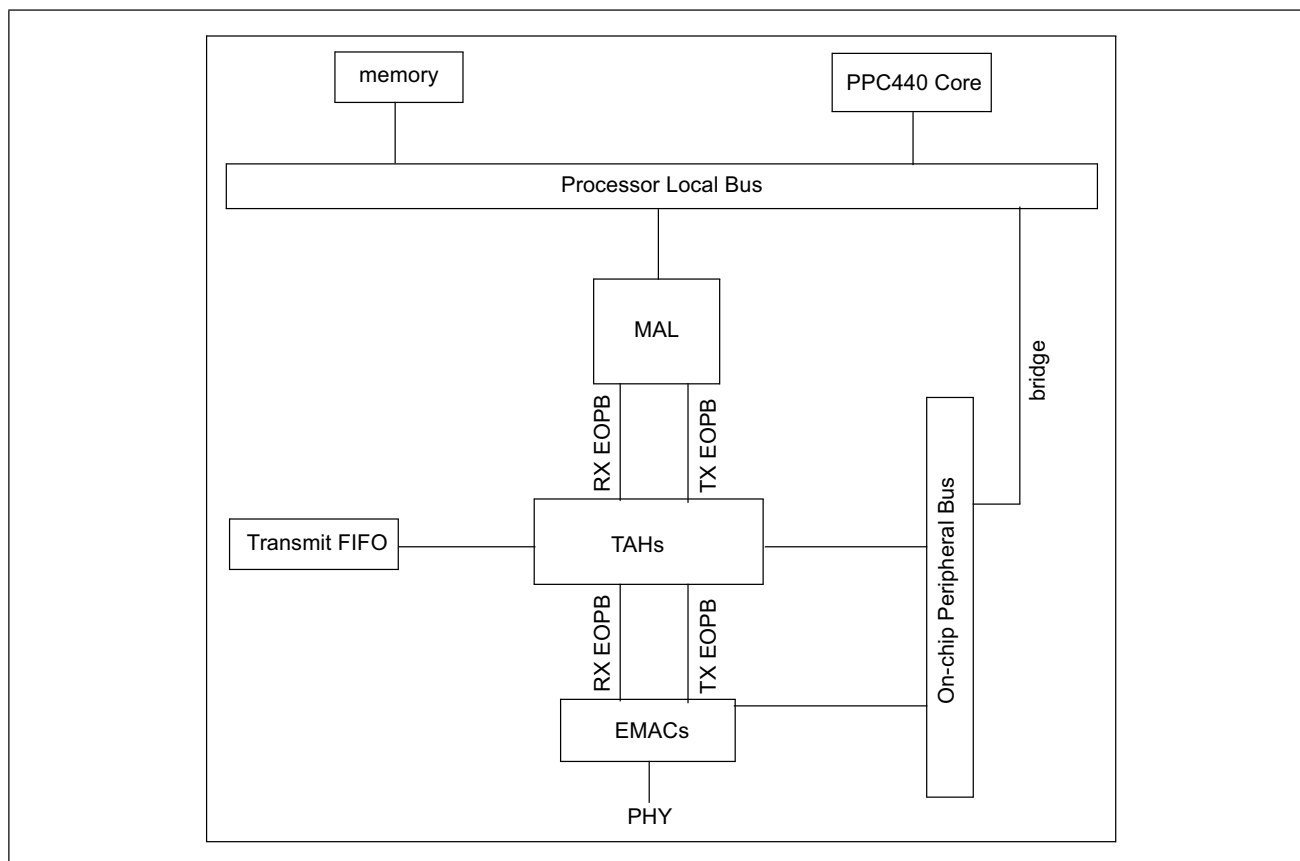
25.1.1 Features

- Verifies checksum for the UDP/TCP/IP headers in the receive path
- Generates checksum for the UDP/TCP/IP headers in the transmit path
- Supports TCP segmentation in the transmit path
- Supports IPv4 headers
- Supports ethernet encapsulation (RFC894)
- Supports IEEE 802.2/802.3 encapsulation (RFC1042)
- Supports VLAN tagged frames according to IEEE 802.3ac
- Inserts/Replaces VLAN Tag for transmit packets (programmable option)
- Inserts/Replaces MAC source address for transmit packets (programmable option)
- Inserts Frame Check Sequence (FCS) control for the transmitted/received packets
- Provides OPB interface for access to internal registers (up to 167 MHz of operational frequency)
- Provides two 128-bit MAL EOPB interfaces (separate transmit and receive channels) with the MAL for moving packets (operates from 33 to 167 Mhz)
- Provides two 128-bit MAL EOPB interfaces (separate transmit and receive channels) with the EMAC for moving packets (operates from 33 to 167 Mhz)
- Supports different transmit FIFO sizes (2K bytes to 16K bytes)
- Supports multiple status mechanism (called “back to back TX status mechanism”) which allows TAH to delay the sending of the current status word or to piggyback the previous status word with the current status word

Preliminary User's Manual

Figure 25-1 illustrates a general system structure overview of an embedded PowerPC processor core integrated with TCP/IP acceleration hardware on ethernet core. To see how both TAHs are implemented in the PPC440GX Embedded Processor see *PPC440GX Block Diagram* on page 52.

Figure 25-1. General PPC440GX Structure with TCP/IP Acceleration Hardware



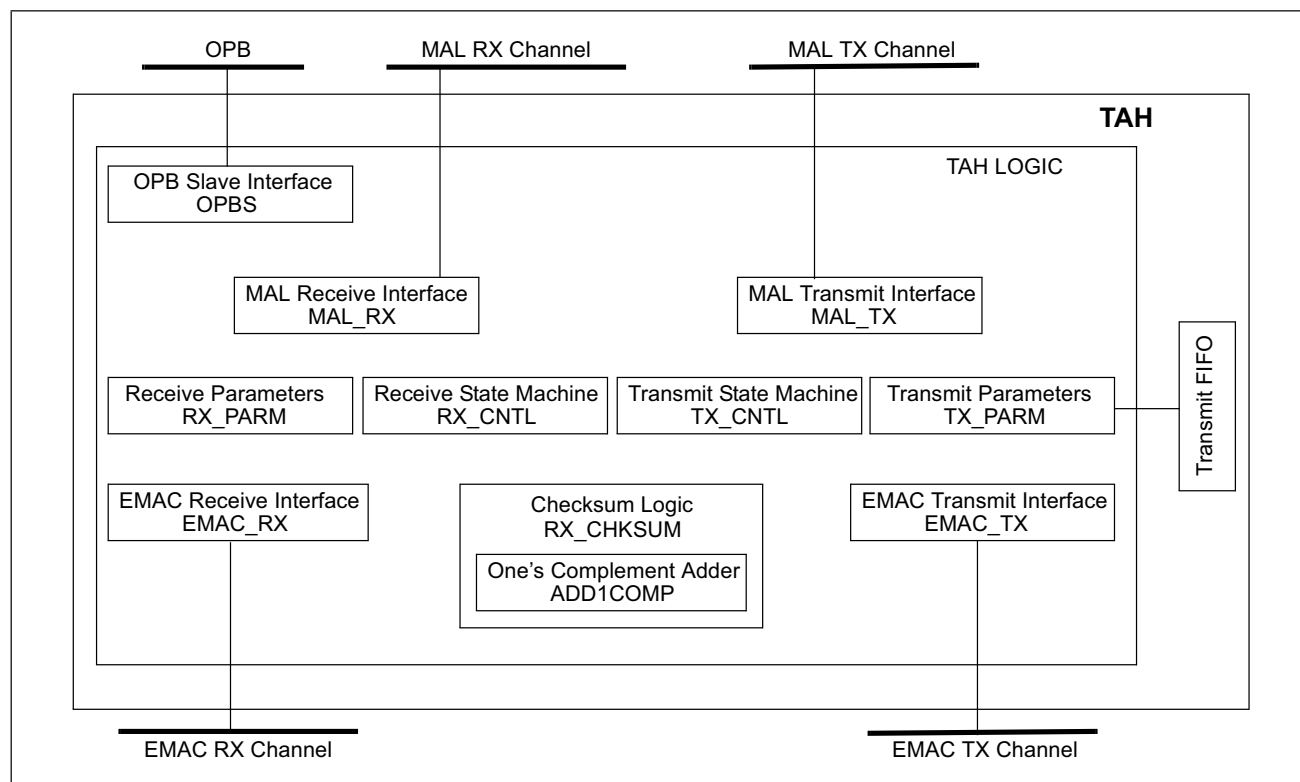
TAH may be connected to a single EOPB bus. On this bus, both the OPB master and MAL are connected as masters. The OPB master will use only the OPB function while the MAL will use the EOPB over this single EOPB bus.

TAH's transmit channel and receive channel operate independently. This means that each of these channels may be connected to a different MAL instance, possibly on a different EOPB (this option can increase system performance especially when Gigabit mode is chosen).

25.2 TAH Operations

The TAH hardware components and its internal structure are illustrated in *Figure 25-2* followed by a brief description of its components.

Figure 25-2. Internal TAH Structure



25.2.1 TAH Hardware Components

TAH hardware components consist of:

- EOPB slave device to MAL
- EOPB master device to EMAC
- Hardware acceleration logic on the transmit path
- Hardware acceleration logic on the receive path
- OPB slave device

25.2.1.1 EOPB Slave Logic to MAL

The MAL slave logic consists of two finite state machines. MAL_TX is the finite state machine for the transmit path and MAL_RX is the finite state machine for the receive path. TAH is not a generic EOPB slave, but rather a dedicated MAL slave.

Preliminary User's Manual

25.2.1.2 EOPB Master Logic to EMAC

As a surrogate for MAL, TAH provides the EOPB master logic in two finite state machines. EMAC_TX is the finite state machine for the transmit path and EMAC_RX is the finite state machine for the receive path. TAH is not a generic EOPB master, but rather a dedicated EMAC master.

25.2.1.3 Hardware Acceleration Logic on the Transmit Path

This is implemented as a finite state machine (TX_CNTL) and associated parameters (TX_PARM). These modules also provide the control signals for the external Transmit FIFO. The checksum logic is contained in ADD1COMP and RX_CHKSUM.

25.2.1.4 Hardware Acceleration Logic on the Receive Path

This is implemented as a finite state machine (RX_CNTL) and associated parameters (RX_PARM). The checksum logic is contained in ADD1COMP and RX_CHKSUM.

25.2.1.5 OPB Slave (OPBS) Logic

The OPB slave (OPBS) logic handles the OPB transactions for accessing TAH's configuration and status registers. It provides two-cycle latency for single data transfer. It does not support burst transfers or the OPB Sequential Address mechanism.

25.2.2 Data Ordering - Transmit and Receive Data Path

TAH performs data transfers between the MAL EOPB and the EMAC EOPB. The EOPB operates using Big Endian format in quadword mode (128 bits). TAH preserves the data ordering between the two interfaces.

25.2.3 TAH Transmit Operation

The transmit part of TAH is responsible for frame transmission from MAL to EMAC. EMAC's TX channel can be configured to work in either of the following two ways:

1. Single packet mode: the Transmit Request bit in Mode Register 1 is '0'. The channel requests a single packet from TAH and then resets its GET_NEW_PACKET bit in the Transmit Mode Register 0 when it receives a TAH2EMAC_TX_STATUS_DONE indication. The channel will only ask for service after the GET_NEW_PACKET bit is set to '1' again.
2. Multiple packet mode: the Transmit Request bit in Mode Register 1 is '1'. Once the channel finishes transferring a packet and receives TAH2EMAC_TX_STATUS_DONE, it will ask TAH for the next packet if there is enough room in the FIFO. The channel will keep asking for more packets until either of the following events occur:
 - The channel receives TAH2EMAC_TX_DSCR_NOT_VALID at which time it will clear its GET_NEW_PACKET bit and wait for software to re-activate it by writing '1' to GET_NEW_PACKET bit.
 - A Transmit Error or an SQE occurs and the corresponding interrupt is not masked in the Interrupt Status Enable Register. After such an error, the channel clears its GET_NEW_PACKET bit and activates its DB bit and the corresponding ERROR bit in the Interrupt Status Register, as soon as it receives TAH2EMAC_TX_STATUS_DONE indication. The channel will not request service again until the GET_NEW_PACKET bit is set and the DB bit is cleared.

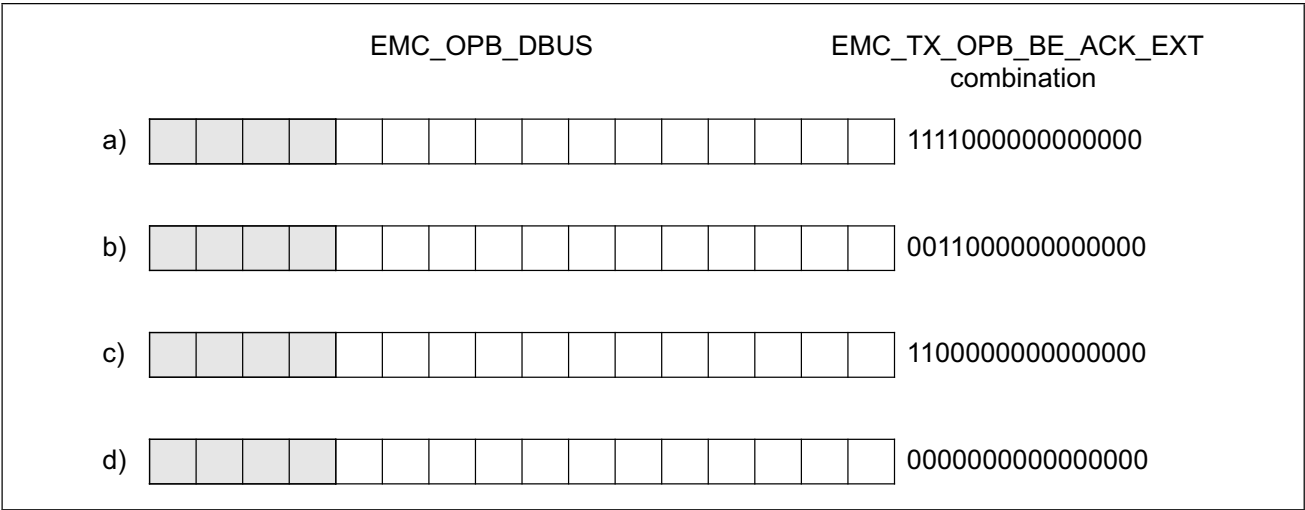
There are two situations in which the EMAC TX channel may request data-transfer from TAH.

1. If the channel is in the Idle Phase (after reset or after the channel received TAH2EMAC_TX_STATUS_DONE), EMAC may assert EMC_TX_FRAME and drive EMC_TX_ARB_LEVEL to a non-idle value.
2. If the channel requested data transfer and received TAH2EMAC_TX_DSCR_NOT_VALID, then EMC_TX_FRAME is already asserted, so when the channel is ready to receive data again, it only needs to drive EMC_TX_ARB_LEVEL to a non-idle value.

At the end of the transmission process, EMAC can be configured to either wait for every status/word before requesting a new packet, or not waiting for status/error word after packet transmission:

1. Waiting for every status/error word before requesting a new packet: The Maximum Waiting Status Words bits in Mode Register 1 are '000'. The channel waits until it gets the status word after EMC_TX_FRAME deassertion, and only after the status word is ready EMC_TX_ARB_LEVEL is driven to a non-idle value, EMC_OPB_XFER_ACK is pulsed, and EMC_TX_OPB_BE_ACK_EXT is "1100000000000000" (see Figure 25-3, case c).
2. Not Waiting for status/error word after packet transmission: The Maximum Waiting Status Words bits in Mode Register 1 are '001'. After EMC_TX_FRAME deassertion, EMC_TX_ARB_LEVEL is driven to a non-idle value, EMC_OPB_XFER_ACK is pulsed, and the EMC_TX_OPB_DBUS can only be assembled in one of the following ways as described in Figure 25-3:

Figure 25-3. EMC_OPB_DBUS Assembly



Note: Shaded boxes in Figure 25-3 indicate the location where the valid status/error words should be placed on the EMC_OPB_DBUS during the status/error/transfer. Cases a, b, c, and d are described below.

- a) - Two valid status/error words - The status/error word of the current transmit packet is placed on the first and second bytes, and the status/error word of the previous transmit packet is placed on the third and fourth bytes.
- b) - One valid status/error word of the previous transmit packet - The status/error word is placed on the third and fourth bytes. (The status of the current packet should be ready after the next transmit packet).
- c) - One valid status/error word of the current transmit packet - The status/error word is placed on the first and second bytes. (The status of the previous packet was sent).
- d) - Null status/error word - no status/error word is sent. (The status of the current packet should be ready after the next transmit packet and the previous packet was sent).

Preliminary User's Manual

The interface between TAH and MAL is similar to the interface between TAH and EMAC as described above. The only difference is in the handling of the status/error word when hardware segmentation is enabled. For more details, see the following section on hardware acceleration support.

25.2.3.1 Hardware Acceleration Support

TAH provides hardware acceleration in the transmit path in the form of hardware checksum generation and TCP segmentation support. Since TAH will modify the packet content, the application software must enable the option in the control information to perform data padding and to add FCS.

When hardware acceleration is enabled, certain restrictions are imposed on the TCP/UDP/IP headers. The restrictions on the IP header fields are listed in *Table 25-1*:

Table 25-1. IP Header Fields in Support of Hardware Acceleration Functions

	Verify Checksum on Receive	Generate Checksum only on Transmit	TCP Segmentation
Version	must be IPv4	must be IPv4	must be IPv4
Header Length	must be 5 (no options allowed)	must be 5 (no options allowed)	must be 5 (no options allowed)
Type of Service	don't care	don't care	don't care
Total Length	don't care	must be more than 40 see note	initial value must be more than 40 (see note at the bottom of the table); actual value used is generated by hardware based on selected segment size (* see note below)
ID	don't care	don't care	generated by hardware based on initial value
Flag	fragment bit must be 0	fragment bit must be 0	fragment bit must be 0
Fragment Offset	must be 0	must be 0	must be 0
Time-to-Live	don't care	don't care	don't care
Protocol	must be TCP or UDP	must be TCP or UDP	must be TCP
Header Checksum	don't care	generated by hardware	generated by hardware
Source IP Address	don't care	don't care	don't care
Destination IP Address	don't care	don't care	don't care
Options	don't care	don't care	ignored

Note: *TAH uses the total length field information in the IP header to determine the number of bytes to be processed. It is very important that the number of bytes in the buffer that are to be transmitted matches exactly the total length in the IP header. If the number of bytes are smaller, an error will be reported and the transmission terminated as soon as the error is detected. (Note that when TCP segmentation is enabled, some segments might have already been sent.) If the number of bytes are larger, the padding bytes must be zeros. Non-zero padding bytes can result in checksum errors detected at the receiving site.

The restrictions on the TCP header fields are listed in *Table 25-2*:

Table 25-2. TCP Header Fields in Support of Hardware acceleration Functions

	Verify Checksum on Receive	Generate Checksum only on Transmit	TCP Segmentation
Source Port Number	don't care	don't care	don't care
Destination Port Number	don't care	don't care	don't care
Sequence Number	don't care	don't care	generated by hardware based on initial value
Acknowledge Number	don't care	don't care	don't care
Header Length	don't care	don't care	first packet uses value provided; subsequent packets use a value of 5
Flags	don't care	don't care	see table note
Window Size	don't care	don't care	don't care
TCP Checksum	don't care	generated by hardware	generated by hardware
Urgent Pointer	don't care	don't care	don't care
Options	don't care	don't care	don't care
Note: When TCP segmentation is enabled, the TCP flags are handled as follows: <ul style="list-style-type: none"> • URG (urgent) - recreated in the first segment only; reset in all other segments • ACK (acknowledge) - recreated in all segments • PSH (push) - recreated in the last segment only; reset in all other segments • RST (reset) - must be zero • SYN (synchronize) - must be zero • FIN (finished) - recreated in the last segment only; reset in all other segments. 			

25.2.3.2 Hardware Checksum Generation

TAH generates the IP, UDP, and TCP checksums for the packet when the appropriate bits in the control information (see the section on MAL TX Control) from MAL are set to enable this feature.

When EMAC requests data transfer from TAH, TAH will in turn request data transfer from MAL. If hardware checksum generation is enabled in the control information returned from MAL, then TAH will store the subsequent data temporarily in the Transmit FIFO while it calculates the checksum. Data is not sent to EMAC until the entire packet has been transferred from MAL to TAH since the checksum must be computed over the entire packet. When data is finally sent to EMAC, the computed checksums will be placed in the appropriate header fields in the packet. TAH parses the incoming data stream to determine the type of data present. The following formats are supported:

- IP version 4
- TCP or UDP
- Ethernet II (DIX) frame format as defined in RFC 894 or IEEE 802 frame format with SNAP header as defined in RFC 1042
- VLAN as defined in IEEE 802.3ac

The size of the packet supported is dependent on the size of the Transmit FIFO. The Transmit FIFO must be at least 80 bytes bigger than the packet size.

Preliminary User's Manual

25.2.3.3 Hardware Segmentation

TAH segments the TCP packet from MAL into smaller packets using the selected Segmentation Size when the appropriate bits in the control information (see the section on MAL TX Control) from MAL are set to enable this feature. By enabling the segmentation support in hardware, the application software can transfer a large block of data, say 32K bytes, and provide a single TCP/IP header template, and have hardware segment the data into multiple packets of the desired segment size, say 1.5K bytes, and insert the necessary TCP/IP header for each packet. The desired segment size is specified in *TAH Segment Size Registers 0:5 (TAHx_SSR0-TAHx_SSR5)* on page 897.

When EMAC requests data transfer from TAH, TAH will in turn request data transfer from MAL. If hardware segmentation is enabled in the control information returned from MAL, then TAH will store the subsequent data temporarily in the Transmit FIFO while it calculates the checksum. This is similar to the procedure for supporting hardware checksum generation. The difference is in the handling of the TCP/IP headers. When segmentation is enabled, the original headers from the packet received from MAL is saved for later use.

New headers computed based on the original headers are stored in the Transmit FIFO instead. The header fields affected are described in *Hardware Acceleration Support* on page 885. One of the fields affected is the IP Length field. This is computed from the selected Segment Size. When the amount of data equal to the selected segment size has been transferred from MAL, TAH will pause the transfer to store the computed checksum in the Transmit FIFO at the location where the headers are stored. TAH will also start transferring the segmented packet from the Transmit FIFO to EMAC. At the same time, TAH will use the stored headers to create the headers for the next packet and continue the process of transferring data from MAL. This process continues until all the data from MAL has been transferred.

As the smaller packets are sent to EMAC, status will be returned for each packet. If error is reported by EMAC, then the transmit operation is terminated and the error is reported to MAL. If no error is reported and there are more segments to be sent for the original packet from MAL, then TAH will not forward the status to MAL since MAL only expects a single status for the entire packet. At the end of the segmentation operation, after the entire packet from MAL has been segmented into smaller packets and sent to EMAC, the final status from EMAC is reported to MAL. For optimum performance, the application software should set the Maximum Waiting Status Words bits in the EMAC Mode Register 1 to '001' (see *Mode Register 1 (EMACx_MR1)* on page 842) to enable Back to Back TX Status Mechanism. This allows the status reporting by GMAC to be overlapped with the data transfer between EMAC and TAH.

TAH provides hardware segmentation support for both Ethernet II format frames and IEEE SNAP format frames. In addition, when Ethernet II format frames are used, TAH also provides hardware segmentation support for Jumbo frames. Note that hardware segmentation for Jumbo frames in the IEEE SNAP format is not supported since there is no established standard.

When hardware segmentation is used for IEEE SNAP format frames, software MUST provide the correct MAC length in the original packet header based on the size of the first segment. Thereafter, hardware will fill in the correct MAC length for subsequent TCP segments.

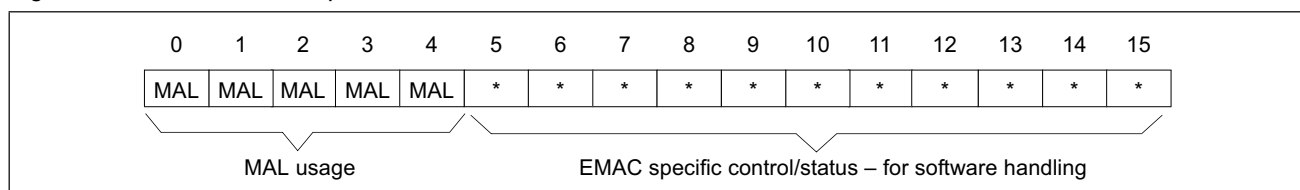
25.2.3.4 MAL Transmit Data

During transmit operations, the low order 3 bits of the MAL address bus (MAL_OPB_ABUS) define the format of the data being transferred from MAL to TAH (see *TX Buffer Descriptor MAL Control* on page 764). The MAL data bus (MAL_OPB_DBUS) contains status from TAH/EMAC for previously processed packets when the low order 3 bits of the MAL address bus are all zeros during the Status phase.

MAL TX Write Data - The MAL data bus (MAL_OPB_DBUS) contains packet data to be transmitted when the low order 3 bits of the MAL address bus are non-zero. The MAL data bus (MAL_OPB_DBUS) contains packet related control information from MAL when the low order 3 bits of the MAL address bus are all zeros during the Frame phase.

25.2.3.5 MAL TX Descriptor Control/Status Field

For each transmitted packet, MAL uses the descriptor control/status field of the buffer descriptor to provide an EMAC with control information (write), and to obtain packet status from the EMAC after transmission is complete (read). Software writes the control bits in the buffer descriptor before packet transmission, and reads the status bits from the buffer descriptor after packet transmission has completed. See *MAL Operation* on page 756 for more information on the buffer descriptor structure.

Figure 25-4. MAL TX Descriptor Control/Status Field

Bits	Bit Name	Bit Description	Mode
0:4	Reserved	Always zero	
TX Control Information (Write Access)			
5	Reserved	Always zero	W
6	Generate FCS	0 FCS is not generated by EMAC. 1 EMAC calculates and adds the FCS field to the packet to be transmitted. This bit must be set to 1 when hardware acceleration bits 12:14 are enabled.	W
7	Generate padding	0 Padding is not generated by EMAC. 1 EMAC adds the padding field to the packet to be transmitted (only when Generate FCS is also set). This bit must be set to 1 when hardware acceleration bits 12:14 are enabled.	W
8	Insert source address	0 EMAC will not insert source address. 1 EMAC inserts the source address field into the packet to be transmitted using the stored value.	W
9	Replace source address	0 EMAC will not replace source address. 1 EMAC replaces the source address field in the packet to be transmitted using the stored value.	W
10	Insert VLAN Tag	0 EMAC will not insert a VLAN tag. 1 EMAC inserts the VLAN Tag field into the packet to be transmitted using the stored value.	W
11	Replace VLAN Tag	0 EMAC will not replace the VLAN tag. 1 EMAC replaces the VLAN Tag field in the packet to be transmitted using the stored value.	W
12:14	Hardware acceleration	000 Hardware acceleration is disabled 001 Hardware checksum generation enabled TCP segmentation enabled using TAHx_SSR0 010 Hardware checksum generation enabled TCP segmentation enabled using TAHx_SSR1 011 Hardware checksum generation enabled TCP segmentation enabled using TAHx_SSR2 100 Hardware checksum generation enabled TCP segmentation enabled using TAHx_SSR3 101 Hardware checksum generation enabled TCP segmentation enabled using TAHx_SSR4 110 Hardware checksum generation enabled TCP segmentation enabled using TAHx_SSR5 111 Hardware checksum generation enabled	W
15	Reserved	Always zero	W

Preliminary User's Manual

Bits	Bit Name	Bit Description	Mode
TX Status Information (Read Access)			
5	Parity Error	0 No parity error occurred. 1 Indicates parity error occurred. This bit is set when TCP segmentation is enabled.	R
6	Bad FCS on transmitted frame	0 Packet transmission OK. 1 Indicates that bad FCS was indicated for transmitted package. This bit is set when TCP segmentation is enabled.	R
7	Transmit Error Detected	0 Packet transmission OK. 1 Indicates that hardware acceleration is enabled but an error has been detected. TAHx_TSR register specifies the source of error.	R
8	Loss of carrier sense	0 No loss of carrier. 1 During the transmission of a frame, the PHY_CRS input was de-asserted after it previously was asserted, or it was not asserted at all. This bit is set when TCP segmentation is enabled.	R
9	Excessive deferral	0 No excessive deferral. 1 Indicates that the current frame has been deferred for an excessive period of time. Applicable only in half duplex mode. The value of this period in bit times is calculated in the following ways: For 10/100 Mbps operation it is: 2 x (maxFrameSize x 8) bit times. For 1000 Mbps operation it is: 2 x (burstlimit + maxFrameSize x 8 + headerSize) bit times. This bit is set when TCP segmentation is enabled.	R
10	Excessive collisions	0 Less than 16 collisions. 1 Indicates that the current frame transmission had ended with a collision on the 16th consecutive attempt. Applicable only in half-duplex mode. Returns '0' in full-duplex mode. This bit is set when TCP segmentation is enabled.	R
11	Late collision	0 No late collision. 1 Frame collided outside of the collision window. Applicable only in half-duplex mode. Returns '0' in full-duplex mode. This bit is set when TCP segmentation is enabled.	R
12	Multiple collision	0 More than 1 but less than 16 collisions did not occur. 1 Transmitted frame collided more than once but less than 16 times. Applicable only in half-duplex mode. Does not result in any loss of data and will not be reported when TCP segmentation is enabled.	R
13	Single collision	0 Single collision did not occur. 1 Activates if transmitted frame collided once. Applicable only in half-duplex mode. Does not result in any loss of data and will not be reported when TCP segmentation is enabled.	R
14	Underrun	0 Underrun did not occur. 1 Frame transmission was aborted because of underrun; data from the Transmit FIFO was not valid in time to allow continuous data transmission on the MII/GMII interface. This bit is set when TCP segmentation is enabled.	R
15	SQE	0 Signal Quality Error did not occur. 1 Signal Quality Error test failed during packet transmission. Applicable only in half -duplex mode during 10 Mbps operation. This bit is 0 in all other modes. This bit is set when TCP segmentation is enabled.	R

TAH does not monitor MAL_TX_OPB_QW_XFER output from MAL because all MAL transfers are quadword EOPB transactions.

EMAC controls the pace of the data prefetch rate by managing the content of the EMC_TX_ARB_LEVEL bus. When hardware acceleration is not enabled, TAH replicates the EMC_TX_ARB_LEVEL signal from the TAH/EMAC EOPB to the TAH/MAL EOPB. When hardware acceleration is enabled, since TAH stores the data in a Transmit FIFO until the entire packet (or segments thereof) has been transferred, TAH only uses the "normal" setting on the TAH2MAL_TX_ARB_LEVEL bus.

If the EMAC transmit channel is idle, EMAC may request service at any time by asserting EMC_TX_FRAME. It may withdraw its request for service only when TAH2EMAC_TX_ACTIVE is asserted. Upon detecting an active request from EMAC in the idle state, TAH will in turn assert TAH2MAL_TX_FRAME to request service from MAL. In the middle of a TCP segmentation operation, TAH will not be in the idle state when EMAC asserts the service request from its idle state. This is because TAH has already received part or all of the packet from MAL and is in the process of breaking the packet into smaller segments for transfer to EMAC. In this case TAH will regenerate the control information sent earlier from MAL, and send the next segmented packet when it is ready.

25.2.3.6 Normal Packet Termination

The last data transfer cycle of the given packet can be determined by examining the least three significant bits of OPB_ABUS. The number of valid bytes within the last data cycle is specified by the content of the MAL_BE_EXT bus.

25.2.3.7 Early Packet Termination in Transmit

EMAC can initiate an early packet termination during a transmit operation before TAH completes the data transfer by deasserting EMC_TX_FRAME. This is typically used when error conditions force the EMAC to abort the transmission. Likewise, TAH can initiate an early packet termination before MAL completes the data transfer by deasserting TAH2MAL_TX_FRAME if EMAC initiates early packet termination.

During TCP segmentation operation, TAH will initiate early packet termination on the MAL interface if EMAC reports any of the following errors:

- Parity Error
- Bad FCS
- Loss of carrier sense
- Excessive Deferral
- Excessive Collisions
- Late Collision
- FIFO Underrun
- Signal Quality Error

In accordance with the OPB specifications, an early packet termination causes the master to read the packet's status and end the transmission, and the slave is allowed to initiate an early packet termination only after the master has written the control information to the slave or after the master has asserted the DSCR_NOT_VALID signal. These are implemented in both the TAH-MAL EOPB and the TAH-EMAC EOPB.

EMAC automatically retransmits frames collided on the MII/GMII interface. The Transmit FIFO always preserves the first 64 (in 10/100 Mbps) or 512 (in Gigabit Ethernet media) bytes of the packet until it receives an indication that the collision window has elapsed. Otherwise, if collision was detected within the collision window, the frame is retransmitted automatically without a new request applied to TAH. The collision information is reported in the status in the form of "Single Collision" and "Multiple Collision". Since these occurrences are not fatal, when TCP segmentation is enabled, these occurrences will not be reported.

Preliminary User's Manual**25.2.3.8 Empty Packet**

EMAC treats empty packet as if a normal packet had been written, but without writing any data to the FIFO. A valid status word with all zeros will be returned after an empty packet. If hardware acceleration is not enabled, TAH will act as a passive conduit and will pass anything between MAL and EMAC, including empty packets. If hardware acceleration is enabled, empty packets will result in an error, since the fields necessary to carry out the hardware acceleration function are missing.

EMAC expects that for quadword-aligned packets, the EOPB master activates the related indication about quadword transfer during the last data transfer, rather than providing an empty packet indication. This is indeed the case for TAH, and TAH expects the same from MAL.

25.2.4 TAH Receive Operation

The receive part of TAH is responsible for transferring packets from EMAC to MAL. If hardware acceleration is enabled, TAH will also verify the TCP/UDP/IP checksum. At the end of the reception process, TAH will insert the checksum status, if any, to the status/error word provided by EMAC before presenting the status to MAL.

25.2.4.1 Normal Operation

EMAC initiates request for service from TAH by asserting EMC_RX_FRAME, and switches the EMC_RX_ARB_LEVEL bus to a (non-idle) request level. As a response, TAH will assert TAH2MAL_RX_FRAME and drive the TAH2MAL_RX_ARB_LEVEL bus to a (non-idle) request level. MAL fetches the buffer descriptor from memory and presents the Control information to TAH by asserting the MAL_RX_ACTIVE signal and the MAL_RX_SELECT signal, and deasserting the MAL_RX_OPB_RNW signal to indicate a write transfer. The control information is passed onto EMAC by TAH using the same protocol. During MAL transactions, the low order 3 bits of OPB_ABUS is used for transaction qualifiers. The two settings defined provide for the transfer of control/status information and data.

EMAC signals its readiness to transfer data to TAH by driving the EMC_RX_ARB_LEVEL bus to a non-idle request level. Likewise, TAH will signal its readiness to transfer data to MAL by driving TAH2MAL_RX_ARB_LEVEL to a non-idle request level. Once TAH has secured the MAL EOPB by observing that MAL has asserted MAL_RX_ACTIVE, MAL_RX_SELECT, and MAL_RX_OPB_RNW, TAH will likewise assert TAH2EMAC_RX_ACTIVE, TAH2EMAC_RX_SELECT, and TAH2EMAC_RX_OPB_RNW. Data is transferred from EMAC to TAH and onto MAL. Along the way TAH will compute the checksum if so specified in the Mode Register.

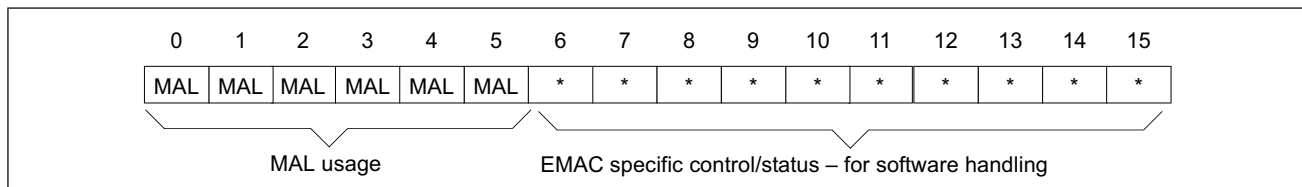
TAH by itself has no constraint on burst boundaries since it is just a passive conduit of data between EMAC and MAL. However, since MAL has a 16-cycle burst boundary, TAH must assure that MAL is available to receive data before accepting data from EMAC. This is achieved by deasserting the control signals to EMAC after every 16 cycles and monitoring the same control signals from MAL. If the control signals are still asserted by MAL, meaning that MAL is able to continue the transfer, then TAH will reassert its control signals to EMAC. Otherwise transfer from EMAC is put on hold until MAL is ready to accept more data.

25.2.4.2 MAL RX Descriptor Status

EMAC reports status to TAH by driving EMC_RX_ARB_LEVEL to the highest level. Status is transferred when the EOPB master asserts the ACTIVE and the SELECT signal.

For each packet that is received, MAL obtains status from EMAC after reception is complete, and writes this information into the buffer descriptor status/control field. Software uses this information to monitor the status of received packets. See *MAL Operation* on page 756 for more information on the buffer descriptor structure.

Figure 25-5. MAL RX Descriptor Control/Status Field



Bits	Bit Name	Bit Description	Mode
0:5	Reserved	Always zero	
RX Status Information (Read Access)			
6	Overrun Error	0 No overrun error. 1 EMAC detected an overrun error. An overrun error occurs if the flow of received data to the RX FIFO is corrupted because of insufficient empty space.	R
7	Pause Packet	0 Received packet is not a control pause packet. 1 Received packet is a control pause packet.	R
8	Bad Packet	0 No packet errors. 1 Early termination caused by packet error.	R
9	Runt Packet	0 Duration of PHY_RX_DV signal OK. 1 Duration of PHY_RX_DV signal greater than ShortEventMax Time constant and less than collision windows.	R
10	Short Event	0 Duration of PHY_RX_DV signal OK. 1 Duration of PHY_RX_DV signal was less than ShortEventMaxTime constant	R
11	Alignment Error	0 Received packet length OK. 1 Received packet length not an integral number of octets.	R
12	Bad FCS	0 FCS OK. 1 The FCS value does not match the FCS value calculated by EMAC.	R
13	Packet Too Long	0 Received packet length OK. 1 Received packet length exceeds maximum packet length. 1518 octets for standard packet. 1522 octets for VLAN tagged packet. 9018 octets for standard jumbo packet. 9022 octets for VLAN tagged jumbo packet. Data following the maximum packet length is not transferred to MAL	R
14:15	EMAC Errors	00 No errors 01 In range error 10 Out of range error 11 Checksum error if TCP/IP/UDP; otherwise checksum is not verified. Note: TAH reports checksum related errors only if checksum verification is enabled in the mode register (TAHx_MR[CVR=1]). TAH will not report checksum related errors if "In range error" or "Out of range error" is reported by EMAC.	R

25.2.4.3 Normal Packet Ending

For the last data cycle, EMAC asserts EMC_RX_LAST_DATA and indicates the valid bytes on the data bus using the EMC_RX_BE_EXT_ACK signal. TAH will repeat these signals on the MAL EOPB using TAH2MAL_RX_LAST_DATA and TAH2MAL_RX_BE_EXT_ACK signals. Following the last data cycle, both EMAC and TAH deasserts the Frame signal.

Preliminary User's Manual

25.2.4.4 Early Packet Termination

TAH does not terminate packet transmission prematurely. If EMAC exercises early packet termination, the error status will be reported to MAL in the same way as for the no error case.

25.2.5 VLAN Support

TAH is able to handle VLAN tagged frames as specified in IEEE P802.3ac.

VLAN tagged frame is an extension of the standard MAC frame. The extension for VLAN tag support consists of a 4-octet VLAN tag inserted between the end of the Source Address and the beginning of the Length/Type field of the MAC frame. This tag consists of two fields.

1. A 2-octet constant Type field value equal to the VLAN Tag Protocol Identifier (0x8100).
2. A 2-octet field containing Tag Control Information (TCI).

Following the VLAN tag is the MAC Client Data and FCS fields of the basic MAC frame. The length of the frame is extended by four octets by the VLAN tag (up to 1522 bytes maximum, and up to 9022 bytes for jumbo operation). The FCS is calculated over all fields from the Destination Address through the end of the MAC client data or Pad (if present); that is, all fields except the preamble, SFD, and FCS).

25.2.6 Jumbo Frame Support

TAH supports Ethernet II format Jumbo frames. This format does not change the definition of any Ethernet or TCP/IP fields. The difference between a standard Ethernet II frame and a Jumbo frame is in the value of the IP length field. The use of Jumbo frames is indicated by the choice of the segment size selected. TAH can support frame sizes up to 16K bytes provided the transmit FIFO is big enough.

25.2.7 Error Handling

During the transmit operation, when an error is detected by TAH and hardware acceleration is enabled, the transfer will be terminated. If the command/data has been received from MAL but has not been transferred to EMAC, then a null transfer to EMAC will be initiated to solicit any previous status outstanding. TAH will set bit 7 of the MAL TX Status port to indicate the error. In addition, an interrupt will be generated if interrupt is not disabled, and the error status is saved in the Transmit Status Register until it is read.

During the receive operation, when an error is detected during checksum verification, the error status is reported in the MAL RX Status port but the transfer is not terminated.

25.2.8 Enabling Hardware Acceleration

Hardware accelerate functions can be enabled on a per-packet basis for transmit operations, and on a per EMAC/TAH basis for receive operations. For each transmitted packet, MAL uses the descriptor control/status field of the buffer descriptor to provide control information to TAH. Hardware acceleration is enabled by setting the hardware acceleration bits (bits 12–14 as described in *MAL TX Descriptor Control/Status Field* on page 888) in the buffer descriptor to a non-zero value.

Note: If the hardware accelerate bits are not set to 000 (some form of hardware accelerate is enabled), bits 6 and 7 must also be set. Failure to do so will cause improper operation. If "Generate FCS" is not set, the FCS will be bad, and EMAC will set bit 6 "Bad FCS on transmitted frame". If "Generate padding" is not set, the frame might not meet the minimum packet size requirement of the network. If this were the case, when the frame is received by the other side, the frame would be considered a "runt frame" and will be reported as such or discarded.

25.3 TAH Registers

This section describes the TAH internal registers. The TAH registers are accessed through the OPB. Any access to the registers should be fullword aligned. Read operations from unused addresses return zeros; write operations to these addresses have no effect. All registers are cleared during power up or when the soft reset is issued except as noted.

Table 25-3. TAH Register Summary

Register	Address	Access	Description	Page
TAH0_REVID	0x1 4000 0B50	R/W	TAH0 Revision ID Register	895
TAH1_REVID	0x1 4000 0D50	R/W	TAH1 Revision ID Register	895
TAH0_MR	0x1 4000 0B60	R/W	TAH0 Mode Register	895
TAH1_MR	0x1 4000 0D60	R/W	TAH1 Mode Register	895
TAH0_SSR0	0x1 4000 0B64	R/W	TAH0 Segment Size Register 0	897
TAH1_SSR0	0x1 4000 0D64	R/W	TAH1 Segment Size Register 0	897
TAH0_SSR1	0x1 4000 0B68	R/W	TAH0 Segment Size Register 1	897
TAH1_SSR1	0x1 4000 0D68	R/W	TAH1 Segment Size Register 1	897
TAH0_SSR2	0x1 4000 0B6C	R/W	TAH0 Segment Size Register 2	897
TAH1_SSR2	0x1 4000 0D6C	R/W	TAH1 Segment Size Register 2	897
TAH0_SSR3	0x1 4000 0B70	R/W	TAH0 Segment Size Register 3	897
TAH1_SSR3	0x1 4000 0D70	R/W	TAH1 Segment Size Register 3	897
TAH0_SSR4	0x1 4000 0B74	R/W	TAH0 Segment Size Register 4	897
TAH1_SSR4	0x1 4000 0D74	R/W	TAH1 Segment Size Register 4	897
TAH0_SSR5	0x1 4000 0B78	R/W	TAH0 Segment Size Register 5	897
TAH1_SSR5	0x1 4000 0D78	R/W	TAH1 Segment Size Register 5	897
TAH0_TSR	0x1 4000 0B7C	R	TAH0 Transmit Status Register	898
TAH1_TSR	0x1 4000 0D7C	R	TAH1 Transmit Status Register	898

Preliminary User's Manual**25.3.1 TAH Revision ID Register (TAHx_REVID)**

TAHx_REVID is the Core Connect revision ID register as defined in the *IBM ASIC Soft Core Release Requirement* document. This is a read-only register containing the revision number and the branch revision number of the core. The values change with each new revision of the core. *Figure 25-6* describes the TAHx_REVID register bits.

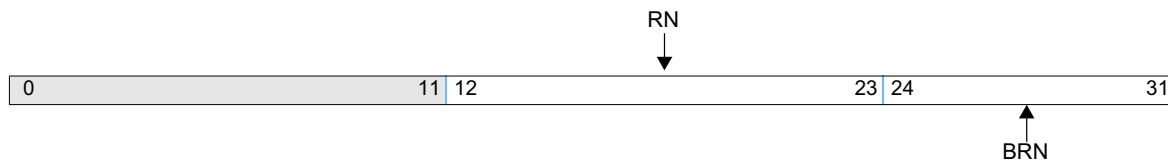


Figure 25-6. TAHx Revision ID Register (TAHx_REVID)

0:11		Reserved	
12:23	RN	Revision Number	Set to 1
24:31	BRN	Branch Revision Number	Set to 0

25.3.2 TAH Mode Register (TAHx_MR)

TAH mode register defines the configuration modes of TAH that may be changed at any time during TAH operation. When TAHx_MR[CVR=1], TAHx verifies the checksum for IP, TCP and UDP frames. When TAHx_MR[SR=1] TAHx activates a soft reset and the bit is cleared after reset is completed.

TAHx_MR[ST] defines the number of bytes (in multiple of 256 bytes) that must be received from MAL when hardware assist is disabled before the send data is forwarded to EMAC. For example, a value of 2 means that 512 bytes would have to be fetched from MAL before the data is sent to EMAC. A larger value would decrease the chance of underrun in EMAC but increase the latency. A smaller value would increase the chance of underrun in EMAC but decrease the latency. A value of zero means that as soon as data is received from MAL, it is forwarded to EMAC. But since transfer to EMAC does not start until TAH has accumulated a full burst (16 cycles, or 256 bytes), a value of zero is functionally equivalent to a value of one in TAHx_MR[ST].

TAHx_MR[TFS] defines the size of the transmit FIFO attached to TAHx in units of 2K bytes. For example, a value of 001 means 2Kbyte FIFO is present, and 101 means a 10Kbyte FIFO is present. If hardware segmentation is enabled, the size of the transmit FIFO must be bigger than the size of the largest segment by 512 bytes. If hardware checksum is enabled, the size of the transmit FIFO must be bigger than the size of the packet by 80 bytes; otherwise TAH terminates the transfer with error if hardware acceleration is enabled. For standard ethernet packets, a 2K-byte transmit FIFO is sufficient. For Alteon style Jumbo packets with a 9K-byte frame size, a 10K-byte transmit FIFO size is sufficient.

Figure 25-7 describes the TAHx_MR register bits.

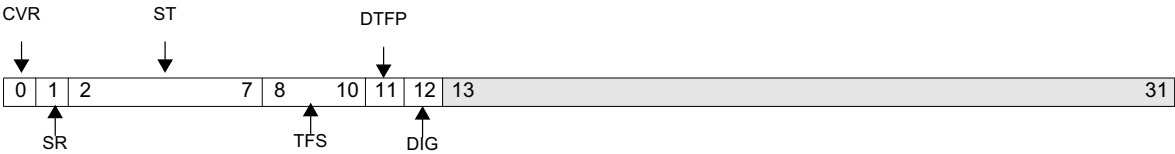


Figure 25-7. TAH Mode Register (TAHx_MR)

0	CVR	Checksum Verification on Receive 0 Checksum for IP, TCP and UDP packets disabled 1 Checksum for IP, TCP and UDP packets enabled
1	SR	TAH Software Reset 0 TAH reset is complete 1 Reset the TAH
2:7	ST	Send Threshold 000000 - 256 bytes 000001 - 256 bytes 000010 - 512 bytes
8:10	TFS	Transmit FIFO Size 001 Transmit FIFO size is 2Kbyte 010 Transmit FIFO size is 4Kbyte 011 Transmit FIFO size is 6Kbyte 100 Transmit FIFO size is 8Kbyte 101 Transmit FIFO size is 10Kbyte Note: The default is 001. Maximum of 10K FIFO is used.
11	DTFP	Disable Transmit FIFO Parity Protection 0 TAH checks for parity errors in the transmit FIFO 1 TAH does not check for parity errors in the transmit FIFO
12	DIG	Disable Interrupt Generation 0 TAH generates interrupts when errors are detected 1 TAH does not generate interrupts when errors are detected
13:31		Reserved

25.3.3 TAH Segment Size Registers 0:5 (TAHx_SSR0-TAHx_SSR5)

TAH segment size registers specify the size of the segment to be used when TCP segmentation is enabled. The size (defined as number of halfwords) of a segment refers to the total number of bytes following the type/length field and before the FCS field in an ethernet packet. It includes the IP headers, the TCP headers, and the data payload. For an IEEE 802 formatted packet, it also includes the 8-byte Logical Link Control (LLC) header and Sub-Network Access Protocol (SNAP) header. It does not include the 6-byte MAC destination address, the 6-byte MAC source address, the 2-byte Type/Length field, and the 4-byte FCS field. It also does not include the 4-byte VLAN tag field. For a standard Ethernet packet, the segment size should be set to 1500. For an Alteon style Jumbo frame, the segment size should be set to 9000.

Note: TAH can accommodate any arbitrary segment size provided that Transmit FIFO is big enough. All segment size registers are initialized to a value of 750, which corresponds to 1500 bytes (size of a standard Ethernet packet).

Figure 25-8 describes the TAHx_SSR0-TAHx_SSR5 register bits.

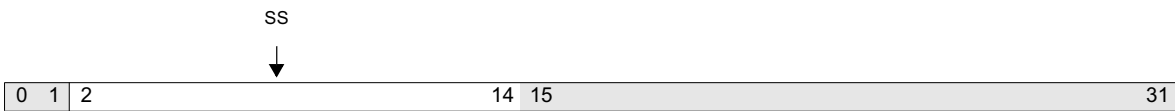


Figure 25-8. TAH Segment Size Register 0:5 (TAHx_SSR0-TAHx_SSR5)

0:1		Reserved
2:14	SS	Segment Size Defines the size of the segment in multiples of 2 bytes to be used when TCP segmentation is enabled.
15:31		Reserved

25.3.4 TAH Transmit Status Register (TAHx_TSR)

TAH Transmit Status Register contains the error status that results in the abnormal termination of a transmit operation when hardware acceleration is enabled. The register can only be written by TAH. The content is cleared when the Transmit Status Register is read.

When hardware checksum is enabled, the size of the transmit FIFO is not more than 80 bytes bigger than the size of the packet. When hardware segmentation is enabled, the size of the transmit FIFO is no more than 512 bytes bigger than the size of a segment.

Figure 25-9 describes the TAHx_TSR register bits.

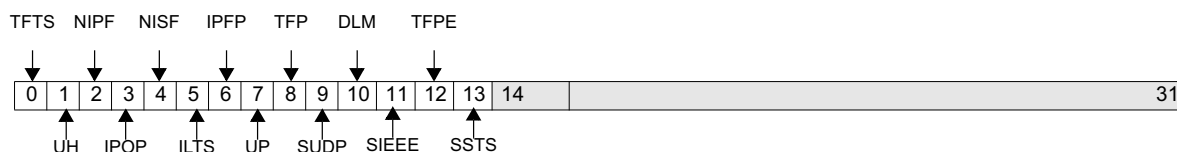


Figure 25-9. TAH Transmit Status Register (TAHx_TSR)

0	TSTS	Transmit FIFO Too Small The size of the transmit FIFO is not more than 80bytes bigger than the size of the packet when hardware checksum is enabled; or the size of the transmit FIFO is no more than 512 bytes bigger than the size of a segment when hardware segmentation is enabled..	
1	UH	Unrecognized Header The packet is in a format that is not supported by hardware	
2	NIPF	Not IP Version 4 Format The packet is not in IP version 4 format	
3	IPOP	IP Option Present IP option is present in the packet	
4	NISF	No IEEE SNAP Format The packet uses the IEEE 802 format but the SNAP header is incorrect	
5	ILTS	IP Length Too Short The total length in the IP header is smaller than 40 bytes.	
6	IPFP	IP Fragment Present The packet is part of an IP fragment.	
7	UP	Unsupported Protocol The protocol in use is neither TCP or UDP.	
8	TFP	TCP Flags Present Some TCP flags other than ACK are set.	
9	SUDP	Segmentation for UDP UDP packets cannot be segmented.	
10	DLM	Data Length Mismatch Amount of send data is smaller than the size of the TCP/IP header or the value in the IP length field.	If more send data is present, the extra data will be discarded with no error.
11	SIEEE	Segmentation for IEEE The size of the segment exceeds 1500 bytes for IEEE formatted packets when hardware segmentation is enabled.	
12	TFPE	Transmit FIFO Parity Error Parity error is detected in the Transmit FIFO.	

Preliminary User's Manual

13	SSTS	Segment Size Too Small The size of the segment is less than 168 bytes for DIX formatted segments or 176 bytes for IEEE formatted segments when hardware segmentation is enabled
14:31		Reserved

25.4 Power-Up and Initialization

To reset or reconfigure TAH, both EMAC and MAL must be in the quiescent state to avoid erroneous situations. There are two types of reset operations that may be applied to EMAC: hard reset and soft reset.

25.4.1 Hard Reset

When RESET input is asserted, TAH aborts all on-going activities unconditionally, initializes all internal state machines, counters, and registers. The reset signal must be asserted for at least two cycles of the slowest clock domain inside TAH in order to be properly recognized. This means the hard reset must be at least 800 ns.

25.4.2 Soft Reset

The software should first reset the appropriate channels in MAL (see *MAL Configuration Register (MAL0_CFG)* on page 776) and then initialize a soft reset by setting the Soft Reset bit in Mode Register to '1'. As a response to the soft reset, TAH aborts all on-going activities unconditionally, initializes all internal state machines, counters, and registers. After TAH finishes all activities related to the soft reset processing, it clears the Soft Reset bit.

Preliminary User's Manual

26. Serial Port Operations

The PPC440GX contains two universal asynchronous receiver/transmitters (UARTs) which provide two full-duplex serial interfaces to support communications with serial peripheral devices. Each UART is compatible with the National Semiconductor (NS) 16750 chip, and includes a 64-byte send and a 64-byte receive FIFO.

Features of the UART include:

- Compatible with the NS 16750
- 64-byte send FIFO, 64-byte receive FIFO
- Full duplex operation
- Programmable baud rate generator
- Supports 5- to 8-bit word size, 1 or 2 stop bits, even, odd, or no parity
- One 8-wire interface (UART0) and one 4-wire interface (UART1)
- Hardware flow control is selectable

The UART performs serial-to-parallel conversion on data characters received from a peripheral device, and parallel-to-serial conversion on data characters received from the processor. The processor can read the complete status of the UART at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the UART, as well as any error conditions, such as parity, overrun, framing, and break interrupt.

This UART is functionally identical to NS16750 in character mode (on power up it will be in this mode), and can be put into FIFO mode to relieve the processor of excessive software overhead. Here, internal FIFOs are activated allowing 64 bytes (plus 3 bits per byte of error data in the RCVR FIFO) to be stored in both receive and transmit modes.

The source of the UART serial clock input is selected in UART Configuration Register 0 and 1 (SDR0_UART0 and 1[U0EC:U1EC]) bits 8. Either the internal serial clock or an external serial clock can be selected. A programmable baud rate generator is included that is capable of dividing the UART serial clock input by a divisor of 1 to ($2^{16} - 1$) and producing the $16\times$ clock required for driving the UART internal transmitter/receiver logic. The internal serial clock input is derived from the PLB clock by a divisor specified in SDR0_UART0 and 1[UDIV]. See *Section 14 Clocking* on page 477 for additional information.

The UART has an interrupt system that can be programmed to the user's requirements, helping to minimize the computing required to handle the communications link. UART interrupts are capable of triggering an interrupt request to the PPC440GX interrupt controller.

26.1 Functional Description

- Runs NS 16750 software
- Registers are identical to the NS16750 register set
- After reset, all registers are identical to the NS16750 register set
- Complete status reporting capability
- Transmitter and receiver are each buffered with 64-byte FIFOs when FIFO mode selected
- Can add/delete standard asynchronous communication bits such as start, stop, and parity to/from the serial data
- When in character mode, holding and shift registers eliminate the need for precise synchronization between the processor and serial data

- Full prioritized interrupt system controls
- Independently controlled transmit, receive, line status, and data set interrupts
- Programmable baud rate generator divides the UART serial clock input by 1 to ($2^{16}-1$) and generates the 16x clock:

$$\text{Baud rate (bps)} = (\text{Serial Clock Input}) / (16 \times \text{Decimal Divisor})$$

- Receiver uses 5-way oversampling as follows: it samples each serial bit five times, and if at least three of the samples are 1's, the bit is determined to be a 1, otherwise it is a 0
- Fully programmable serial-interface characteristics:
 - 5-, 6-, 7-, or 8-bit characters
 - Even, odd, or no parity bit generation and detection
 - 1-, 1.5-, or 2-stop bit generation
 - Variable baud rate
- Line break generation and detection, and false start bit detection
- Internal diagnostic capability:
 - Loopback controls for communications link fault isolation
 - Break, parity, overrun, framing error simulation

26.2 Serial Port Clocking

Each of the UARTs is independently configured to use either the on-chip baud rate clock, or can be clocked from the shared external signal pin UARTSerClk. The on-chip serial clock source is generated by dividing the PLB clock and is an integer (n) fraction of the PLB clock, where n is in the range from 1 to 256.

Thus, the on-chip generated UART clock frequencies are:

$$\text{UART0 Serial Clock} = \text{PLBCLK} / \text{SDR0_UART0[U0DIV]}$$

$$\text{UART1 Serial Clock} = \text{PLBCLK} / \text{SDR0_UART1[U1DIV]}$$

and the UART Serial Clock rate for the external clock is:

$$\text{UART0 and/or UART1 SerialClock} = \text{UARTSerClk clock signal}$$

Note that if the UARTSerClk signal input is not used, it must be pulled-up and not left to float. See the Data Sheet section entitled Signal Functional Description for complete details.

The choice of serial clock frequency affects the serial communications error rate. If an external clock of 1.8432 MHz (or some multiple of this frequency) is used, the error rate approaches zero. However, when using the internally generated clock only certain clock frequencies are possible, which results in a small, non-zero error rate in all cases.

The optimum serial clock frequency is determined from the following relationship:

$$\text{Serial Clock} = \text{Baud Rate} \times 16 \times \text{UART Divisor}$$

Preliminary User's Manual

Acceptable baud rates are always integral multiples of 300 (for example, $1200 = 4 \times 300$). Table 26-1 shows optimum UART divisor and divide ratios for a range of possible baud rates. This information is provided for various clock frequencies. Note that the serial clock frequency must be less than half the OPB frequency. The UART divisor is programmed in UARTx_DLM and UARTx_DLL (see *Divisor Latch LSB and MSB Registers (UARTx_DLL, UARTx_DLM)* on page 914). The value range is 1 to $(2^{16}-1)=65535$.

Table 26-1. Baud Rate Settings

Desired Baud Rate (bps)	PLB:UART Divide Ratio	Serial Clock Frequency (MHz)	Minimum OPB Frequency (MHz)	UART Divisor	Actual Baud Rate (bps)	Error (%)
PLB Clock = 100MHz						
1200	12	8.3333	16.9491	434	1200.08	0.006
2400	14	7.1429	14.4927	186	2400.15	0.01
4800	14	7.1429	14.4927	93	4800.31	0.01
9600	7	14.2860	29.4117	93	9600.61	0.01
19200	13	7.6923	15.6250	25	19230.77	0.16
28800	31	3.2258	6.4935	7	28801.84	0.01
33600	31	3.2258	6.4935	6	33602.15	0.01
38400	27	3.7037	7.4626	6	38580.25	0.47
57600	27	3.7037	7.4626	4	57870.37	0.47
115200	27	3.7037	7.4626	2	115740.74	0.47
307200	10	10.0000	20.4081	2	312500.00	1.73 ¹
PLB Clock = 125 MHz						
1200	31	4.0322	8.1301	210	1200.08	0.006
2400	31	4.0322	8.1301	105	2400.15	0.01
4800	22	5.6818	11.4943	74	4798.83	-0.02
9600	22	5.6818	11.4943	37	9597.67	-0.02
19200	11	11.3636	23.2558	37	19195.33	-0.02
28800	16	7.3529	14.9254	16	28722.43	0.27
33600	29	4.3103	8.6956	8	33674.57	0.22
38400	29	4.3103	8.6956	7	38485.22	0.22
57600	17	7.3529	14.9254	8	57444.85	-0.27
115200	17	7.3529	14.9254	4	114889.70	-0.27
307200	25	5.0000	10.1010	1	312500	1.73 ¹
PLB Clock =133 MHz						
1200	14	9.5238	19.4175	496	1200.07	0.006
2400	14	9.5238	19.4175	248	2400.15	0.01
4800	14	9.5238	19.4175	124	4800.31	0.01
9600	14	9.5238	19.4175	62	9600.61	0.01
19200	14	9.5238	19.4175	31	19201.23	0.01
28800	17	7.8431	15.0376	17	28835.06	0.12
33600	31	4.3011	8.4033	8	33602.15	0.01
38400	31	4.3011	8.4033	7	38402.46	0.01

Table 26-1. Baud Rate Settings (Continued)

Desired Baud Rate (bps)	PLB:UART Divide Ratio	Serial Clock Frequency (MHz)	Minimum OPB Frequency (MHz)	UART Divisor	Actual Baud Rate (bps)	Error (%)
57600	29	4.5977	9.2807	5	57471.26	-0.22
115200	24	5.5556	11.2359	3	115740.74	0.47
307200	27	4.9383	9.9751	1	308641.97	0.47
PLB Clock = 150MHz						
1200	31	4.8387	9.7719	252	1200.08	0.006
2400	31	4.8387	9.7719	126	2400.15	0.01
4800	31	4.8387	9.7719	63	4800.31	0.01
9600	16	9.3750	17.9641	61	9605.53	0.06
19200	14	10.7143	20.4081	35	19132.65	-0.35
28800	25	6.0000	12.1457	13	28846.15	0.16
33600	31	4.838	17.5054	16	33602.15	0.01
38400	27	5.5556	11.2359	9	38580.25	0.47
57600	27	5.5556	11.2359	6	57870.37	0.47
115200	27	5.5556	11.2359	3	115740.74	0.47
307200	31	4.8387	9.7719	1	302419.35	-1.56 ¹
PLB Clock = 166.66 MHz						
1200	31	5.3763	10.8695	280	1200.08	0.01
2400	31	5.3763	10.8695	140	2400.15	0.01
4800	31	5.3763	10.8695	70	4800.31	0.01
9600	31	5.3763	10.8695	35	9600.61	0.01
19200	32	5.2083	10.5263	17	19148.28	-0.27
28800	19	8.7719	17.8571	19	28855.03	0.19
33600	31	5.3763	10.8695	10	33602.15	0.01
38400	17	9.8039	20.0000	16	38296.57	-0.27
57600	30	5.5556	11.2359	6	57870.37	0.47
115200	30	5.5556	11.2359	3	115740.74	0.47
307200	17	9.8039	20.0000	2	306372.55	-0.27
1. This amount of error may cause framing errors.						

Preliminary User's Manual**26.3 UART Registers**

UART registers are accessed via memory to 0x14000_0xYY where X=2 for UART0 and X=3 for UART1.

Table 26-2. UART Configuration Registers

Register	Address	Access	Description	Page
UARTx_RBR	0x14000_0X00 ¹	R	UART x Receiver Buffer Register	906
UARTx_THR	0x14000_0X00 ¹	W	UART x Transmitter Holding Register	906
UARTx_IER	0x14000_0X01 ¹	R/W	UART x Interrupt Enable Register	906
UARTx_IIR	0x14000_0X02	R	UART x Interrupt Identification Register	907
UARTx_FCR	0x14000_0X02	W	UART x FIFO Control Register	909
UARTx_LCR	0x14000_0X03	R/W	UART x Line Control Register	910
UARTx_MCR	0x14000_0X04	R/W	UART x Modem Control Register	911
UARTx_LSR	0x14000_0X05	R/W	UART x Line Status Register	912
UARTx_MSR	0x14000_0X06	R/W	UART x Modem Status Register	913
UARTx_SCR	0x14000_0X07	R/W	UART x Scratch Register	914
UARTx_DLL	0x14000_0X00 ¹	R/W	UART x Divisor Latch (LSB)	914
UARTx_DLM	0x14000_0X01 ¹	R/W	UART x Divisor Latch (MSB)	914

1. UARTx_LCR[DLAB] controls the function accessed through registers 0x14000_0X00 and 0x14000_0X01. When UARTx_LCR[DLAB] is 0, access is enabled to the Receiver/Transmitter registers and the Interrupt Enable register. When UARTx_LCR[DLAB] is a 1, access is enabled to the Divisor Latch registers.

The system programmer may access any of the UART registers via the processor. These registers control all UART operations, including transmission and reception of data. In PPC440GX there are two UARTs, designated 0 (8-wire interface) and 1 (4-wire interface). In the following sections, the registers are specified with a generic name where x represents either 0 or 1. For example, the Line Control Register appears as a UARTx_LCR.

For UART0, the eight available signals are Rx, Tx, DCD, DSR, CTS, DTR, RTS, and RI. The OUT1 and OUT2 signals are not available for UART0.

For UART1, two of the four wires are TX and RX. The remaining two wires can be programmed as a combination of DTR and DSR, or CTS and RTS in SDR0_PFC1[U1ME]. DCD, RI, OUT1, and OUT2 are not available on the 4-wire interface.

26.3.1 Receiver Buffer Registers (UARTx_RBR)

Figure 26-1 describes UARTx_RBR bit definitions.



Figure 26-1. UART Receiver Buffer Registers (UARTx_RBR)

0:7		Data bit
Note: UARTx_RBR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

26.3.2 Transmitter Holding Registers (UARTx_THR)

Figure 26-2 describes UARTx_THR bit definitions.



Figure 26-2. UART Transmitter Holding Registers (UARTx_THR)

0:7		Data bit
Note: UARTx_THR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

26.3.3 Interrupt Enable Registers (UARTx_IER)

Five UART interrupts on four priority levels are enabled via the Interrupt Enable Register, UARTx_IER. Any of the five interrupts can be used to surface a UART interrupt to the PPC440GX interrupt controller. Each interrupt can be enabled by setting its appropriate bit. Resetting UARTx_IER[4:7] totally disables the UART interrupt system. Disabling an interrupt prevents it from being shown as active in the UARTx_IIR and prevents it from signaling a UART interrupt to the PPC440GX interrupt controller. See *Table 26-3 Interrupt Priority Level* on page 907. Figure 26-3 describes UARTx_IER bit definitions.

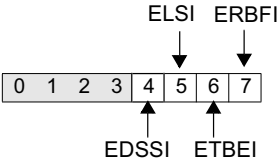


Figure 26-3. UART Interrupt Enable Registers (UARTx_IER)

0:3		Reserved	Always 0.
4	EDSSI	Modem Status Interrupt 0 Disable modem status interrupt 1 Enable modem status interrupt	
5	ELSI	Receiver Line Status Interrupt enable 0 Disable receiver line status interrupt 1 Enable receiver line status interrupt	

Preliminary User's Manual

6	ETBEI	Transmitter Holding Register Empty Interrupt enable 0 Disable transmitter holding register empty interrupt 1 Enable transmitter holding register empty interrupt	
7	ERBFI	Received Data Available Interrupt enable 0 Disable received data available interrupt 1 Enable received data available interrupt	In FIFO mode, timeout interrupts follow the enable/disable state of ERBFI.
Note: UARTx_IER is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

Interrupt Identification Registers (UARTx_IIR)

The UART prioritizes interrupts into four levels which are recorded in the Interrupt Identification Register. The interrupt types in the order of their priority are as follows:

1. Receiver line status
2. Received data available and character timeout indication
3. Transmitter holding register empty
4. Modem status

Table 26-3 lists the interrupt priority levels.

Table 26-3. Interrupt Priority Level

IIR Bit 4	IIR Bit 5	IIR Bit 6	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	1	1	1	Receiver Line Status	Overrun, Parity or Framing Error, or Break Interrupt.	Read LSR.
0	1	0	2	Received Data Available	Receiver data available or trigger level reached.	Read RBR, or FIFO drops below trigger level.
1	1	0	2	Character Timeout Indication	No characters have been removed from or input to the receiver FIFO during the last four character times and it contains at least one character during this time.	Read RBR.
0	0	1	3	Transmitter Holding Register Empty	Transmitter Holding Register Empty.	Read IIR (if source of interrupt) or write THR.
0	0	0	4	Modem Status	Clear to Send, Data Set Ready, Ring Indicator or Data Carrier Detect.	Read MSR.

When the processor accesses UARTx_IIR, the UART records new interrupts, but does not change its current contents until the access by the processor is complete. The UART indicates the highest priority interrupt pending to the PPC440GX interrupt controller via the IIR. *Figure 26-4* describes UARTx_IIR bit definitions.

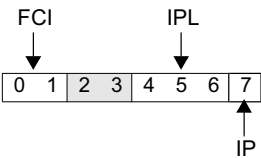


Figure 26-4. UART Interrupt Identification Registers (UARTx_IIR)

0:1	FCI	FIFO Control Indicator 00 FIFOs disabled (UARTx_FCR[FC] = 0) 01 Reserved 10 Reserved 11 FIFOs enabled (UARTx_FCR[FC] = 1)
2:3		Reserved
4:6	IPL	Interrupt Priority Level 000 Priority level 4 001 Priority level 3 010 Priority level 2 011 Priority level 1 100 Reserved 101 Reserved 110 Priority level 2 111 Reserved Note: Priority 1 is highest priority.
7	IP	Interrupt Pending 0 Interrupt is pending 1 No interrupt pending When set to 0, IIR contents can be used as a pointer to the appropriate interrupt service routine.
Note: UARTx_IIR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

Preliminary User's Manual**26.3.4 FIFO Control Registers (UARTx_FCR)**

The FIFO control register has the same address as the IIR and is a write-only register. This register is used to perform FIFO control operations such as selecting the type of DMA signaling, setting the receiver FIFO trigger levels, clearing the FIFOs, and enabling the FIFO.

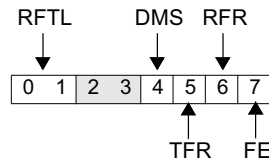


Figure 26-5. UART FIFO Control Registers (UARTx_FCR)

0:1	RFTL	Receiver FIFO Trigger Level 00 01 byte 01 16 bytes 10 32 bytes 11 56 bytes	
2:3		Reserved	
4	DMS	DMA Mode Select 0 Mode 0 = single transfer 1 Mode 1 = multiple transfers	Select single or multiple transfer mode if UARTx_FCR[7] = 1.
5	TFR	Transmitter FIFO Reset 0 Operation complete 1 Reset the transmitter FIFO	A 1 written to this bit clears all bytes in the transmitter FIFO and resets all of its counter logic to 0. The transmitter shift register is not cleared. This bit is self-clearing.
6	RFR	Receiver FIFO Reset 0 Operation complete 1 Reset the receiver FIFO	A 1 written to this bit clears all bytes in the receiver FIFO and resets all of its counter logic to 0. The receiver shift register is not cleared. This bit is self-clearing.
7	FE	FIFO Enable 0 Disable FIFOs 1 Enable FIFOs	When set to 1, both the receiver and transmitter FIFOs are enabled. When set to 0, both receiver and transmitter FIFOs are reset. Data is automatically cleared from both FIFOs when changing to and from FIFO and 16450 modes. Programming other bits will be ignored if this bit is not a 1.
Note: UARTx_FCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

26.3.5 Line Control Registers (UARTx_LCR)

The system programmer uses the line control register (LCR) to specify the format of the asynchronous data communications exchange and to set the Divisor Latch Access bit. The contents of the LCR can also be read by the processor. The read capability simplifies system programming, and eliminates the need for separate storage of the line characteristics in system memory.

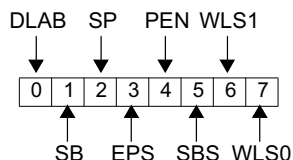


Figure 26-6. UART Line Control Registers (UARTx_LCR)

0	DLAB	Divisor Latch Access Bit 0 Address RBR, THR and IER with LTADR2-0 for read or write operation 1 Address Divisor Latches with LTADR2-0 for read or write operation	
1	SB	Set Break 0 Disable Break 1 Enable Break	Causes a break condition to be transmitted to the UART when the core is receiving. SOUT is forced to the spacing state (0). This bit acts only on SOUT and has no effect on the transmitter logic.
2	SP	Sticky Parity 0 Disable sticky parity 1 Enable sticky parity	If UARTx_LCR[EPS] = 1 and UARTx_LCR[PEN] = 1, the parity bit is transmitted and checked as 0. If UARTx_LCR[EPS] = 0 and UARTx_LCR[PEN] = 1, the parity bit is transmitted and checked as 1.
3	EPS	Even Parity Select 0 Generate odd parity 1 Generate even parity	This bit is significant only if UARTx_LCR[PEN] = 1.
4	PEN	Parity Enable 0 Disable parity checking 1 Enable parity checking	
5	SBS	Stop Bit Select 0 Characters have 1 stop bit 1 Characters have 1.5 or 2 stop bits	If UARTx_LCR[WPS1,WPS0] = 00, characters have 1.5 stop bits. For any other value of UARTx_LCR[WPS1,WPS0], characters have 2 stop bits. The receiver checks the first stop bit only, regardless of how many stop bits are selected.
6	WLS1	Word Length Select Bits 1 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters	
7	WLS0	Word Length Select Bits 0 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters	

Note: UARTx_LCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.

Preliminary User's Manual**26.3.6 Modem Control Registers (UARTx_MCR)**

The interface between the modem, data set, or peripheral device emulating a modem, and the UART, is controlled by the Modem Control Register (UARTx_MCR).

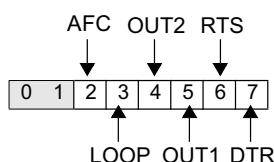


Figure 26-7. UART Modem Control Registers (UARTx_MCR)

0:1		Reserved	Always 0.
2	AFC	Auto Flow Control 0 Hardware flow control disabled 1 Hardware flow control enabled	
3	LOOP	Loopback Mode 0 Disabled 1 Enabled	<p>Provides a local loopback feature for diagnostic testing of the UART. The following occurs:</p> <ol style="list-style-type: none"> 1. SOUT is set to the marking state (logic 1) SIN is disconnected. 2. The output of the transmitter shift register feeds the input of the receiver shift register. 3. The four modem control inputs \overline{DSR}, \overline{CTS}, \overline{RI}, and \overline{DCD} are disconnected. 4. The four modem control outputs \overline{DTR}, \overline{RTS}, $\overline{OUT1}$, and $\overline{OUT2}$ are set to a logic 1 (their inactive state). 5. The four modem control outputs are connected internally to the four modem control inputs. <p>Transmitted data is immediately received to verify the UART transmit and receive data paths.</p> <p>Receiver and transmitter interrupts are operational. Their sources are external to the UART. Also operational are the modem control interrupts, but their source is the low-order 4 bits of UARTx_MCR instead of the modem control inputs to the UART. UARTx_IER still controls the interrupts.</p>
4	OUT2	User Output 2 0 $\overline{OUT2}$ inactive (1) 1 $\overline{OUT2}$ active (0)	The $\overline{OUT2}$ bit may be written and read but it provides no function
5	OUT1	User Output 1 0 $\overline{OUT1}$ inactive (1) 1 $\overline{OUT1}$ active (0)	The $\overline{OUT1}$ bit may be written and read but it provides no function
6	RTS	Request To Send 0 \overline{RTS} inactive (1) 1 \overline{RTS} active (0)	
7	DTR	Data Terminal Ready 0 \overline{DTR} inactive (1) 1 \overline{DTR} active (0)	

Note: UARTx_MCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.

26.3.7 Line Status Registers (UARTx_LSR)

Information concerning the data transfer is held for the processor in this register. Bits 3 through 6 are conditions that produce a receiver line status interrupt whenever the condition corresponding to the active bit is detected and the interrupt is enabled. This register is intended for read operations only and writing is not recommended.

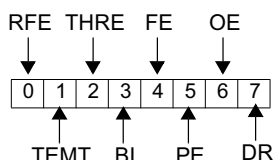


Figure 26-8. UART Line Status Registers (UARTx_LSR)

0	RFE	<p>Receiver FIFO Error Indicator</p> <p>0 In FIFO mode, reset to 0 when the processor reads the UARTx_LSR, provided there are no subsequent errors in the FIFO.</p> <p>1 There are one or more instances of parity error, framing error or break indication in the FIFO.</p>	Always 0 in 16450 mode.
1	TEMT	<p>Transmitter Empty Indicator</p> <p>0 Reset to 0 whenever the THR or the transmitter shift register contain a character. In FIFO mode, it is reset to 0 whenever the transmitter FIFO or the transmitter shift register contain a character.</p> <p>1 Set to 1 when the THR and the Transmitter shift register are both empty. In FIFO mode, it is set to 1 when the transmitter FIFO and the transmitter shift register are both empty.</p>	
2	THRE	<p>Transmitter Holding Register Empty Indicator</p> <p>0 Concurrent reset to 0 with the loading of the THR by the processor. In FIFO mode it is reset to 0 when at least one byte is written to the transmitter FIFO.</p> <p>1 Set to 1 when the UART is ready to accept a new character for transmission. In FIFO mode, this bit is set when the transmitter FIFO is empty.</p>	When UARTx_IER[THRE] = 1, the UART issues an interrupt to the PPC440GX interrupt controller. This bit is set to 1 when a character is transferred from the THR to the transmitter shift register.
3	BI	<p>Break Interrupt Indicator.</p> <p>0 Reset to 0 whenever processor reads Line Status Register (LSR).</p> <p>1 Set to 1 whenever the received data input is held at the spacing level (0) for longer than a full word transmission time.</p>	The full word transmission time is the time required for the start bit, data bits (can be 5–8 bits), parity and stop bits. In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Only one 0 character is loaded into the receiver FIFO when a break occurs. After the next valid start bit is received and has gone into the marking state, the next character transfer is enabled. Error causes a Receiver Line Status Interrupt.
4	FE	<p>Framing Error Indicator.</p> <p>0 Reset to 0 whenever processor reads LSR.</p> <p>1 Set to 1 whenever stop bit following the last data bit or parity bit is detected as 0 (spacing level). Indicates that a valid stop bit was not found in the received character.</p>	Error causes a Receiver Line Status Interrupt.

Preliminary User's Manual

5	PE	Parity Error Indicator. 0 Reset to 0 whenever processor reads UARTx_LSR. 1 Indicates that the received data character does not have the correct parity as determined by the even parity select bit (UARTx_LCR[EPS]). Set to 1 upon detection of a parity error.	In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Error causes a Receiver Line Status Interrupt.
6	OE	Overrun Error Indicator. 0 Reset to 0 whenever processor reads UARTx_LSR. 1 Data in the RBR was read by the processor before the next character was transferred into the UARTx_RBR, hence the original data was lost.	In FIFO mode, if the incoming data continues to fill the FIFO beyond the trigger level, an OE occurs only after the FIFO is completely full and the entire next character has been received in the receiver shift register. The processor is informed of the OE immediately upon occurrence. The character in the shift register will be overwritten and will not be transferred to the FIFO. Error causes a Receiver Line Status Interrupt.
7	DR	Receiver Data Ready Indicator. 0 Reset to 0 when all data has been read from the receiver FIFO or the UARTx_RBR. 1 An entire incoming character has been received into the UARTx_RBR or receiver FIFO.	

Note: UARTx_LSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.

26.3.8 Modem Status Registers (UARTx_MSR)

The processor can monitor the present state of the modem (or peripheral device) control lines by reading the Modem Status Register (UARTx_MSR). In addition, the UARTx_MSR has four bits to indicate if any of the modem (or peripheral device) control lines have changed state.

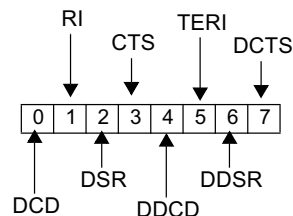


Figure 26-9. UART Modem Status Registers (UARTx_MSR)

0	DCD	Data Carrier Detect	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT2].
1	RI	Ring Indicator	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT1].
2	DSR	Data Set Ready	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[DTR].
3	CTS	Clear To Send	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[RTS].
4	DDCD	Delta Data Carrier Detect 0 Set when processor reads the Modem Status Register 1 DCD input changed state	Indicates that the $\overline{\text{DCD}}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
5	TERI	Trailing Edge of Ring Indicator 0 Set when processor reads the Modem Status Register 1 $\overline{\text{RI}}$ input changed from 0 to 1	Indicates that the $\overline{\text{RI}}$ input to the UART changed from 0 to 1 since the processor last read the Modem Status Register. A modem status interrupt is generated.

6	DDSR	Delta Data Set Ready 0 Set when processor reads the Modem Status Register 1 DSR input changed state	Indicates that the $\overline{\text{DSR}}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
7	DCTS	Delta Clear To Send 0 Set when processor reads the Modem Status Register 1 CTS input changed state	Indicates that the $\overline{\text{CTS}}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
Note: UARTx_MSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

26.3.9 Scratchpad Registers (UARTx_SCR)

A scratchpad register intended for use by the programmer as a temporary data location is provided in this UART. It does not control the UART operation in any way.



Figure 26-10. Scratchpad Registers (UARTx_SCR)

0:7		Data bits
Note: UARTx_SCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

26.3.10 Divisor Latch LSB and MSB Registers (UARTx_DLL, UARTx_DLM)

The divisor latches are used to program the UART divisor used in generating the baud clock. A 16-bit divisor may be programmed through these registers. Access to these registers is provided by setting UARTx_LCR[DLAB] = 1. These registers have a power-on reset value of 0.



Figure 26-11. UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM)

0:7		Data bits
Note: UARTx_DLM is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		



Figure 26-12. UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)

8:15		Data bits
Note: UARTx_DLL is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

Preliminary User's Manual

The UART divisor is calculated using the following formula:

$$\text{UART Divisor} = \text{Serial Input Clock} / (16 \times \text{Baud Rate})$$

For example, if the serial input clock = 11.0592MHz and a baud rate of 9600bps is required:

$$\text{UART Divisor} = \text{Serial Input Clock} / (16 \times \text{Baud Rate})$$

$$= 11,059,200 / (16 \times 9600)$$

$$= 72 = 0x48$$

For this example, UARTx_DLM should be programmed to 0 and UARTx_DLL register should be programmed to 0x48. Due to the error introduced by rounding, some baud rates cannot be generated at certain serial input clock frequencies. *Table 26-4* lists some common baud rates and their corresponding Divisor Latch register values with a serial input clock of 11.0592MHz.

Table 26-4. Divisor Latch Settings for Certain Baud Rates

Baud Rate (bps)	Divisor Latch MSB	Divisor Latch LSB
9600	0x00	0x48
19200	0x00	0x24
28800	0x00	0x18
38400	0x00	0x12
57600	0x00	0x0C

26.4 FIFO Operation

This section discusses the various modes of operation including interrupt mode, polled mode, and sleep mode.

26.4.1 Interrupt Mode

26.4.1.1 Receiver

Receiver interrupts occur as described below when the receiver FIFO and receiver interrupts are enabled by setting `UARTx_FCR[FE] = 1` and `UARTx_IER[ERBFI] = 1`.

The received data available indicator is issued when the number of characters in the FIFO has reached the trigger level programmed into `UARTx_FCR`. This indicator is reset to 0 when the FIFO character count drops below this trigger level.

The receiver line status interrupt (`UARTx_IIR = 0xC6`) is a top priority interrupt, whereas the received data available interrupt (`UARTx_IIR = 0xC4`) is a second priority interrupt.

Data Ready (`UARTx_LSR[DR]`) is set as soon as a character is transferred from the shift register to the receiver FIFO. This bit is reset when the FIFO is empty.

Receiver timeout interrupts will occur as described below when the receiver FIFO and receiver interrupts are enabled by setting `UARTx_FCR[FE] = 1` and `UARTx_IER[ERBFI] = 1`.

A FIFO timeout will occur when:

At least one character is in the receiver FIFO, no serial characters have been received for four serial character time periods, and the processor has not read the FIFO for four serial character time periods. A serial character time period is as follows:

$$1/(\text{baud rate}) \times (\# \text{ start bits} + \text{word length} + \# \text{ parity bits} + \# \text{ stop bits})$$

For example, the serial character time period for an 8-bit word with one parity bit, two stop bits at 56K baud is as follows:

$$1/(56000) \times (1 + 8 + 1 + 2) = 214.3 \mu\text{s}$$

So the timeout would occur after 857.1 μs , if the above conditions hold.

When a timeout interrupt has occurred, it is cleared and the timer is reset when the processor reads one character from the receiver FIFO.

When a timeout interrupt has not occurred, the timer is reset after a new serial character is received or the processor reads the receiver FIFO.

Preliminary User's Manual

26.4.1.2 Transmitter

Transmitter interrupts occur, as described below, when the transmitter FIFO and transmitter interrupts are enabled by setting `UARTx_FCR[FE] = 1` and `UARTx_IER[ETBEI] = 1`.

The transmitter holding register interrupt (`UARTx_IIR = 0xC2`) occurs when transmit FIFO is empty, and is cleared as soon as the transmitter holding register is written to or the IIR is read. One to 64 characters may be written to the transmitter FIFO while servicing this interrupt.

The transmitter FIFO empty indications are delayed by one character time minus the last stop bit time whenever the following event occurs: `UARTx_LSR[THRE] = 1` and there were less than two bytes simultaneously present in the transmit FIFO since the last `UARTx_LSR[THRE] = 1`. If `UARTx_FCR[FE] = 1` (FIFOs enabled), the first transmitter interrupt after changing `UARTx_FCR[FE]` is immediate.

Receiver FIFO trigger level interrupts, received data available interrupts, and character timeouts all have equivalent second interrupt priority. Current transmitter holding register empty interrupt and Transmit FIFO empty have equivalent third interrupt priority.

26.4.2 Polled Mode

When `UARTx_FCR[FE] = 1` (FIFOs enabled), and `UARTx_IER[5:7]` are all set to 0 (interrupts disabled), the UART is in FIFO polled mode of operation. The receiver and transmitter are controlled separately, so either can be in polled mode of operation. In polled mode, the user program must check the `UARTx_LSR` to see the status of the receiver and/or transmitter.

`UARTx_LSR[3:6]` specifies which errors (if any) have occurred. Character status errors are handled in the same way as in interrupt mode. Since `UARTx_IER[ELSI] = 0`, the IIR is not affected. `UARTx_LSR[DR]` is set as long as there is at least one character in the receiver FIFO. `UARTx_LSR[THRE]` indicates if the transmitter FIFO is empty. `UARTx_LSR[TEMT]` indicates if the transmitter FIFO and the transmitter shift register are empty. `UARTx_LSR[RFE]` indicates if there are any errors in the receiver FIFO.

In FIFO polled mode, there are no character timeout or trigger levels; however, the FIFOs are still capable of holding characters.

26.4.3 UART and Sleep Mode

Both UARTs can be placed in sleep mode via the UART sleep bits in the `CPM0_ER` register (`CPM0_ER[UART0:UART1]`). The most common usage would be to save a little power if one or both of the UARTs were not going to be used.

Using sleep mode dynamically requires careful software control to make sure the UARTs are idle before putting them to sleep.

26.5 DMA Operation

The DMA controller can be configured to perform DMA operations using UART0 and UART1, which appear as an 8-bit peripherals to the DMA controller. When selected, the UART receiver is internally wired to the DMAReq and DMAAck signals of DMA channel 2, and the transmitter is internally wired to the DMAReq and DMAAck signals of DMA channel 3.

The UART can be operated in FIFO mode or non-FIFO mode. In FIFO mode, the transfers can be done as single transfers (DMA mode 0) or multiple transfers (DMA mode 1), depending on the setting of the DMS field in the FIFO Control Register (FCR). In non-FIFO mode, DMA transfers are performed using single transfers, using the UART's DMA mode 0. This section describes proper UART0 DMA programming. For more information on general DMA programming, see *Direct Memory Access Controller* on page 695.

26.5.1 UART Configuration Registers (SDR0_UART0 and SDR0_UART1)

Figure 26-13 and Figure 26-14 describe SDR0_UART0 and SDR0_UART1 register bits.

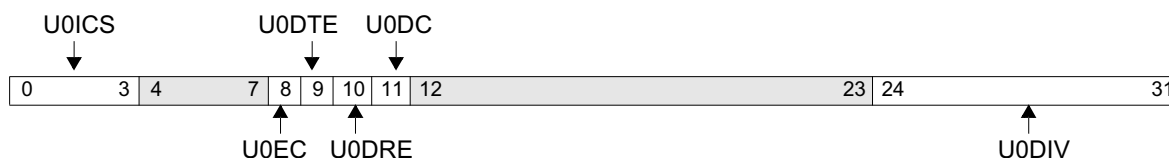


Figure 26-13. UART Configuration Register 0 (SDR0_UART0)

0:3	U0ICS	UART 0 Internal Clock Source 0000 Reserved 0001 Reserved 0010 UART 0 internal clock source = PLB 0011 Reserved
4:7		Reserved
8	U0EC	UART 0 EXT Clock Enable 0 UART0 uses the internal serial clock 1 UART0 uses the external serial clock
9	U0DTE	UART 0 DMA Transmit Enable 0 UART0 DMA transmit is disabled 1 UART0 DMA transmit is enabled
10	U0DRE	UART 0 DMA Receive Enable 0 UART0 DMA receive is disabled 1 UART0 DMA receive is enabled
11	U0DC	UART 0 DMA Clear 0 U0DTE and U0DRE are not cleared when UART receives a corresponding terminal count. 1 U0DTE and U0DRE are cleared when UART receives a corresponding terminal count.
12:23		Reserved
24:31	U0DIV	UART 0 Divider 0000_0000 - divider = 256 0000_0001 - divider = 1 0000_0010 - divider = 2 1111_1111 - divider = 255 The source for SDR0_UART0[U0DIV] is indicated by SDR0_UART0[U0ICS]

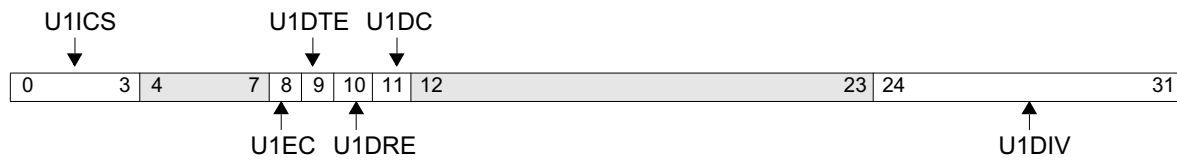
Preliminary User's Manual

Figure 26-14. UART Configuration Register 1 (SDR0_UART1)

0:3	U1ICS	UART 1 Internal Clock Source 0000 Reserved 0001 Reserved 0010 UART 1 internal clock source = PLB 0011 Reserved
4:7		Reserved
8	U1EC	UART 1 EXT Clock Enable 0 UART1 uses the internal serial clock 1 UART1 uses the external serial clock
9	U1DTE	UART 1 DMA Transmit Enable 0 UART1 DMA transmit is disabled 1 UART1 DMA transmit is enabled
10	U1DRE	UART 1 DMA Receive Enable 0 UART1 DMA receive is disabled 1 UART1 DMA receive is enabled
11	U1DC	UART 1 DMA Clear 0 U1DTE and U1DRE are not cleared when UART receives a corresponding terminal count. 1 U1DTE and U1DRE are cleared when UART receives a corresponding terminal count.
12:23		Reserved
24:31	U1DIV	UART 1 Divider 0000_0000 - UART1 divisor = 256 0000_0001 - UART1 divisor = 1 0000_0010 - UART1 divisor = 2 1111_1111 - UART1 divisor = 255 The source for SDR0_UART1[U1DIV] is indicated by SDR0_UART1[U1ICS]

26.5.2 Transmitter DMA Mode

The four DMA channels on the PPC440GX Embedded Processor can be used to move data to and from the two UARTs. The UART0 transmit channel is tied to DMA channel 1 and the UART1 transmit channel is tied to DMA channel 3. Use of the DMA with the UARTs is controlled by the UxDTE bits of the SDR0_UART0 register.

When the DMA mode bit in the UART FIFO control register, UARTx_FCR[DMS], is 0 the DMA request goes active when the FIFOs are disabled or the FIFOs are enabled and there are no characters in the TX FIFO or Transmit Holding Register (THR). Once activated, the DMA request goes inactive after the first character is loaded into the TX FIFO or THR.

When UARTx_FCR[DMS] is 1 the DMA request goes active, when FIFOs are enabled and there is at least one unfilled position in the TX FIFO. DMA request goes inactive when the TX FIFO is completely full. To operate in this mode the corresponding PPC440GX Embedded Processor DMA Channel Control Register must be configured to accept DMA requests from an internal source. Setting the Peripheral Location (PL) bit of DMA0_CRx to 1 configures the DMA channel to accept DMA requests from UARTs on the OPB.

Table 26-5 lists required register settings for UART0 transmit transfers. The UART0 transmit channel is tied to DMA channel 1. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controller* on page 695 for more information.

Table 26-5. UART 0 Transmitter DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART0[U0DTE]=1	UART0 DMA transmit channel is enabled using DMA channel 1
SDR0_UART0[U0DC]	Set to 0 to not clear SDR0_UART0[DTE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA0_CR1[TD]=0	DMA channel 1 transfer direction is from memory to peripheral.
DMA0_CR1[PL]=1	DMA channel 1 peripheral is on the OPB (UART0).
DMA0_CR1[PW]=000	Peripheral width is byte (8 bits).
DMA0_CR1[TM]=00	DMA Channel 1 is in peripheral mode.
DMA0_CR1[PWC]=000010	Peripheral wait cycles, how long the internal DMAAck is active. Three cycles are required.
DMA0_CR1[PHC]=000	Peripheral hold cycles are 0.
DMA0_CR1[ETD]=1	EOT/TC is programmed as terminal count output.
UART0_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.

Note: When using DMA channel 1 for UART0 transmitter transfers, external DMA transfers cannot be performed on this channel.

Preliminary User's Manual

Table 26-6 lists required register settings for UART 1 transmit transfers. The UART1 transmit channel is tied to DMA channel 3. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controller* on page 695 for more information.

Table 26-6. UART 1 Transmitter DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART1[U1DTE]=1	UART 1 DMA transmit channel is enabled using DMA channel 3.
SDR0_UART1[U1DC]	Set to 0 to not clear SDR0_[DTE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA0_CR3[TD]=0	DMA channel 3 transfer direction is from memory to peripheral.
DMA0_CR3[PL]=1	DMA channel 3 peripheral is on the OPB (UART0).
DMA0_CR3[PW]=000	Peripheral width is byte (8 bits).
DMA0_CR3[TM]=00	DMA channel 3 is in peripheral mode.
DMA0_CR3[PWC]=000010	Peripheral wait cycles, how long the internal DMAck is active. Three cycles are required.
DMA0_CR3[PHC]=000	Peripheral hold cycles are 0.
DMA0_CR3[ETD]=1	EOT/TC is programmed as terminal count output.
UART1_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.

Note: When using DMA channel 3 for UART 1 transmitter transfers, external DMA transfers cannot be performed on this channel.

26.5.3 Receiver DMA Mode

The four DMA channels on the PPC440GX Embedded Processor can be used to move data to and from the two UARTs. The UART0 receive channel is tied to DMA channel 0 and the UART1 receive channel is tied to DMA channel 2. Use of the DMA with the UARTs is controlled by the UxDRE bits of the SDR0_UART0 register.

When the DMA mode bit in the UART FIFO control register, UARTx_FCR[DMS], is 0 the DMA request goes active when there is at least one character in the RX FIFO or Receive Buffer Register (RBR). Once activated, the DMA request goes inactive when there are no more characters in the FIFO or RBR.

When UARTx_FCR[DMS] is 1 the DMA request goes active when the FIFOs are enabled and the trigger level or the timeout has been reached. DMA request goes inactive when there are no more characters in the FIFO or RBR. To operate in this mode the corresponding PPC440GX Embedded Processor DMA Channel Control Register must be configured to accept DMA requests from an internal source. Setting the Peripheral Location (PL) bit of DMA0_CRx to 1 configures the DMA channel to accept DMA requests from UARTs on the OPB.

Table 26-7 lists required register settings for UART0 receive transfers. The UART0 receive channel is tied to DMA channel 0. Other DMA registers and register fields must be programmed appropriately, see *Direct Memory Access Controller* on page 695 for more information.

Table 26-7. UART0 Receiver DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART0[U0DRE]=1	UART0 DMA receiver channel is enabled using DMA channel 0.
SDR0_UART0[U0DC]	Set to 0 to not clear SDR0_UART0[DRE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA0_CR0[TD]=1	DMA channel 0 transfer direction is from peripheral to memory.
DMA0_CR0[PL]=1	DMA channel 0 peripheral is on the OPB (UART0).
DMA0_CR0[PW]=000	Peripheral width is byte (8 bits).
DMA0_CR0[TM]=00	DMA channel 0 is in peripheral mode.
DMA0_CR0[PWC]=000010	Peripheral wait cycles, how long the internal DMAAck is active. Three cycles are required.
DMA0_CR0[PHC]=000	Peripheral hold cycles are 0.
DMA0_CR0[ETD]=1	EOT/TC is programmed as terminal count output.
UART0_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.

Note: When using DMA channel 0 for UART0 receiver transfers, external DMA transfers cannot be performed on this channel.

Preliminary User's Manual

Table 26-8 lists required register settings for UART1 receive transfers. The UART1 receive channel is tied to DMA channel 2. Other DMA registers and register fields must be programmed appropriately, see “Direct Memory Access Controller” on page -695 for more information.

Table 26-8. UART1 Receiver DMA Mode Register Field Settings

Register [Field]	Meaning
SDR0_UART1[U1DRE]=1	UART1 DMA receiver channel is enabled using DMA channel 2.
SDR0_UART1[U1DC]	Set to 0 to not clear SDR0_UART1[DRE] enable when terminal count is reached, set to 1 to clear enable when terminal count is reached.
DMA0_CR2[TD]=1	DMA Channel 2 transfer direction is from peripheral to memory.
DMA0_CR2[PL]=1	DMA Channel 2 peripheral is on the OPB (UART0).
DMA0_CR2[PW]=00	Peripheral width is byte (8 bits).
DMA0_CR2[TM]=00	DMA Channel 2 is in peripheral mode.
DMA0_CR2[PWC]=000010	Peripheral wait cycles, how long the internal DMAAck is active. Three cycles are required.
DMA0_CR2[PHC]=000	Peripheral hold cycles are 0.
DMA0_CR2[ETD]=1	EOT/TC is programmed as terminal count output.
UART1_FCR[DMS]	Set to 0 for a single DMA transfer or 1 for multiple DMA transfers.

Note: When using DMA channel 2 for UART1 receiver transfers, external DMA transfers cannot be performed on this channel.

Preliminary User's Manual

27. IIC Bus Interface

The PPC440GX Embedded Processor provides two inter-integrated circuit (IIC) bus interfaces, IIC0 and IIC1. These interfaces comply with the specifications contained in the Phillips ® Semiconductors document The I2C bus and how to use it (including specifications) (1995 update).

Each IIC bus has a two wire, bi-directional, open-drain, low-speed serial interface. The serial clock (IIC0SCLK and IIC1SCLK) and serial data (IIC0SDA and IIC1SDA) lines are bidirectional, to support multiple bus masters and to mix high- and low-speed devices on the same bus.

Each IIC interface (referred to as IIC to distinguish it from the Phillips I 2 C bus) supports the following standard and enhanced features:

- 100-kHz and 400-kHz operation
- 8-bit data transfers
- 7-bit and 10-bit addressing
- Slave transmitter and receiver
- Master transmitter and receiver
- Multiple bus masters

27.1 Addressing

The IIC interfaces support 7-bit and 10-bit addressing for master and slave transfers. Addressing is described in detail in *IICx Low Master Address Register (IICx_LMADR)* on page 930, *IICx High Master Address Register (IICx_HMADR)* on page 931, *IICx Low Slave Address Register (IICx_LSADR)* on page 939, and *IICx High Slave Address Register (IICx_HSADR)* on page 940.

Descriptions of addressing modes and address formats follow.

27.1.1 Addressing Modes

For master transfers, the address mode (AMD) field of the IIC Control register (IICx_CNTL) controls whether 7-bit or 10-bit addresses are used. If IICx_CNTL[AMD] = 0, addresses contain 7 bits; if IICx_CNTL[AMD] = 1, addresses contain 10 bits.

For slave transfers, the contents of the IICx High Slave Address register (IICx_HSADR) determines whether 7-bit or 10-bit addressing is used. If IICx_HSADR = 0b00000000, 7-bit addressing is used. If 10-bit addressing is to be used for slave transfers, IICx_HSADR = 0b11110yyx, where yy contains the high-order bits of the 10-bit address, and x is a don't care.

Programming Note: For slave transfers, IICx_CNTL[AMD] does not control addressing mode.

27.1.2 Seven-Bit Addresses

Figure 27-1 illustrates a 7-bit address. For master transfers, the address bits 0 through 6 (A0:A6) are read from IICx_LMADR. For slave transfers, A0:A6 are read from IICx_LSADR. Bit 7 of address byte 0 contains a transfer type bit provided by the IIC interface.

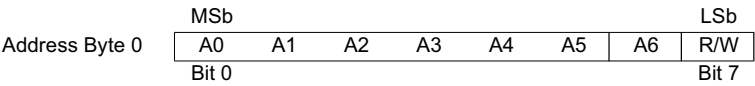


Figure 27-1. 7-Bit Addressing

27.1.3 Ten-Bit Addresses

Figure 27-2 illustrates a 10-bit address. A0:A1 of address byte 0 are read from IICx_HMADR[A6:A7] (for master transfers) or IICx_HSADR[A6:A7] (for slave transfers). These are the two highest-order address bits transmitted on the IIC bus. Bit 7 of address byte 0 contains a transfer type bit provided by the IIC interface.

For 10-bit addressing for master or slave transfers, respectively, IICx_HMADR[A0:A4] and IICx_HSADR [A0:A4] must contain 0b11110.

The low-order byte of the 10-bit address, contained in A0:A7 of address byte 1, are read from IICx_LMADR or IICx_LSADR for master or slave transfers, respectively.

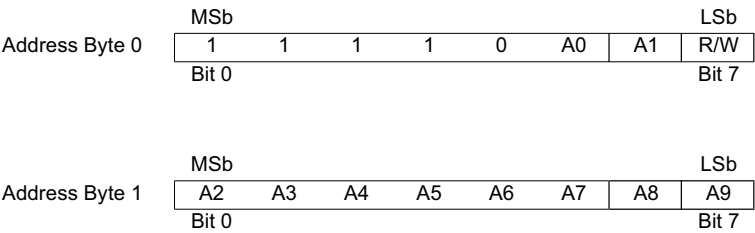


Figure 27-2. 10-Bit Addressing

Preliminary User's Manual**27.2 IIC Registers**

IICx registers are accessed via memory locations 0x1 4000 0XYY, where X=4 for the IIC0 interface and X=5 for the IIC1 interface. YY identifies the register within the IIC0 or IIC1.

Table 27-1 lists the IICx registers. Descriptions of the registers, in the listed order, follow in *IIC Register Descriptions* on page 928.

Table 27-1. IIC Registers

Register	Description	Address	Offset	Access	Effect of Reset	Bits	Page
IICx_MDBUF	IICx Master Data Buffer	0x14000 0X00	0x0	R/W	Cleared	8,16	928
IICx_SDBUF	IICx Slave Data Buffer	0x14000 0X02	0x2	R/W	Cleared	8,16	929
IICx_LMADR	IICx Low Master Address	0x14000 0X04	0x4	R/W	No	8	930
IICx_HMADR	IICx High Master Address	0x14000 0X05	0x5	R/W	No	8	931
IICx_CNTL	IICx Control	0x14000 0X06	0x6	R/W	Cleared	8	932
IICx_MDCNTL	IICx Mode Control	0x14000 0X07	0x7	R/W	Cleared	8	933
IICx_STS	IICx Status	0x14000 0X08	0x8	R/W	Cleared	8	935
IICx_EXTSTS	IICx Extended Status	0x14000 0X09	0x9	R/W	Cleared	8	937
IICx_LSADR	IICx Low Slave Address	0x14000 0X0A	0xA	R/W	No	8	939
IICx_HSADR	IICx High Slave Address	0x14000 0X0B	0xB	R/W	No	8	940
IICx_CLKDIV	IICx Clock Divide	0x14000 0X0C	0xC	R/W	Cleared	8	941
IICx_INTRMSK	IICx Interrupt Mask	0x14000 0X0D	0xD	R/W	Cleared	8	943
IICx_XFRCNT	IICx Transfer Count	0x14000 0X0E	0xE	R/W	Cleared	8	944
IICx_XTCNTLSS	IICx Extended Control and Slave Status	0x14000 0X0F	0xF	R/W	Cleared	8	945
IICx_DIRECTCNTL	IICx Direct Control	0x14000 0X10	0x10	R/W	0x0f	4	947

27.3 IIC Register Descriptions

The following sections contains the bit definitions for the various registers in the IIC interface.

27.3.1 IICx Master Data Buffer (IICx_MDBUF)

The IICx Master Data Buffer (IICx_MDBUF) is a 1-byte × 4-byte first-in/first-out (FIFO) buffer. Data placed in IICx_MDBUF is written onto the IIC bus when performing a master write operation. Data received from the IIC bus is read from IICx_MDBUF when performing a master read operation.

Figure 27-3 describes the IICx_MDBUF register bits.



Figure 27-3. IICx Master Data Buffer (IICx_MDBUF)

0		Data bit
1		Data bit
2		Data bit
3		Data bit
4		Data bit
5		Data bit
6		Data bit
7		Data bit

Halfword reads from and writes to the IIC bus assume the MSB comes first. For halfword writes, the first byte written to the IICx_MDBUF is the MSB, byte 0. The followed byte written is the LSB, byte 1. For halfword reads, the first byte received is MSB, byte 0, and the following byte is LSB, byte 1.

If a byte is read from the IICx_MDBUF while the FIFO is empty, obsolete data is read. Check the master buffer status, IICx_STS[MDBS], before reading IICx_MDBUF.

If a byte is written to IICx_MDBUF while the FIFO is full, the byte is discarded and not placed into the FIFO. Check the master buffer full status, IICx_STS[MDBF], before writing IICx_MDBUF.

IICx_MDBUF is cleared (flushed and set to empty) whenever the IIC interface is reset, or IICx_MDCNTL[FMDB] = 1.

Preliminary User's Manual**27.3.2 IICx Slave Data Buffer (IICx_SDBUF)**

The IICx Slave Data Buffer (IICx_SDBUF) is a 1-byte 4-byte first-in/first-out (FIFO) buffer. The data contained in IICx_SDBUF is either received from the IIC bus when the IIC interface is addressed as a slave during a write, or is written on the IIC bus when the IIC interface is addressed as a slave during a read operation.

Figure 27-4 describes the IICx_SDBUF register bits.

0	7
---	---

Figure 27-4. IICx Slave Data Buffer (IICx_SDBUF)

0		Data bit
1		Data bit
2		Data bit
3		Data bit
4		Data bit
5		Data bit
6		Data bit
7		Data bit

Halfword reads from and writes to the IIC bus assume the MSB comes first. For halfword writes, the first byte written to the IICx_SDBUF is the MSB, byte 0. The followed byte written is the LSB, byte 1. For halfword reads, the first byte received is MSB, byte 0, and the following byte is LSB, byte 1.

If a byte is read from the IICx_SDBUF while the FIFO is empty, obsolete data is read. Check the slave buffer status, IICx_XTCNTLSS[SDBD], before reading IICx_SDBUF.

If a byte is written to IICx_SDBUF while the FIFO is full, the byte is discarded and not placed into the FIFO. Check the slave buffer full status, IICx_XTCNTLSS[SDBF], before writing IICx_SDBUF.

IICx_SDBUF is cleared (flushed and set to empty) whenever the IIC interface is reset, or IICx_MDCNTL[FSDB] = 1.

27.3.3 IICx Low Master Address Register (IICx_LMADR)

The IICx Low Master Address (IICx_LMADR) and IICx High Master Address Register (IICx_HMADR) form addresses that the IIC interface transmits on the IIC bus.

When in 7-bit address mode, IICx_CNTL[AMD] = 0, IICx_LMADR[A0:A6] is the address transmitted on the IIC bus; IICx_LMADR[A7] is a don't care. When in 10-bit address mode, IICx_CNTL[AMD] = 1, IICx_LMADR[A0:A7] is the second byte of the address transmitted on the IIC bus.

Figure 27-5 describes the IICx_LMADR register bits.

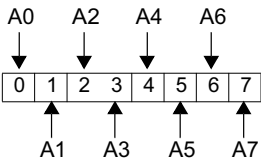


Figure 27-5. IICx Low Master Address Register (IICx_LMADR)

0	A0	Address bit 0
1	A1	Address bit 1
2	A2	Address bit 2
3	A3	Address bit 3
4	A4	Address bit 4
5	A5	Address bit 5
6	A6	Address bit 6 LSb for 7-bit addresses
7	A7	Address bit 7 LSb for 10-bit addresses; don't care for 7-bit addresses

Preliminary User’s Manual

27.3.4 IICx High Master Address Register (IICx_HMADR)

IICx High Master Address Register (IICx_HMADR) provides the upper address bits in 10-bit addressing mode.

When in 10-bit address mode, IICx_CNTL[AMD] = 1, IICx_HMADR must be programmed to 0b1111 0yyz, where yy are the high-order bits of a 10-bit address and z is a don't care.

IICx_HMADR[A5:A6] are the two most significant bits of the 10-bit address. IICx_HMADR[A7] is a don't care.

Figure 27-6 describes the IICx_HMADR register bits.

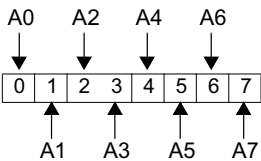


Figure 27-6. IICx High Master Address Register (IICx_HMADR)

0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

27.3.5 IICx Control Register (IICx_CNTL)

The IICx Control Register (IICx_CNTL) starts and stops IIC interface master transfers on the IIC bus. When a transfer begins, the IIC interface uses the values in IICx_CNTL to determine the type and size of the transfer.

Programming Note: IICx_CNTL *must* be the last register programmed. Whenever IICx_CNTL[PT] = 1, the IIC interface attempts to perform the requested transfer using values set in other registers.

During and after transfers, the IICx_STS and IICx_EXTSTS registers can be read to determine the state of the IIC interface and the IIC bus.

Only IICx_CNTL[PT] is cleared when a requested master transfer is complete; the remaining bits are not affected.

Figure 27-7 describes the IICx_CNTL register bits.

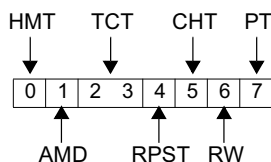


Figure 27-7. IICx Control Register (IICx_CNTL)

0	HMT	Halt Master Transfer 0 Normal transfer operation. 1 Issue Stop signal on the IIC bus as soon as possible to halt master transfer.	If no transfer is in progress, no action is taken. IICx_CNTL[PT] needs not be set. If IICx_MDCNTL[EINT] = 1, an interrupt is generated.
1	AMD	Addressing Mode 0 Use 7-bit addressing. 1 Use 10-bit addressing.	Does not affect slave transfers.
2:3	TCT	Transfer Count 00 Transfer one byte. 01 Transfer two bytes. 10 Transfer three bytes. 11 Transfer four bytes.	
4	RPST	Repeated Start 0 Normal start operation 1 Use repeated Start function to start transfer.	
5	CHT	Chain Transfer 0 Transfer is only or last transfer. 1 Transfer is one of a sequence of transfers (but not last in sequence).	Completion of a requested transfer causes a Stop condition to be generated on the IIC bus.
6	RW	Read/Write 0 Transfer is a write. 1 Transfer is a read.	
7	PT	Pending Transfer 0 Most recent requested transfer is complete. 1 Start transfer if bus is free.	

Preliminary User's Manual

Table 27-2 summarizes IIC interface operation for settings of IICx_CNTL[HMT, RPST, CHT, PT]. x represents a don't care in the RPST, CHT and PT columns.

Table 27-2. IIC Response to IICx_CNTL Field Settings

IICx_CNTL Fields				Resulting Action on IIC Bus and Inside IIC Interface
HMT	RPST	CHT	PT	
0	x	x	0	No action taken
0	0	1	1	Start, Transfer, ACK on last byte, Pause
0	0	0	1	Start, Transfer, NACK on last byte, Stop
1	x	x	x	NACK on current byte, Stop
0	1	x	1	Start, Transfer, NACK on last byte, Wait

Settings of IICx_CNTL[HMT, RPST, CHT, PT] result in the following actions.

Start	IIC Start condition generated, if the IIC interface was stopped or waiting.
Stop	IIC Stop condition generated; IIC interface enters the Stop condition.
ACK	IIC Acknowledge condition generated.
NACK	IIC Not Acknowledge condition generated, if performing a read.
Transfer	Requested bytes are transferred.
Pause	IIC interface enters the Pause state.
Wait	IIC interface enters the Wait state.

IICx_CNTL[HMT] overrides IICx_CNTL[RPST, CHT].

IICx_CNTL[RPST, PT] overrides IICx_CNTL[CHT].

27.3.6 IICx Mode Control Register (IICx_MDCNTL)

The IICx Mode Control Register (IICx_MDCNTL) sets the major modes of operation on the IIC bus. In addition, IICx_MDCNTL can force the data buffers into the empty state.

In typical applications, IICx_MDCNTL is configured once, during software initialization. Applications providing complex error handling may reconfigure this register more often.

Programming Note: IICx_CLKDIV must be initialized before IICx_MDCNTL. IICx_LSADR and IICx_HSADR should also be configured before IICx_MDCNTL.

Note that the IIC hardware does not implement time-out functions on the IIC bus. Such functions must be implemented, in software, by setting IICx_CNTL[HMT] = 1, or setting IICx_XTCNTLSS[SRST] = 1.

Regarding IICx_MDCNTL[HSCL], a "slave not ready" condition occurs during a slave receive operation, if there is no space in the slave data buffer at the start of a write operation, or if the slave data buffer fills during the write. In a slave transmit operation, a slave not ready condition occurs if there is no data in the slave data buffer at the start of a read operation, or if the slave data buffer becomes empty during the read.

Using IICx_MDCNTL[HSCL] to handle slave not ready conditions can affect system performance. A slave holding the IIC_SCL signal low guarantees data delivery to or from the requesting master, but prevents other masters from performing transfers over the IIC bus. There is no general rule for handling slave not ready conditions; each system has its own requirements.

Figure 27-8 describes the IICx_MDCNTL register bits.

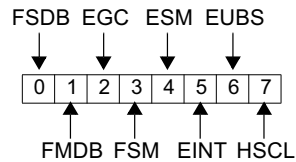


Figure 27-8. IICx Mode Control Register (IICx_MDCNTL)

0	FSDb	Flush Slave Data Buffer 0 Normal operation 1 Set slave data buffer to empty.	Cleared after buffer is emptied.
1	FMDB	Flush Master Data Buffer 0 Normal operation 1 Set master data buffer to empty.	Cleared after buffer is emptied.
2	EGC	Enable General Call 0 Ignore general call on IIC bus. 1 Respond to general call on IIC bus.	IICx_MDCNTL[ESM] overrides this field; if IICx_MDCNTL[ESM] = 1, a general call is ignored.
3	FSM	Fast/Standard Mode 0 IIC transfers run at 100 kHz (standard mode). 1 IIC transfers run at 400 kHz (fast mode).	
4	ESM	Enable Slave Mode 0 Slave transfers are ignored. 1 Slave transfers are enabled.	Program IICx_LSADR and IICx_HSADR before setting this field.
5	EINT	Enable Interrupt 0 Interrupts are disabled. 1 Enables interrupts for interrupts enabled in IICx_NTRMSK.	
6	EUBS	Exit Unknown IIC Bus State 0 Normal operation. 1 IIC bus control state machine exits unknown bus state, if in an unknown state.	If the IIC bus control state machine is in a known state, setting IICx_MDCNTL[EUBS] = 1 has no effect.
7	HSCL	Hold IIC Serial Clock Low 0 If slave is not ready, issue a NACK in response to slave transfer request. 1 If slave is not ready, hold the IICsCL signal low until slave is ready.	This field is used only when in slave mode.

Preliminary User's Manual**27.3.7 IICx Status Register (IICx_STS)**

The IICx Status register (IICx_STS) contains the state of the IIC interface and the status of any previously requested master transfers.

During and after transfers, software can read the IICx_STS and IICx_EXTSTS registers to determine the state of the IIC interface and the IIC bus.

Programming Note: IICx_STS should be the first register read by an interrupt or error handler routine. IICx_STS can also be read in a polling loop if the IIC interrupts are not used.

All IICx_STS bit fields except for IICx_STS[SSS] must be cleared before requesting another master transfer. IICx_STS[SSS] is not affected by master transactions.

Figure 27-9 describes the IICx_STS register bits.

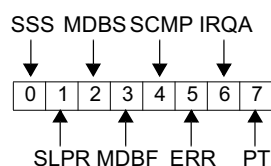


Figure 27-9. IICx Status Register (IICx_STS)

0	SSS	Slave Status Set 0 No slave operations are in progress. 1 Slave operation is in progress.	Read-only; this field is set when any of the following fields are set: IICx_XTCNTLSS[SRG, SRS, SWC, SWS].
1	SLPR	Sleep Request 0 Normal operation. 1 Sleep mode (CPR0_ER[IIC] = 1).	Read-only. The IIC interface is awakened when a start signal is detected on the IIC bus or when the CPR0_ER[IICx] is cleared.
2	MDBS	Master Data Buffer Status 0 Master data buffer is empty. 1 Master data buffer contains data.	Read-only.
3	MDBF	Master Data Buffer Full 0 Master data buffer is not full. 1 Master data buffer is full.	Read-only.
4	SCMP	Stop Complete 0 No request to halt transfer, or master data transfer, is complete. 1 Request to halt transfer, or master data transfer, is complete.	To clear IICx_STS[SCMP], set IICx_STS[SCMP] = 1.
5	ERR	Error 0 No error has occurred. 1 One of the following fields is set: IICx_EXTSTS[LA, ICT, XFRA] = 1.	Read-only.
6	IRQA	IRQ Active 0 No IIC interrupt has been sent to the universal interrupt controller (UIC). 1 An IIC interrupt has been sent to the UIC.	To clear IICx_STS[IRQA], set IICx_STS[IRQA] = 1. If IICx_MDCNTL[EINT] = 0, then IICx_STS[IRQA] is not set.
7	PT	Pending Transfer 0 No transfer is pending, or transfer is in progress. 1 Transfer is pending.	Read-only.

The Error and Pending Transfer, IICx_STS[ERR, PT], bit fields indicate the success or failure of the requested transfer. Table 27-3 lists the transfer status for all combinations of the IICx_STS[ERR,PT] bit fields.

Table 27-3. IICx_STS[ERR, PT] Decoding

ERR	PT	Status
0	0	Requested transfer completed without errors
0	1	Requested transfer is in progress; no errors were detected
1	0	Requested transfer is complete, but not all data was transferred
1	1	Requested transfer is in progress; but an error was detected

Programming Note: No action regarding a master transfer should be taken unless all pending transfers have completed, IICx_STS[PT] = 0.

If an error requires the IIC interface to send a Stop, the Stop Complete bit field is set, IICx_STS[SCMP] = 1. Note that slave operations should be serviced regardless of the state of a requested master transfer.

IIC1_MDCNTL[EUBS] must be set after a reset before IIC interface can be placed in sleep mode. The IIC interface is placed in sleep mode by setting the CPM0_ER[IICx] via software. Awakening the IIC interface is possible directly through software by clearing the CPM0_ER[IICx] or indirectly by detecting a Start condition on the IIC bus. When a Start condition is detected, the IIC interface is awakened, clearing the CPM0_ER[IICx] and the IICx_STS[SLPR].

The IICx_STS[MDBS, MDBF] contain the current status of the Master Data Buffer, IICx_MDBUF. When the IICx_MDBUF contains data, IICx_STS[MDBS] is set. When the IICx_MDBUF is full, IICx_STS[MDBF] is set.

The state of the IICx_MDBUF is not instantly recorded by the IICx_STS[MDBS, MDBF]. The delay depends on the size of the buffer access. For halfword accesses, these fields are valid on the third OPB clock following the transfer. For byte accesses, these fields are valid on the second OPB clock following the transfer.

Preliminary User's Manual**27.3.8 IICx Extended Status Register (IICx_EXTSTS)**

The IICx Extended Status register (IICx_EXTSTS) reports additional IIC status.

During and after transfers, software can read the IICx_STS and IICx_EXTSTS registers to determine the state of the IIC interface and the IIC bus.

Figure 27-10 describes the IICx_EXTSTS register bits.

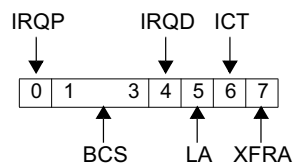


Figure 27-10. IICx Extended Status Register (IICx_EXTSTS)

0	IRQP	<p>IRQ Pending</p> <p>0 No IRQ is pending.</p> <p>1 An IRQ is active, another IRQ is on-deck, and another interrupt-generating condition has occurred.</p>	<ul style="list-style-type: none"> IICx_EXTSTS[IRQP] might be set momentarily while an IRQ moves from the Pending to the On-deck state. An interrupt remains pending, IICx_EXTSTS[IRQP]=1, until the current on-deck interrupt becomes active, IICx_STS[IRQD]=0 and IICx_STS[IRQA]=1. Writing 1 to IICx_EXTSTS[IRQP] clears the field. When the IIC interrupt is disabled, IICx_MDCNTL[IRQP] = 0, IICx_EXTSTS[IRQP] should be ignored.
1:3	BCS	<p>Bus Control State</p> <p>000 Unused; if this value is read an error occurred.</p> <p>001 Slave-selected state; the IIC interface has detected and decoded a slave transfer request on the IIC bus.</p> <p>010 Slave Transfer state; the IIC interface has detected but has not decoded a slave transfer request on the IIC bus.</p> <p>011 Master Transfer state; entered after a master transfer request has started on the IIC bus.</p> <p>100 Free Bus state; the bus is free and no transfer request is pending.</p> <p>101 Busy Bus state; the bus is busy.</p> <p>110 Unknown state; value after IIC reset.</p> <p>111 Unused; if this value is read an error occurred.</p>	<p>Read-only.</p>

4	IRQD	<p>IRQ On-Deck</p> <p>0 No IRQ is on-deck. 1 An interrupt is active, and another interrupt-generating condition has occurred.</p>	<ul style="list-style-type: none"> • IICx_EXTSTS[IRQD] might be set momentarily while an IRQ moves from the On-deck to the Active state. • An interrupt remains on-deck, IICx_EXTSTS[IRQD] = 1, until the current active interrupt is no longer active, IICx_STS[IRQA] = 0. • If IICx_EXTSTS[IRQP] = 1, IICx_EXTSTS[IRQD] is set on the next OPB clock. • Writing 1 to IICx_EXTSTS[IRQD] clears the field. • When the IIC interrupt is disabled, IICx_MDCNTL[IRQP]=0, IICx_EXTSTS[IRQD] should be ignored.
5	LA	<p>Lost Arbitration</p> <p>0 Normal operation. 1 Loss of arbitration has ended the requested master transfer.</p>	<ul style="list-style-type: none"> • If arbitration is lost, any requested master transaction may have terminated prematurely. Read data may be incomplete and not all write data may have been written. • If arbitration is lost during a repeat start, the master may not own the IIC bus.
6	ICT	<p>Incomplete Transfer</p> <p>0 Normal operation. 1 Some of the bytes of the requested master transfer were not transferred.</p>	For an incomplete transfer, read the transfer count, IICx_XFRCNT, to determine how bytes were transferred.
7	XFRA	<p>Transfer Aborted</p> <p>0 No transfer is pending, or transfer is in progress. 1 A requested master transfer was aborted by a NACK during the transfer of the address byte, or was aborted because arbitration was lost. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.</p>	Transfer aborted. When set to a 1, a requested master transfer was aborted by a NOT acknowledge during the transfer of the address byte. It is also set to a 1 when a requested master transfer loses data. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.

IICx_EXTSTS[IRQP, IRQD] and IICx_STS[IRQA] provide a FIFO for storing interrupts. A new interrupt is considered pending, and remains pending while an on-deck interrupt is present. Once the on-deck interrupt becomes active, the pending interrupt moves on-deck, and remains on-deck until there is no active interrupt. When the active interrupt is cleared, the on-deck (initially pending) interrupt becomes active.

Programming Note: An active interrupt remains active until software clears it.

IICx_EXTSTS[BCS] indicates the state of the IIC interface. The field is read-only.

IICx_EXTSTS[LA, ICT, XFRA] are cleared when IICx_EXTSTS[XFRA] = 1.

Writing 1 to IICx_EXTSTS[IRQP, IRQD, LA, ICT, XFRA] clears these fields.

Preliminary User’s Manual

27.3.9 IICx Low Slave Address Register (IICx_LSADR)

The IICx Low Slave Address Register (IICx_LSADR) and IICx_ High Slave Address Register (IICx_HSADR) program the slave address of the IIC interface. IICx_HSADR is used only for 10-bit addressing, and is not programmed in 7-bit addressing mode.

When 7-bit addressing is used, IICx_LSADR is written with the slave address; IICx_HSADR must be written with zeros. For 7-bit addressing, IICx_LSADR[A0:A6] contain the address transmitted on the IIC bus; IICx_LSADR[A7] is a don't care.

When 10-bit addressing is used, IICx_LSADR[A0:A7] contain the second address byte transmitted on the IIC bus.

Figure 27-11 describes the IICx_LSADR register bits.

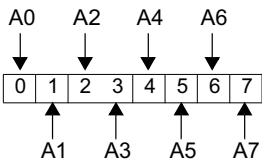


Figure 27-11. IICx Low Slave Address Register (IICx_LSADR)

0	A0	Address bit 0
1	A1	Address bit 1
2	A2	Address bit 2
3	A3	Address bit 3
4	A4	Address bit 4
5	A5	Address bit 5
6	A6	Address bit 6 LSb for 7-bit addresses
7	A7	Address bit 7 LSb for 10-bit addresses; don't care for 7-bit addresses

27.3.10 IICx High Slave Address Register (IICx_HSADR)

For 7-bit addressing, set IICx High Slave Address Register (IICx_HSADR) to 0.

To enable 10-bit slave addressing, IICx_HSADR must be programmed to 0b1111 0yyx, where yy are the high-order bits of a 10-bit address and x is a don't care.

Programming Note: IICx_HSADR is used only for 10-bit addressing, and should be set to 0 for 7-bit addressing mode.

Thus, in 10-bit address mode, IICx_HSADR[A6:A7] contain the two highest -order bits of the 10-bit address; IICx_HSADR[A7] is a don't care. IICx_LSADR contains the low-order byte of the 10-bit address.

Figure 27-12 describes the IICx_HSADR register bits.

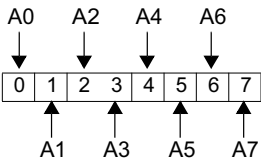


Figure 27-12. IICx High Slave Address Register (IICx_HSADR)

0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

Thus, in 10-bit address mode, bits 0:6 are used to decode the first address byte that was transmitted on the IIC bus, and bit 7 is in a don't care.

Preliminary User's Manual

27.3.11 IICx Clock Divide Register (IICx_CLKDIV)

The IICx Clock Divide Register (IICx_CLKDIV) establishes a reference between the OPB clock and the IIC bus serial clock.

Programming Note: IICx_CLKDIV must be initialized before IICx_MDCTRL. Until IICx_CLKDIV is initialized, all IIC bus activity is ignored.

Figure 27-13 describes the IICx_CLKDIV register bits.

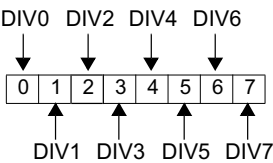


Figure 27-13. IICx Clock Divide Register (IICx_CLKDIV)

0	DIV0	Divisor bit 0
1	DIV1	Divisor bit 1
2	DIV2	Divisor bit 2
3	DIV3	Divisor bit 3
4	DIV4	Divisor bit 4
5	DIV5	Divisor bit 5
6	DIV6	Divisor bit 6
7	DIV7	Divisor bit 7

IICx_CLKDIV divides PPC440GX Embedded Processor’s on-chip peripheral bus (OPB) clock to form the base clock for the IIC bus.

Table 22-4 lists the divisor values for several OPB frequency ranges. These divisor values apply for standard and fast mode. Select the divisor value by matching the OPB clock frequency to the corresponding frequency range in Table 22-4. For example, if the OPB clock frequency is 50MHz, select a divisor value of 0x4.

Table 27-4. IICx Clock Divide Programming

OPB Frequency Range (MHz)	DivisorValue
20	0x1
$20 < f \leq 30$	0x2
$30 < f \leq 40$	0x3
$40 < f \leq 50$	0x4
$50 < f \leq 60$	0x5
$60 < f \leq 70$	0x6
$70 < f \leq 80$	0x7
$80 < f \leq 90$	0x8
$90 < f \leq 100$	0x9
$100 < f \leq 110$	0xA
$110 < f \leq 120$	0xB
$120 < f \leq 130$	0xC
$130 < f \leq 140$	0xD
$140 < f \leq 150$	0xE

Preliminary User's Manual**27.3.12 IICx Interrupt Mask Register (IICx_INTRMSK)**

The IICx Interrupt Mask Register (IICx_INTRMSK) specifies which conditions can generate an IIC interrupt when the IIC interrupt is enabled, IICx_MDCNTL[EINT]=1.

Figure 27-14 describes the IICx_INTRMSK register bits.

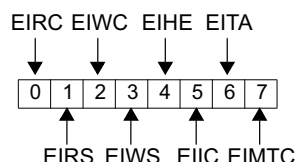


Figure 27-14. IICx Interrupt Mask Register (IICx_INTRMSK)

0	EIRC	Enable IRQ on Slave Read Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave read on the IIC bus. IICx_XTCNTLSS[SR] = 1 indicates a Slave Read Complete.
1	EIRS	Enable IRQ on Slave Read Needs Service 0 Disable 1 Enable	The interrupt is activated upon receipt of a slave read on the IIC bus and the slave buffer was empty or went empty and more data was requested on the IIC bus. Note: IICx_XTCNTLSS[SRS] = 1 indicates a Slave Read Needs Service.
2	EIWC	Enable IRQ on Slave Write Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave write on the IIC bus. Note: IICx_XTCNTLSS[SWC] = 1 indicates a Slave Write Complete.
3	EIWS	Enable IRQ on Slave Write Needs Service 0 Disable 1 Enable	The interrupt is activated when the slave buffer becomes full during a slave write on the IIC bus. Note: IICx_XTCNTLSS[SWS] = 1 indicates a Slave Write Needs Service.
4	EIHE	Enable IRQ on Halt Executed 0 Disable 1 Enable	
5	EIIC	Enable IRQ on Incomplete Transfer 0 Disable 1 Enable	
6	EITA	Enable IRQ on Transfer Aborted 0 Disable 1 Enable	
7	EIMTC	Enable IRQ on Requested Master Transfer Complete 0 Disable 1 Enable	

27.3.13 IICx Transfer Count Register (IICx_XFRCNT)

The IICx Transfer Count Register (IICx_XFRCNT) reports the number of bytes transferred on the IIC bus during a master or a slave operation.

Figure 27-15 describes the IICx_XFRCNT register bits.

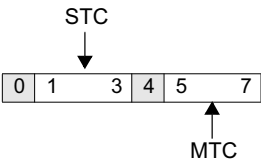


Figure 27-15. IICx Transfer Count Register (IICx_XFRCNT)

0		Reserved
1:3	STC	Slave Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved
4		Reserved
5:7	MTC	Master Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved

IICx_XFRCNT[MTC] is cleared when there is a pending transfer, IICx_CNTL[PT] = 1.

IICx_XFRCNT[STC] is cleared when:

- A slave operation starts on the IIC bus
- Software indicates the slave does not need service by clearing IICx_XTCNTLSS[SRS] or IICx_XTCNTLSS[SWS].

Preliminary User's Manual**27.3.14 IICx Extended Control and Slave Status Register (IICx_XTCNTLSS)**

The IICx Extended Control and Slave Status Register (IICx_XTCNTLSS) provides additional control of IIC interface functions and reports the status of slave operations.

Figure 27-16 describes the IICx_XTCNTLSS register bits.

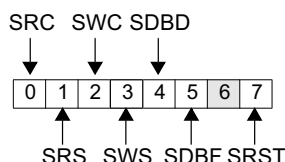


Figure 27-16. IICx Extended Control and Slave Status Register (IICx_XTCNTLSS)

0	SRC	<p>Slave Read Complete</p> <p>0 Normal operation, or IICx_MDCNTL[HSCL] = 0, IICx_SDBUF is empty, and a read operation is in progress.</p> <p>1 A NACK or Stop condition was received over the IIC bus, or a repeated Start condition ended a read operation.</p>	Check whether the read operation emptied IICx_SDBUF.
1	SRS	<p>Slave Read Needs Service</p> <p>0 Normal operation or slave read does not need service.</p> <p>1 IICx_SDBUF is empty, and a read operation was requested on the IIC bus.</p> <p>The set condition may also indicate that IICx_SDBUF is empty due to a slave read and additional data is requested by the master.</p>	<p>1. If IICx_MDCNTL[HSCL]=0 and IICx_SDBUF contains no data, the slave will issue a NACK and IICx_XTCNTLSS[SRS] is set.</p> <p>2. If IICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master request additional data.</p> <p>3. If IICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains no data, the slave will hold IIC_SCL low to indicate the slave is busy. IICx_XTCNTLSS[SRS] is set until the IICx_SDBUF is filled. Once filled, IIC_SCL is released, IICx_XTCNTLSS[SRS] is cleared, and the slave sends the data.</p> <p>4. If IICx_MDCNTL[HSCL]=1, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master requests additional data.</p>
2	SWC	<p>Slave Write Complete</p> <p>0 Normal operation or slave write in progress.</p> <p>1 A Stop signal was received during a write operation, or a repeated Start condition ended a write operation.</p>	
3	SWS	<p>Slave Write Needs Service</p> <p>0 Normal operation or slave write does not need service.</p> <p>1 IICx_SDBUF is full during a slave write.</p>	<p>1. If IICx_MDCNTL[HSCL] = 1 and IICx_SDBUF is full, the slave will hold IIC_SCL low to indicate the slave is busy. IICx_XTCNTLSS[SWS] is set until IICx_SDBUF is empty. Once empty, IIC_SCL is released, IICx_XTCNTLSS[SWS] is cleared, and the slave receives the data.</p> <p>2. If IICx_MDCNTL[HSCL] = 0 and IICx_SDBUF is full, the slave will issue a NACK and IICx_XTCNTLSS[SWS] is set.</p>
4	SDBD	<p>Slave Data Buffer Has Data</p> <p>0 IICx_SDBUF is empty</p> <p>1 IICx_SDBUF contains data</p>	Read-only

5	SDBF	Slave Data Buffer Full 0 IICx_SDBUF is not full 1 IICx_SDBUF is full	Read-only
6		Reserved	
7	SRST	Soft Reset 0 Normal operation 1 Soft reset	

Writing a 1 to IICx_XTCNTLSS[SRX, SRS, SWC, SWS] clears these fields.

The IICx_XTCNTLSS[SBSS, SDBF] contain the current status of the Slave Data Buffer, IICx_SDBUF. When the IICx_SDBUF contains data, IICx_XTCNTLSS[SDBD] is set. When the IICx_SDBF is full, IICx_XTCNTLSS[SDBF] is set.

The state of the IICx_SDBUF is not instantly recorded by the IICx_XTCNTL[SDBD, SDBF]. The delay depends on the size of the buffer access. For half-word accesses, these fields are valid on the third OPB clock following the transfer. For byte accesses, these fields are valid on the second OPB clock following the transfer.

If any of the following fields: IICx_XTCNTLSS[SRX, SRS, SWC, SWS] = 1 and IICx_MDCNTL[HSCL] = 0; no new slave operations will be accepted over the IIC bus. A NACK is issued until IICx_XTCNTLSS[SRS] or IICx_XTCNTLSS[SRS], and IICx_XTCNTLSS[SRS] or IICx_XTCNTLSS[SRS], are cleared.

Soft reset, IICx_XTCNTLSS[SRST], provides a last means of recovery from IIC interface or IIC bus failure. Once enabled, soft reset completely resets the IIC interface. All IIC registers are affected. All transmissions from the IIC interface are terminated. Enabling soft reset during an IIC transmission may improperly terminate the transmission and hang the IIC bus.

Preliminary User's Manual**27.3.15 IICx Direct Control Register (IICx_DIRECTCNTL)**

The IICx Direct Control Register (IICx_DIRECTCNTL), which controls and monitors the IIC serial clock (IICSCL) and serial data (IICSDA) signal, is used for error recovery when a malfunction is detected on the IIC interface.

Figure 27-17 describes the IICx_DIRECTCNTL register bits.

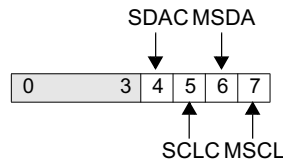


Figure 27-17. IICx Direct Control Register (IICx_DIRECTCNTL)

0:3		Reserved
4	SDAC	IICSDA Output Control Directly controls the IICSDA output. 0 IICSDA is a logic 0 1 IICSDA is a logic 1
5	SCLC	IICSCL Output Control Directly controls the IICSCL output 0 IICSCL is a logic 0 1 IICSCL is a logic 1
6	MSDA	Monitor IICSDA Used to monitor the IICSDA input 0 IICSDA is a logic 0 1 IICSDA is a logic 1 Read-only
7	MSCL	Monitor IICSCL. Used to monitor the IICSCL input. 0 IICSCL is a logic 0 1 IICSCL is a logic 1 Read-only

IICx_DIRECTCNTL[SDAC, SCC] can be written to control the IICSDA and IICSCL signals. When controlling the IICSDA and IICSCL signals directly, the IIC controller must be placed in the reset state, IICx_XTCNTL[SRST] = 1.

IICx_DIRECTCNTL[MSDA, MSC] are used to verify that IICx_DIRECTCNTL[SDAC, SCC] were written successfully, and that the IICSCL signals can be controlled. If IICx_DIRECTCNTL[MSDA, MSC] do not correspond to IICx_DIRECTCNTL[SDAC, SCC], respectively, toggle IICSCL repeatedly to regain control.

IICx_DIRECTCNTL[SDAC, SCC, MSDA, MSC] = 1 after a chip or system reset. A Soft Reset, IICx_XTCNTLSS[SRST] = 1, does not affect the state of IICx_DIRECTCNTL.

27.4 Interrupt Handling

Service request on the IIC interface can be monitored by polling the IICx_STS[SSS, PT] or by using the IIC interrupt to the UIC. When the IICx_MDCNTL[EINT] is enabled, the IIC interface generates an interrupt to the UIC if an unmasked IIC interrupt condition occurs. The interrupt is recorded by the UIC if UIC0_ER[IICx] is enabled. The conditions that generate an IIC interrupt to the UIC are determined by the interrupt mask settings in IICx_INTMSK.

The IIC interface can queue up to three interrupts since there is one interrupt for master operations and two for slave operations. The current interrupt is referred to as the *active* interrupt. The first interrupt in the queue is the on-deck interrupt; the second queued interrupt is called the pending interrupt. The queue holds multiple interrupts until the active interrupt is cleared by writing a 1 to IICx_STS[IRQA]. When an active interrupt is cleared, the on-deck interrupt becomes the active interrupt and the pending interrupt becomes the on-deck interrupt.

When multiple interrupts occur, the status of the active interrupt and the queued interrupt merge making it impossible to determine which of the conditions originated the active interrupt. An interrupt handle should therefore save the contents of the status registers (IICx_STS and IICx_XTCNTLSS) and handle all conditions set. Once handled, the status conditions can be cleared by overwriting the status registers with their saved content. If another IIC interrupt condition occurs before clearing the status register, the status bit of this condition is preserved since status bits are cleared when written with a 1.

When multiple interrupts occur, the status of the active interrupt and the queued interrupt merge making it impossible to determine which of the conditions originated the active interrupt. An interrupt handle should therefore save the contents of the status registers (IICx_STS and IICx_XTCNTLSS) and handle all conditions set. Once handled, the status conditions can be cleared by overwriting the status registers with their saved content. If another IIC interrupt condition occurs before clearing the status register, the status bit of this condition is preserved since status bits are cleared when written with a 1.

Under certain conditions, the IIC interface merges slave read (write) needs service and slave read (write) complete interrupts into one interrupt. If a slave read (write) needs service interrupt is active, or queued, and a slave read (write) complete interrupt occurs, and IICx_XTCNTLSS has not yet been read, the two interrupts are merged into a single interrupt. This merge function is performed in the IIC interface logic, and is not under software control.

27.5 General Considerations

1. After a reset, the IIC interface enters the unknown IIC bus state. This state is exited when either activity is seen on the bus or when the exit unknown IIC bus state bit (in the mode control register), is set to a 1. If the IIC interface is being used in a single master system as the master, then the exit unknown IIC bus state bit must be used to force the logic out of the unknown state.
2. Once a byte is written into either the master or slave buffer, a total of four OPB clock periods must occur before the data can be read. Flushing the master or slave buffer also requires four OPB clock periods to complete.
3. IICx_DIRECTCNTL [MSDA, MSC] are used to verify that IICx_DIRECTCNTL [SDAC, SCC] were written successfully, and that the IIC_SCL signals can be controlled. If IICx_DIRECTCNTL [MSDA, MSC] do not correspond to IICx_DIRECTCNTL [SDAC, SCC], respectively, toggle IIC_SCL repeatedly to regain control.
4. The master and slave buffers are 4×1 byte-wide FIFOs. Exercise care when using master and slave buffers. As an example, consider the case where one byte of data is written on the IIC bus. The data is first written into the master or slave buffer. After four OPB clock cycles the data is placed on the IIC bus. There is no way to verify data in the buffer without disturbing the IIC transaction. The act of verification requires removing the data. If the data is removed, invalid data is placed on the IIC bus.

Preliminary User's Manual

5. Use care when monitoring the IICx_XCNTLSS[SDBD] or to IICx_STS[MDBS] to determine when data is present. These bits are set to 1 when the buffer contains data in any stage. Consider the case where the master buffer is empty prior to being loaded with a byte received over the IIC bus. The byte enters the fourth stage of the buffer and the IICx_STS[MDBS] is set to 1. Stages 1, 2, and 3 do not contain data. Therefore, the data is not available for four OPB clock cycles. Any attempt to prematurely read the data yields invalid data.
6. When responding to a slave needs service request, manage the data first. Read data out of the slave buffer, IICx_SDB, for slave reads or write data into the IICx_SDB for slave writes. Next clear the slave needs service request. For reads, clear IICx_XCNTLSS[SRS]. For writes, clear IICx_XCNTLSS[SWS]. Last, clear the active interrupt, IICx_STS[IRQA] = 0.
7. There is no timeout function implemented in the IIC interface. If this type of error recovery function is needed, it must be implemented in software.
8. Avoid the situations listed in Section 7.2 of the *Phillips Semiconductors I²C Specification*, dated 1995. For your convenience, the section is summarized as follows:

If multiple masters can be simultaneously involved in a transfer to the same address, or device, then the design of the system must be done in such a way that arbitration between:

- A repeated Start condition and a data bit does not occur.
- A Stop condition and a data bit does not occur.
- A repeated Start condition and a Stop condition does not occur.

An example of this situation occurs if one master writes 1 byte while another master writes 2 bytes to the same device.

Preliminary User's Manual

28. GPIO Operations

This chapter describes the General Purpose I/O (GPIO) controller located on the on-chip peripheral bus (OPB) of the PPC440GX. Thirty two of the functional I/O were implemented using the GPIO controller. Function pins that are not used in the application can be used as GPIOs.

28.1 GPIO Controller Overview

The GPIO Controller is an OPB macro that controls up to 32 bidirectional module I/O pins with user-programmable functions. To reduce the quantity of module I/O on the PPC440GX package there are no dedicated general purpose I/Os. Each of the GPIOs has been assigned a function pin such as an interrupt. However, if not all of the pins are used functionally they can be used as GPIO. The pin function control register 0 (SDR0_PFC0) controls whether a pin is used for a defined function or as a GPIO. See *Pin Function Control Register 0 (SDR0_PFC0)* on page 953

All module I/O inputs are synchronized to the OPBCLK before being stored in the Input register. All registers, except the Input Register, are both read and write accessible. The Input register is read-only. Registers provide direct control of all GPIO Controller functions. All register bits and core input and output signals maintain a bit-for-bit correspondence. For example, GPIO_Out[20] is controlled by GPIO0_OR Register Bit 20.

28.2 Features

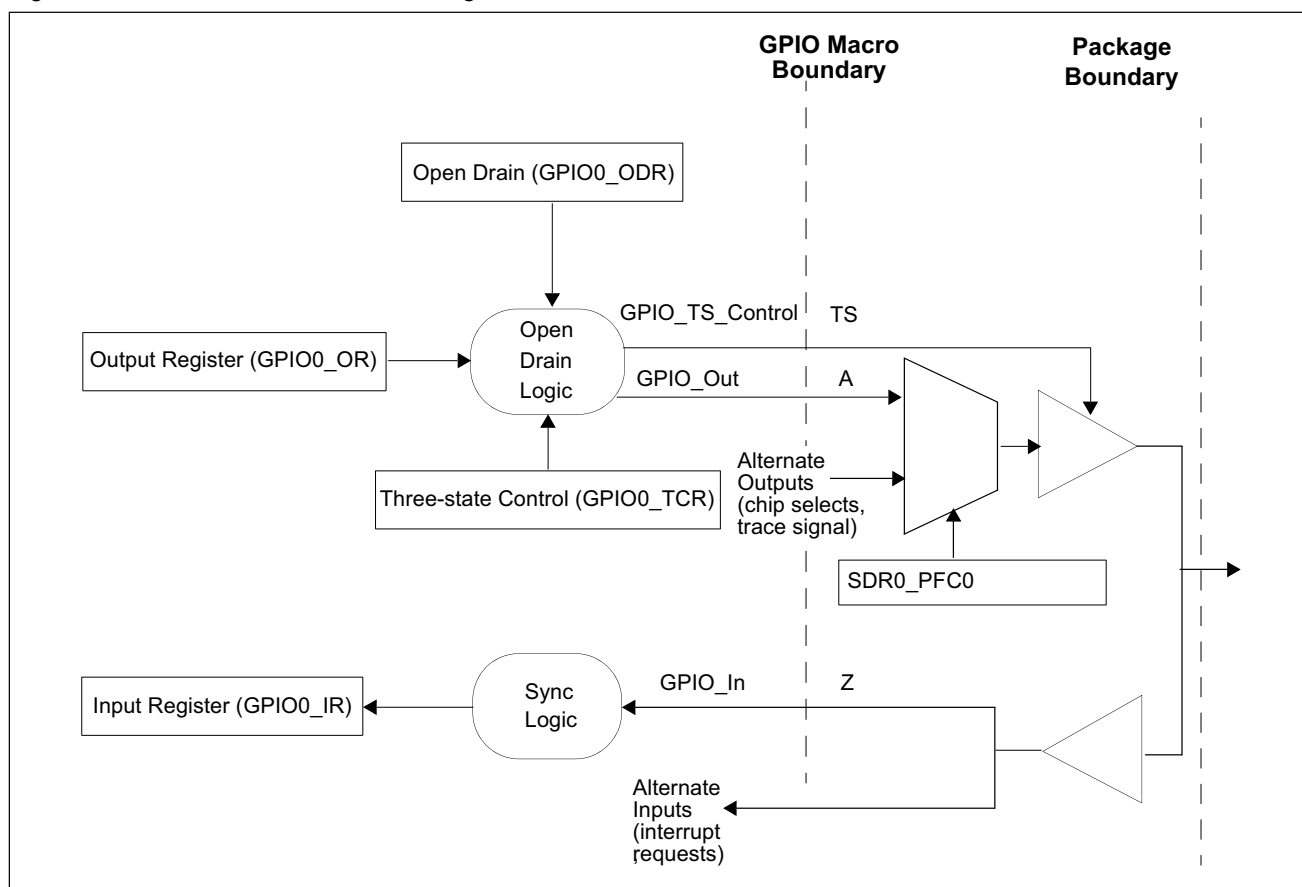
The GPIO Controller has the following features:

- Direct control of all functions from registers programmed via memory-mapped addresses
- Control of 32 bidirectional GPIO module pins
 - Each GPIO output has programmable three-state control
 - Each GPIO output is separately programmable to emulate an open drain driver
 - Each GPIO input is observable from a register bit

28.3 GPIO Interface Signals

The selection between a pin being a designated function or a GPIO is controlled by the pin function control register 0 (SDR0_PFC0). Once a pin is selected as a GPIO, it is controlled by a corresponding bit in 4 registers; the GPIO Input Register (GPIO_IR), GPIO Output Register, GPIO_OR, GPIO Output Tri-state Control register, GPIO_TCR, and GPIO Output Open Drain Control register, GPIO_ODR. *Figure 28-1* shows how the GPIO macro is multiplexed with a function pin.

Figure 28-1. GPIO Functional Block Diagram



Note 1: All the registers within the GPIO controller are synchronous with the OPBCLK, and all GPIO inputs are synchronized to OPBCLK before being stored. Pin multiplexing is controlled by the setting of SDR0_PFC0[0:18] as follows:

- Function Inputs - UART, Interrupts
- Function Outputs - UART, Trace

Note 2: For GPIO bits 0-17 these pins default to function pins. If these pins are used as GPIO the default state must be accounted for in the design. For example GPIO bits 0-10 default to inputs as interrupts. If they are used as GPIO inputs simply setting the appropriate bits in the SDR0_PFC0 register is sufficient. However, if they are used as GPIO outputs they will power-up as floating and may need to have a pull-up or pull-down, to maintain an inactive state on attached logic, until they can be configured by software. Conversely GPIO bits that default to function outputs may conflict with external logic if they are used as inputs; until they can be configured.

Preliminary User's Manual

28.3.1 External Module Signals

The GPIO signals do not have dedicated module pins. The module pins, used by the GPIO signals, are shared (multiplexed) with other signals. The pin function control register 0 (SDR0_PFC0) controls the way in which the signal on a shared pin is interpreted. The following sections provide details of the SDR0_PFC0 register bit settings and the function of the GPIO registers.

The GPIO signals are multiplexed with eleven interrupt signals, four UART1 signals and two IIC1 signals. Multiplexing is controlled by the setting of SDR0_PFC0[0:18] as follows. In all cases when a pin is used as a GPIO, it is necessary to configure the appropriate bit in this register as well as the GPIO registers located in the GPIO controller.

28.3.1.1 Pin Function Control Register 0 (SDR0_PFC0)

Figure 28-2 describes SDR0_PFC0 register bits.

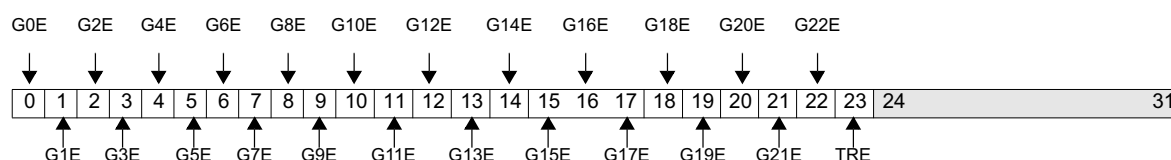


Figure 28-2. Pin Function Control Register 0 (SDR0_PFC0)

0	G0E	GPIO 0 Enable 0 Enable interrupt IRQ00 as an interrupt 1 Enable interrupt IRQ00 as GPIO0
1	G1E	GPIO 1 Enable 0 Enable interrupt IRQ01 as an interrupt 1 Enable interrupt IRQ01 as GPIO1
2	G2E	GPIO 2 Enable 0 Enable interrupt IRQ02 as an interrupt 1 Enable interrupt IRQ02 as GPIO2
3	G3E	GPIO 3 Enable 0 Enable interrupt IRQ03 as an interrupt 1 Enable interrupt IRQ03 as GPIO3
4	G4E	GPIO 4 Enable 0 Enable interrupt IRQ04 as an interrupt 1 Enable interrupt IRQ04 as GPIO4
5	G5E	GPIO 5 Enable 0 Enable interrupt IRQ05 as an interrupt 1 Enable interrupt IRQ05 as GPIO5
6	G6E	GPIO 6 Enable 0 Enable interrupt IRQ06 as an interrupt 1 Enable interrupt IRQ06 as GPIO6
7	G7E	GPIO 7 Enable 0 Enable interrupt IRQ07 as an interrupt 1 Enable interrupt IRQ07 as GPIO7
8	G8E	GPIO 8 Enable 0 Enable interrupt IRQ08 as an interrupt 1 Enable interrupt IRQ08 as GPIO8
9	G9E	GPIO 9 Enable 0 Enable interrupt IRQ09 as an interrupt 1 Enable interrupt IRQ09 as GPIO9

10	G10E	GPIO 10 Enable 0 Enable interrupt IRQ010 as an interrupt 1 Enable interrupt IRQ010 as GPIO10	
11	G11E	GPIO 11 Enable 0 Enable GPIO11 as GMCTxClk or TBIRxClk1 1 Enable GPIO11 as GPIO11	
12	G12E	GPIO 12 Enable 0 Enable UART1RX as UART1RX 1 Enable UART1RX as GPIO 12	
13	G13E	GPIO 13 Enable 0 Enable UART1TX as UART1TX 1 Enable UART1TX as GPIO 13	
14	G14E	GPIO 14 Enable 0 Enable UART1DSR_CTS as UART1DSR_CTS 1 Enable UART1DSR_CTS as GPIO 14	
15	G15E	GPIO 15 Enable 0 Enable UART1RTS_DTR as UART1RTS_DTR 1 Enable UART1RTS_DTR as GPIO 15	
16	G16E	GPIO 16 Enable 0 Enable IIC1SCL as IIC1SCL 1 Enable IIC1SCL as GPIO 16	This is an open drain driver and therefore needs to be pulled up.
17	G17E	GPIO 17 Enable 0 Enable IIC1SDA as IIC1SDA 1 Enable IIC1SDA as GPIO 17	This is an open drain driver and therefore needs to be pulled up.
18	G18E	GPIO 18 Enable 0 Select TrcBS0 as external interrupt 13 1 Select TrcBS0 as GPIO 18 if SDR0_PFC0[TRE]=0. Select TrcBS0 as TrcBS0 (CPU trace) if SDR0_PFC0[TRE]=1	
18:22		GPIO Trace Enable 0 Enable GPIO's 1 Enable CPU trace (RISCTrace support)	
19	G19E	GPIO 19 Enable 0 Select TrcBS1 as external interrupt 14 1 Select TrcBS1 as GPIO 19 if SDR0_PFC0[TRE]=0. Select TrcBS1 as TrcBS1 (CPU trace) if SDR0_PFC0[TRE]=1	
20	G20E	GPIO 20 Enable 0 Select TrcBS2 as external interrupt 15 1 Select TrcBS2 as GPIO 20 if SDR0_PFC0[TRE]=0. Select TrcBS2 as TrcBS2 (CPU trace) if SDR0_PFC0[TRE]=1	
21	G21E	GPIO 21 Enable 0 Select TrcES0 as external interrupt 16 1 Select TrcES0 as GPIO 21 if SDR0_PFC0[TRE]=0. Select TrcES0 as TrcES0 (CPU trace) if SDR0_PFC0[TRE]=1	
22	G22E	GPIO 22 Enable 0 Select TrcES1 as external interrupt 17 1 Select TrcES1 as GPIO 22 if SDR0_PFC0[TRE]=0. Select TrcES1 as TrcES1 (CPU trace) if SDR0_PFC0[TRE]=1	

Preliminary User's Manual

23	TRE	GPIO Trace Enable 0 Enable GPIO's 18-26 1 Enable CPU trace (RISCTrace support)	If SDR0_PFC0[TRE]=0 Set SDR0_PFC0[G18E]=1 to enable GPIO 18 Set SDR0_PFC0[G19E]=1 to enable GPIO 19 Set SDR0_PFC0[G20E]=1 to enable GPIO 20 Set SDR0_PFC0[G21E]=1 to enable GPIO 21 Set SDR0_PFC0[G22E]=1 to enable GPIO 22 If SDR0_PFC0[TRE]=1 Set SDR0_PFC0[G18E]=1 Set SDR0_PFC0[G19E]=1 Set SDR0_PFC0[G20E]=1 Set SDR0_PFC0[G21E]=1 Set SDR0_PFC0[G22E]=1
24:31		Reserved	

28.4 GPIO Registers

All GPIO registers are 32-bit memory-mapped registers and are accessed via load/store instructions at the address of the register. The GPIO0_IR register is read-only; all other registers are both read and write accessible. The registers are only active for a particular bit when that bit is selected as a GPIO in the SDR0_PFC0 register.

Table 28-1 contains a summary of all GPIO registers.

Table 28-1. GPIO Register Summary

Register	Description	Address	Access	Page
GPIO0_OR	GPIO Output Register	0x1 40000700	R/W	956
GPIO0_TCR	GPIO Three-State Control Register	0x1 40000704	R/W	956
GPIO0_ODR	GPIO Open Drain Register	0x1 40000718	R/W	957
GPIO0_IR	GPIO Input Register	0x1 4000071C	R	958

28.4.1 GPIO Register Reset Values

When a system reset occurs, each register in the GPIO macro, except GPIO0_IR, is reset to 0. All outputs are in the high-impedance state. GPIO0_IR is not reset because it is always clocked and always follows the GPIO_In state.

Note: The SDR0_PFC0 register needs to be initialized since it is not initialized by reset. During system or chip reset this register defaults to 0x0000 2000. However its value is not affected by a core reset.

28.4.2 GPIO Output Register (GPIO0_OR)

The state of each bit in the GPIO0_OR Register may be reflected in the corresponding GPIO controller macro output signal GPIO_Out. *Figure 28-3* describes GPIO0_OR register bits.



Figure 28-3. GPIO Output Register (GPIO0_OR)

0:31	GPIO0_OR register bits
------	------------------------

28.4.3 GPIO Three-State Control Register (GPIO0_TCR)

Each bit in the GPIO0_TCR Register controls the corresponding GPIO_TS_Control macro output signal. When 0, GPIO_TS_Control forces the module I/O drivers into the high-impedance state. There is a bit-to-bit correspondence between the bits in the GPIO0_TCR register and the GPIO and alternate signals. Setting a bit to 1 in GPIO0_TCR enables the associated output driver. Clearing the bit to 0 forces the corresponding output into high impedance state.

The state of the GPIO_Out signal driving Pin A (Driver Data Input) of the module I/O three-state driver is irrelevant when the driver is in the high impedance state. High impedance take precedence over data output signals. *Figure 28-4* describes GPIO0_TCR register bits.

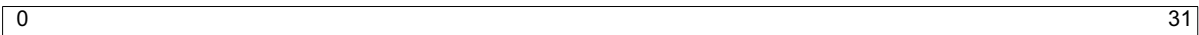


Figure 28-4. GPIO Three-State Register (GPIO0_TCR)

0:31	GPIO0_TCR register bits
------	-------------------------

28.4.4 GPIO Open Drain Register (GPIO0_ODR)

The GPIO Open Drain Register configures the module I/O three-state driver to emulate open drain drivers on a bit-by-bit basis. This is done by controlling the GPIO_Out and GPIO_TS_Control signals, via the GPIO0_OR and GPIO0_TCR registers, respectively. *Figure 28-5* describes GPIO0_ODR register bits.

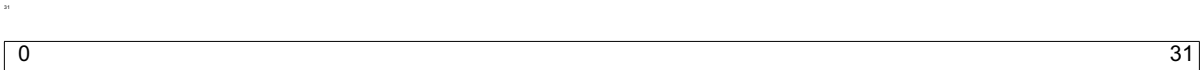


Figure 28-5. GPIO Open Drain Register (GPIO0_ODR)

0:31	GPIO0_ODR register bits
------	-------------------------

There is a bit-to-bit correspondence between the bits in the GPIO0_ODR register and the GPIO and alternate signals. When programmed to 1, each bit in the GPIO0_ODR register forces the corresponding GPIO_Out signal to 0 and the corresponding GPIO_TS_Control signal to the inverted GPIO_Out signal. In this case, the value of the corresponding bit in the GPIO0_TCR register is ignored. The open drain logic is shown in the table below.

When emulating an open drain driver, the module I/O driver never drives a 1 level. It either drives a 0 level or it is in the high impedance state emulating an open drain 1 level.

Table 28-2. GPIO0_ODR Control Logic

GPIO0_ODR Bit	GPIO0_OR Bit	GPIO_Out Macro Output	GPIO0_TCR Bit	GPIO_TS_Control Macro Output	Module I/O Three-State Driver
0	x	x	0	0	Forced to high-impedance state
0	0	0	1	1	Driving 0
0	1	1	1	1	Driving 1
1	0	0	x	1	Driving 0
1	1	0	x	0	Forced to high-impedance state

28.4.5 GPIO Input Register (GPIO0_IR)

The state of each bit in the GPIO0_IR Register reflects the corresponding GPIO Controller macro input signal GPIO_In. All GPIO_In signals are synchronized to OPBCLK before being stored in the GPIO0_IR Register. The GPIO0_IR Register is read-only and does not change during a read access.

All bi-directional three-state drivers must be in the high-impedance state in order to receive valid data as input from off chip. *Figure 28-6* describes GPIO0_IR register bits.

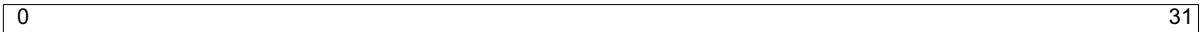


Figure 28-6. GPIO Input Register (GPIO0_IR)

0:31	GPIO register bits
------	--------------------

Preliminary User's Manual

29. External Bus Master Interface

The On-Chip Peripheral Bus (OPB) attached External Bus Master Interface (EBMI) controller transfers data between the OPB and the External Peripheral Bus (EXPB) under the direction of an external master device. The EBMI is a slave on the EXPB and a master on the OPB. This controller is necessary for any implementation of external master devices that must access system resources (SDRAM memory, on-chip SDRAM, PCI address space, etc.) on the OPB or Processor Local Bus (PLB).

In PPC440GX the EBMI must share the EXPB with an on-chip external bus controller (EBC) and must arbitrate for access to the external bus. The EBMI has a device control register (DCR) interface to allow a processor to read and write internal EBMI registers specific to the EBMI. An external master may also read and write internal EBMI registers using the EXPB special cycle.

29.1 Features

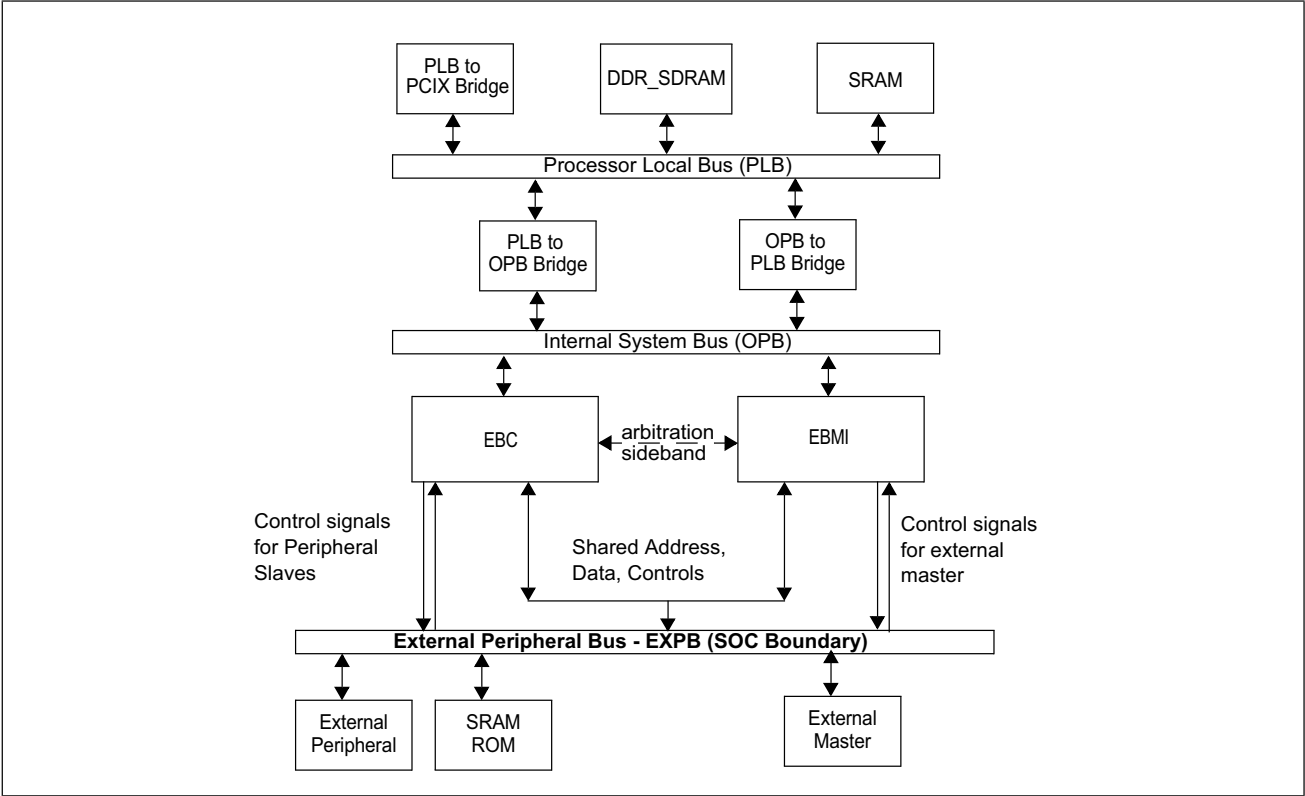
The EBMI controller has the following features:

- EXPB slave device; OPB master device
- 32-bit data/ 36-bit address OPB master interface
 - Single word (32-bit) halfword, byte, or burst reads
 - Single word, halfword, byte, and burst writes
 - 32/16/8-bit OPB slave support
- Programmable 20 to 32-bit EXPB address size
- 32-bit EXPB data interface with support for 8, 16, or 32-bit external master
- Data packing on burst writes, up to 8 words
- Read prefetching support: no prefetching, 1 word, 8 word, 16 word
- Dual 32-byte buffers
- Support for OPB at 1, 2, 3, or 4 times the frequency of the external bus
- Directly supports one external master
- Class 2 sleep support
- Programmable by external master

29.2 Physical Implementation

Figure 29-1 provides an example of EBMI as implemented in a system-on-a-chip environment.

Figure 29-1. EXPB Implementation in a SOC



Preliminary User's Manual

29.2.1 Signals

Figure 29-2 illustrates the signal I/O for the EBMI core.

Figure 29-2. EBMI Core I/O Diagram

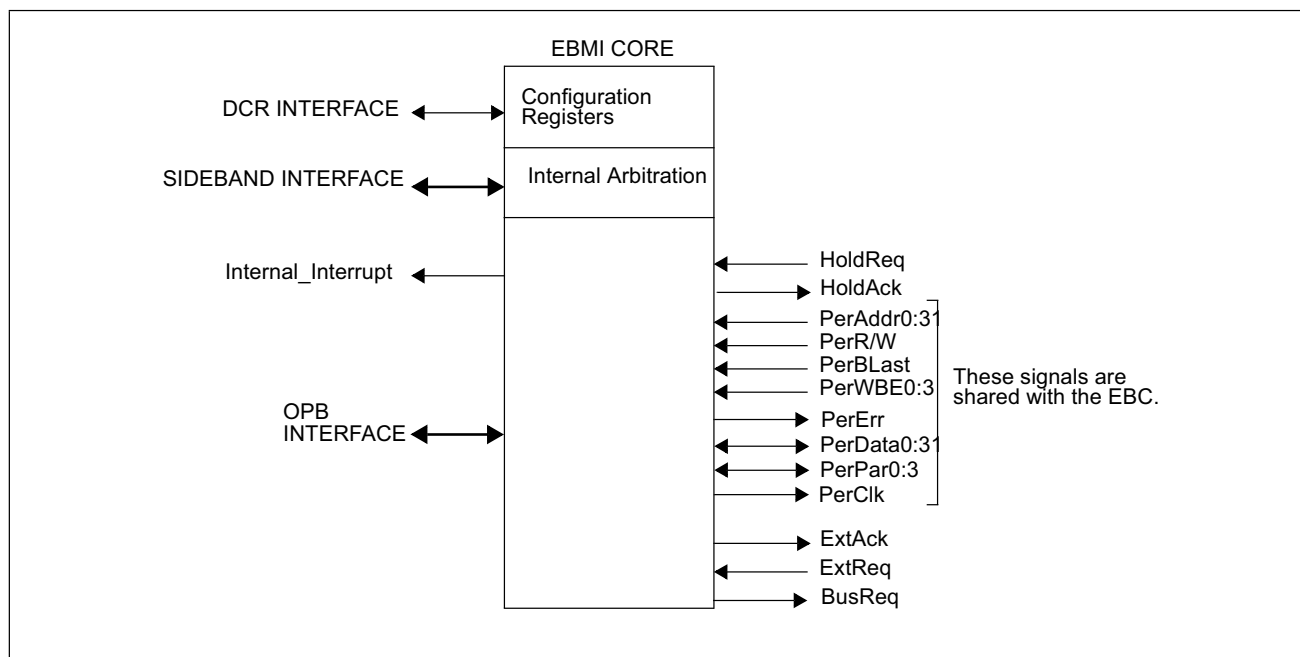


Table 29-1 provides a summary of all EXPB signals at the SOC I/Os used by the external master, followed by a brief description and page reference for detailed functional description.

Note: All EBMI I/Os are synchronous to the rising edge of PerClk.

Table 29-1. Summary of EXPB Signals for External Master

Signal Name	External Master State when HoldAck=0	Active State	I/O	Description	Page
PerAddr0:31	high-Z	—	O	External master address bus. PerAddr0 is the most significant bit.	962
PerData0:31	high-Z	—	I/O	External master data bus. PerData0 is the most significant bit.	962
PerPar0:3	high-Z	—	I/O	External master parity bus. The EBMI implements odd parity.	962
PerR/W	high-Z	—	O	External master read not write	962
PerBLast	high-Z	Low	O	External bus burst Last	962
PerWBE0:3	high-Z	Low	O	External master byte enable input	963
HoldReq	operable	High	O	External master hold request	964
HoldAck	operable	High	I	External master hold acknowledge (grant)	964
ExtAck	operable	Low	I	External bus cycle transfer acknowledge	964
ExtReq	operable	Low	O	External bus cycle request valid	964
PerErr	high-Z	High	I	External bus read data error and parity error indication	964
BusReq	operable	High	I	Bus preempt request from the EBCO	964
PerClk	EXT	—	I	Output from PPC440GP	965

29.2.1.1 PerAddr0:31 (External Master Address Bus)

The EXPB address is forwarded to the internal system bus. A master may choose to implement any number of address bits between 20 and 32. The upper address bits are appended with the appropriate value as programmed in EBM0_UAR to match the 36-bit bus width of the internal system bus.

During Special Cycles, the address bus is used to select internal EBMI facilities and is not forwarded onto the internal system bus.

The least significant bit of the address is bit 31. For a 32-bit address external master, the most significant bit of the address bus is bit 0. For a 28-bit address external master, the most significant bit of the address bus is bit 4. For a 20-bit address external master the most significant bit of the address bus is bit 12.

The address must be driven with $\overline{\text{ExtReq}}$ and held until $\overline{\text{ExtAck}}$ is observed active. The lower two bits of the address must point to the address of the first byte being transferred. During multi-beat transfers, PerAddr0:31 is ignored by the EBMI after the first $\overline{\text{ExtAck}}$ is driven active.

The external master must not allow its address to exceed its upper address range during multi-beat operations. For example, if a 32-bit external master with 20 bits of addressing begins a multi-beat operation at 0xFFFF4, it must terminate the burst after 3 beats.

29.2.1.2 PerData0:31 (External Master Data Bus)

The EXPB data bus is used to transfer data between the internal system bus or internal EBMI facilities and the external master. Bit 0 is the most significant bit, and bit 31 is the least significant bit. When subdivided into bytes, Bits 0:7 represent the most significant byte on the bus.

A master may interface to the EXPB as a byte, halfword, or word width device. A byte master uses PerData0:7 and PerPar0. A halfword master uses PerData0:15 and PerPar0:1. A word master uses PerData0:31 and PerPar0:3. The EBMI must be programmed via EBM0_CTL[EXSZ] to indicate the size of the external master. When interfaced to byte and half word external masters, the EBMI ignores the unused byte lanes.

The external master must drive write data coincident with $\overline{\text{ExtReq}}$ when it is asserted. Write data must remain driven until the EBMI asserts $\overline{\text{ExtAck}}$. For multi-beat transfers, the data must switch to the next beat the cycle after $\overline{\text{ExtAck}}$ has been asserted.

29.2.1.3 PerPar0:3 (External Master Data Bus Parity)

All EXPB data transfers (including Special Cycles) must include odd-byte parity if parity checking is enabled. Parity checking is enabled by programming EBM0_CTL[PDIS]=0.

29.2.1.4 PerR/W (External Master Read Not Write)

When $\overline{\text{ExtReq}}$ is asserted, the external master must drive this signal after the master has been granted the bus (as indicated by HoldAck=1) to indicate whether the current operation is a read (high) or a write (low).

29.2.1.5 PerBLast (External Bus Burst Last)

This signal must be asserted by the external master to indicate the last beat of a multi-beat transfer. It is asserted with the last beat of write data or when requesting the last beat of read data. If the transfer is a single beat operation, PerBLast must be asserted with the assertion of $\overline{\text{ExtReq}}$.

Preliminary User's Manual

29.2.1.6 PerWBE0:3 (External Master Byte Enable Input)

These signals must be asserted with $\overline{\text{ExtReq}}$ to indicate which bytes to write for a single-beat write operation. A 1-byte master uses only PerWBE0. A 2-byte master uses only PerWBE0:1. A 4-byte master uses all 4 bits.

At least one byte enable bit used by a master must be asserted for single-beat read operations in order to distinguish it from a Special Cycle read.

For multi beat operations ($\overline{\text{PerBLast}}$ asserted when $\overline{\text{ExtReq}}$ is asserted), the $\overline{\text{PerWBE0:3}}$ bits used by the external master must be zero.

To indicate a Special Cycle, all byte enable bits used by the master are de-asserted, and $\overline{\text{PerBLast}}$ is asserted when $\overline{\text{ExtReq}}$ is asserted.

Table 29-2 below shows the allowable $\overline{\text{PerWBE0:3}}$, bus address, and master size combinations for write operations.

Table 29-2. Allowable PerWBE/Address/Master Size Combinations

PerAddr30:31	$\overline{\text{PerWBE0:3}}$	Number of Bytes Xfered	Size of Master (bytes)	Location of Data on EXPB
00	(0:3) = 0000	4	4	PerData0:31
01	(0:3) = 1000	3	4	PerData8:31
00	(0:3) = 0001	3	4	PerData0:23
10	(0:3) = 1100	2	4	PerData16:31
01	(0:3) = 1001	2	4	PerData8:23
00	(0:3) = 0011	2	4	PerData0:15
11	(0:3) = 1110	1	4	PerData24:31
10	(0:3) = 1101	1	4	PerData16:23
01	(0:3) = 1011	1	4	PerData8:15
00	(0:3) = 0111	1	4	PerData0:7
aa ¹	(0:3) = 1111	4	4	PerData0:31
00	(0:1) = 00	2	2	PerData0:15
00	(0:1) = 01	1	2	PerData0:7
01	(0:1) = 10	1	2	PerData8:15
10	(0:1) = 00	2	2	PerData0:15
10	(0:1) = 01	1	2	PerData0:7
11	(0:1) = 10	1	2	PerData0:15
aa ¹	(0:1) = 11	2	2	PerData0:15
00	(0) = 0	1	1	PerData0:7
01	(0) = 0	1	1	PerData0:7
10	(0) = 0	1	1	PerData0:7
11	(0) = 0	1	1	PerData0:7
aa ¹	(0) = 1	1	1	PerData0:7

Note: 1. Lower 2-bits of the EBMI offset address for the Special Cycle address.

29.2.1.7 HoldReq (External Master Hold Request)

This signal is asserted by the external master to indicate that it requests ownership of the EXPB. This signal must remain active while a transfer is in progress and must not be deasserted when ExtReq is active. After completing at least one transfer, the external master should get off the bus as soon as possible upon seeing BusReq go active. This is done to avoid affecting system performance if the CPU is doing reads/writes to the EBC.

This signal must remain asserted for at least one cycle after the deassertion of ExtReq at the end of a burst write operation when parity checking is enabled in the EBMI_CTL register.

29.2.1.8 HoldAck (External Master Hold Acknowledge (Grant))

This signal is asserted by the EBMI to indicate that the external master has been granted ownership of the EXPB. This signal remains asserted until the cycle after the external master has de-asserted HoldReq. This signal is always driven by the EBMI. EBM0_CTL[EXSC] must be initialized with a DCR write to indicate the master size before EBMI asserts HoldAck.

29.2.1.9 ExtReq (External Bus Cycle Request Valid)

The external master asserts ExtReq to indicate the start of a new transfer. PerAddr0:31, PerR/W, PerBLast, and PerWBEn bits used by the master must be valid for this transfer when ExtReq is asserted. For write operations, PerData and PerPar bits used by the external master must also be valid. ExtReq must remain asserted until the data transfer has completed as indicated by ExtAck asserted by the EBMI. The external master must deassert ExtReq when the data transfer has completed.

29.2.1.10 ExtAck (External Bus Cycle Transfer Acknowledge)

This signal is asserted by the EBMI to indicate that read data is available or that write data has been accepted. This signal is always driven by the EBMI. For each data transfer, ExtAck is active for only one PerClk cycle.

29.2.1.11 PerErr (External Bus Parity Error Indication)

PerErr is used to signal write parity errors only when parity checking is enabled by setting EBM0_CTL[PPEN]=1. For a single-beat transfer, PerErr is asserted by the EBMI the same cycle ExtAck is asserted to indicate that a parity error has been detected on the data write operation.

For a multi-beat write transfer, PerErr is asserted by the EBMI two cycles after ExtAck is asserted for the data beat that had the error.

This signal is also asserted by the EBMI when ExtAck is asserted to indicate an error occurred during the read operation. If the current operation is a multi-beat transfer, PerErr continues to be asserted with ExtAck for all subsequent elements of the burst until the transfer is terminated by the de-assertion of ExtReq.

29.2.1.12 BusReq (Bus Preempt Request from the EBC)

This signal is asserted by the EBMI to indicate that the EBC needs ownership of the bus. When this is asserted, the external master should complete the current transaction if it is doing a single-beat operation and de-assert HoldReq. If the external master is doing a burst operation, the burst should be terminated as soon as possible. This signal is always driven by the EBMI.

Failure to relinquish bus ownership when BusReq is asserted can adversely affect system performance. Deadlock occurs if the master never relinquishes the bus.

Preliminary User's Manual**29.2.1.13 PerClk (EXPB Clock)**

PerClk is an output from the PPC440GX and is not used by the EBMI core. All EXPB transactions occur synchronous to PerClk.

29.2.2 Arbitration

To gain control of the EXPB, the external master requests the bus by placing an active level on the HoldReq input. The EBMI indicates that the bus is available by activating the HoldAck signal to the external master. The external master must not deassert HoldReq until HoldAck has been asserted. The external master must keep HoldReq asserted for as long as it requires the EXPB. The external master may de-assert HoldReq the same cycle it de-asserts ExtReq.

The PLB-to-OPB bridge will not accept a read or burst write command from the CPU or any other PLB master unless it can guarantee that the operation can be forwarded and accepted by the EBC. The EBC may not accept an operation off the OPB unless it also has ownership of the EXPB; consequently, the EBMI and the PLB-to-OPB bridge logic communicates with an internal arbitration protocol to ensure that the EXPB bus is free to be used by the EBC when a read or burst write command is accepted by the PLB-to-OPB bridge. The external master participates in this protocol by relinquishing the EXPB as soon as possible after BusReq is asserted by the EBC. The EBM0_FAIR register allows software to tune this protocol to meet system needs.

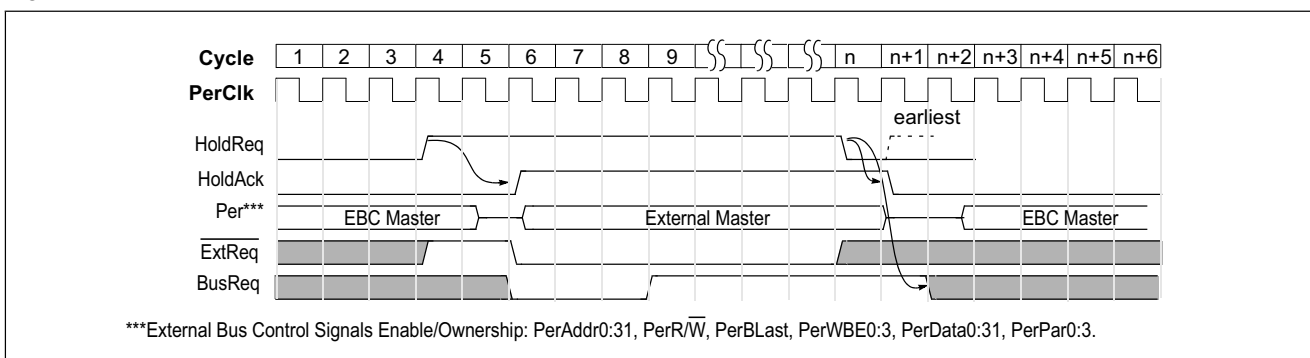
BusReq is asserted by the EBMI to indicate that the EBC requires control of the EXPB. The EBC cannot reclaim ownership of the EXPB until the external master releases it by negating HoldReq.

Note: The BusReq may be asserted regardless of bus ownership.

There is no requirement for when the external master must relinquish the bus after BusReq has been asserted; however, system performance may be compromised if other devices are not allowed fair access to the EXPB. The external master must complete the current transfer and relinquish the bus by deasserting HoldReq when BusReq is asserted.

The external master must High-Z the PerAddr0:31, PerR/W, PerBLast, PerWBE0:3, PerData0:31, and PerPar0:3 bus signals when HoldAck is deasserted (the cycle after HoldReq is deasserted). Figure 29-3 shows the EXPB bus arbitration protocol. The EBMI takes a minimum of two cycles to assert HoldAck after HoldReq is asserted; however, the assertion of HoldAck potentially takes much longer if the EBC has ownership of the EXPB.

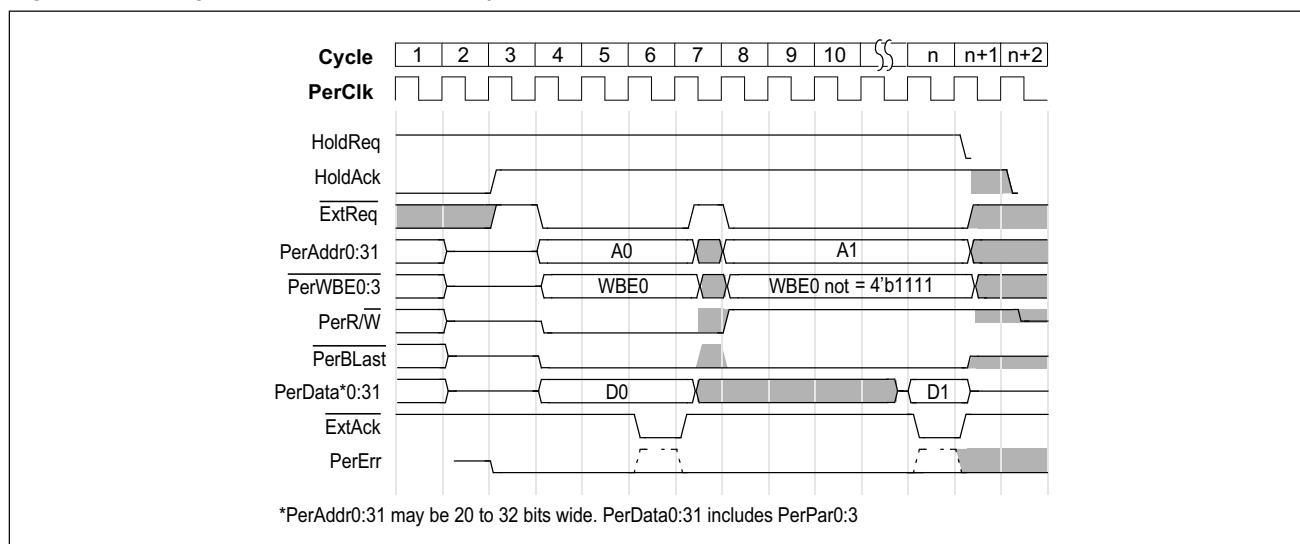
Figure 29-3. EXPB Bus Arbitration



29.2.3 Single Transfers

A single-beat write or read transfer is indicated by the external master by the assertion of $\overline{\text{PerBLast}}$ when $\overline{\text{ExtReq}}$ is asserted. The external master drives PerAddr0:31 , PerWBE0:3 , and PerR/W with $\overline{\text{ExtReq}}$. Write data and data parity (if enabled) must also be driven with $\overline{\text{ExtReq}}$ for a write operation.

Figure 29-4. Single Beat Write Followed by Read



The EBMI interface indicates that the write data has been accepted or that read data is available by asserting $\overline{\text{ExtAck}}$ for one cycle.

Single-beat read operations normally cause a 4-byte read operation to be forwarded to the OPB. If the single-beat prefetch mode is disabled in the EBMI, the EBMI reads only the bytes that are indicated in the PerWBE0:3 bits. At least one PerWBE0:3 bit must be asserted; otherwise, the operation is decoded as a Special Cycle read operation. When the read data is available, the EBMI drives the read data with $\overline{\text{ExtAck}}$ asserted. For more information, see the `EBM0_CTL` register.

When using single beat reads to poll a memory location, the external master must force the read valid bits to reset to avoid reading stale data from the buffer (see *Read Valid Bits* on page 971).

29.2.4 Burst Transfers

A multi-beat (burst) write or read transfer is indicated by the external master deasserting $\overline{\text{PerBLast}}$ at the start of the transfer. Burst transfers must always occur on an address boundary equal to the size of the bus master. The PerWBE0:3 bits used by the external master must be zero during a burst transfer. The external master must drive/accept the next beat of data onto the bus the cycle after $\overline{\text{ExtAck}}$ is asserted by the EBMI. The EBMI may assert $\overline{\text{ExtAck}}$ in the cycle following the external master driving $\overline{\text{ExtReq}}$. The external master must assert $\overline{\text{PerBLast}}$ to indicate that the last beat of data is available/wanted and that the burst transfer will terminate.

The EBMI may not terminate the burst data transfer; however, it will pace the data transfer by delaying the assertion of $\overline{\text{ExtAck}}$. On a burst write operation, the EBMI will assert $\overline{\text{ExtAck}}$ for multiple cycles until either the write data buffer has been filled or $\overline{\text{PerBLast}}$ is asserted. If the data transfer exceeds the size of the data buffers, the EBMI will deassert $\overline{\text{ExtAck}}$ when the buffer boundary is reached. On a burst read operation, the EBMI will assert $\overline{\text{ExtAck}}$ for multiple cycles until no more read data is available or $\overline{\text{PerBLast}}$ is asserted. The EBMI will re-assert $\overline{\text{ExtAck}}$ when more read data has been fetched from the OPB and is available. See *Section 29.3 EBMI Buffer Management* on page 970 for more information.

Preliminary User's Manual

When write data parity checking is enabled in the EBMI, the PerErr is asserted by the EBMI for a burst write operation two cycles after ExtAck is asserted if a parity error is detected on the burst write data transfer. The external master must keep HoldReq asserted for one cycle after deasserting EXT_Req_N after a burst write operation if parity checking is enabled. The external master should monitor PerErr for two cycles after the end of a burst write transfer to check for a parity error indication if parity checking is enabled in the EBM0_CTL register .

Figure 29-5. Burst Write

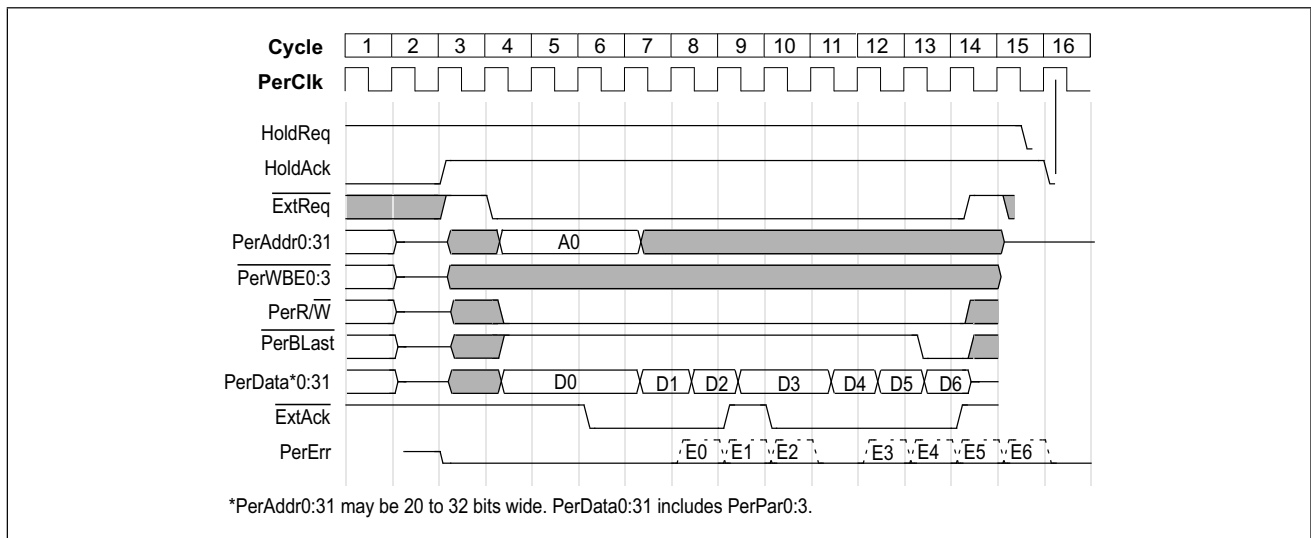
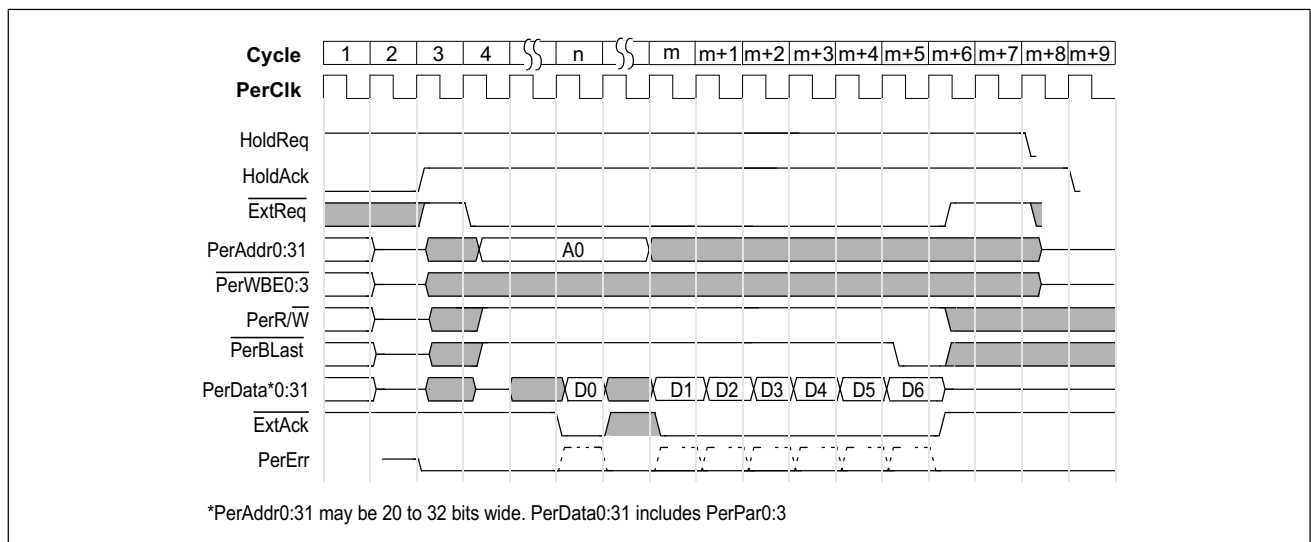


Figure 29-6. Burst Read



29.2.5 Special Cycle

A Special Cycle is executed by the external master to access internal EBMI configuration and status registers. An external master performs a Special Cycle transfer by asserting ExtReq while at the same time driving all implemented PerWBE0:3 bits to high when is asserted and driving PerBLast low. The lower 10-bits of PerAddr0:31 are used to select a specific register.

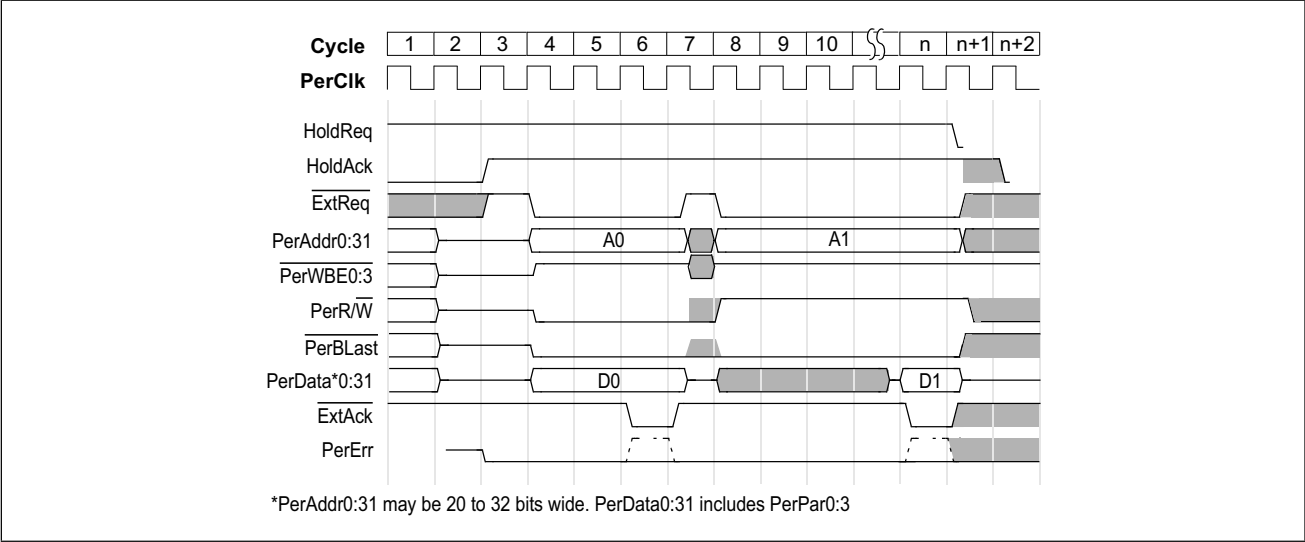
A Special Cycle is always a 32-bit data operation that is used to read or write the facilities implemented as DCR accessible registers in the EBMI. 16-bit masters must execute two single-beat Special Cycles in a row using the same address and PerWBE0:3 driven to 2'b11. 8-bit masters must execute four single-beat Special Cycles in a row using the same address and PerWBE0 driven to 1'b1. There must be at least one idle cycle between the individual special cycle transfers. EBM0_BESR[4] is set if a 16-bit or 8-bit external master inserts an OPB transfer request between the required 2 or 4 special cycle operations. The unexpected OPB transfer request is forwarded to the OPB as normal; however, future special cycle operations may complete incorrectly.

Only the lower 10-bits of the address are used for the Special Cycle to access the DCR addressable facilities within the EBMI. The upper address bits are ignored.

If data parity checking is enabled in the EBMI and a parity error is detected during a Special Cycle write operation, the target register is not written and the EBMI asserts PerErr coincident with ExtAck. The write operation will not execute to the DCR register. 8- and 16-bit external masters must complete all 32 bits of the Special Cycle write even if an error has been indicated by the EBMI when the error is not on the last data item of the 32-bit special cycle write.

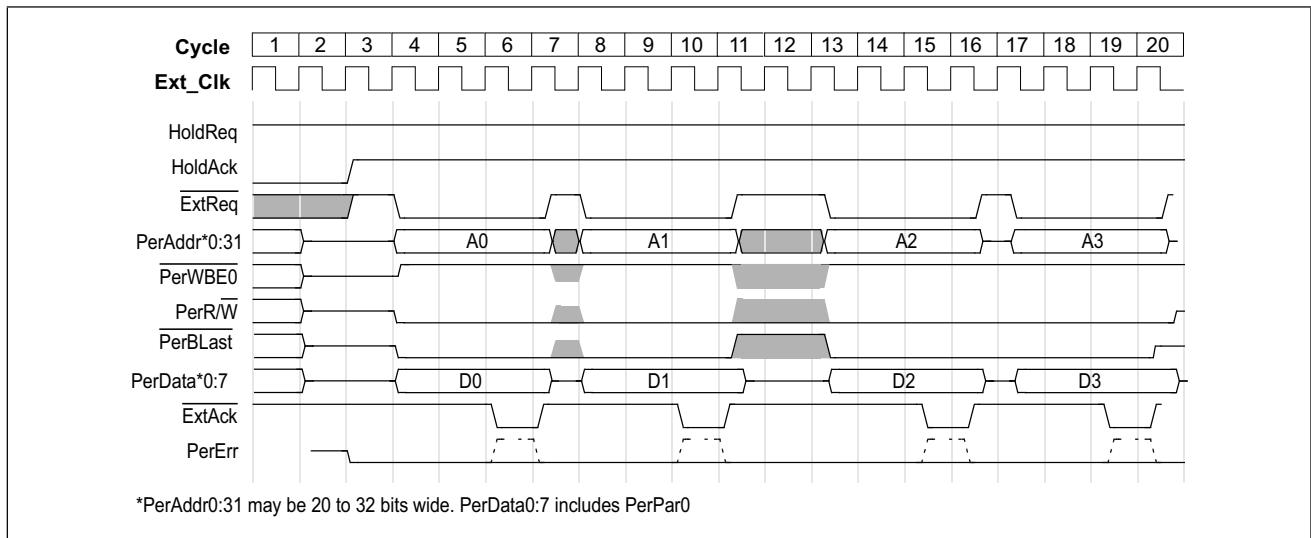
Figure 29-7 below shows an external 32-bit master performing a special cycle write followed by a Special Cycle read. Figure 29-8 below shows special cycle operations being performed by an 8-bit external master.

Figure 29-7. Special Cycle Write followed by Read



Preliminary User's Manual

Figure 29-8. Special Cycle Write By 8-Bit Master

**29.2.6 Error Reporting**

The PerErr pin is driven by the EBMI to indicate a detected parity error condition on a data write from the external master. All external masters are required to generate parity for write data when parity checking is enabled in the EBMI. The EBMI is not informed if a parity error is detected on the read data; however, the system architecture may require that the external master indicate that a parity error was detected/driven by the EBMI. If the EBMI detects a parity error during a write operation and EBM0_BEMR[EPE]=0, an interrupt is generated.

The PerErr pin is driven by the EBMI to also indicate that an error occurred during the data read transaction. Possible sources for a read error would be an uncorrectable ECC error or a read from an address that does not exist. ExtAck is asserted with PerErr to allow the data transfer to complete. The external master should not use the read data if PerErr is asserted. If burst prefetching is enabled (EBM0_CTL[BPF]), PerErr may be asserted for good data if other unused data later in the buffer received an error.

An external interrupt pin is one method that an external master can use to indicate to the CPU that it detected a read parity error.

29.2.7 Interrupt Signal

The EBMI is connect to the UIC1_Irq[6]. An active “high” interrupt is asserted to the system logic when an unmasked error has been detected and is driven directly out of a latch. The signal remains asserted until the error condition is cleared by writing zeros to EBM0_BESR or by masking the error condition in EBM0_BEMR.

29.3 EBMI Buffer Management

The EBMI has two 32-byte data buffers. External master read operations read data from one buffer while the other is being filled from the OPB. External master write operations write data into one buffer while the other is written to its destination.

29.3.1 Read Operations

For read operations, the EBMI marks one buffer as the current read buffer. Read operations use data from the current buffer and do not issue another read if the current read buffer address matches the word address on the EXPB and the requested data bytes or the entire buffer is marked valid. When the current read buffer address does not match the EXPB word address or if the data in the buffer is not marked valid, the read operation forces a fetch from the address indicated by the external master. If the fetch is for a single-beat read, the EXPB word address becomes the current read buffer address and the appropriate data bytes are marked valid when the fetch is complete. If the fetch is for a burst read, the EXPB word address becomes the starting current read buffer address and the entire buffer is marked valid when the fetch is complete. When the entire buffer is marked valid, an EXPB word address matches the current read buffer address if it falls within the 32 or 16 byte range of the buffer.

29.3.1.1 OPB Read Prefetch

Burst read requests may cause a read prefetch at the start of the request to fill a read buffer. The size of the prefetch is programmed by software using EBM0_CTL[BPF]. The prefetch is programmed to 32 or 16 bytes and is used to optimize internal bus utilization. The EBMI fetches the next buffer of data when the EBMI has asserted ExtAck for the last beat of data in the current read buffer. The buffer burst prefetch may also be disabled with EBM0_CTL[BPF] so only one beat of data the size of the transfer on the EXPB is fetched at a time.

If the burst prefetch ahead mode bit BPFA is enabled in EBM0_CTL and prefetching is enabled, the EBMI continues to prefetch ahead for burst read operations as long as PerBLast is deasserted. The EBMI prefetches ahead into the other buffer for the next 32 or 16 bytes of data as soon as possible after initial prefetch has been initiated. When the data from the initial prefetch has been returned to the external master, the EBMI continues to assert ExtAck and return data from the next buffer if the data is ready and prefetched ahead into the other buffer that was emptied.

A single word (4-bytes) is read at the word address indicated by the external master to service single-beat, non-burst requests. This can be disabled in EBM0_CTL with mode bit SPFD to allow only exact reads using the EXPB write byte enables.

The best EXPB bandwidth can be achieved for burst read operations if BPFA is enabled with RDER (Ready Early Indication) in EBM0_CTL. If mode bit BPFA is disabled, the EBMI will not prefetch the next data buffer until all data from the initial prefetch has been returned to the external master.

Preliminary User's Manual

29.3.2 Read Valid Bits

The EBMI maintains the read valid bits for each buffer. The read valid bits indicate which bytes at the current read buffer address are valid and if the entire buffer is valid. The prefetched read data in a buffer is used only if the appropriate read valid bits are set. The read valid bits are set when the fetch read data is available in the buffer. The read valid bits are reset under the following conditions:

- Any single-beat or burst write, or Special Cycle write operation resets all read valid bits in both buffers.
- The beginning of a new bus tenure (HoldAck just asserted) resets all read valid bits in both buffers unless EBM0_CTL[DRST]=1.
- Any single-beat read operation that requires a new buffer fetch resets all read valid bits in both buffers.
- A read buffer that received an internal failure indication from the OPB has its read valid bits cleared. When the current EXPB transaction is ended. This will force a new OPB read operation if the read is repeated.

29.3.3 Read Early Mode

To reduce read latency for burst read operations, the EBMI may assert $\overline{\text{ExtAck}}$ to the external master as soon as the first beat of read data is available. To enable this function, software must set RDER=1 in EBM0_CTL to enable the read early mode. To avoid a buffer underrun condition, the EBMI core de-asserts $\overline{\text{ExtAck}}$ and 'paces' the burst read operation if the burst read is suspended or delayed.

29.3.4 Write Operations

Single beat write operations ping/pong between the two EBMI buffers. Data is not packed into the buffer and is delivered to its destination as soon as it is acknowledged on the EXPB.

Burst write operations are packed into the buffer until either 32 bytes of data have been received or the burst transfer has completed on the EXPB.

29.4 EBMI Registers

The EBMI registers listed in *Table 29-4* are accessed indirectly through the EBM0_CFGADDR and EBM0_CFGDATA registers (listed on *Table 29-3*) using the mtdcr and mfdcr instructions. A read-modify-write sequence should be used to write registers with reserved fields. During this sequence, software must not alter the contents of any reserved fields.

Table 29-3. EBMI Access Registers

Mnemonic	Name	DCR Number	Access	Page
EBM0_CFGADDR	EBMI Address Register	0x14	R/W	972
EBM0_CFGDATA	EBMI Data Register	0x15	R/W	973

Table 29-4. External Bus Configuration and Status Registers

Mnemonic	Name	Offset	Access	Page
EBM0_CTL	EBMI Control Register	0x00	R/W	973
EBM0_LCNT	EBMI OPB Latency Count Register	0x01	R/W	975
EBM0_BEAR	EBMI Bus Error Address Register	0x02	R/W	975
EBM0_BESR	EBMI Bus Error Status Register	0x03	R/W	976
EBM0_BEMR	EBMI Bus Error Mask Register	0x04	R/W	977
EBM0_UAR	EBMI OPB Upper Address Register	0x05	R/W	977
EBM0_UAM	EBMI OPB Upper Address Mask	0x06	R/W	978
EBM0_SLPMD	EBMI Sleep Mode Register	0x07	R/W	979
EBM0_FAIR	EBMI Fairness Control Register	0x08	R/W	981
EBM0_CID	EBMI Core ID Register	0x11	R	980

An external master may also use a special cycle read or write operation on the EXPB to access the EBMI registers. PerAddr27:31 is used to select the register. If a special cycle operation occurs during the same cycle as DCR operation from the CPU, the DCR operation has priority and executes first.

Note: Indirect addressing is not used for special cycle operations, and all operations are targeted to the EBMI registers only.

29.4.1 EBMI Configuration Address Register (EBM0_CFGADDR)

EBM0_CFGADDR described in Figure 29-9 is written by software to indirectly address the EBMI registers.

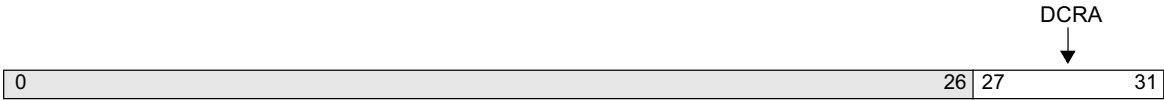


Figure 29-9. EBMI Configuration Address Register (EBM0_CFGADDR)

0:26		
27:31	DCRA	DCR Address Offset

Preliminary User's Manual**29.4.2 EBM0 Configuration Data Register (EBM0_CFGDATA)**

EBM0_CFGDATA described in *Figure 29-10* is a data port written by software after it writes data to read or write the other EBM0 registers.

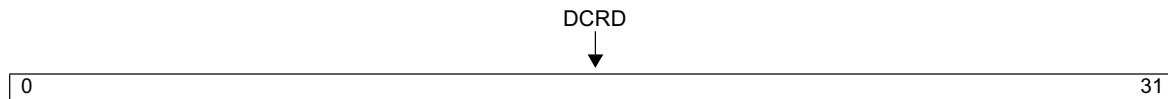


Figure 29-10. EBM0 Configuration Data Register (EBM0_CFGDATA)

0:31	DCRD	DCR Data Port
------	------	---------------

29.4.3 EBM0 Control Register (EBM0_CTL)

Figure 29-11 describes the EBM0_CTL register bits.

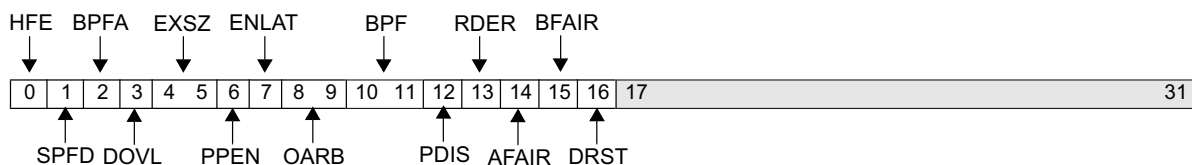


Figure 29-11. EBM0 Control Register (EBM0_CTL)

0	HFE	Hold First Error 0 EBM0_BEAR and EBM0_BESR contain information from first error detected. 1 EBM0_BEAR and EBM0_BESR contain information from last error detected.	
1	SPFD	Single Beat Word Prefetch Disable 0 4-byte read on OPB for any single-beat EXPB read request. 1 1 to 4- byte read on OPB for any single-beat EXPB read request, according to EXPB PerWBE0:3 signals.	This bit must be set if the EBM0 is to read only the bytes that are requested for a single beat read.
2	BPFA	Burst Prefetch Ahead 0 Data is read from OPB into the data buffer when the data is required. 1 Data is read from OPB into the data buffer before the data is required, while the 1st buffer is being emptied.	This bit should be set by software if the external master routinely does long or many sequential burst read operations longer than the burst prefetch size of the buffer. The EBM0 will prefetch 1 buffer ahead to maintain EXPB bus bandwidth. This bit is ignored if the Burst Prefetch mode below is set to '10' to disable the prefetch.

Preliminary User's Manual

3	DOVL	<p>Disable HoldA Arbitration Overlap</p> <p>0 HoldAck asserted in response to an active HoldReq asserted by external master immediately. There is a possible 2-cycle delay if a Special Cycle operation is in progress from the previous tenure.</p> <p>1 HoldAck is not asserted in response to an active HoldReq from the external master until all pending OPB operations and Special Cycle operations from a previous bus tenure are complete.</p>	This bit should be left as 0 and the EBM0_FAIR register used to tune arbitration.
4:5	EXSZ	<p>External Master Data Bus Size (ext_mst_size_l2)</p> <p>00 8-bit data bus, PerData0:7</p> <p>01 16-bit data bus, PerData0:15</p> <p>10 32-bit data bus, PerData0:31</p> <p>11 No external master</p>	Software must initialize this register to indicate the size of the external master data bus.
6	PPEN	<p>Enable second 32-byte buffer</p> <p>0 Second buffer disabled. Use only 1 buffer.</p> <p>1 Second buffer enabled. Ping/Pong enabled.</p>	
7	ENLAT	<p>Enable OPB Latency Counter</p> <p>0 OPB Latency counter is disabled</p> <p>1 OPB Latency counter is enabled</p>	This bit enables the EBM0_LCNT register function.
8:9	OARB	<p>OPB Arbitration Control</p> <p>Must be 0b01</p>	
10:11	BPF	<p>Burst Prefetch</p> <p>00 8-beat (32-byte) read</p> <p>01 4-beat (16-byte) read</p> <p>10 1-beat (4/2/1-byte) read (no prefetch, read byte size of master)</p> <p>11 Reserved, unused</p>	<p>For most applications, this should remain at '00.'</p> <p>If BPF=2'b10, the request will be forwarded exactly as received if SPFD=1.</p> <p>These bits control the number of bytes prefetched on a burst read from the external master.</p>
12	PDIS	<p>EXPB Parity Checking Disabled</p> <p>0 Write data parity is checked on EXPB</p> <p>1 Write data parity is not checked on EXPB</p>	Field enabled parity checking is only for external master transfers. This bit must be set by software if the external master does not support parity on PerData.
13	RDER	<p>Enable Read Early Indication</p> <p>0 Burst read data returned to external master when burst read from OPB is complete.</p> <p>1 Burst read data returned to external master beginning when first beat from OPB is complete.</p>	This mode bit is used to reduce latency for read burst operations.
14	AFAIR	Arbitration Fairness Counter Disable	This bit disables the AFCNT in the EBM0_FAIR register.
15	BFAIR	BReq Fairness Counter Disable	This bit disables the BFCNT in the EBM0_FAIR register.
16	DRST	Disable Reset Valid for New Tenure	This bit disables the clearing of the read buffer valid bits at the start of a new external master bus tenure. This bit should be set if the external master does long burst reads but often has to give up the EXPB before receiving all the data because Bus-Req is asserted.
17:31		Reserved	

29.4.4 EBMI OPB Latency Count Register (EBM0_LCNT)

The latency counter is the upper 6 bits of the initial load of a 10-bit OPB bus latency counter. The counter is a programmable timer that limits the EBMI latency on the OPB bus. The lowest four low-order bits of the latency count are hard-wired to zero, so the minimum latency count is 16.

When the latency counter is enabled in EBM0_CTL[ENLAT], the EBMI is transferring data on the OPB, and another OPB master requests ownership of the bus, the latency counter decrements with each data transfer. When the counter decrements to zero, the EBMI must relinquish control of the OPB as soon as possible. *Figure 29-12* describes the EBM0_LCNT register bits.



Figure 29-12. EBMI OPB Latency Count Register (EBM0_LCNT)

0:5	LCNT	Latency Counter This is the upper 6 bits of the initial load of a 10-bit OPB bus latency counter.
6:31		Reserved

29.4.5 EBMI Bus Error Address Register (EBM0_BEAR)

EBM0_BEAR contains either the OPB or EXPB address of the access where a data bus error occurred. EBM0_BEAR is written when a data access error occurs and its contents are locked if the Hold First Error (HFE) bit in EBM0_CTL is set. If the HFE bit is not set, then EBM0_BEAR continues to be updated with the address of any subsequent cycle in which an error occurs. If an OPB bus error and an EXPB error occur in the same cycle, this register is updated with the EXPB bus address. *Figure 29-13* describes the EBM0_BEAR register bits.

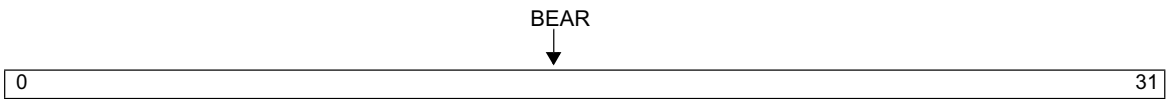


Figure 29-13. Peripheral Bus Error Address Register (EBM0_BEAR)

0:31	BEAR	Bus Error Address The contents of this register are valid when any bit is set in the EBM0_BESR.
------	------	--

29.4.6 EBM0 Bus Error Status Register (EBM0_BESR)

EBM0_BESR records the occurrence and type of errors for transactions attempted on behalf of the external master. It is possible to have multiple errors on a single transfer. If multiple errors occur on a transfer, multiple error bits may be set if the HFE bit in EBM0_CTL is not set. If the HFE bit is set, then only the first error detected is reported.

If a read operation is in progress when an error occurs, any data in the read buffer is returned to the external master with PerErr asserted when ExtAck is asserted until the transfer is terminated by the external master. Figure 29-14 describes the EBM0_BESR register bits.

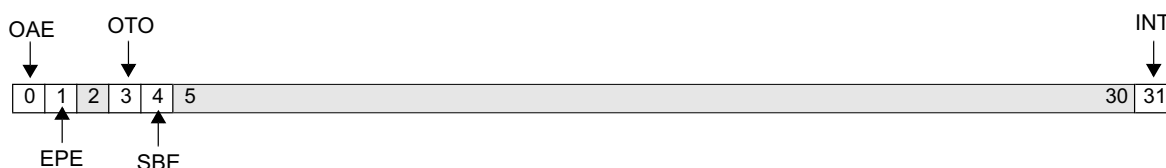


Figure 29-14. EBM0 Bus Error Status Register (EBM0_BESR)

0	OAE	OPB Bus Ack Error 0 No ack error detected 1 Ack error detected	This bit is set when the EBM0 detects an OPB error indication asserted during a data transfer on the OPB.
1	EPE	EXPB Bus Parity Error 0 No parity error detected 1 Parity error detected	This bit is set when the EBM0 detects a parity error on incoming write data from the external master. The transfer with the parity error is not sent onto the OPB. Burst write operations must fill the buffer/terminate before they are sent onto the OPB. Consequently, none of the write data in the write buffer is forwarded to the OPB if a parity error is detected. All burst write data after the parity error is also discarded.
2		Reserved	
3	OTO	OPB Bus Timeout Error 0 No timeout detected 1 Timeout indication received when EBM0 is reading or writing data on the OPB.	The current transfer on the OPB is terminated.
4	SBE	EXPB Special Cycle Error 0 No Special Cycle error detected. 1 External master has issued a non-Special Cycle operation when a Special Cycle operation was expected.	This error can only occur when using 16 or 8-bit external masters since the Special Cycle operation takes multiple transfers.
5:30			
31	INT	Interrupt Asserted 0 Interrupt line is not asserted. 1 Interrupt line is asserted to the chip internal interrupt controller.	This bit is a logical OR of all the other bits in the EBM0_BESR that are not masked in the EBM0_BEMR. This bit is not writable by software.

29.4.7 EBM0 Bus Error Mask Register (EBM0_BEMR)

EBM0_BEMR has the same bit definitions as EBM0_BESR. This register is used to mask the reporting of each error condition as an interrupt. The appropriate bit in EBM0_BESR is still set, but an interrupt is not asserted for that error if the corresponding bit is a 1 in EBM0_BEMR. Figure 29-15 describes the EBM0_BEMR register bits.



Figure 29-15. EBM0 Bus Error Mask Register (EBM0_BEMR)

0:5	EMSK	Bus Error Mask 0 Reporting of this error is not masked 1 Reporting of this error is masked.
6:31		Reserved

29.4.8 EBM0 OPB Upper Address Register (EBM0_UAR)

EBM0_UAR contains the upper bits[0:15] of the OPB address that are appended to the EXPB address to create a 36-bit OPB address. Bits [4:15] of the upper address correspond to PerAddr[0:11]. This register is used in conjunction with EBM0_BEMR. Figure 29-16 describes the EBM0_UAR register bits.

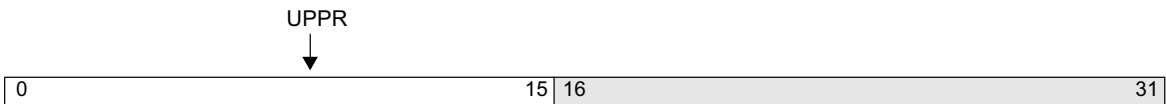


Figure 29-16. EBM0 Upper Address Register (EBM0_UAR)

0:15	UPPR	OPB Upper Address	This register must be initialized by software.
16:31		Reserved	

29.4.9 EBM0 OPB Upper Address Mask Register (EBM0_UAM)

EBM0_UAM is used to indicate the bits from EBM0_UAR that should be used to pre-pend to the EXPB address to create the 36-bit OPB address. Note that the upper 4 bits of EBM0_UAM are always ignored, since the EXPB interface supports a maximum of 32 bits (see *Table 29-5 Valid EBM0_UAM Contents*). *Figure 29-17* describes the EBM0_UAM register bits.

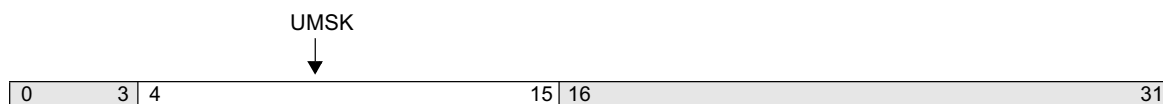


Figure 29-17. EBM0 Upper Address Mask (EBM0_UAM)

0:3		Reserved	
4:15	UMSK	OPB Upper Address Mask	This register must be initialized by software if the external master provides less than a complete 32-bit address.
16:31		Reserved	

Table 29-5. Valid EBM0_UAM Contents

UMSK(4:15)	OPB Address(0:35)	Number of EXPB Address Bits
000000000000	EBM0_UAR[0:3] PerAddr0:31	32
100000000000	EBM0_UAR[0:4] PerAddr1:31	31
110000000000	EBM0_UAR[0:5] PerAddr2:31	30
111000000000	EBM0_UAR[0:6] PerAddr3:31	29
111100000000	EBM0_UAR[0:7] PerAddr4:31	28
111110000000	EBM0_UAR[0:8] PerAddr5:31	27
111111000000	EBM0_UAR[0:9] PerAddr6:31	26
111111100000	EBM0_UAR[0:10] PerAddr7:31	25
111111110000	EBM0_UAR[0:11] PerAddr8:31	24
111111111000	EBM0_UAR[0:12] PerAddr9:31	23
111111111100	EBM0_UAR[0:13] PerAddr10:31	22
111111111110	EBM0_UAR[0:14] PerAddr11:31	21
111111111111	EBM0_UAR[0:15] PerAddr12:31	20

Preliminary User's Manual

29.4.10 EBM Sleep Mode Register (EBM0_SLPMD)

EBM0_SLPMD is used to enable the sleep mode and to program the number of OPB clock cycles to wait for the sleep request to be asserted after all of the requirements to go to sleep have been met.

EBMI sleep requirements are EBM0_SLPMD[SLEN]=1, HoldReq=0, pending MFDCR and MTDCR operations have completed, and all pending special cycle operations and OPB operations have completed. EBMI deasserts its sleep request immediately when the sleep requirements are no longer met.

Note: The minimum granularity of the idle timer is 32 clock cycles.

A value of 0 in the sleep counter (if bits 0:4 are left 0) waits 32 clocks from the time that the requirements to go to sleep have been met before asserting the sleep request signal.

A value of 1 in the sleep counter (after programming the register bits 0:4 to 1) waits 64 clocks from the time that the requirements to go to sleep have been met before making a sleep request to the clock and power management unit.

Figure 29-18 describes the EBM0_SLPMD register bits.

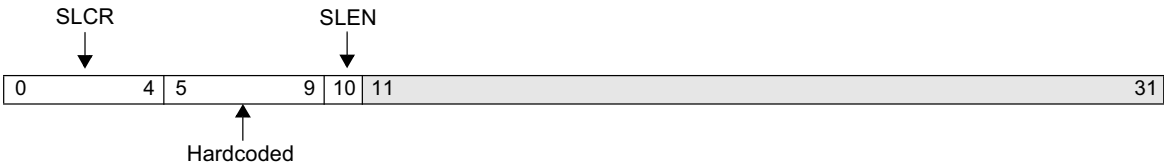


Figure 29-18. EBM Sleep Mode Register (EBM0_SLPMD)

0:4	SLCR	Programmable timer values
5:9		Hardcoded to 1s
10	SLEN	Sleep mode enable bit 0 Sleep request not asserted 1 Sleep request asserted when sleep requirements are met Software must set this bit for the EBMI to assert its sleep request and be put to sleep.
11:31		Reserved

29.4.11 EBMI Core ID Register (EBM0_CID)

EBM0_CID described in *Figure 29-19* indicates the core ID which includes the technology, core number, and library revision number assigned to this core. Writes to this register are ignored.

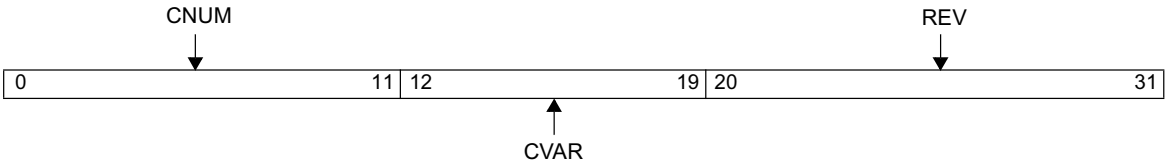


Figure 29-19. EBMI Core ID Register 0 (EBM0_CID)

0:11	CNUM	Core Number	'325'
12:19	CVAR	Core Variation	'01'
20:31	REV	Revision Number	This value is the IBM SCCS revision number tracked and modified by the core designer.

Preliminary User's Manual**29.4.12 EBMI Fairness Control Register (EBM0_FAIR)**

EBM0_FAIR described in *Figure 29-20* is used to tune the arbitration protocol between the EBMI and other internal SOC cores when trying to gain ownership of the EXPB.

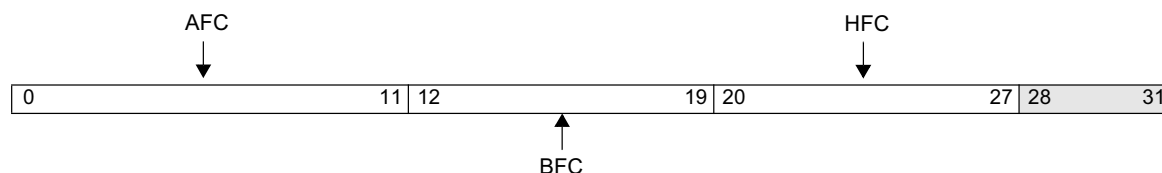


Figure 29-20. EBMI Fairness Control Register 0 (EBM0_FAIR)

0:11	AFC	Arbitration Fairness Count	These bits indicate the number of OPB clocks EBMI will wait before asserting a signal to stop PLB-to-OPB traffic when an external master request is pending. Under most conditions, these bits should be set to a high value to lower the priority of the external master. These bits are used to tune the priority of requests from the PLB to the OPB vs the priority of requests from the external master. If PLB-to-OPB traffic needs a relatively high priority, then a high value (for example x'F00') is written into this register by software.
12:19	BFC	BusReq Fairness Count	These bits indicate the number of PerClk cycles EBMI will wait before asserting BusReq to the external master. The BusReq fairness counter will begin decrementing when the EBCO has relinquished EXPB bus ownership. If BFC=0, BusReq may be asserted the same cycle as HoldAck. Under some conditions, the EBC may request ownership of the external bus and assert BusReq before the external master has asserted ExtReq. Setting the BFC bits to a non-zero value may allow the external master to make forward progress (if it is sensitive to this condition) by delaying BusReq until after the external master has asserted ExtReq.
20:27	HFC	HoldReq Fairness Count	These bits indicate the number of PerClk cycles EBMI will wait before forwarding HoldReq to the EBCO for the external master to gain bus ownership. This register is set to a high value to prioritize DMA transfers to and from the EBCO since DMA transfers are not affected by AFC.
28:31		Reserved	

Preliminary User's Manual

30. External Bus Controller

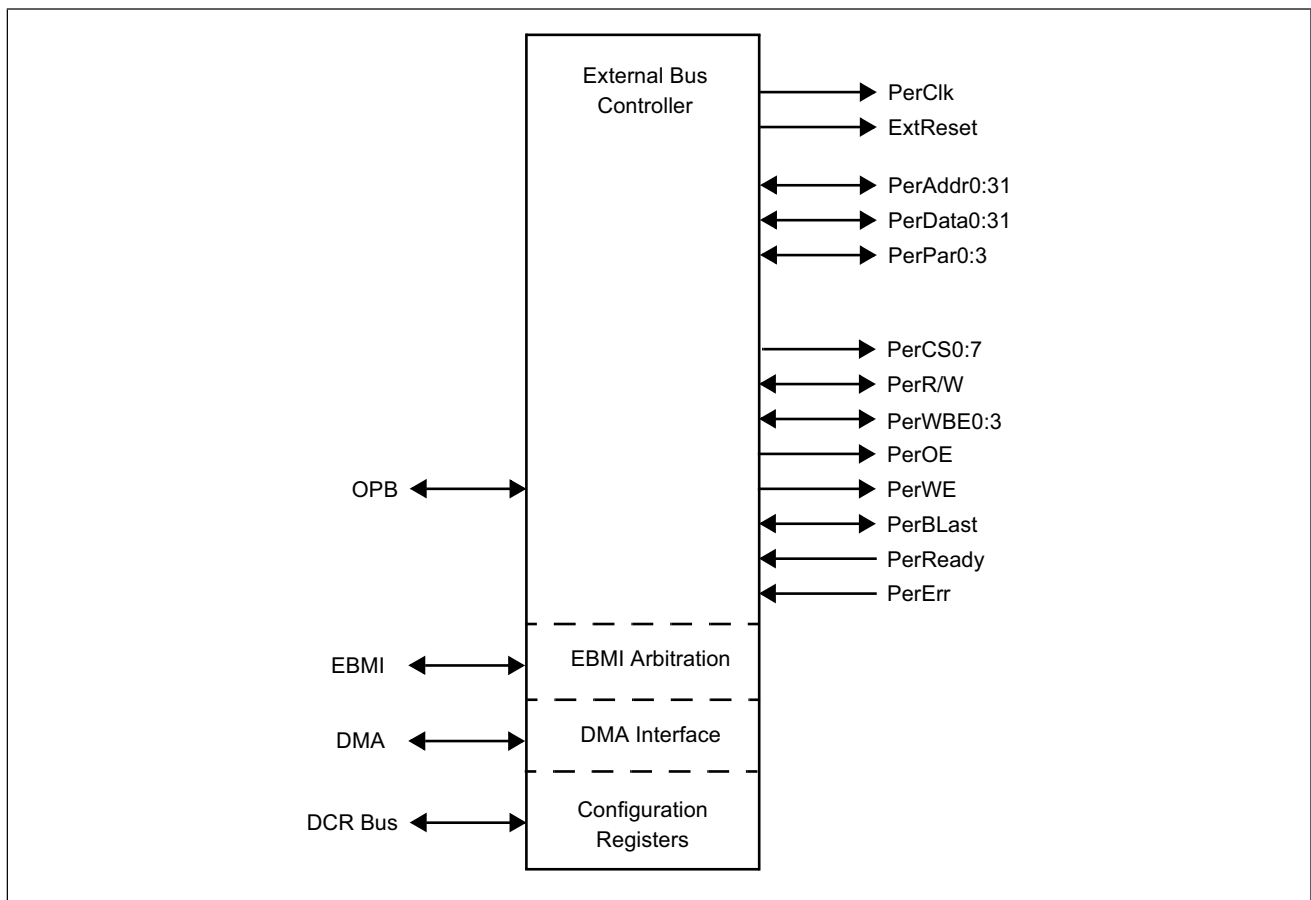
The PPC440GX External Bus Controller attached to the OPB (EBC) provides direct attachment for most SRAM/Flash type memory and peripheral devices. The interface minimizes the amount of external glue logic needed to communicate with memory and peripheral devices. This reduces the embedded system device count, circuit board area, and cost.

To eliminate off-chip address decoding, the EBC provides eight programmable chip selects that enable system designers to locate memory and peripherals within the PPC440GX memory map. Chip select, data bus, and associated control signal timings are programmable for both single and burst transfers. For peripherals with variable timing requirements, the EBC supports device-paced transfers with optional bus-timeout. System design is further simplified through dynamic bus sizing, which supports seamlessly attaching 8-, 16-, and 32-bit wide memories and peripherals. Whenever a size mismatch exists between a read or write operation and the externally attached device, the EBC automatically packs or unpacks data as appropriate.

30.1 Interface Signals

Figure 30-1 illustrates the signal I/O between the EBC and the external peripheral bus.

Figure 30-1. External Bus Controller Signals



The usage along with the state of these signals during and after a reset is as follows:

Table 30-1. EBC Signal Usage and State During/Following a Chip or System Reset

Signal	$\overline{\text{EBCOReset}} = 0$	$\overline{\text{EBCOReset}} = 1$	Usage
PerClk ¹	See Note 1	Toggling	Peripheral bus clock. During an EBC transfer, all EBC signal transitions and data sampling occurs synchronous to PerClk.
$\overline{\text{ExtReset}}$	0	1	Peripheral reset for use by slaves and external bus masters.
PerAddr0:31	High impedance	Last Address	Peripheral address bus. PerAddr0 is the most significant bit.
PerData0:31	High impedance	00000000	Peripheral data bus. PerData0 is the most significant bit.
PerPar0:3	High impedance	0000	Peripheral parity bus. The EBC implements odd parity.
$\overline{\text{PerCS0:7}}$	High impedance	1	Chip selects. Use CS0 to select the device when attaching boot device to the peripheral bus.
PerR/W	High impedance	1	Read not write.
$\overline{\text{PerWBE0:3}}$	High impedance	1	Write byte enables or read/write byte enables.
$\overline{\text{PerOE}}$	High impedance	1	Output enable.
$\overline{\text{PerWE}}$	High impedance	1	Write enable. $\overline{\text{PerWE}}$ is low whenever any bit in $\overline{\text{PerWBE0:3}}$ is low and $\text{PerR/W} = 0$.
$\overline{\text{PerBLast}}$	High impedance	1	Burst Last. Active during non-burst operations and the last transfer of a burst access.
PerReady	Input	Input	An input to allow external peripherals to perform device-paced transfers.
PerErr	Input	Input	Peripheral data error input. Sampled during the data transfer only.

Note 1: PerClk is stable a minimum of 20 μS prior to the deassertion of $\overline{\text{ExtReset}}$, assuming SysClk is 66 MHz or lower.

30.1.1 Interfacing to Byte, Halfword, and Word Devices

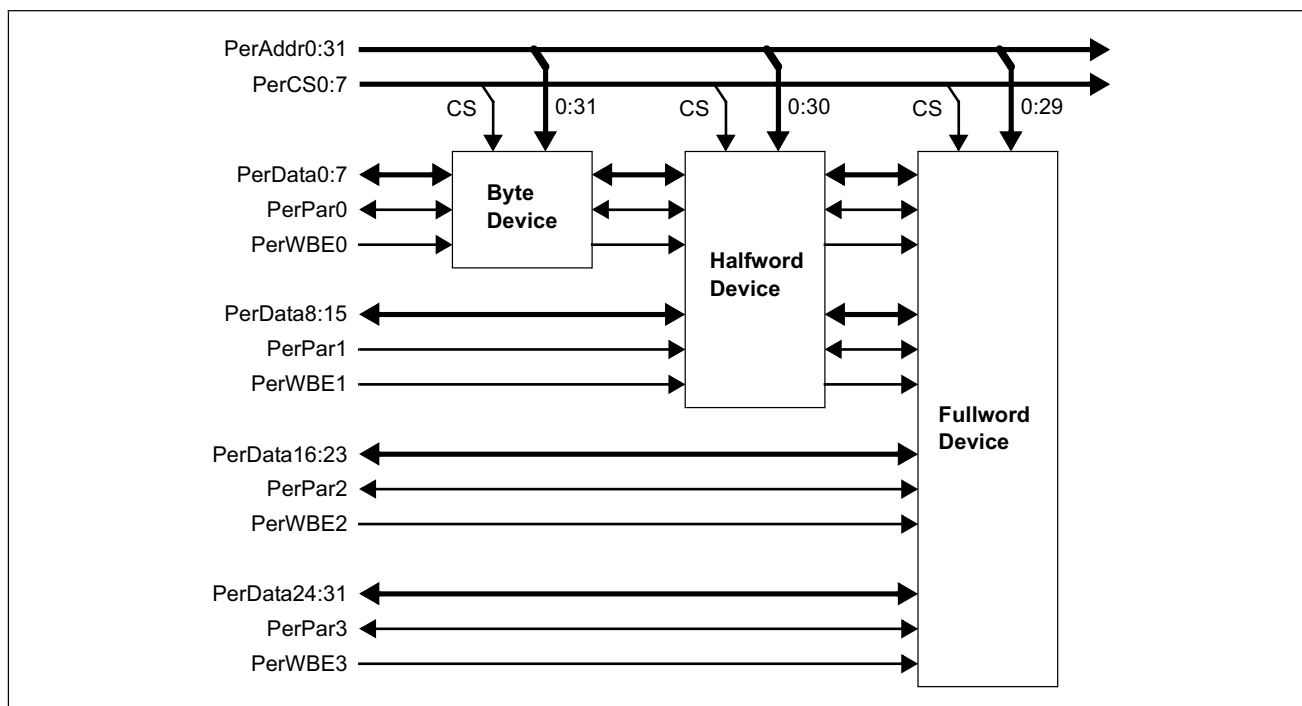
Figure 30-2 illustrates how to interface from byte, halfword, and word devices to the peripheral data bus. When devices are connected in this way, the EBC supports burst transfers and automatically converts read and write operations to the data width of the external device. As shown in Figure 30-2, halfword devices should not connect to PerAddr31. Similarly, a 32-bit device does not require either PerAddr30 or PerAddr31. Instead, the active byte lanes should be inferred from $\overline{\text{PerWBE0:3}}$, the read/write byte enables.

When a large number of byte and halfword devices are attached to the peripheral data bus, the capacitive loading on byte lane 0 (and byte lane 1, if many halfword devices are used) will be much larger than the loading on byte 3, possibly resulting in unacceptable timing performance on byte 0 or byte 1.

Preliminary User's Manual

If a bank register is configured as word-wide, then byte-wide devices may be attached to the bus in any byte lane (and accessed using byte loads and stores). Similarly, if a bank register is configured as word-wide, then halfword-wide devices may be attached to the bus in the byte 0/byte 1 lane, or in the byte 2/byte 3 lane, and accessed using halfword loads and stores. External logic may be required to develop additional control signals if the data bus is utilized in this manner.

Figure 30-2. Attachment of Devices of Various Widths to the Peripheral Data Bus



30.1.2 Driver Enables

As shown in *Table 30-2*, the output enables for the peripheral address, data, and most of the EBC control signals are configurable. For systems that do not use an external master or where the external master does not directly control devices on the peripheral bus, setting `EBC0_CFG[EMC] = 1` eliminates the need for pull-up resistors on `PerCS0:7`. If the EBCO shares the external bus with an external master it will high impedance all outputs when the external master wins the arbitration for the bus. This bit allows the EBCO to continue driving the Chip Selects only while the external master owns the external bus.

Pullups are also unnecessary on the remainder of the EBC control signals when `EBC0_CFG[ATC] = 1`, `EBC0_CFG[DTC] = 1`, and `EBC0_CFG[CTC] = 1`.

Pullups are required if `EBC0_CFG[ATC] = 0`, `EBC0_CFG[DTC] = 0`, and `EBC0_CFG[CTC] = 0` because if the EBCO stops driving `PerAddr` for a sufficiently long time, `PerAddr` can charge/discharge to the threshold voltage of the receivers on the bus. This has the potential to cause the receivers to oscillate, creating excessive noise and/or low current draw.

Both chip and system resets set the output enable control bits `EBC0_CFG[ATC] = 1`, `EBC0_CFG[DTC] = 1`, and `EBC0_CFG[CTC] = 1`. In most applications, clearing `ATC`, `DTC`, or `CTC` is not recommended. If `EBC0_CFG[CTC] = 0`, EBC control signals can transition from the active state to high impedance without first being driven inactive. To prevent this, all peripheral banks must be configured with at least one hold cycle, `EBC0_BnAP[TH] > 0`.

Table 30-2. Effect of Driver Enable Programming on EBC Signal States

EBC Operation	$\overline{\text{ExtReset}}$ PerClk	$\overline{\text{PerCS0:7}}$	PerAddr0:31	$\overline{\text{PerR/W}}$ $\overline{\text{PerWBE0:3}}$ $\overline{\text{PerOE}}$ $\overline{\text{PerBLast}}$ $\overline{\text{PerWE}}$	PerData0:31 PerPar0:3
Reset	Driven	Driven	Driven	Driven	Driven
Idle	Driven	EBC0_CFG[CTC] ¹	EBC0_CFG[ATC] ¹	EBC0_CFG[CTC] ¹	EBC0_CFG[DTC] ¹
Read	Driven	EBC0_BnAP[OEO] ²	EBC0_BnAP[OEO] ²	EBC0_BnAP[OEO] ²	High impedance
Write	Driven	EBC0_BnAP[OEO] ²	EBC0_BnAP[OEO] ²	EBC0_BnAP[OEO] ²	EBC0_BnAP[OEO, OEN] ²
External master ownership	Driven	EBC0_CFG[EMC]	High impedance	High impedance	High impedance
DMA peripheral transfer	Driven	Driven	High impedance	Driven	Driven by DMA Controller

Note 1: $\overline{\text{SysReset}}$ active.

Note 1: If the EBC0_CFG bit is set, the signal is driven to the appropriate state during the indicated EBC operation. Otherwise, the I/O is high impedance.

Note 2: The OEO has an effect on driver enables as follows:

For PerData0:31 and PerPar0:3,

- If EBC0_CFG[OEO] = 1, data is driven coincident with address for writes.
- If EBC0_CFG[OEO] = 0, data is driven EBC0_BnAP[OEN] cycles after PerCSn is asserted.

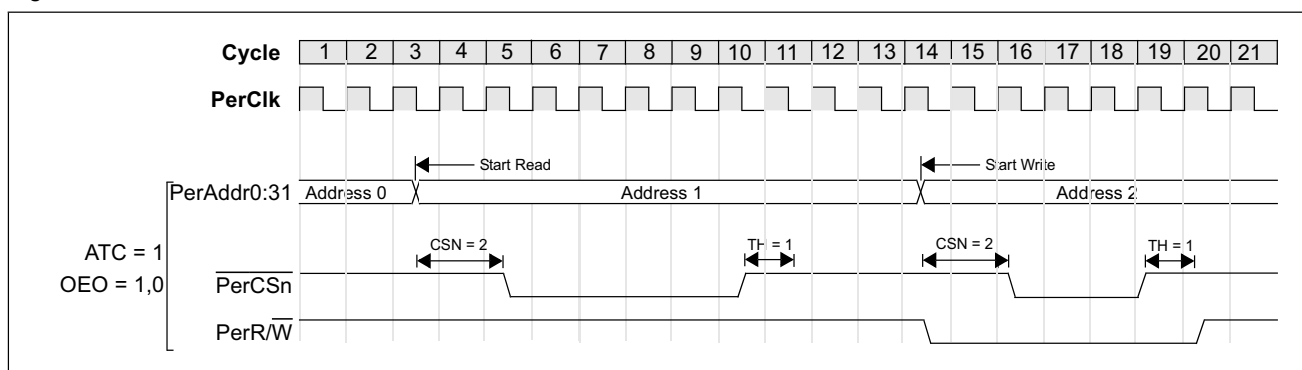
For $\overline{\text{PerCS0:7}}$, $\overline{\text{PerAddr0:31}}$, $\overline{\text{PerR/W}}$, $\overline{\text{PerWBE0:3}}$, $\overline{\text{PerOE}}$, $\overline{\text{PerBLast}}$, and $\overline{\text{PerWE}}$

- If EBC0_CFG[OEO] = 1, they are driven beginning late in the cycle prior to the address.
- If EBC0_CFG[OEO] = 0, they are driven coincident with the address.

30.1.2.1 Effect of ATC and OEO On Address Bus Driver

If ATC = 1 and OEO = 0,1 as described in Figure 30-3, then the external bus is always driven when the EBCO is idle, except when an external master owns the bus.

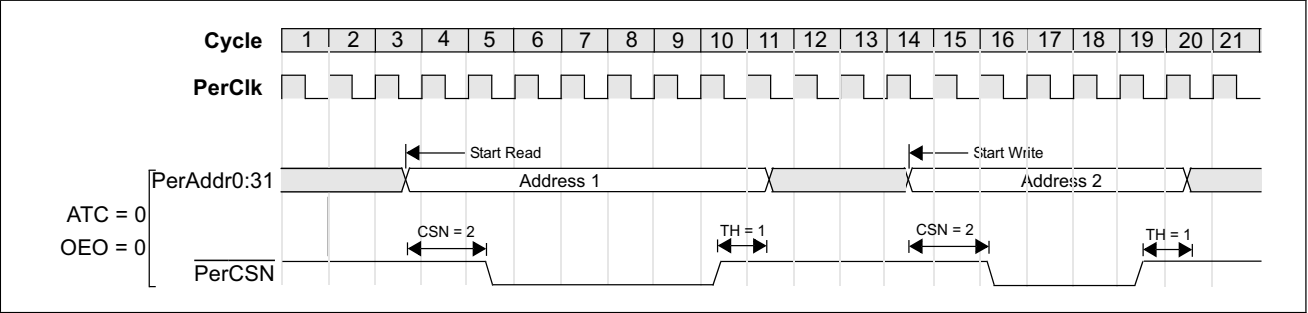
Figure 30-3. ATC = 1, OEO = 0,1



Preliminary User's Manual

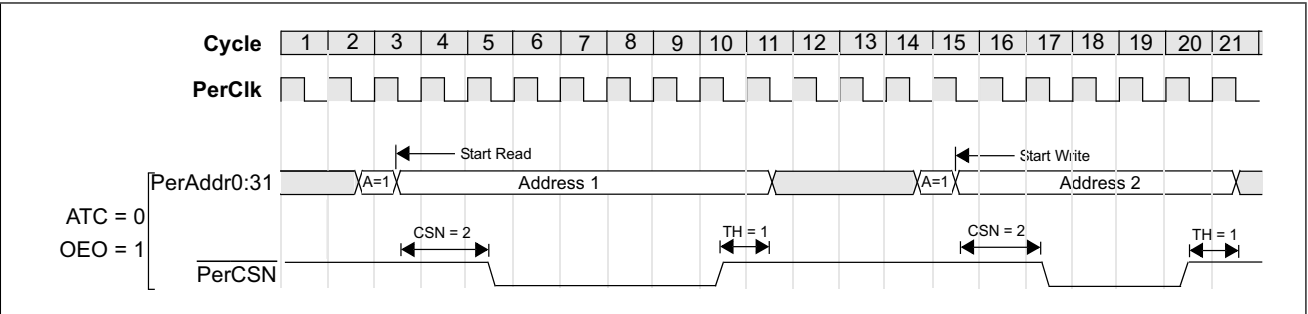
If ATC = 0 and OEO = 0, as described in *Figure 30-4*, then the external bus address driver enables goes active when the cycle starts (off clock edge of first cycle) on the external bus.

Figure 30-4. ATC = 0, OEO = 0



If ATC = 0 and OEO = 1, as described in *Figure 30-5*, the external bus address driver enable goes active prior to the first clock of the cycle on the external bus.

Figure 30-5. ATC = 0, OEO = 1



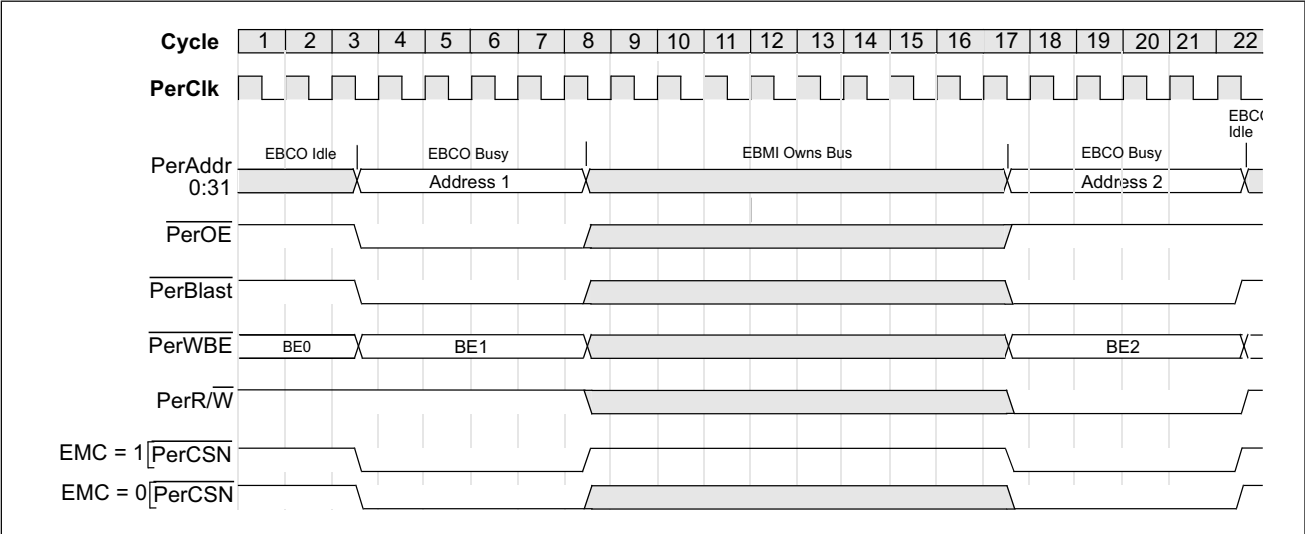
30.1.2.2 CTC, OEO, and EMC Effect on Control Signal Drivers

The control signals include $\overline{\text{PerCSN}}$, $\overline{\text{PerOE}}$, $\overline{\text{PerBlast}}$, $\overline{\text{PerR/W}}$, and $\overline{\text{PerWBE}}$.

If CTC = 1, as described in *Figure 30-6*, then the control signals are always driven when the EBCO is idle except when the external master owns the bus. If the external master owns the bus, all control signals except $\overline{\text{PerCSN}}$ are high impedance. The $\overline{\text{PerCSN}}$ driver depends on the EMC value.

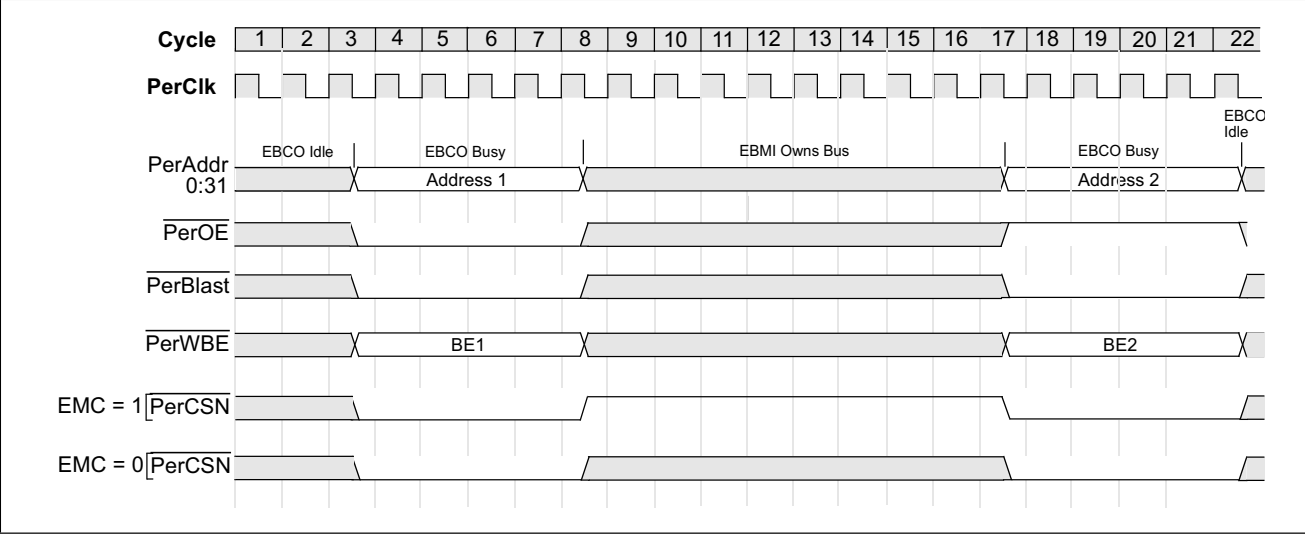
If EMC = 1, the $\overline{\text{PerCSN}}$ driver continues to be driven; otherwise, the $\overline{\text{PerCSN}}$ is high impedance. The OEO has no effect when CTC = 1.

Figure 30-6. CTC = 1



If CTC = 0 and OEO = 0, as described in *Figure 30-7*, then the control signal driver goes active when the cycle starts on the external bus (off clock edge of first cycle on the external bus).

Figure 30-7. CTC = 0, OEO = 0



If CTC = 0 and OEO = 1, as described in *Figure 30-8*, then the control signal drive enable goes active prior to the first clock of the cycle on the external bus.

The diagram illustrates the timing of EBC and EBMI bus operations over 22 clock cycles. The signals shown are:

- Cycle:** 1 to 22.
- PerClk:** Clock signal.
- PerAddr 0:31:** Address bus. It shows EBCO Idle, EBCO Busy (Address 1), EBMI Owns Bus, EBCO Busy (Address 2), and EBCO Idle. The address value A=0 is shown for the first EBCO Busy period, and A=1 for the second.
- PerOE:** Output Enable signal, active low.
- PerBlast:** Burst signal, active low.
- PerWBE:** Write Enable signal, active low. It shows BE=0, BE1, BE=0, and BE2.
- EMC = 1:** Signal for the first EBCO Busy period (Address 1).
- EMC = 0:** Signal for the second EBCO Busy period (Address 2).

If DTC = 1, as described in *Figure 30-9*, then PerData and PerPar are always driven when the EBCO is idle except when an external master owns the bus. OEO has no effect when DTC = 1.

The diagram illustrates the timing of an EBCO read/write operation over 21 clock cycles. The signals shown are:

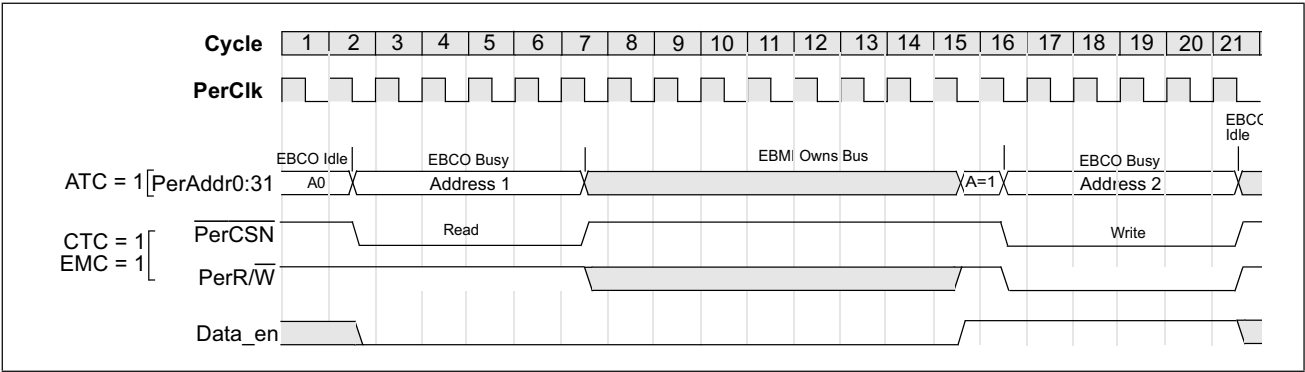
- PerClk**: Clock signal.
- ATC = 1[PerAddr0:31]**: Address bus signal. It shows **A0** in cycle 1, **Address 1** from cycle 2 to 16, and **Address 2** from cycle 17 to 20.
- CTC = 1** and **EMC = 1**: Active-low chip select and enable signals, both active from cycle 1 to 20.
- PerCSN**: Active-low chip select signal, active from cycle 1 to 20.
- Read**: Active-low read signal, active from cycle 1 to 16.
- Write**: Active-low write signal, active from cycle 17 to 20.
- Data_en**: Active-low data enable signal, active from cycle 1 to 20.

The diagram is divided into four phases:

- EBCO Idle**: Cycle 1.
- EBCO Busy**: Cycles 2 to 16.
- EBM! Owns Bus**: Cycles 17 to 20.
- EBCO Busy**: Cycle 21.

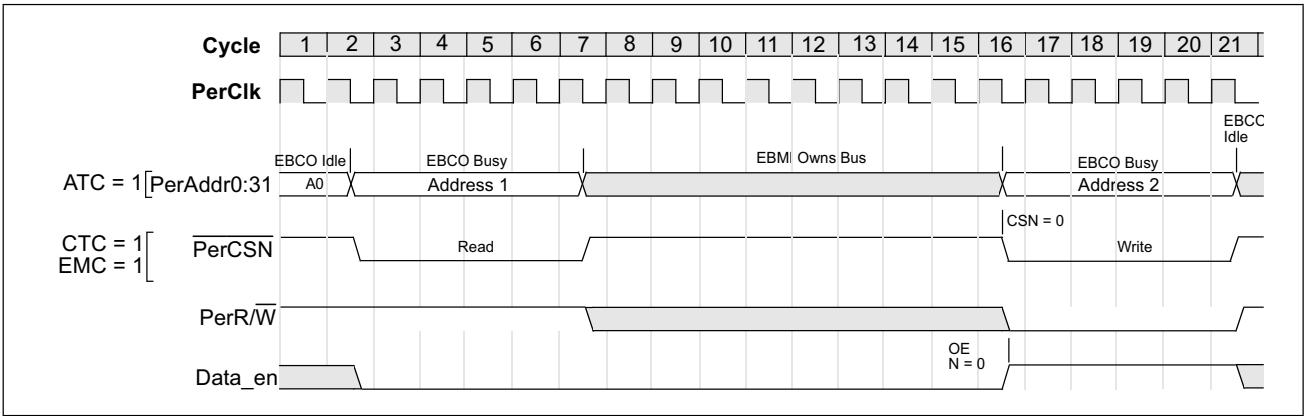
If DTC = 0 and OEO = 1, as described in Figure 30-10 then the external data bus driver enable goes active at the first clock of the cycle on the external bus.

Figure 30-10. DTC = 0, OEO = 1



If DTC = 0 and OEO = 0, as described in Figure 30-11, then the external data bus driver enable is controlled on a bank-by-bank basis by the EBC0_BnAP[OEN] parameter.

Figure 30-11. DTC = 0, OEO = 0



Preliminary User's Manual**30.2 Non-Burst Peripheral Bus Transactions**

The timing of the $\overline{\text{PerCSn}}$, $\overline{\text{PerOE}}$, and $\overline{\text{PerWBE0:3}}$ signals is programmable via the Peripheral Bank Access Parameter (EBC0_BnAP) registers. For non-burst transfers, the access parameter registers control the peripheral bus timing as follows:

- $\overline{\text{PerCSn}}$ becomes active 0-3 PerClk cycles (EBC0_BnAP[CSN]) after the address is driven.
- $\overline{\text{PerOE}}$ is driven low 0-3 PerClk cycles (EBC0_BnAP[OEN]) after $\overline{\text{PerCSn}}$ is active.
- $\overline{\text{PerBLast}}$ is active throughout the entire transfer and is driven high during the programmed hold time (EBC0_BnAP[TH]).
- $\overline{\text{PerWBE0:3}}$ can be either write byte enables or read and write enables.

If EBC0_BnAP[BEM] = 0, $\overline{\text{PerWBE0:3}}$ are write byte enables and:

- $\overline{\text{PerWBE0:3}}$ goes active 0-3 (EBC0_BnAP[WBNI]) PerClk cycles after $\overline{\text{PerCSn}}$ becomes active.
- $\overline{\text{PerWBE0:3}}$ becomes inactive 0-3 (EBC0_BnAP[WBF]) PerClk cycles before $\overline{\text{PerCSn}}$ becomes inactive.

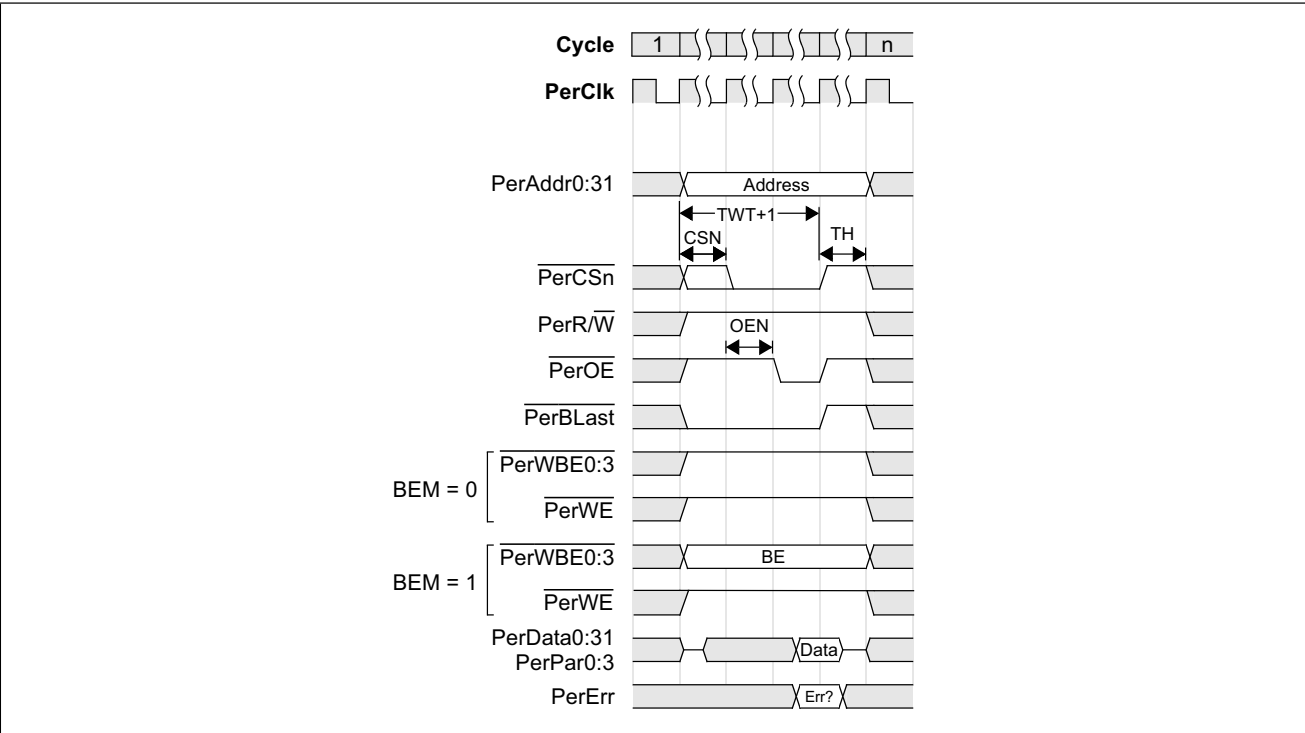
If EBC0_BnAP[BEM] = 1, $\overline{\text{PerWBE0:3}}$ are read/write byte enables and have timing identical to the peripheral address bus. In this case the EBC0_BnAP[WBNI] and EBC0_BnAP[WBF] parameters are ignored.

- 1–256 PerClk cycles (EBC0_BnAP[TWT] + 1) after the address became valid:
 - If EBC0_CFG[CTC] = 1 or EBC0_BnAP[TH] > 0, $\overline{\text{PerCSn}}$ is driven high.
 - If EBC0_CFG[CTC] = 0 and EBC0_BnAP[TH] = 0, $\overline{\text{PerCSn}}$ transitions directly from logic 0 to the high-impedance state.
- The fields EBC0_BnAP[TWT, CSN, OEN, WBNI, WBF] in are not independent. For non-burst configured banks (EBC0_BnAP[BME] = 0) that are also non-device-paced (EBC0_BnAP[RE] = 0), it is required that $\text{TWT} \geq \text{CSN} + \text{MAX}(\text{EBC0_BnAP}[\text{OEN}, \text{WBNI}]) + \text{EBC0_BnAP}[\text{WBF}]$.
- The hold time, EBC0_BnAP[TH], is programmable from 0 to 7 PerClk cycles. During the hold time, the peripheral address bus remains driven with the last address, and all control signals are actively driven high. If the operation was a write, the peripheral data bus continues driving the last data value.
- There is no guarantee of dead cycles between transfers on the peripheral interface for one OPB transfer. If the OPB request size is greater than the bank size or the OPB request size is the same or less than the bank size but with the OPB seqaddr signal asserted, if the EBCO does packing data for read and unpacking data for write, and if the number of hold cycles is programmed to 0 (EBC0_BnAP[TH] = 0 and EBC0_BnAP[CSN] = 0), then:
 - $\overline{\text{PerCSn}}$ may not go inactive between the back-to-back transfers.
 - If EBC0_BnAP[OEN] = 0, $\overline{\text{PerOE}}$ may not become inactive between the back-to-back transfers.
 - If EBC0_BnAP[WBNI] = 0 and EBC0_BnAP[WBF] = 0, $\overline{\text{PerWBE0:3}}$ may not go inactive between the back-to-back transfers.
- There will always be a dead cycle between transfers on the peripheral interface that are initiated for separate OPB transfer.

30.2.1 Single Read Transfer

Figure 30-12 shows the peripheral interface timing for a single read transfer from a non-burst enabled ($EBC0_BnAP[BME] = 0$) bank. The transaction begins with the address being driven. Since this is a single transfer, $PerBLast$ is also driven active along with the address. If byte enable mode is enabled for the bank ($EBC0_BnAP[BEM] = 1$), the byte enables are also output concurrently on $PerWBE0:3$. $PerCSn$ then becomes active $EBC0_BnAP[CSN]$ cycles after the address, while $PerOE$ goes low $EBC0_BnAP[OEN]$ cycles after $PerCSn$. The EBC then waits until $EBC0_BnAP[TWT] + 1$ cycles have elapsed since the start of the transaction and then reads the data bus and the peripheral error input, $PerErr$. If parity checking is enabled ($EBC0_BnAP[PAR] = 1$), the parity is also read at this time. The EBC then drives $PerCSn$, $PerOE$, and $PerBLast$ high and waits $EBC0_BnAP[TH]$ cycles.

Figure 30-12. Single Read Transfer



Preliminary User's Manual

30.2.2 Single Write Transfer

Figure 30-13 shows the peripheral interface timing for a single write transfer to a non-burst enabled ($\text{EBC0_BnAP[BME]} = 0$) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is also driven active along with the address. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address. At this point the signalling sequence depends on whether or not byte enable mode is enabled for the bank.

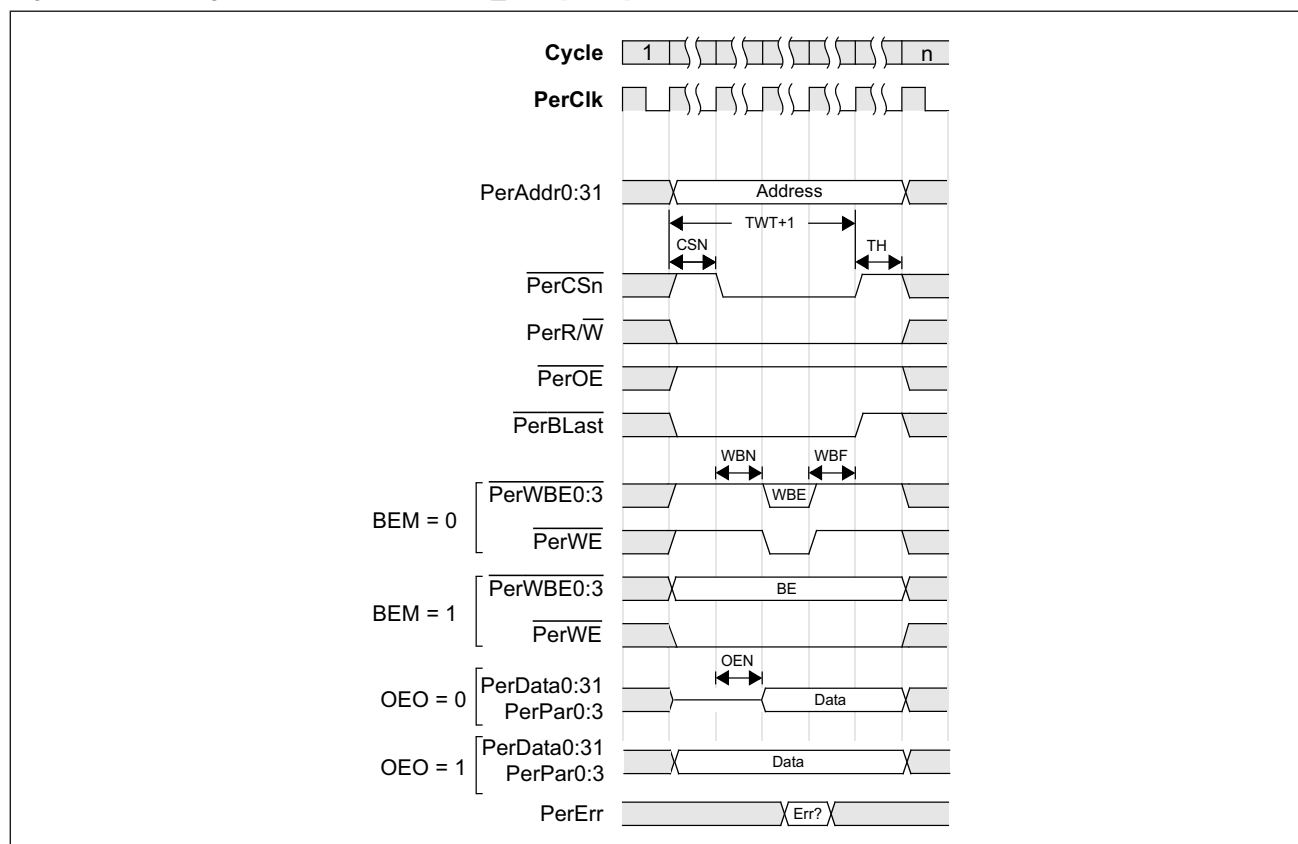
- If $\text{EBC0_BnAP[BEM]} = 0$, byte enable mode is disabled and the $\overline{\text{PerWBE0:3}}$ are write byte enables. The appropriate write byte enables go low EBC0_BnAP[WBW] cycles after $\overline{\text{PerCSn}}$. The EBC then waits until $(\text{EBC0_BnAP[TWT]} - \text{EBC0_BnAP[WBW]} + 1)$ cycles have elapsed since the start of the transaction, then drives all the $\overline{\text{PerWBE0:3}}$ inactive.
- If $\text{EBC0_BnAP[BEM]} = 1$, the $\overline{\text{PerWBE0:3}}$ lines are byte enables and have the same timing as the peripheral address bus.

OEO also has an effect on PerData and PerPar .

- If $\text{EBC0_CFG[OEO]} = 1$, PerData and PerPar become actively driven the same time as the address.
- If $\text{EBC0_CFG[OEO]} = 0$, PerData and PerPar become actively driven EBC0_BnAP[OEN] after $\overline{\text{PerCSn}}$ falls.

After $\text{EBC0_BnAP[TWT}+1]$ cycles elapse from the start of transfer, $\overline{\text{PerCSn}}$ and $\overline{\text{PerBLast}}$ are driven high. The EBC then waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

Figure 30-13. Single Write Transfer, $\text{EBC0_CFG[OEO]} = 0/1$



30.3 Burst Transactions

Bursting is controlled on a per-bank basis by the burst mode enable bit in the EBC0_BnAP registers. When enabled (EBC0_BnAP[BME] = 1) this mode activates bursting for all OPB sequential transfers to the EBC, and all packing and unpacking operations.

OPB sequential transfers to the OPB occur when the CPU performs cache line transfers to the EBC or when the CPU or any other PLB master performs fixed length burst transfers to the EBC.

Under many conditions, the EBC will execute extra read transfers on the peripheral interface for a sequential OPB burst operation. For example, if the transfer size on the OPB is less than or equal to the bank size, the EBC will do extra read(s) on the peripheral interface before the OPB termination can be detected. If extra read transfers are not allowed for a specific peripheral device (a FIFO for example), the EBC must be programmed to support fixed length bursts in EBC0_BnAP[BCE]. If the fixed length burst count is complete and the OPB operation has not terminated, the EBC will execute another fixed length burst read. The fixed length burst will terminate early if an OPB termination is detected.

Note: Access to FIFO devices must start and end on a word boundary.

When bursting is enabled:

- $\overline{\text{PerCSn}}$ becomes active 0-3 (EBC0_BnAP[CSN]) PerClk cycles after the address becomes valid.
- $\overline{\text{PerCSn}}$ is no longer actively driven:
 - 1-32 (EBC0_BnAP[FWT] + 1) PerClk cycles after the address becomes valid when a single transfer occurs to a burst-enabled bank.
 - 1-8 (EBC0_BnAP[BWT] + 1) PerClk cycles after the last address becomes valid during a burst:
 - If EBC0_CFG[CTC] = 1 or EBC0_BnAP[TH] > 0, $\overline{\text{PerCSn}}$ is driven high.
 - If EBC0_CFG[CTC] = 0 and EBC0_BnAP[TH] = 0, $\overline{\text{PerCSn}}$ transitions directly from logic 0 to the high-impedance state.
- During read operations, $\overline{\text{PerOE}}$ is driven low 0-3 (EBC0_BnAP[OEN]) PerClk cycles after $\overline{\text{PerCSn}}$ is active. $\overline{\text{PerOE}}$ goes inactive when $\overline{\text{PerCSn}}$ goes inactive.
- For bursts, the EBC drives a new address (EBC0_BnAP[FWT] + 1) + $n \times$ (EBC0_BnAP[BWT] + 1) cycles after the start of the transaction, where $n = 0, 1, 2, \dots$
- During write operations, the write data is driven concurrent with each address except the first one.
- For the first address:
 - If EBC0_CFG[OE0] = 1, PerData and PerPar become actively driven the same time as the address.
 - If EBC0_CFG[OE0] = 0, PerData and PerPar become actively driven EBC0_BnAP[OEN] after $\overline{\text{PerCSn}}$ falls.
- $\overline{\text{PerWBE0:3}}$ can be either write byte enables or read and write enables.

If EBC0_BnAP[BEM] = 0, $\overline{\text{PerWBE0:3}}$ are write byte enables and:

- For the first transfer of a burst or a single transfer to a burst enabled bank, the appropriate write byte enables go low 0-3 (EBC0_BnAP[WBNI]) cycles after $\overline{\text{PerCSn}}$ becomes active. The EBC then waits until EBC0_BnAP[FWT] – EBC0_BnAP[WBF] + 1 cycles have elapsed since the start of the transaction and drives $\overline{\text{PerWBE0:3}}$ inactive.
- The remaining transfers of the burst are similar, except that $\overline{\text{PerWBE0:3}}$ becomes active when each new address is driven on the interface. The $\overline{\text{PerWBE0:3}}$ signals remain low for (EBC0_BnAP[BWT] + 1) – EBC0_BnAP[WBNI] – EBC0_BnAP[WBF] cycles.

Preliminary User's Manual

If $\text{EBC0_BnAP[BEM]} = 1$, $\overline{\text{PerWBE0:3}}$ are byte enables that have timing identical to the peripheral address bus. In this case, EBC0_BnAP[WBNI] and EBC0_BnAP[WBFI] are ignored.

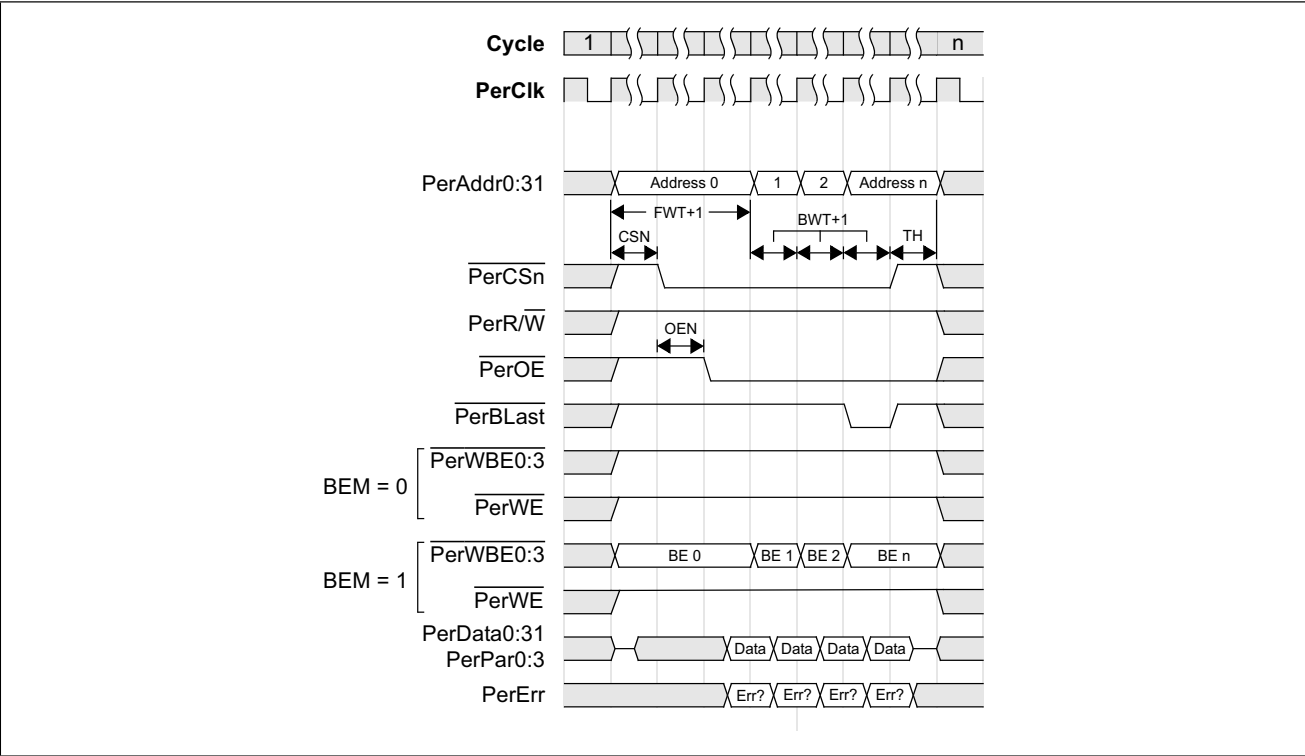
- $\overline{\text{PerBLast}}$ is active throughout the entire last (or only) transfer of a burst operation and is deactivated during the programmed hold time (EBC0_BnAP[TH]).
- EBC0_BnAP[CSNI] , WBNI , OENI apply to the first (or only) transfer of a burst, while EBC0_BnAP[WBFI] applies to all transfers. It is required that $\text{FWT} \geq \text{CSNI} + \text{MAX}(\text{EBC0_BnAP[OENI]}, \text{WBNI}) + \text{EBC0_BnAP[WBFI]}$, and $\text{EBC0_BnAP[BWT]} \geq \text{WBFI}$.
- Hold time (EBC0_BnAP[TH]) is programmable from 0 to 7 cycles. During the hold time, the peripheral address bus remains driven, and all control signals are driven inactive. If the operation was a write, the peripheral data bus continues driving the write data.
- There is no guarantee of dead cycles between transfers on the peripheral interface for one OPB transfer. If the OPB request size is greater than the bank size or the OPB request size is the same or less than the bank size but with the OPB seqaddr signal asserted, if the EBCO does packing data for read and unpacking data for write, and if the number of hold cycles is programmed to 0 ($\text{EBC0_BnAP[TH]} = 0$ and $\text{EBC0_BnAP[CSNI]} = 0$), then:
 - $\overline{\text{PerCSNI}}$ may not go inactive between the back-to-back transfers.
 - If $\text{EBC0_BnAP[OENI]} = 0$, $\overline{\text{PerOE}}$ may not become inactive between the back-to-back transfers.
 - If $\text{EBC0_BnAP[WBNI]} = 0$ and $\text{EBC0_BnAP[WBFI]} = 0$, $\overline{\text{PerWBE0:3}}$ may not go inactive between the back-to-back transfers.
- There will always be a dead cycle between transfers on the peripheral interface that are initiated for separate OPB transfers.

30.3.1 Burst Read Transfer

Figure 30-14 shows the peripheral interface timing for a burst read transfer from a burst enabled ($\text{EBC0_BnAP[BME]} = 1$) bank. The transaction begins with the address being driven. If byte enable mode is enabled for the bank ($\text{EBC0_BnAP[BEM]} = 1$) the byte enables are also output concurrently on $\overline{\text{PerWBE0:3}}$. $\overline{\text{PerCSNI}}$ then becomes active EBC0_BnAP[CSNI] cycles after the address, while $\overline{\text{PerOE}}$ goes low EBC0_BnAP[OENI] cycles after $\overline{\text{PerCSNI}}$. The EBC waits until $\text{EBC0_BnAP[FWT]}+1$ cycles have elapsed since the start of the transaction and then reads the data bus and the peripheral error input (PerErr). If parity checking is enabled ($\text{EBC0_BnAP[PEN]} = 1$), the parity is read at the same time.

The next address of the burst is then driven, and after $EBC0_BnAP[BWT]+1$ cycles, the EBC performs the next read. The remaining items in the burst are read in the same manner, except that $\overline{PerBLast}$ is active during the last data element. The EBC then drives \overline{PerCSn} , \overline{PerOE} and $\overline{PerBLast}$ high and waits $EBC0_BnAP[TH]$ cycles before allowing any pending transfers to occur.

Figure 30-14. Burst Read Transfer



Preliminary User's Manual

30.3.2 Burst Write Transfer

Figure 30-15 shows the peripheral interface timing for a burst write transfer to burst enabled ($EBC0_BnAP[BME] = 1$) bank. The transaction begins with the address being driven. At this point, the signalling sequence depends on whether byte enable mode is enabled for the bank.

- If $EBC0_BnAP[BEM] = 0$, byte enable mode is disabled, and $\overline{PerWBE0:3}$ are write byte enables. In this case, the appropriate write byte enables go low $EBC0_BnAP[WBN]$ cycles after \overline{PerCSn} . The EBC then waits until $(EBC0_BnAP[FWT] + 1) - EBC0_BnAP[WBF]$ cycles have elapsed since the start of the transaction and drives $\overline{PerWBE0:3}$ inactive. $EBC0_BnAP[WBF]$ cycles are then allowed to elapse, after which, the address and data are output for the second element in the burst. As shown in Figure 30-15, the EBC transfers the subsequent data items in a similar manner.
- If $EBC0_BnAP[BEM] = 1$, the $\overline{PerWBE0:3}$ lines are byte enables and have the same timing as the peripheral address bus. In this configuration, external logic may be necessary to latch write data at the appropriate times. The driving of write data and data parity is controlled by $EBC0_BnAP[OEN]$ when $EBC0_CFG[OEO] = 0$.

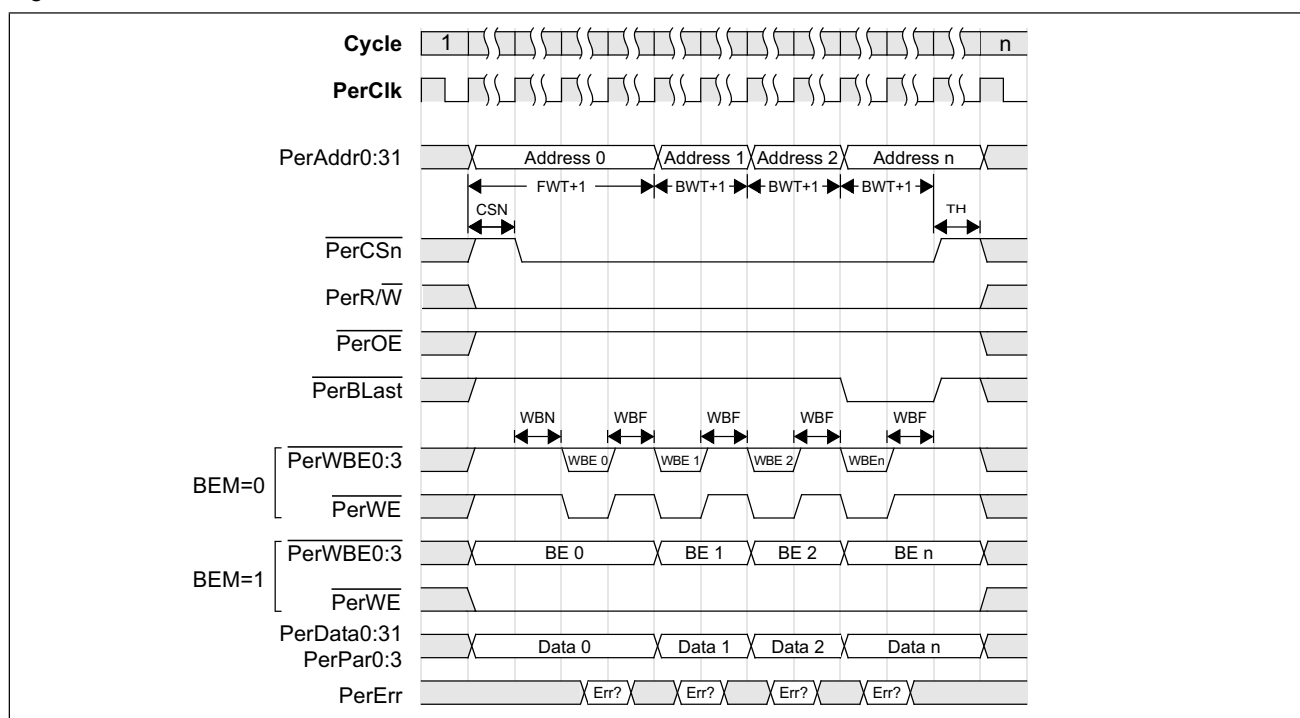
$\overline{PerBLast}$ goes low at the beginning of the last transfer to indicate that the burst is ending. The EBC then drives \overline{PerCSn} and $\overline{PerBLast}$ high and waits $EBC0_BnAP[TH]$ cycles before allowing any pending transfers to occur.

$\overline{PerData0:31}$ and $\overline{PerPar0:3}$ depend on OEO and OEN for their first data transfers as follows:

- If $EBC0_CFG[OEO] = 1$, $\overline{PerData0:31}$ and $\overline{PerPar0:3}$ become actively driven at the same time as the address.
- If $EBC0_CFG[OEO] = 0$, $\overline{PerData0:31}$ and $\overline{PerPar0:3}$ become actively driven $EBC0_BnAP[OEN]$ after \overline{PerCSn} falls.

$\overline{PerData0:31}$ and $\overline{PerPar0:3}$ are asserted at the same time as the address.

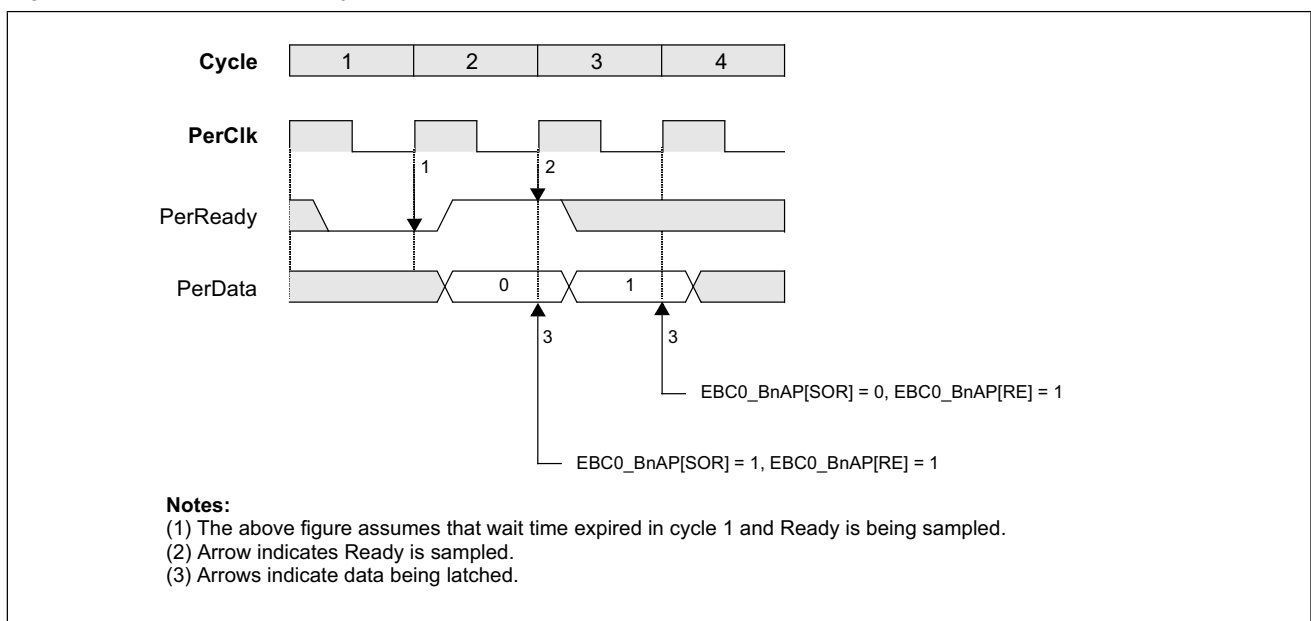
Figure 30-15. Burst Write Transfer



30.4 Device-Paced Transfers

The EBC supports interfacing to peripheral and memory devices with variable timings via device-paced transfers. An EBC bank is configured for device-paced transfers by programming `EBC0_BnAP[RE]=1`. During a device-paced transfer, the EBC monitors the ready input (`PerReady`) and inserts wait states until the external device is ready or a timeout occurs. When using the ready input modes: Sample On Ready enabled and Sample On Ready disabled. The selection of these modes is controlled on a per-bank basis by `EBC0_BnAP[SOR]`. When Sample On Ready is enabled, (`EBC0_BnAP[SOR] = 1`) data is transferred on the `PerClk` rising edge where `PerReady` is sampled active. When Sampling On Ready is disabled (`EBC0_BnAP[SOR] = 0`), `PerReady` sampled active causes the data transfer to occur in the next cycle, which results in an additional cycle of wait time. *Figure 30-16* illustrates available ready modes and latch times transfer.

Figure 30-16. Available Ready Modes and Latch Times



- For burst disabled banks (`EBC0_BnAP[BME] = 0`), sampling of the `PerReady` input starts `EBC0_BnAP[TWT]` cycles after the beginning of the transfer. Wait states are inserted and sampling continues once per cycle until either `PerReady` is high when sampled or a timeout occurs.
- For burst enabled banks (`EBC0_BnAP[BME] = 1`), sampling of the `PerReady` input starts `EBC0_BnAP[FWT]` `PerClk` cycles after the beginning of the first transfer of a burst and `EBC0_BnAP[BWT]` cycles after the beginning of subsequent transfers of the burst.
 - If `EBC0_BnAP[SOR] = 0`, sampling continues every other cycle after the address goes active and the wait period (either `TWT` for non-burst, or `FWT` or `BWT` for burst bank) expires until either `PerReady` is sampled high or a timeout occurs. If `EBC0_BnAP[FWT] = EBC0_BnAP[BWT] = 0`, two-cycle transfers occur.
 - If `EBC0_BnAP[SOR] = 1`, sampling continues once per cycle after the address goes active and the wait period (either `TWT` for non-burst, or `FWT` or `BWT` for burst bank) expires until either `PerReady` is sampled high or a timeout occurs. If `EBC0_BnAP[FWT] = EBC0_BnAP[BWT] = 0`, single-cycle transfers occur.
- When `EBC0_BnAP[SOR] = 1`, data transfer occurs at the same `PerClk` edge where `PerReady` is sampled active. In contrast, if `EBC0_BnAP[SOR] = 0`, the data transfer occurs in the next cycle.
- When `EBC0_BnAP[SOR] = 1`, if the hold time is set to zero (`EBC0_BnAP[TH] = 0`), the programmed hold time is ignored, and the EBC performs the transaction with one hold cycle.

Preliminary User's Manual

- When EBC0_BnAP[RE] = 1, the write byte enable off parameter must be programmed to zero (EBC0_BnAP[WBF] = 0). The Write Byte Enables (/PerWBE0:3) are not deasserted between data transfers but change as needed with the address.

The EBC may be programmed to wait only a limited time for PerReady to become active, or it may be programmed for unlimited wait. If EBC0_CFG[PTD] = 1, timeouts are disabled and the EBC waits indefinitely for an active PerReady.

If EBC0_CFG[PTD] = 0, device-paced timeouts are enabled, and the EBC only waits for the number of PerClk cycles programmed in EBC0_CFG[RTC]. The timeout counter is reset whenever the peripheral address changes. In this manner each data element within a device-paced burst transaction is treated separately for the purposes of determining whether a timeout error occurs. If PerReady does not become active before the timeout counter reaches the value programmed into EBC0_CFG[RTC], the transfer is aborted and an error is signalled. See *Error Reporting* on page 1012 for details on how timeout errors are logged.

The required relationship between the access bank parameters changes for device-paced banks, which require that the bank timing wait parameters meet the following relationships:

For non-burst configured banks:

$$\text{EBC0_BnAP[TWT]} > \text{EBC0_BnAP[CSN]} + \text{MAX}(\text{EBC0_BnAP[OEN, WBN]}).$$

For burst configured bank:

For first transfer:

$$\text{EBC0_BnAP[FWT]} > \text{EBC0_BnAP[CSN]} + \text{MAX}(\text{EBC0_BnAP[OEN, WBN]}).$$

For following transfers:

$$\text{EBC0_BnAP[BWT]} > \text{EBC0_BnAP[WBF]}$$

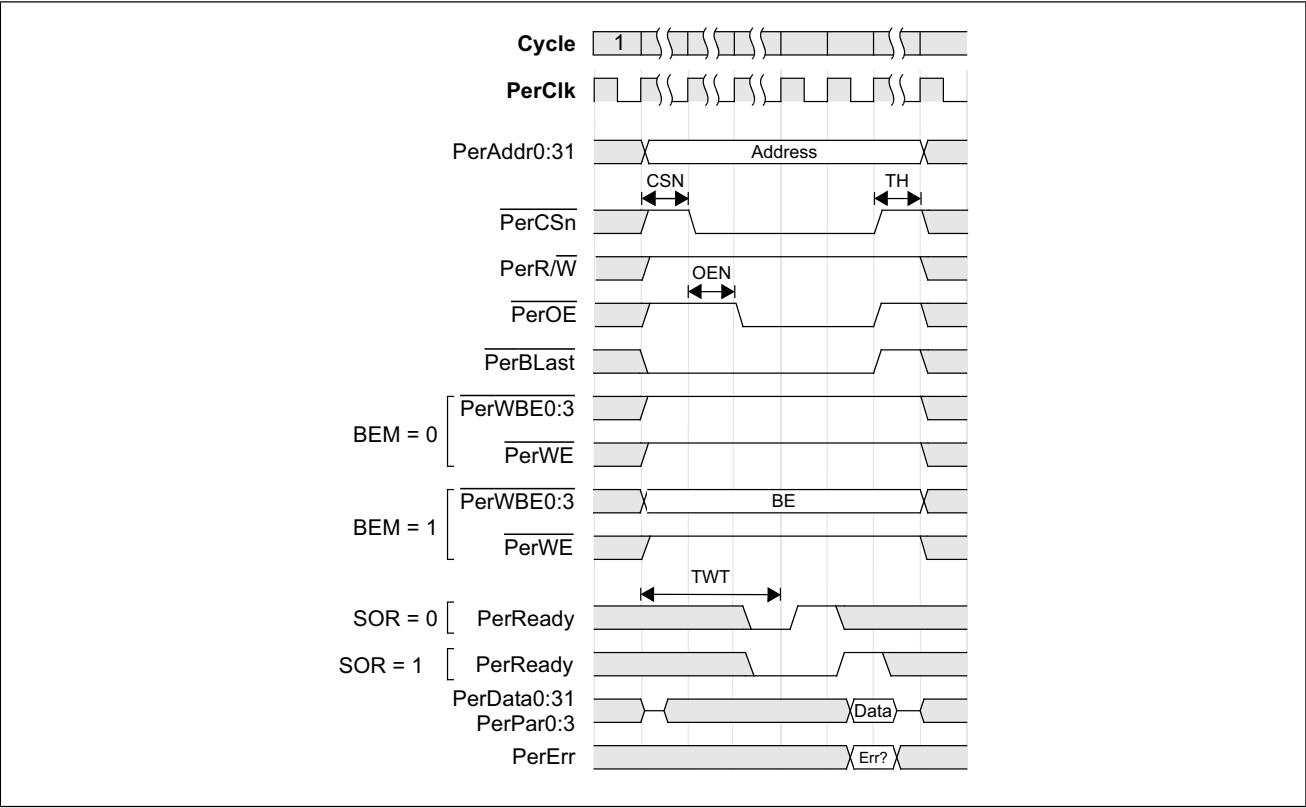
30.4.1 Device-Paced Single Read Transfer

Figure 30-17 shows the peripheral interface timing for a device-paced single read transfer to a burst disabled (EBC0_CFG[BME] = 0) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is driven active along with the address. If byte enable mode is enabled for the bank (EBC0_BnAP[BEM] = 1), the byte enables are also output concurrently on PerWBE0:3. PerCSn then becomes active EBC0_BnAP[CSN] cycles after the address, while PerOE goes low EBC0_BnAP[OEN] cycles after PerCSn.

The EBC then waits until EBC0_BnAP[TWT] cycles have elapsed since the start of the transaction and then begins sampling PerReady. If device-paced timeouts are disabled (EBC0_CFG[PTD] = 1), the EBC waits indefinitely for PerReady to become active; otherwise, the EBC waits only EBC0_CFG[RTC] cycles from the start of the transaction until logging a timeout error.

Once PerReady is sampled active and if Sample On Ready is disabled (EBC0_BnAP[SOR] = 0), the EBC waits one more cycle. The EBC then samples the data bus and the peripheral error input (PerErr). If parity checking is enabled (EBC0_BnAP[PEN] = 1), the parity is also read at this time. The EBC then drives PerCSn, PerOE, and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending EBC transfers to occur.

Figure 30-17. Device-Paced Single Read Transfer



Preliminary User's Manual**30.4.2 Device-Paced Single Write Transfer**

Figure 30-18 shows the peripheral interface timing for a device-paced single write transfer to a burst disabled ($\text{EBC0_BnAP[BME]} = 0$) bank. The transaction begins with the address being driven. Since this is a single transfer, PerBLast is also driven active along with the address. At this point, the signalling sequence depends on whether byte enable mode is enabled for the particular bank.

- If $\text{EBC0_BnAP[BEM]} = 0$, byte enable mode is disabled and the $\overline{\text{PerWBE0:3}}$ are write byte enables. The appropriate write byte enables go low EBC0_BnAP[WBn] cycles after $\overline{\text{PerCSn}}$ went low. $\overline{\text{PerWBE0:3}}$ return high on the same PerClk edge that the write data is transferred (see below).
- If $\text{EBC0_BnAP[BEM]} = 1$, the $\overline{\text{PerWBE0:3}}$ lines are byte enables and have the same timing as the peripheral address bus.

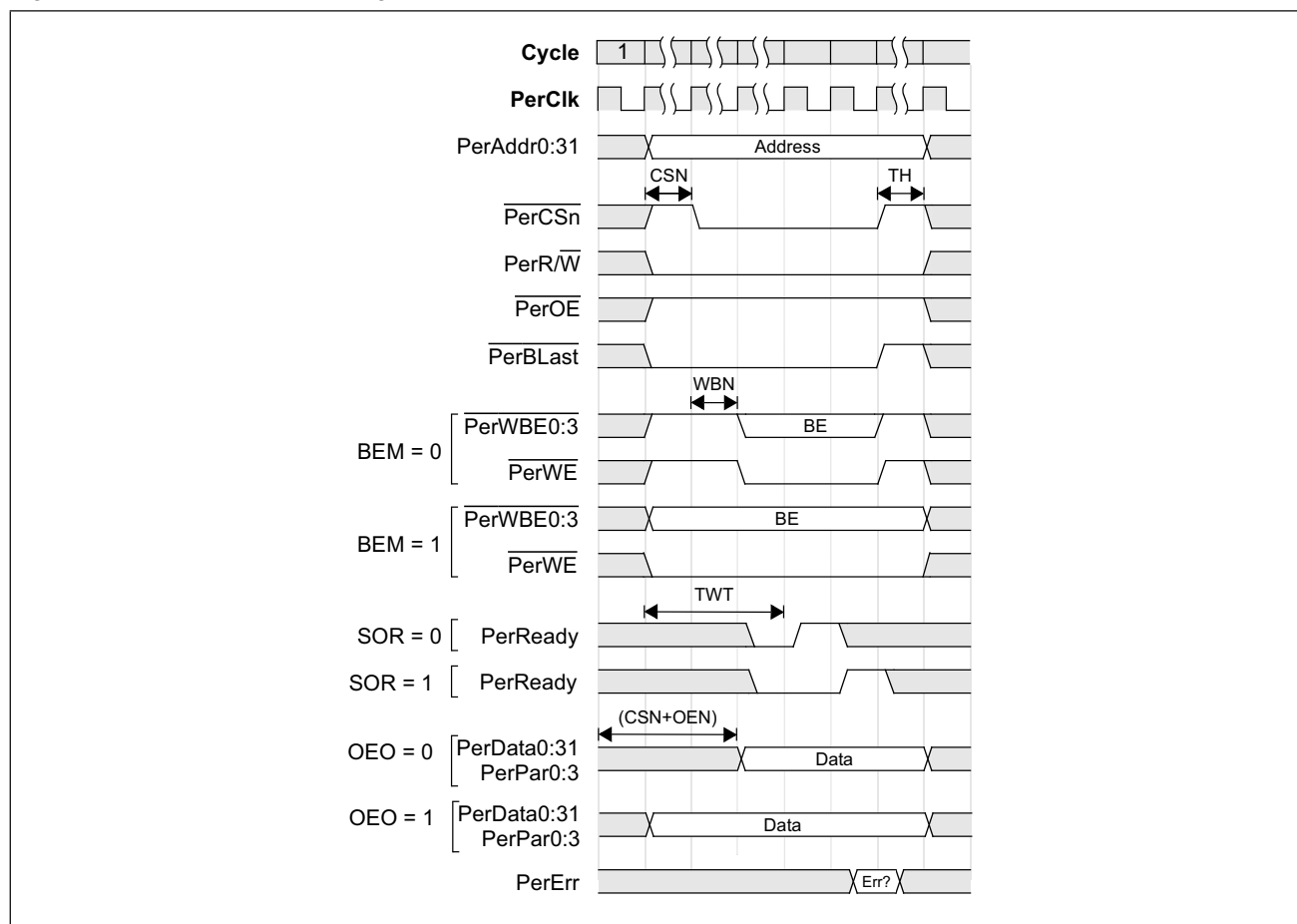
EBC0_CFG[OEO] also has an effect on PerData and PerPar as follows:

- If $\text{EBC0_CFG[OEO]} = 1$, PerData and PerPar become actively driven at the same time as the address.
- If $\text{EBC0_CFG[OEO]} = 0$, PerData and PerPar become actively driven EBC0_BnAP[OEN] cycles after $\overline{\text{PerCSn}}$ falls.

The EBC then waits until EBC0_BnAP[TWT] cycles have elapsed since the start of the transaction and then begins sample PerReady . If device-paced timeouts are disabled ($\text{EBC0_CFG[PTD]} = 1$), the EBC waits indefinitely for PerReady to become active. Otherwise, the EBC waits only EBC0_CFG[RTC] cycles from the start of the transaction until logging a timeout error.

If PerReady is sampled active and Sample On Ready is disabled ($\text{EBC0_BnAP[SOR]} = 0$), the EBC waits one more cycle. At this point, the write transfer occurs, and the EBC reads the peripheral error input (PerErr). The EBC then drives PerCSn, PerOE, and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending EBC transfers to occur.

Figure 30-18. Device-Paced Single Write Transfer



Preliminary User's Manual

30.4.3 Device-Paced Burst Read Transfer

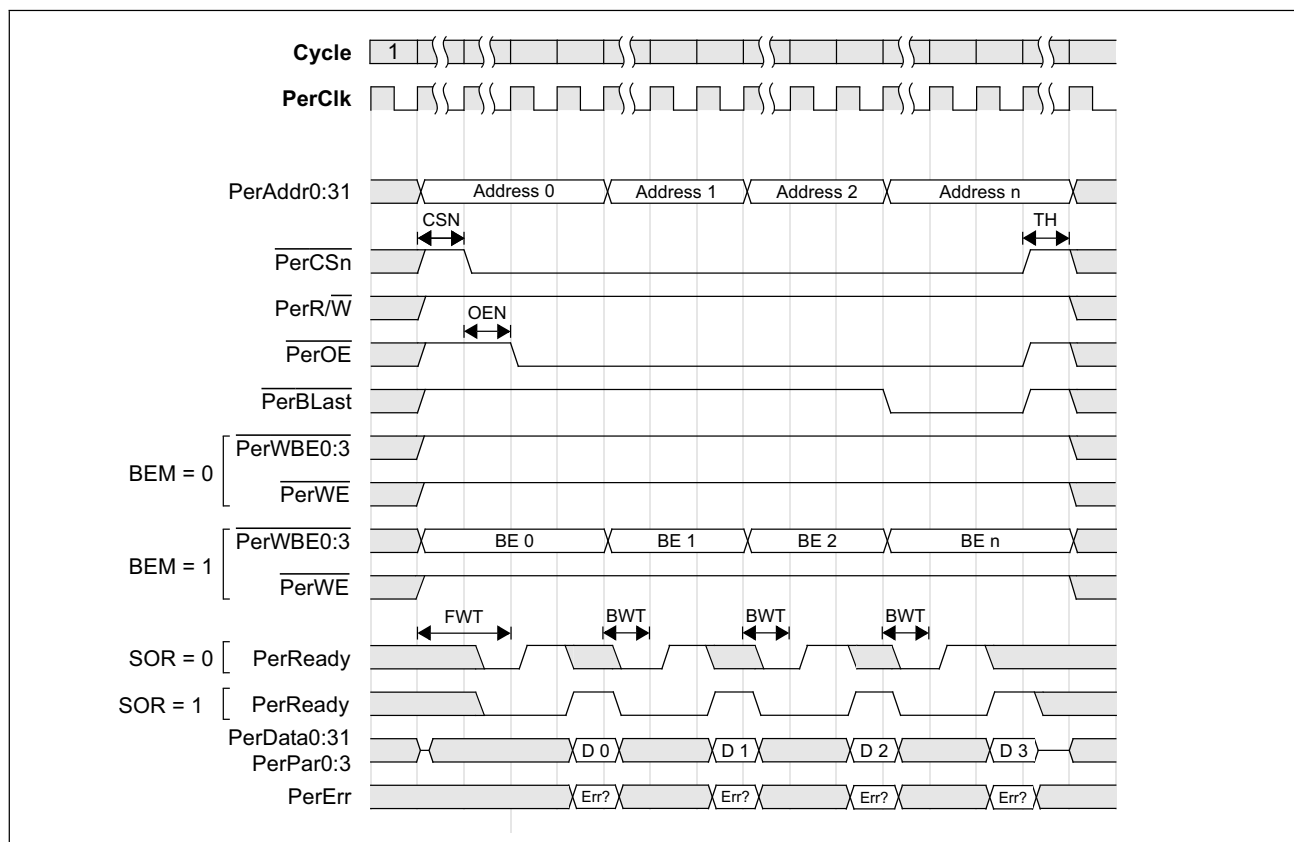
Figure 30-19 shows the peripheral interface timing for a device-paced burst read transfer from a burst enabled ($EBC0_BnAP[BME] = 1$) bank. The transaction begins with the address being driven. If byte enable mode is enabled for the bank ($EBC0_BnAP[BEM] = 1$), the byte enables are also output concurrently on $PerWBE0:3$. $PerCSn$ then becomes active $EBC0_BnAP[CSN]$ cycles after the address, while $PerOE$ goes low $EBC0_BnAP[OEN]$ cycles after $PerCSn$. The EBC then waits until $EBC0_BnAP[FWT]$ cycles have elapsed since the start of the transaction and begins sampling $PerReady$.

If device-paced timeouts are disabled ($EBC0_CFG[PTD] = 1$), the EBC waits indefinitely for $PerReady$ to become active; otherwise, the EBC waits only $EBC0_CFG[RTC]$ cycles from the start of each new address until logging a timeout error.

If $PerReady$ is sampled active and Sample On Ready is disabled ($EBC0_BnAP[SOR] = 0$), the EBC waits one more cycle before sampling read data. The EBC then reads the data bus and the peripheral error input ($PerErr$). If parity checking is enabled ($EBC0_BnAP[PEN] = 1$), the parity is also read. See *Burst Transactions* on page 994 for information on extra reads during burst operations.

The next address of the burst is then driven, and after $EBC0_BnAP[BWT]$ cycles, the EBC waits for $PerReady$ as described above. The remaining items in the burst are read in this same manner, except that $PerBLast$ is active during the last data element. The EBC then drives $PerCSn$, $PerOE$, and $PerBLast$ high and waits $EBC0_BnAP[TH]$ cycles before allowing any pending transfers to occur.

Figure 30-19. Device-Paced Burst Read Transfer



30.4.4 Device-Paced Burst Write Transfer

Figure 30-20 shows the peripheral interface timing for a device-paced burst write transfer to a burst enabled ($EBC0_BnAP[BME] = 1$) bank. The transaction begins with the address being driven. \overline{PerCSn} then becomes active $EBC0_BnAP[CSN]$ cycles after the address. At this point, the signalling sequence depends on whether or not byte enable mode is enabled for the bank.

- If byte enable mode is disabled ($EBC0_BnAP[BEM] = 0$), $\overline{PerWBE0:3}$ are write byte enables. In this case, the appropriate write byte enables go low $EBC0_BnAP[WBn]$ cycles after \overline{PerCSn} becomes active for the first element in a burst and at the same time as each new address for the remainder of the burst. If $EBC0_BnAP[WBn] < 0$, $\overline{PerWBE0:3}$ is driven inactive on the same $PerClk$ edge that write data is transferred (see Figure 30-20); otherwise, $\overline{PerWBE0:3}$ remains low for all data elements in the burst.
- If $EBC0_BnAP[BEM] = 1$, $\overline{PerWBE0:3}$ are byte enables and have the same timing as $PerAddr0:31$.

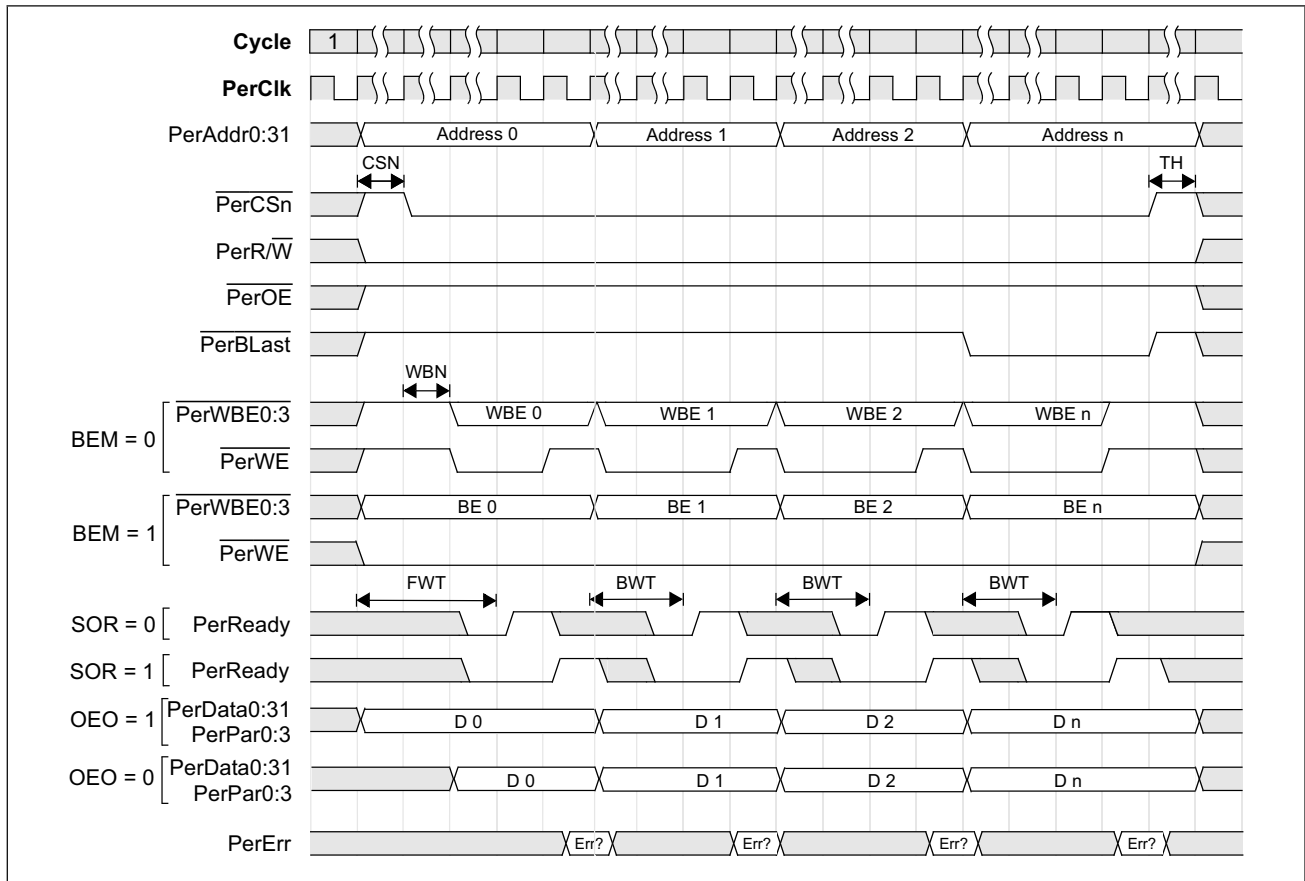
The EBC then waits until $EBC0_BnAP[FWT]$ cycles have elapsed since the start of the transaction and begins sampling $PerReady$. If device-paced timeouts are disabled ($EBC0_CFG[PTD] = 1$), the EBC waits indefinitely for $PerReady$; otherwise, the EBC waits only $EBC0_CFG[RTC]$ cycles from the start of the transaction until logging a timeout error.

If $PerReady$ is sampled active and Sample On Ready is disabled ($EBC0_BnAP[SOR] = 0$), the EBC waits one more cycle. At this point, the write transfer occurs, and the EBC reads the peripheral error input ($PerErr$).

Preliminary User's Manual

The next address of the burst is then driven, and after EBC0_BnAP[BWT] cycles, the EBC waits for PerReady as described above. The remaining items in the burst are transferred in this same manner, except that PerBLast is active for the last data element. The EBC then drives PerCSn, PerOE, and PerBLast high and waits EBC0_BnAP[TH] cycles before allowing any pending transfers to occur.

Figure 30-20. Device-Paced Burst Write Transfer



30.5 EBC Registers

All EBC configuration and status registers are accessed using the **mtdcr** and **mfdcr** instructions. Access to these registers is performed using an indirect addressing method through the EBC0_CFGADDR and EBC0_CFGDATA registers (see *Table 30-3*).

Table 30-3. EBC DCR Addresses

Mnemonic	Register	DCR Number	Access	Page
EBC0_CFGADDR	EBC Address Register	0x012	R/W	1007
EBC0_CFGDATA	EBC Data Register	0x013	R/W	1007

Table 30-4 lists the indirectly accessed EBC configuration and status registers.

Table 30-4. External Bus Configuration and Status Registers

Mnemonic	Name	Offset	Access	Page
EBC0_B0CR	Peripheral Bank 0 Configuration Register	0x00	R/W	1007
EBC0_B1CR	Peripheral Bank 1 Configuration Register	0x01	R/W	1007
EBC0_B2CR	Peripheral Bank 2 Configuration Register	0x02	R/W	1007
EBC0_B3CR	Peripheral Bank 3 Configuration Register	0x03	R/W	1007
EBC0_B4CR	Peripheral Bank 4 Configuration Register	0x04	R/W	1007
EBC0_B5CR	Peripheral Bank 5 Configuration Register	0x05	R/W	1007
EBC0_B6CR	Peripheral Bank 6 Configuration Register	0x06	R/W	1007
EBC0_B7CR	Peripheral Bank 7 Configuration Register	0x07	R/W	1007
EBC0_B0AP	Peripheral Bank 0 Access Parameter Registers	0x10	R/W	1009
EBC0_B1AP	Peripheral Bank 1 Access Parameter Registers	0x11	R/W	1009
EBC0_B2AP	Peripheral Bank 2 Access Parameter Registers	0x12	R/W	1009
EBC0_B3AP	Peripheral Bank 3 Access Parameter Registers	0x13	R/W	1009
EBC0_B4AP	Peripheral Bank 4 Access Parameter Registers	0x14	R/W	1009
EBC0_B5AP	Peripheral Bank 5 Access Parameter Registers	0x15	R/W	1009
EBC0_B6AP	Peripheral Bank 6 Access Parameter Registers	0x16	R/W	1009
EBC0_B7AP	Peripheral Bank 7 Access Parameter Registers	0x17	R/W	1009
EBC0_BEAR	Bus Error Address Register	0x20	R	1013
EBC0_BESR	Peripheral Bus Error Status Register	0x21	R/W	1014
EBC0_CFG	External Bus Configuration Register	0x23	R/W	1015
EBC0_CID	EBC ID Register	0x24	R	1016

To access one of these registers, software must first write the address offset into the EBC0_CFGADDR register. The target register can then be read or written through the EBC0_CFGDATA DCR address. The following PowerPC code illustrates this procedure by writing the EBC0_BnCR register and then reading back the written value.

```

li      r3,EBC0_B0CR           ! address offset
lis     r4,<config upper>       ! upper 16-bits of configuration data
ori     r4,r4,<config lower>    ! lower 16-bits of configuration data
mtdcr   EBC0_CFGADDR,r3        ! set offset addr
mtdcr   EBC0_CFGDATA,r4        ! write config data
mfdcr   r5,EBC0_CFGDATA        ! read back config data

```

Register bits marked as reserved must only be written to their defined reset value.

Preliminary User's Manual**30.5.1 EBC Address Register (EBC0_CFGADDR)**

Figure 30-21 shows the EBC0_CFGADDR bit definitions. This register is written by software to indirectly address the EBC registers.



Figure 30-21. EBC Address Register (EBC0_CFGADDR)

0:26		Reserved
27:31	DCRA	DCR Address Offset

30.5.2 EBC Data Register (EBC0_DATA)

This register is a data port written by software after it writes the EBC0_CFGADDR to read or write the other EBC registers.

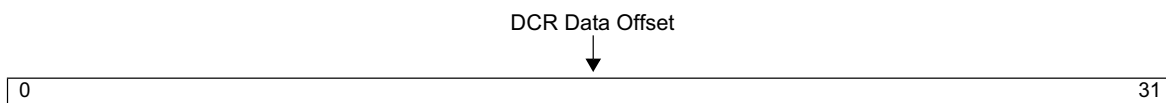


Figure 30-22. EBC Data Register (EBC0_CFGDATA)

0:31	DCRD	DCR Data Port
------	------	---------------

30.5.3 Peripheral Bank Configuration Registers (EBC0_B0CR-EBC0_B7CR)

EBC0_B0CR-EBC0_B7CR must be configured to enable memory in each respective bank. Boot ROM, if installed, must be attached to bank 0 and must use CS0.

If a boot ROM is present, Bank 0 is automatically configured using the core strapping pins. After system reset, the EBC0_B0CR[BAS] is loaded with a value of 0xFFE, EBC0_B0CR[BS] is loaded with a value of 0b001, and EBC0_B0CR[BU] is loaded with a value of 0b01. The EBC0_B0CR[BW] is loaded with a value of CCR_CPU_bROM_Size[0:2]. Software may reconfigure EBC0_B0CR later if necessary.

EBC0_BnCR[BAS] sets the base address for a peripheral device. The bank starting address must be a multiple of the bank size programmed in EBC0_BnCR[BS]. EBC0_BnCR[BAS] is compared to bits 0:11 of the address. If the address is within the range of an EBC0_BnCR[BAS] field, the associated bank is enabled for the transaction.

Multiple bank registers must not be programmed with the same base address or as overlapping banks. An attempt to use such overlapping banks is not recorded as an error and may cause contention on the external bus.

EBC0_BnCR[BS] sets the number of bytes which the bank may access, beginning with the base address set in EBC0_BnCR[BAS].

EBC0_BnCR[BU] protects banks of physical devices from read or write accesses.

When a write access is attempted to an address within the range of EBC0_BnCR[BAS], and the bank is designated as read-only, a protection error occurs. Also, when a read access is attempted to an address within the range of EBC0_BnCR[BAS] field, and the bank is designated as write-only, a protection error occurs. The address of the attempted access is logged in EBC0_BEAR and type of error is logged in EBC0_BESR.

EBC0_BnCR[BW] controls the width of region accesses. If devices are attached to the data bus, the EBC automatically packs read data and unpacks write data when a data transfer size mismatch exists.

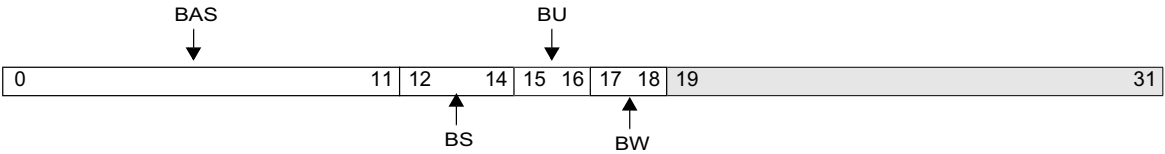


Figure 30-23. Peripheral Bank Configuration Registers (EBC0_B0CR-EBC0_B7CR)

0:11	BAS	Base Address Select	Specifies the bank starting address, which must be a multiple of the bank size.
12:14	BS	Bank Size 000 1 MB bank 001 2 MB bank 010 4 MB bank 011 8 MB bank 100 16 MB bank 101 32 MB bank 110 64 MB bank 111 128 MB bank	
15:16	BU	Bank Usage 00 Disabled 01 Read-only 10 Write-only 11 Read/Write	Specifies the type of accesses allowed for the bank. A protect error occurs if a write is attempted to a read-only bank or a read is attempted from a write-only bank.
17:18	BW	Bus Width 00 8-bit bus 01 16-bit bus 10 Reserved 11 32-bit bus	
19:31		Reserved	

Preliminary User's Manual**30.5.4 Peripheral Bank 0–7 Access Parameters (EBC0_B0AP-EBC0_B7AP)**

The EBC0_B0AP-EBC0_B7AP registers contain access parameters for their associated banks.

EBC0_BnAP[BME] controls bursting for external devices and all packing and unpacking operations. If EBC0_BnAP[BME] = 1, bursting is enabled. When bursting is enabled, EBC0_BnAP[CSN, OEN, and FWT] apply only to the first transfer, while EBC0_BnAP[BWT and WBN] apply during all remaining transfers of the burst.

EBC0_BnAP[TWT] specifies the number of wait states taken by each transfer to the bank. The number of cycles from address valid to the deassertion of $\overline{\text{PerCSn}}$ is $(1 + \text{EBC0_BnAP[TWT]})$, where $0 \leq \text{TWT} \leq 255$. This field is used for non-burst configured banks (EBC0_BnAP[BME] = 0).

EBC0_BnAP[FWT] specifies the number of wait states to be taken by the first access to the bank during a burst configured bank (EBC0_BnAP[BME] = 1). During a burst, the number of cycles from the first address valid to the second address is $(1 + \text{EBC0_BnAP[FWT]})$, where $0 \leq \text{FWT} \leq 31$.

EBC0_BnAP[BWT] specifies the number of wait states to be taken by accesses beyond the first during a burst transfer to a burst enabled bank (EBC0_BnAP[BME] = 1). On burst accesses, except for the last, the number of cycles from address valid to the next valid address on each burst access is $(1 + \text{EBC0_BnAP[BWT]})$, where $0 \leq \text{BWT} \leq 7$. On the last burst access, the number of cycles from address valid to the deassertion of $\overline{\text{PerCSn}}$ is $(1 + \text{EBC0_BnAP[BWT]})$, where $0 \leq \text{BWT} \leq 7$.

EBC0_BnAP[CSN] specifies the chip select turn on delay relative to the address. $\overline{\text{PerCSn}}$ may turn on coincident with the address or be delayed by 1, 2, or 3 PerClk cycles.

EBC0_BnAP[BCE] controls burst read from a peripheral device so extra reads do not occur. If a bank is enabled for burst and EBC0_BnAP[BCE] = 1, EBC can burst for fixed lengths bursts of the programmed value. If a bank is enabled for burst and EBC0_BnAP[BCE] = 0, EBC runs burst cycles on the external bus until the OPB_SeqAddr signal goes away and may do extra read operations.

EBC0_BnAP[BCT] specifies the number of transfers from the peripheral device for a decoded OPB sequential read when EBC0_BnAP[BCE] = 1. If the OPB sequential transfer is longer than the number of transfers programmed, the EBC has enough time to pack the data and continues transferring until the OPB sequential transfer is deasserted. If the OPB sequential transfer is shorter than the number of transfers programmed, the EBCO divides the transfer into several fixed length transfers. It then continues to do fixed length transfers until the OPB sequential transfer is deasserted. *Table 30-5* shows the actual number of bytes transferred, depending upon the size of the peripheral device and EBC0_BnAP[BCT].

Table 30-5. Fixed Length Burst Count Transfer Size

EBC0_BnAP[BCT10:11]	EBC0_BnCR[BW17:18]	Number of Bytes Read
0b00	8-bit	2
0b00 0b01	16-bit 8-bit	4
0b00 0b01 0b10	32-bit 16-bit 8-bit	8
0b01 0b10 0b11	32-bit 16-bit 8-bit	16
0b10 0b11	32-bit 16-bit	32
0b11	32-bit	64

Preliminary User's Manual

EBC0_BnAP[OEN], for read operations, specifies when the output enable signal, $\overline{\text{PerOE}}$, is asserted for read operations relative to the chip select signal. If EBC0_BnAP[OEN] = 0, $\overline{\text{PerOE}}$ is asserted coincident with the chip select. If EBC0_BnAP[OEN] = 1, 2, or 3, $\overline{\text{PerOE}}$ is delayed by 1, 2, or 3 PerClk cycles.

EBC0_BnAP[OEN], for write operations, specifies when the peripheral data bus is driven relative to the chip select signal when EBC0_CFC [OEO] = 0.

EBC0_BnAP[WBN], when EBC0_BnAP[BEM]=0, specifies when the write byte enables, $\overline{\text{PerWBE0:3}}$, are asserted relative to the chip select signal. If EBC0_BnAP[WBN] = 0, $\overline{\text{PerWBE0:3}}$ turns on coincident with the chip select. If EBC0_BnAP[WBN] = 1, 2, or 3, $\overline{\text{PerWBE0:3}}$ is delayed 1, 2, or 3 PerClk cycles from the chip select.

EBC0_BnAP[WBF], when EBC0_BnAP[BEM]=0, specifies when the write byte enables are deasserted relative to the deassertion of the chip select signal. If EBC0_BnAP[WBF] = 0, $\overline{\text{PerWBE0:3}}$ goes high coincident with the chip select signal. If EBC0_BnAP[WBF] = 1, 2, or 3, then $\overline{\text{PerWBE0:3}}$ turns off 1, 2, or 3 PerClk cycles before the turn-off of the chip select signal.

Programming Note: It is an error to set EBC0_BnAP[WBF] > EBC0_BnAP[BWT]; moreover, for device-paced transfers (EBC0_BnAP[RE] = 1), EBC0_BnAP[WBF] must be 0.

EBC0_BnAP[TH] specifies the number of PerClk cycles (0–7) that the peripheral bus is held idle after the deassertion of PerCSn. During these cycles, the address bus and data bus are active and PerR/W is valid. During the hold time, chip select, output enable, and write byte enables are inactive. If Ready Mode is used (EBC0_BnAP[RE] = 1) with sample on ready (EBC0_BnAP[SOR] = 1), EBC0_BnAP[TH] must be set to at least 1.

EBC0_BnAP[RE] controls the use of the PerReady input signal. If EBC0_BnAP[RE] = 0, the PerReady input is ignored and no additional wait states are inserted into bus transactions. If EBC0_BnAP[RE] = 1, the PerReady input is examined after the wait period expires; additional wait states are inserted if the PerReady input is 0. The maximum number of wait states in each new address is determined by the settings of EBC0_CFG[PTD, RTC]. If EBC0_CFG[PTD] = 1, the ready timeout function is disabled, and the PPC440GX waits indefinitely until PerReady = 1. If EBC0_CFG[PTD] = 0, the PPC440GX waits the number of cycle indicated by EBC0_CFG[RTC] for PerReady to become active. If PerReady does not become active in the allotted time, the address of the error is logged in EBC0_BEAR and the type of error is captured in EBC0_BESR.

EBC0_BnAP[SOR] controls the location of the data transfer cycle with respect to the PerReady input. If EBC0_BnAP[SOR] = 1 the data transfer occurs on the same PerClk edge that PerReady is sampled active, whereas if EBC0_BnAP[SOR] = 0 the data transfer occurs one cycle later.

EBC0_BnAP[BEM] controls whether $\overline{\text{PerWBE0:3}}$ is active during writes or for both reads and writes.

EBC0_BnAP[PEN] enables odd parity generation and checking.

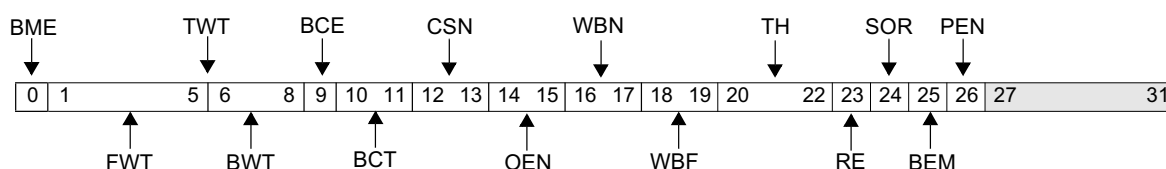


Figure 30-24. Peripheral Bank Access Parameters (EBC0_B0AP-EBC0_B7AP)

0	BME	Burst Mode Enable 0 Burst mode disabled 1 Burst mode enabled	
1:8	TWT	Transfer Wait 0 to 255 PerClk cycles	If bursting is disabled, BME=0, wait states on all non-burst transfers.

Preliminary User's Manual

1:5	FWT	First Wait 0 to 31 PerClk cycles	If bursting is enabled, BME=1, number of wait states on the first transfer of a burst.
6:8	BWT	Burst Wait 0 to 7 PerClk cycles	If bursting is enabled, BME=1, number of wait states on non-first transfers of a burst.
9	BCE	Fixed Length Burst Reads 0 Disabled 1 Enabled	If BCE=1, all burst read operations consist of exactly BCT transfers. When BCE=0, burst read operations may read more than the requested amount of data.
10:11	BCT	Fixed Length Burst Count 00 2 transfers 01 4 transfers 10 8 transfers 11 16 transfers	The amount of data transferred depends upon the programmed bank width, EBC0_BnCR[BW].
12:13	CSN	Chip Select On Timing 0 to 3 PerClk cycles.	<u>Number</u> of cycles from peripheral address driven to PerCSn low.
14:15	OEN	Output Enable On Timing 0 to 3 PerClk cycles	Number of cycles from $\overline{\text{PerCSn}}$ low to $\overline{\text{PerOE}}$ low.
16:17	WBN	Write Byte Enable On Timing 0 to 3 PerClk cycles	If BEM=0, number of cycles from $\overline{\text{PerCSn}}$ low to $\overline{\text{PerWBE0:3}}$ active.
18:19	WBF	Write Byte Enable Off Timing (If bursting and ready pin input are disabled) 0 to 3 PerClk cycles	If BEM=0 and RE=0, number of cycles $\overline{\text{PerWBE0:3}}$ becomes inactive prior to PerCSn inactive.
20:22	TH	Transfer Hold 0 to 7 PerClk cycles	Contains the number of hold cycles inserted at the end of a transfer. Hold cycles insert idle bus cycles between transfers to enable slow peripherals to remove data from the data bus before the next transfer begins.
23	RE	Ready Enable 0 PerReady input is disabled 1 PerReady input is enabled	Set for Device Paced Transfers
24	SOR	Sampling of Ready for Device Paced 0 Ready is asserted by the device one clock before data is ready on the external bus 1 Ready is asserted by the device on the same clock that data is ready on the external bus	
25	BEM	Byte Enable Mode 0 $\overline{\text{PerWBE0:3}}$ pins are only active on write cycles 1 $\overline{\text{PerWBE0:3}}$ pins are active for both read and write cycles	
26	PEN	Parity Enable 0 Disable Parity checking 1 Enable Parity checking	The EBCO implements odd parity.
27:31		Reserved	

30.5.5 Error Reporting

The EBC monitors four kinds of errors when performing read and write transfers. Of these four, bank protect and external bus errors are always checked, while timeout and read parity error checking must be enabled via DCR-mapped configuration registers. When an enabled error is detected by the EBC, the error condition is logged into the EBC0_BESR, the address is logged into EBC0_BEAR, and an interrupt is asserted to the system interrupt controller for one PerClk cycle. See *Peripheral Bus Error Status Register (EBC0_BESR)* on page 1014 for more detailed error handling information.

30.5.5.1 Protect Error

The Requested read or write operation violates the bank usage programmed in EBC0_BnCR[BU]. For example, in all cases of a write attempt to read-only bank, no external bus activity occurs.

30.5.5.2 External Bus Error

The PerErr input was sampled active during the data transfer cycle of a read or write operation. The associated data is read or written as usual.

30.5.5.3 Timeout Error

This error is possible during memory operations when both PerReady sampling is enabled (EBC0_BnAP[RE] = 1) and device paced timeouts are enabled (EBC0_CFG[PTD] = 0). Whenever the peripheral address bus changes, the EBC begins counting PerClk cycles. If the count reaches the value represented by EBC0_CFG[RTC], a timeout error occurs. Note that timeout errors are not possible during the peripheral portion of DMA transfers.

30.5.5.4 Parity Error

A parity error indicates that the parity calculated for the read data did not match the parity read. Parity generation and checking is enabled for memory operations by setting EBC0_BnAP[PEN] = 1 and for DMA peripheral transfers by programming DMA0_CRn[PCE] = 1.

30.5.5.5 Error Locking

EBC0_CFG[HFE] controls the locking of error information in the EBC0_BESR and EBC0_BEAR.

- If EBC0_CFG[HFE] = 0, the contents of the EBC0_BEAR and the EBC0_BESR reflect the most recent error.
- If EBC0_CFG[HFE] = 1, only the first error detected is reported in the EBC0_BESR and EBC0_BEAR. Subsequent errors are not permitted to overwrite the information detailing the first error. When software processes an error, it must clear all EBC0_BESR bits by writing 0 to allow another error to be logged.

30.5.5.6 Peripheral Bus Error Address Register (EBC0_BEAR)

The Peripheral Bus Error Address Register (EBC0_BEAR) is a 32-bit register containing the address of the access where a data bus error occurred. If EBC0_CFG[HFE] = 1, enabling error locking, and the EBC0_BEAR is not already locked, the contents of EBC0_BEAR are locked until the Peripheral Bus Error Status Register (EBC0_BESR) is cleared. The contents of the EBC0_BEAR are accessed indirectly through EBC0_CFGADDR and EBC0_CFGDATA registers using the **mfdcr** and **mtdcr** instructions.

Precise address capture in the EBC0_BEAR when a parity error occurs only applies to banks with at least one cycle of hold time (EBC0_BnAP[TH] > 0). This is because parity errors are not calculated until the cycle after the data was valid on the external bus, and the EBC0_BEAR is loaded with the address on the bus during that cycle. If the device timings do not include at least one hold cycle, the EBC0_BEAR may capture the next address in a burst or the address of the next transaction.

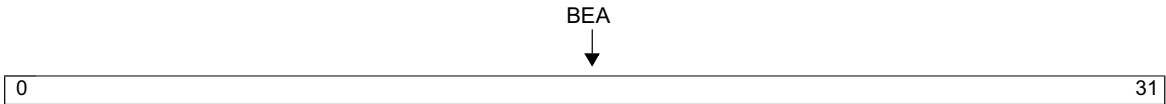


Figure 30-25. Peripheral Bus Error Address Register (EBC0_BEAR)

0:31	BEA	Bus Error Address
------	-----	-------------------

30.5.5.7 Peripheral Bus Error Status Register (EBC0_BESR)

The Peripheral Bus Error Status Register (EBC0_BESR) records the occurrence and type of errors for transactions attempted on behalf of the OPB master. The OPB master may be the DMA controller or the PLB-to-OPB Bridge on behalf of the CPU, PCIX, or any other master. The contents of EBC0_BESR are accessed indirectly through EBC0_CFGADDR and EBC0_CFGDATA registers using the **mfocr** and **mtocr** instructions.

It is possible to have both a parity error and a bus error during the same data transfer. If this occurs, the bus error is detected first and EBC0_BESR and EBC0_BEAR are updated. In the next cycle after the parity error is detected, and, if error locking is not enabled, the error is logged in EBC0_BESR (Refer to *Error Locking* on page 1012).

When an external bus input error is detected during an EBC read operation, the EBC0_BESR[EBE] bit is set. If the error is detected during a DMA read operation, the DMA controller logs the error in its status register and generates an interrupt, if interrupts are enabled. If the error is detected during a PLB master read operation, the error is recorded in the appropriate master's POB0_GESR0 field on the PLB-to-OPB Bridge, and the master is signalled that an error occurred.

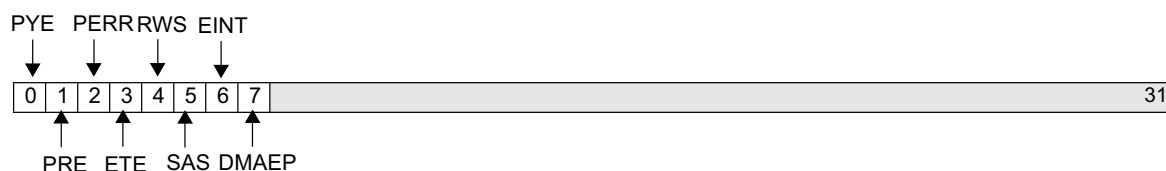


Figure 30-26. Bus Error Status Register (EBC0_BESR)

0	PYE	Parity Error Detected on Read 0 No Parity Error Detected 1 Parity Error Detected	O_ebco_opb_errAck is asserted.
1	PRE	Bank Protection Error 0 No Protection Error 1 Protection Error	O_ebco_opb_errAck is asserted for read or write.
2	PERR	PerErr 0 No PerErr Detected during EBCO transaction. 1 PerErr Detected during EBCO transaction.	If PerErr is detected at any other time during a read, errAck is not asserted. ErrAck is not asserted for write operations.
3	ETE	External Bus Timeout Error 0 No External Bus Timeout Error Detected 1 External Bus Timeout Error Detected	See PDT bit in Figure 30.5.6 on page 1015 for timeout enable information.
4	RWS	Read/write Error Status 0 Errant operation was a write operation 1 Errant operation was a read operation	
5	SAS	Sequential Address status 0 Sequential Address Not Active during error detection 1 Sequential Address Active during error detection	
6	EINT	Error Interrupt This bit is a logical "OR" of the four different errors that may be reported and indicates an interrupt has been asserted to the Interrupt Controller	
7	DMAEP	DMA External Peripheral Error 0 No error 1 Error	DMA external peripheral error detected during EBCO transaction.
8:31		Reserved	

Preliminary User's Manual**30.5.6 EBC Configuration Register (EBC0_CFG)**

The contents of EBC0_CFG are accessed indirectly through the EBC0_CFGADDR and EBC0_CFGDATA registers using the **mfdcr** and **mtddcr** instructions.

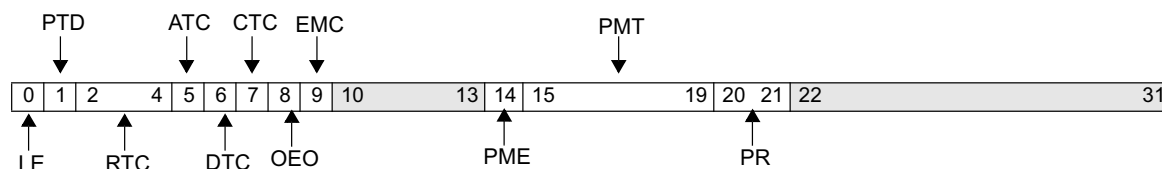


Figure 30-27. EBC Configuration Register (EBC0_CFG)

0	LE	Lock Error 0 Do not lock error 1 Lock error	This bit is used for error locking in the EBC. If this bit is set, then the EBC will latch the first error detected and save it with associated parameters and address. If this bit is not set, then the EBC will continue to latch new errors and associated parameters and addresses that will overwrite previous errors.
1	PTD	Device-Paced Time-out Disable 0 Enabled time-outs 1 Disable time-outs	This bit is valid only when EBC0_BnAP[RE]=1. If PTD=1, the EBC waits indefinitely for assertion of PerReady during device-paced accesses.
2:4	RTC	Ready Timeout Count 000 16 PerClk cycles 001 32 PerClk cycles 010 64 PerClk cycles 011 128 PerClk cycles 100 256 PerClk cycles 101 512 PerClk cycles 110 1024 PerClk cycles 111 2048 PerClk cycles	When PTD=0, the number of clock cycles from the assertion of PerAddr0:31 until a time-out error occurs.
5	ATC	Address Bus High Impedance Control 0 External address bus is high impedance when EBC is idle 1 External address bus drive previous value when EBC is idle and has ownership of the peripheral interface	Default after reset is ATC=1. For details on how the ATC affects driver enables, See <i>Driver Enables</i> on page 985.
6	DTC	Data Bus High Impedance Control 0 External data bus is high impedance when EBC is idle 1 External data bus drive previous write data value when EBC is idle and has ownership of the peripheral interface	Default after reset is DTC=1. For details on how the DTC affects driver enables, See <i>Driver Enables</i> on page 985.
7	CTC	Control Signal High Impedance Control 0 External bus Control Signals are high impedance when EBC is idle 1 External bus Control Signals are driven inactive and held when EBC is idle and has ownership of the peripheral interface	Default after reset is CTC=1. For details on how the CTC affects driver enables, See <i>Driver Enables</i> on page 985.
8	OEO	External Bus Override High Impedance Control 0 External Bus Output Enable Override Disabled 1 External Bus Output Enable Override Enabled	Default after reset is OEO=1. For details on how the OEO affects driver enables, See <i>Driver Enables</i> on page 985.

9	EMC	External Master High Impedance Control 0 High impedance all EBC outputs when HOLD_ACK = 1. 1 High impedance all EBC outputs when HOLD_ACK = 1 except PerCS0:7 are always driven.	Default after reset is EMC=1. For details, See <i>Driver Enables</i> on page 985.
10:13		Reserved	
14	PME	Power Management Enable 0 Disabled 1 Enabled	
15:19	PMT	Power Management Timer 0-31	The EBC makes a sleep request to the Clock and Power Management unit when PME=1 and the EBC has been idle for 32*PMT PerClk cycles.
20:21	PR	Pending Request Timer 00 -16 01 - 32 10 - 64 11 - 128	Number of PerClk cycles to wait for a retried OPB operation ¹ to return before relinquishing external bus ownership back to the external master.
22:31		Reserved	

1. A retried OPB operation occurs as follows:

The EBMI owns the external bus and there is an OPB requested EBCO transfer. The EBCO retries that request. When the EBMI releases the external bus, the EBCO owns it again. The EBCO has several previously retried OPB operations to complete; however, some of the retried OPB operations have not returned and, at the same time, no other OPB requests target the EBCO.

The EBCO enters the “Pending request” state. To detect this condition, the EBCO has a programmable 8-bit timer (PR). The PR loads with its programmed initialized value (16, 32, 64, 128) and begin decrementing each OPB cycle. When the PR reaches zero, the EBCO exits the “Pending Request” state and stops waiting for OPB commands to return.

30.5.7 EBC Core ID Register (EBC0_CID)

This register indicates the core ID. The core ID includes the technology, core number, and library revision number for this core. Writes to this register will be ignored.

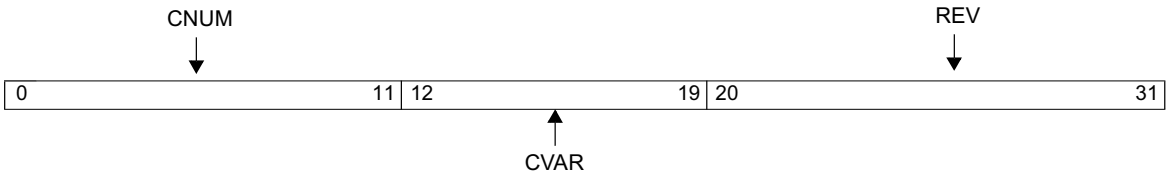


Figure 30-28. EBCO Core ID Register 0 (EBC0_CIDn)

0:11	CNUM	Core Number	'324
12:19	CVAR	Core Variation	'01'
20:31	REV	Revision Number	This value is the IBM SCCS revision number tracked and modified by the core designer.

Part V. Reference

Preliminary User's Manual

31. Instruction Set

Descriptions of the PPC440GX Embedded Processor instructions follow. Each description contains the following elements:

- Instruction names (mnemonic and full)
- Instruction syntax
- Instruction format diagram
- Pseudocode description
- Prose description
- Registers altered

Where appropriate, instruction descriptions list invalid instruction forms and exceptions, and provide programming notes.

Table 31-1 summarizes the PPC440GX Embedded Processor instruction set by category.

Table 31-1. Instruction Categories

Category	Sub-Category	Instruction Types
Integer	Integer Storage Access	load, store
	Integer Arithmetic	add, subtract, multiply, divide, negate
	Integer Logical	and, andc, or, orc, xor, nand, nor, xnor, extend sign, count leading zeros
	Integer Compare	compare, compare logical
	Integer Trap	trap
	Integer Rotate	rotate and insert, rotate and mask
	Integer Shift	shift left, shift right, shift right algebraic
	Integer Select	select operand
Branch		branch, branch conditional, branch to link, branch to count
Processor Control	Condition Register Logical	crand, crandc, cror, crorc, crnand, crnor, crxor, crxnor
	Register Management	move to/from SPR, move to/from DCR, move to/from MSR, write to external interrupt enable bit, move to/from CR
	System Linkage	system call, return from interrupt, return from critical interrupt, return from machine check interrupt
	Processor Synchronization	instruction synchronize
Storage Control	Cache Management	data allocate, data invalidate, data touch, data zero, data flush, data store, instruction invalidate, instruction touch
	TLB Management	read, write, search, synchronize
	Storage Synchronization	memory synchronize, memory barrier
Allocated	Allocated Arithmetic	multiply-accumulate, negative multiply-accumulate, multiply halfword
	Allocated Logical	detect left-most zero byte
	Allocated Cache Management	data congruence-class invalidate, instruction congruence-class invalidate
	Allocated Cache Debug	data read, instruction read

31.1 Instruction Set Portability

To support embedded real-time applications, the PPC440GX implements the defined instruction set of the Book-E Enhanced PowerPC Architecture, with the exception of those operations which are defined for 64-bit implementations only, and those which are defined as floating-point operations. Support for the floating-point operations is provided via the auxiliary processor interface, while the 64-bit operations are not supported at all. See *Instruction Classes* on page 168 for more information on the support for defined instructions within the PPC440GX Embedded Processor.

The PPC440GX also implements a number of instructions that are not part of PowerPC Book-E architecture, but are included as part of the PPC440GX Embedded Processor. Architecturally, they are considered allocated instructions, as they use opcodes which are within the allocated class of instructions, which the PowerPC Book-E architecture identifies as being available for implementation-dependent and/or application-specific purposes. However, all of the allocated instructions which are implemented within the PPC440GX are “standard” for IBM PowerPC 400 Series family of embedded controllers, and are not unique to the PPC440GX Embedded Processor.

The allocated instructions implemented within the PPC440GX Embedded Processor are divided into four sub-categories, and are shown in *Table 31-2*. Programs using these instructions may not be portable to other PowerPC Book-E implementations.

Table 31-2. Allocated Instructions

Arithmetic			Logical	Cache Management	Cache Debug
Multiply-Accumulate	Negative Multiply-Accumulate	Multiply Halfword			
macchw[o][.] macchws[o][.] macchwsu[o][.] macchwu[o][.] machhw[o][.] machhws[o][.] machhwsu[o][.] machhwu[o][.] maclhw[o][.] maclhws[o][.] maclhwsu[o][.] maclhwu[o][.]	nmacchw[o][.] nmacchws[o][.] nmachhw[o][.] nmachhws[o][.] nmaclhw[o][.] nmaclhws[o][.]	mulchw[.] mulchwu[.] mulhhw[.] mulhhwu[.] mullhw[.] mullhwu[.]	dImzb[.]	dccci iccci	dcread icread

31.2 Instruction Formats

For more detailed information about instruction formats, including a summary of instruction field usage and instruction format diagrams for the PPC440GX, see *Instruction Formats* on page 1783.

Instructions are four bytes long. Instruction addresses are always word-aligned.

Instruction bits 0 through 5 always contain the primary opcode. Many instructions have an extended opcode field as well. The remaining instruction bits contain additional fields. All instruction fields belong to one of the following categories:

- Defined

These instruction fields contain values, such as opcodes, that cannot be altered. The instruction format diagrams specify the values of defined fields.

- Variable

Preliminary User's Manual

These fields contain operands, such as general purpose register specifiers and immediate values, each of which may contain any one of a number of values. The instruction format diagrams specify the field names of variable fields.

- Reserved

Bits in a reserved field should be set to 0. In the instruction format diagrams, reserved fields are shaded.

If any bit in a defined field does not contain the specified value, the instruction is illegal and an Illegal Instruction exception type Program interrupt occurs. If any bit in a reserved field does not contain 0, the instruction form is invalid and its result is architecturally undefined. Unless otherwise noted, the PPC440GX will execute all invalid instruction forms without causing an Illegal Instruction exception.

31.3 Pseudocode

The pseudocode that appears in the instruction descriptions provides a semi-formal language for describing instruction operations.

The pseudocode uses the following notation:

+	Twos complement addition
%	Remainder of an integer division; $(33 \% 32) = 1$.
$\overset{u}{<}, \overset{u}{>}$	Unsigned comparison relations
(GPR(<i>r</i>))	The contents of GPR <i>r</i> , where $0 \leq r \leq 31$.
(RA 0)	The contents of the register RA or 0, if the RA field is 0.
(Rx)	The contents of a GPR, where <i>x</i> is A, B, S, or T
0bn	A binary number
0xn	A hexadecimal number
<, >	Signed comparison relations
=	Assignment
=, ≠	Equal, not equal relations
CEIL(<i>x</i>)	Least integer $\geq x$.
CIA	Current instruction address; the 32-bit address of the instruction being described by a sequence of pseudocode. This address is used to set the next instruction address (NIA). Does not correspond to any architected register.
DCR(DCRN)	A Device Control Register (DCR) specified by the DCRF field in an mfdcr or mtdcr instruction
EA	Effective address; the 32-bit address, derived by applying indexing or indirect addressing rules to the specified operand, that specifies an location in main storage.
EXTS(<i>x</i>)	The result of extending <i>x</i> on the left with sign bits.
FLD	An instruction or register field
FLD _b	A bit in a named instruction or register field
FLD _{b,b,...}	A list of bits, by number or name, in a named instruction or register field
FLD _{b:b}	A range of bits in a named instruction or register field
GPR(<i>r</i>)	General Purpose Register (GPR) <i>r</i> , where $0 \leq r \leq 31$.
GPRs	RA, RB, ...
MASK(MB,ME)	Mask having 1s in positions MB through ME (wrapping if MB > ME) and 0s elsewhere.

MS(addr, n)	The number of bytes represented by <i>n</i> at the location in main storage represented by <i>addr</i> .
NIA	Next instruction address; the 32-bit address of the next instruction to be executed. In pseudocode, a successful branch is indicated by assigning a value to NIA. For instructions that do not branch, the NIA is CIA +4.
PC	Program counter.
REG[FLD, FLD ...]	A list of fields in a named register
REG[FLD:FLD]	A range of fields in a named register
REG[FLD]	A field in a named register
REG _b	A bit in a named register
REG _{b₁,b₂,...}	A list of bits, by number or name, in a named register
REG _{b₁:b₂}	A range of bits in a named register
RESERVE	Reserve bit; indicates whether a process has reserved a block of storage.
ROTL((RS),n)	Rotate left; the contents of RS are shifted left the number of bits specified by <i>n</i> .
SPR(SPRN)	A Special Purpose Register (SPR) specified by the SPRF field in an mf spr or mt spr instruction
c _{0:3}	A four-bit object used to store condition results in compare instructions.
do	Do loop. “to” and “by” clauses specify incrementing an iteration variable; “while” and “until” clauses specify terminating conditions. Indenting indicates the scope of a loop.
if...then...else...	Conditional execution; if <i>condition</i> then <i>a</i> else <i>b</i> , where <i>a</i> and <i>b</i> represent one or more pseudocode statements. Indenting indicates the ranges of <i>a</i> and <i>b</i> . If <i>b</i> is null, the else does not appear.
instruction(EA)	An instruction operating on a data or instruction cache block associated with an EA.
leave	Leave innermost do loop or do loop specified in a leave statement.
n	A decimal number
ⁿ b	The bit or bit value <i>b</i> is replicated <i>n</i> times.
xx	Bit positions which are don't-cares.
	Concatenation
×	Multiplication
÷	Division yielding a quotient
⊕	Exclusive-OR (XOR) logical operator
–	Twos complement subtraction, unary minus
¬	NOT logical operator
^	AND logical operator
∨	OR logical operator

Preliminary User's Manual

31.3.1 Operator Precedence

Table 31-3 lists the pseudocode operators and their associativity in descending order of precedence:

Table 31-3. Operator Precedence

Operators	Associativity
REG _b , REG[FLD], function evaluation	Left to right
n_b	Right to left
\neg , $-$ (unary minus)	Right to left
\times , \div	Left to right
$+$, $-$	Left to right
\parallel	Left to right
$=$, \neq , $<$, $>$, $\overset{u}{<}$, $\overset{u}{>}$	Left to right
\wedge , \oplus	Left to right
\vee	Left to right
\leftarrow	None

31.4 Register Usage

Each instruction description lists the registers altered by the instruction. Some register changes are explicitly detailed in the instruction description (for example, the target register of a load instruction). Some instructions also change other registers, but the details of the changes are not included in the instruction descriptions. Common examples of these kinds of register changes include the Condition Register (CR) and the Integer Exception Register (XER). For discussion of the CR, see *Condition Register (CR)* on page 183. For discussion of the XER, see *Integer Exception Register (XER)* on page 187.

add
Add

31.5 Alphabetical Instruction Listing

The following pages list the instructions, both defined and allocated, which are implemented within the PPC440GX.

add	RT, RA, RB	OE=0, Rc=0
add.	RT, RA, RB	OE=0, Rc=1
addo	RT, RA, RB	OE=1, Rc=0
addo.	RT, RA, RB	OE=1, Rc=1

31	RT	RA	RB	OE	266	Rc
0	6	11	16	21 22		31

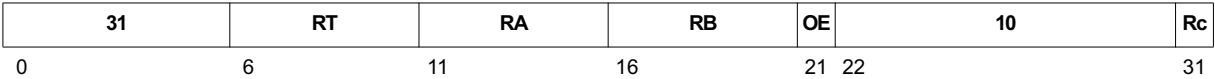
$(RT) \leftarrow (RA) + (RB)$

The sum of the contents of register RA and the contents of register RB is placed into register RT.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

addc	RT, RA, RB	OE=0, Rc=0
addc.	RT, RA, RB	OE=0, Rc=1
addco	RT, RA, RB	OE=1, Rc=0
addco.	RT, RA, RB	OE=1, Rc=1



```
(RT) ← (RA) + (RB)
if (RA) + (RB) > 232 - 1 then
  XER[CA] ← 1
else
  XER[CA] ← 0
```

The sum of the contents of register RA and register RB is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the add operation.

Registers Altered

- RT
- XER[CA]
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

adde

Add Extended

Preliminary User's Manual

adde	RT, RA, RB	OE=0, Rc=0
adde.	RT, RA, RB	OE=0, Rc=1
addeo	RT, RA, RB	OE=1, Rc=0
addeo.	RT, RA, RB	OE=1, Rc=1

31	RT	RA	RB	OE	138	Rc
0	6	11	16	21 22		31

```

(RT) ← (RA) + (RB) + XER[CA]
if (RA) + (RB) + XER[CA]  $\overset{u}{>} 2^{32} - 1$  then
    XER[CA] ← 1
else
    XER[CA] ← 0

```

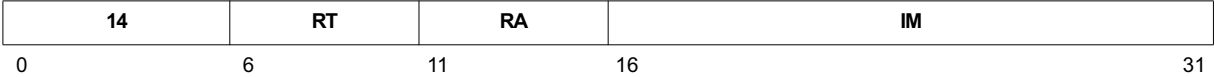
The sum of the contents of register RA, register RB, and XER[CA] is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the add operation.

Registers Altered

- RT
- XER[CA]
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

addi RT, RA, IM



$(RT) \leftarrow (RA|0) + \text{EXTS}(IM)$

If the RA field is 0, the IM field, sign-extended to 32 bits, is placed into register RT.

If the RA field is nonzero, the sum of the contents of register RA and the contents of the IM field, sign-extended to 32 bits, is placed into register RT.

Registers Altered

- RT

Programming Note

To place an immediate, sign-extended value into the GPR specified by RT, set RA = 0.

Table 31-4. Extended Mnemonics for addi

Mnemonic	Operands	Function	Other Registers Altered
la	RT, D(RA)	Load address (RA ≠ 0); D is an offset from a base address that is assumed to be (RA). (RT) ← (RA) + EXTS(D) Extended mnemonic for addi RT,RA,D	
li	RT, IM	Load immediate. (RT) ← EXTS(IM) Extended mnemonic for addi RT,0,IM	
subi	RT, RA, IM	Subtract EXTS(IM) from (RA 0). Place result in RT. Extended mnemonic for addi RT,RA,-IM	

addic

Add Immediate Carrying

Preliminary User's Manual**addic** RT, RA, IM

```

(RT) ← (RA) + EXTS(IM)
if (RA) + EXTS(IM)  $\geq 2^{32} - 1$  then
    XER[CA] ← 1
else
    XER[CA] ← 0

```

The sum of the contents of register RA and the contents of the IM field, sign-extended to 32 bits, is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the add operation.

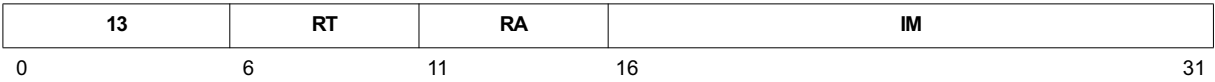
Registers Altered

- RT
- XER[CA]

Table 31-5. Extended Mnemonics for *addic*

Mnemonic	Operands	Function	Other Registers Altered
subic	RT, RA, IM	Subtract EXTS(IM) from (RA) Place result in RT; place carry-out in XER[CA]. <i>Extended mnemonic for</i> addic RT,RA,-IM	

addic. RT, RA, IM



```
(RT) ← (RA) + EXTS(IM)
if (RA) + EXTS(IM) > 232 - 1 then
    XER[CA] ← 1
else
    XER[CA] ← 0
```

The sum of the contents of register RA and the contents of the IM field, sign-extended to 32 bits, is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the add operation.

Registers Altered

- RT
- XER[CA]
- CR[CR0]

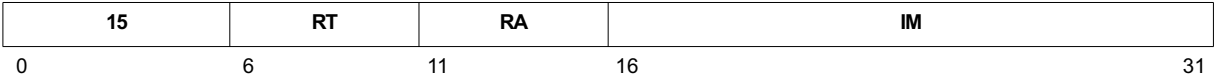
Programming Note

addic. is one of three instructions that implicitly update CR[CR0] without having an RC field. The other instructions are **andi.** and **andis..**

Table 31-6. Extended Mnemonics for **addic.**

Mnemonic	Operands	Function	Other Registers Altered
subic.	RT, RA, IM	Subtract EXTS(IM) from (RA). Place result in RT; place carry-out in XER[CA]. <i>Extended mnemonic for addic. RT,RA,-IM</i>	CR[CR0]

addis RT, RA, IM



$(RT) \leftarrow (RA|0) + (IM \parallel 16_0)$

If the RA field is 0, the IM field is concatenated on its right with sixteen 0-bits and placed into register RT.

If the RA field is nonzero, the contents of register RA are added to the contents of the extended IM field. The sum is stored into register RT.

Registers Altered

- RT

Programming Note

An **addi** instruction stores a sign-extended 16-bit value in a GPR. An **addis** instruction followed by an **ori** instruction stores an arbitrary 32-bit value in a GPR, as shown in the following example:

addis RT, 0, high 16 bits of value
ori RT, RT, low 16 bits of value

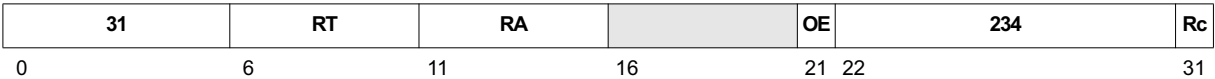
Table 31-7. Extended Mnemonics for addis

Mnemonic	Operands	Function	Other Registers Altered
lis	RT, IM	Load immediate shifted. $(RT) \leftarrow (IM \parallel 16_0)$ Extended mnemonic for addis RT,0,IM	
subis	RT, RA, IM	Subtract $(IM \parallel 16_0)$ from $(RA 0)$. Place result in RT. Extended mnemonic for addis RT,RA,-IM	

Preliminary User's Manual

Add to Minus One Extended

addme	RT, RA	OE=0, Rc=0
addme.	RT, RA	OE=0, Rc=1
addmeo	RT, RA	OE=1, Rc=0
addmeo.	RT, RA	OE=1, Rc=1



```
(RT) ← (RA) + XER[CA] + (−1)
if (RA) + XER[CA] + 0xFFFF FFFF ≥ 232 − 1 then
    XER[CA] ← 1
else
    XER[CA] ← 0
```

The sum of the contents of register RA, XER[CA], and −1 is placed into register RT.
XER[CA] is set to a value determined by the unsigned magnitude of the result of the add operation.

Registers Altered

- RT
- XER[CA]
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Invalid Instruction Forms

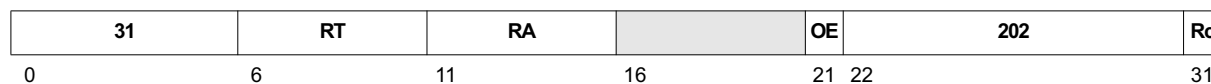
- Reserved fields

addze

Add to Zero Extended

Preliminary User's Manual

addze	RT, RA	OE=0, Rc=0
addze.	RT, RA	OE=0, Rc=1
addzeo	RT, RA	OE=1, Rc=0
addzeo.	RT, RA	OE=1, Rc=1



```

(RT) ← (RA) + XER[CA]
if (RA) + XER[CA]  $\geq 2^{32} - 1$  then
    XER[CA] ← 1
else
    XER[CA] ← 0

```

The sum of the contents of register RA and XER[CA] is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the add operation.

Registers Altered

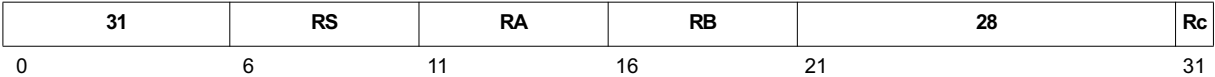
- RT
- XER[CA]
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Invalid Instruction Forms

- Reserved fields

Preliminary User's Manual

and RA, RS, RB Rc=0
and. RA, RS, RB Rc=1



$(RA) \leftarrow (RS) \wedge (RB)$

The contents of register RS are ANDed with the contents of register RB; the result is placed into register RA.

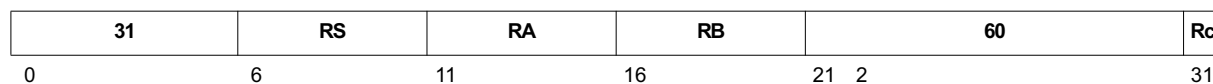
Registers Altered

- RA
- CR[CR0] if Rc contains 1

andc

AND with Complement

andc	RA,RS,RB	Rc=0
andc.	RA,RS,RB	Rc=1

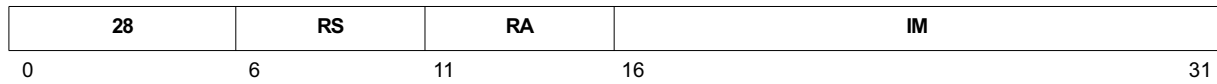


$$(RA) \leftarrow (RS) \wedge \neg(RB)$$

The contents of register RS are ANDed with the ones complement of the contents of register RB; the result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

Preliminary User's Manual**andi.** RA, RS, IM

$$(RA) \leftarrow (RS) \wedge ({}^{16}0 \parallel IM)$$

The IM field is extended to 32 bits by concatenating 16 0-bits on its left. The contents of register RS is ANDed with the extended IM field; the result is placed into register RA.

Registers Altered

- RA
- CR[CR0]

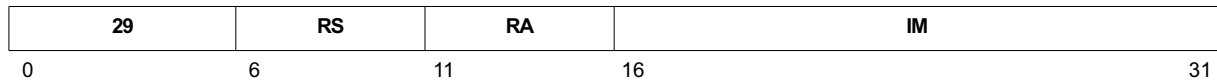
Programming Note

The **andi.** instruction can test whether any of the 16 least-significant bits in a GPR are 1-bits.

andi. is one of three instructions that implicitly update CR[CR0] without having an Rc field. The other instructions are **addic.** and **andis..**

andis.

AND Immediate Shifted

andis. RA, RS, IM

$$(RA) \leftarrow (RS) \wedge (IM \parallel 16_0)$$

The IM field is extended to 32 bits by concatenating 16 0-bits on its right. The contents of register RS are ANDed with the extended IM field; the result is placed into register RA.

Registers Altered

- RA
- CR[CR0]

Programming Note

The **andis.** instruction can test whether any of the 16 most-significant bits in a GPR are 1-bits.

andis. is one of three instructions that implicitly update CR[CR0] without having an Rc field. The other instructions are **addic.** and **andi..**

Preliminary User's Manual

b
Branch

b	target	AA=0, LK=0
ba	target	AA=1, LK=0
bl	target	AA=0, LK=1
bla	target	AA=1, LK=1



```
If AA = 1 then
    LI ← target6:29
    NIA ← EXTS(LI || 20)
else
    LI ← (target – CIA)6:29
    NIA ← CIA + EXTS(LI || 20)
if LK = 1 then
    (LR) ← CIA + 4
PC ← NIA
```

The next instruction address (NIA) is the effective address of the branch target. The NIA is formed by adding a displacement to a base address. The displacement is obtained by concatenating two 0-bits to the right of the LI field and sign-extending the result to 32 bits.

If the AA field contains 0, the base address is the address of the branch instruction, which is the current instruction address (CIA). If the AA field contains 1, the base address is 0.

Instruction execution resumes with the instruction at the NIA.

If the LK field contains 1, then (CIA + 4) is placed into the LR.

Registers Altered

- LR if LK contains 1

bc

Branch Conditional

Preliminary User's Manual

bc	BO, BI, target	AA=0, LK=0
bca	BO, BI, target	AA=1, LK=0
bcl	BO, BI, target	AA=0, LK=1
bcla	BO, BI, target	AA=1, LK=1

16	BO	BI	BD	AA	LK
0	6	11	16	30	31

```

if  $BO_2 = 0$  then
     $CTR \leftarrow CTR - 1$ 
if  $(BO_2 = 1 \vee ((CTR = 0) = BO_3)) \wedge (BO_0 = 1 \vee (CR_{BI} = BO_1))$  then
    if  $AA = 1$  then
         $BD \leftarrow target_{16:29}$ 
         $NIA \leftarrow EXTS(BD \parallel 2_0)$ 
    else
         $BD \leftarrow (target - CIA)_{16:29}$ 
         $NIA \leftarrow CIA + EXTS(BD \parallel 2_0)$ 
    else
         $NIA \leftarrow CIA + 4$ 
    if  $LK = 1$  then
         $(LR) \leftarrow CIA + 4$ 
     $PC \leftarrow NIA$ 

```

If BO_2 contains 0, the CTR decrements, and the decremented value is tested for 0 as part of the branch condition. In this case, BO_3 indicates whether the test for 0 must be true or false in order for the branch to be taken. If BO_2 contains 1, then the CTR is neither decremented nor tested as part of the branch condition.

If BO_0 contains 0, then the CR bit specified by the BI field is compared to BO_1 as part of the branch condition. If BO_0 contains 1, then the CR is not tested as part of the branch condition, and the BI field is ignored.

The next instruction address (NIA) is either the effective address of the branch target, or the address of the instruction after the branch, depending on whether the branch is taken or not. The branch target address is formed by adding a displacement to a base address. The displacement is obtained by concatenating two 0-bits to the right of the BD field and sign-extending the result to 32 bits.

If the AA field contains 0, the base address is the address of the branch instruction, which is the current instruction address (CIA). If the AA field contains 1, the base address is 0.

BO_4 affects branch prediction, a performance-improvement feature. See *Branch Prediction* on page 181 for a complete discussion.

Instruction execution resumes with the instruction at the NIA.

If the LK field contains 1, then $(CIA + 4)$ is placed into the LR.

Registers Altered

- CTR if BO_2 contains 0
- LR if LK contains 1

Table 31-8. Extended Mnemonics for bc, bca, bcl, bcla

Mnemonic	Operands	Function	Other Registers Altered
bdnz	target	Decrement CTR; branch if CTR \neq 0. <i>Extended mnemonic for</i> bc 16,0,target	
bdnza		<i>Extended mnemonic for</i> bca 16,0,target	
bdnzl		<i>Extended mnemonic for</i> bcl 16,0,target	(LR) \leftarrow CIA + 4.
bdnzla		<i>Extended mnemonic for</i> bcla 16,0,target	(LR) \leftarrow CIA + 4.
bdnzf	cr_bit, target	Decrement CTR. Branch if CTR \neq 0 AND CR _{cr_bit} = 0. <i>Extended mnemonic for</i> bc 0,cr_bit,target	
bdnzfa		<i>Extended mnemonic for</i> bca 0,cr_bit,target	
bdnzfl		<i>Extended mnemonic for</i> bcl 0,cr_bit,target	(LR) \leftarrow CIA + 4.
bdnzfla		<i>Extended mnemonic for</i> bcla 0,cr_bit,target	(LR) \leftarrow CIA + 4.
bdnzt	cr_bit, target	Decrement CTR. Branch if CTR \neq 0 AND CR _{cr_bit} = 1. <i>Extended mnemonic for</i> bc 8,cr_bit,target	
bdnzta		<i>Extended mnemonic for</i> bca 8,cr_bit,target	
bdnztl		<i>Extended mnemonic for</i> bcl 8,cr_bit,target	(LR) \leftarrow CIA + 4.
bdnztla		<i>Extended mnemonic for</i> bcla 8,cr_bit,target	(LR) \leftarrow CIA + 4.
bdz	target	Decrement CTR; branch if CTR = 0. <i>Extended mnemonic for</i> bc 18,0,target	
bdza		<i>Extended mnemonic for</i> bca 18,0,target	
bdzl		<i>Extended mnemonic for</i> bcl 18,0,target	(LR) \leftarrow CIA + 4.
bdzla		<i>Extended mnemonic for</i> bcla 18,0,target	(LR) \leftarrow CIA + 4.
bdzf	cr_bit, target	Decrement CTR Branch if CTR = 0 AND CR _{cr_bit} = 0. <i>Extended mnemonic for</i> bc 2,cr_bit,target	
bdzfa		<i>Extended mnemonic for</i> bca 2,cr_bit,target	
bdzfl		<i>Extended mnemonic for</i> bcl 2,cr_bit,target	(LR) \leftarrow CIA + 4.
bdzfla		<i>Extended mnemonic for</i> bcla 2,cr_bit,target	(LR) \leftarrow CIA + 4.

bc

Branch Conditional

Preliminary User's Manual

Table 31-8. Extended Mnemonics for bc, bca, bcl, bcla (Continued)

Mnemonic	Operands	Function	Other Registers Altered
bdzt	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR _{cr_bit} = 1. <i>Extended mnemonic for</i> bc 10,cr_bit,target	
bdzta		<i>Extended mnemonic for</i> bca 10,cr_bit,target	
bdztl		<i>Extended mnemonic for</i> bcl 10,cr_bit,target	(LR) ← CIA + 4.
bdztla		<i>Extended mnemonic for</i> bcla 10,cr_bit,target	(LR) ← CIA + 4.
beq	[cr_field,] target	Branch if equal. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bc 12,4*cr_field+2,target	
beqa		<i>Extended mnemonic for</i> bca 12,4*cr_field+2,target	
beql		<i>Extended mnemonic for</i> bcl 12,4*cr_field+2,target	(LR) ← CIA + 4.
beqla		<i>Extended mnemonic for</i> bcla 12,4*cr_field+2,target	(LR) ← CIA + 4.
bf	cr_bit, target	Branch if CR _{cr_bit} = 0. <i>Extended mnemonic for</i> bc 4,cr_bit,target	
bfa		<i>Extended mnemonic for</i> bca 4,cr_bit,target	
bfl		<i>Extended mnemonic for</i> bcl 4,cr_bit,target	LR
bfla		<i>Extended mnemonic for</i> bcla 4,cr_bit,target	LR
bge	[cr_field,] target	Branch if greater than or equal. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+0,target	
bgea		<i>Extended mnemonic for</i> bca 4,4*cr_field+0,target	
bgel		<i>Extended mnemonic for</i> bcl 4,4*cr_field+0,target	LR
bgela		<i>Extended mnemonic for</i> bcla 4,4*cr_field+0,target	LR

Table 31-8. Extended Mnemonics for bc, bca, bcl, bcla (Continued)

Mnemonic	Operands	Function	Other Registers Altered
bgt	[cr_field,] target	Branch if greater than. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bc 12,4*cr_field+1,target	
bgtb		<i>Extended mnemonic for</i> bca 12,4*cr_field+1,target	
bgtl		<i>Extended mnemonic for</i> bcl 12,4*cr_field+1,target	LR
bgtla		<i>Extended mnemonic for</i> bcla 12,4*cr_field+1,target	LR
ble	[cr_field,] target	Branch if less than or equal. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+1,target	
bleb		<i>Extended mnemonic for</i> bca 4,4*cr_field+1,target	
blel		<i>Extended mnemonic for</i> bcl 4,4*cr_field+1,target	LR
blela		<i>Extended mnemonic for</i> bcla 4,4*cr_field+1,target	LR
blt	[cr_field,] target	Branch if less than Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bc 12,4*cr_field+0,target	
bltb		<i>Extended mnemonic for</i> bca 12,4*cr_field+0,target	
bltl		<i>Extended mnemonic for</i> bcl 12,4*cr_field+0,target	(LR) ← CIA + 4.
bltla		<i>Extended mnemonic for</i> bcla 12,4*cr_field+0,target	(LR) ← CIA + 4.
bne	[cr_field,] target	Branch if not equal. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+2,target	
bneb		<i>Extended mnemonic for</i> bca 4,4*cr_field+2,target	
bnel		<i>Extended mnemonic for</i> bcl 4,4*cr_field+2,target	(LR) ← CIA + 4.
bnela		<i>Extended mnemonic for</i> bcla 4,4*cr_field+2,target	(LR) ← CIA + 4.

bc

Branch Conditional

Preliminary User's Manual

Table 31-8. Extended Mnemonics for bc, bca, bcl, bcla (Continued)

Mnemonic	Operands	Function	Other Registers Altered
bng	[cr_field,] target	Branch if not greater than. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+1,target	
nga		<i>Extended mnemonic for</i> bca 4,4*cr_field+1,target	
ngl		<i>Extended mnemonic for</i> bcl 4,4*cr_field+1,target	(LR) ← CIA + 4.
ngla		<i>Extended mnemonic for</i> bcla 4,4*cr_field+1,target	(LR) ← CIA + 4.
bnl	[cr_field,] target	Branch if not less than; use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+0,target	
bnla		<i>Extended mnemonic for</i> bca 4,4*cr_field+0,target	
bnll		<i>Extended mnemonic for</i> bcl 4,4*cr_field+0,target	(LR) ← CIA + 4.
bnlla		<i>Extended mnemonic for</i> bcla 4,4*cr_field+0,target	(LR) ← CIA + 4.
bns	[cr_field,] target	Branch if not summary overflow. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+3,target	
nsa		<i>Extended mnemonic for</i> bca 4,4*cr_field+3,target	
nsll		<i>Extended mnemonic for</i> bcl 4,4*cr_field+3,target	(LR) ← CIA + 4.
nslla		<i>Extended mnemonic for</i> bcla 4,4*cr_field+3,target	(LR) ← CIA + 4.
bnu	[cr_field,] target	Branch if not unordered. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+3,target	
nu		<i>Extended mnemonic for</i> bca 4,4*cr_field+3,target	
bnul		<i>Extended mnemonic for</i> bcl 4,4*cr_field+3,target	(LR) ← CIA + 4.
bnula		<i>Extended mnemonic for</i> bcla 4,4*cr_field+3,target	(LR) ← CIA + 4.

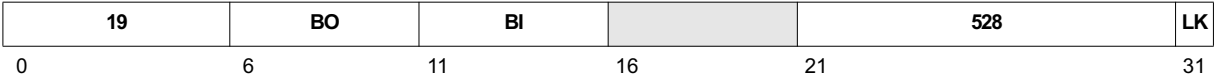
Table 31-8. Extended Mnemonics for bc, bca, bcl, bcla (Continued)

Mnemonic	Operands	Function	Other Registers Altered
bso	[cr_field,] target	Branch if summary overflow. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bc 12,4*cr_field+3,target	
boa		<i>Extended mnemonic for</i> bca 12,4*cr_field+3,target	
sol		<i>Extended mnemonic for</i> bcl 12,4*cr_field+3,target	(LR) ← CIA + 4.
sola		<i>Extended mnemonic for</i> bcla 12,4*cr_field+3,target	(LR) ← CIA + 4.
bt	cr_bit, target	Branch if CR _{cr_bit} = 1. <i>Extended mnemonic for</i> bc 12,cr_bit,target	
bta		<i>Extended mnemonic for</i> bca 12,cr_bit,target	
btl		<i>Extended mnemonic for</i> bcl 12,cr_bit,target	(LR) ← CIA + 4.
btla		<i>Extended mnemonic for</i> bcla 12,cr_bit,target	(LR) ← CIA + 4.
bun	[cr_field], target	Branch if unordered. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bc 12,4*cr_field+3,target	
buna		<i>Extended mnemonic for</i> bca 12,4*cr_field+3,target	
bunl		<i>Extended mnemonic for</i> bcl 12,4*cr_field+3,target	(LR) ← CIA + 4.
bunla		<i>Extended mnemonic for</i> bcla 12,4*cr_field+3,target	(LR) ← CIA + 4.

bcctr

Branch Conditional to Count Register

bcctr BO, BI LK=0
bcctrl BO, BI LK=1



```

if (BO0 = 1 ∨ (CRBI = BO1)) then
    NIA ← CTR0:29 || 20
else
    NIA ← CIA + 4
if LK = 1 then
    (LR) ← CIA + 4
PC ← NIA
    
```

If BO₀ contains 0, then the CR bit specified by the BI field is compared to BO₁ as part of the branch condition. If BO₀ contains 1, then the CR is not tested as part of the branch condition, and the BI field is ignored.

The next instruction address (NIA) is either the effective address of the branch target, or the address of the instruction after the branch, depending on whether the branch is taken or not. The branch target address is formed by concatenating two 0-bits to the right of the 30 most significant bits of the CTR.

BO₄ affects branch prediction, a performance-improvement feature. See *Branch Prediction* on page 181 for a complete discussion.

Instruction execution resumes with the instruction at the NIA.

If the LK field contains 1, then (CIA + 4) is placed into the LR.

Registers Altered

- LR if LK contains 1

Invalid Instruction Forms

- Reserved fields
- If BO₂ contains 0, the instruction form is invalid, and the result of the instruction (in particular, the branch target address and whether or not the branch is taken) is undefined. The architecture does not permit the combination of decrementing the CTR as part of the branch condition, together with using the CTR as the branch target address.

Table 31-9. Extended Mnemonics for bcctr, bcctrl

Mnemonic	Operands	Function	Other Registers Altered
bcctr		Branch unconditionally to address in CTR. <i>Extended mnemonic for</i> bcctr 20,0	
bcctrl		<i>Extended mnemonic for</i> bcctrl 20,0	(LR) ← CIA + 4.

Table 31-9. Extended Mnemonics for bcctr, bcctrl (Continued)

Mnemonic	Operands	Function	Other Registers Altered
beqctr	[cr_field]	Branch, if equal, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+2	
beqctrl		<i>Extended mnemonic for</i> bcctrl 12,4*cr_field+2	(LR) ← CIA + 4.
bfctr	cr_bit	Branch, if CR _{cr_bit} = 0, to address in CTR. <i>Extended mnemonic for</i> bcctr 4,cr_bit	
bfctrl		<i>Extended mnemonic for</i> bcctrl 4,cr_bit	(LR) ← CIA + 4.
bgectr	[cr_field]	Branch, if greater than or equal, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+0	
bgectrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+0	(LR) ← CIA + 4.
bgtctr	[cr_field]	Branch, if greater than, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+1	
bgtctrl		<i>Extended mnemonic for</i> bcctrl 12,4*cr_field+1	(LR) ← CIA + 4.
blectr	[cr_field]	Branch, if less than or equal, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+1	
blectrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+1	(LR) ← CIA + 4.
bltctr	[cr_field]	Branch, if less than, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+0	
bltctrl		<i>Extended mnemonic for</i> bcctrl 12,4*cr_field+0	(LR) ← CIA + 4.
bnctr	[cr_field]	Branch, if not equal, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+2	
bnctrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+2	(LR) ← CIA + 4.
bngctr	[cr_field]	Branch, if not greater than, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+1	
bngctrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+1	(LR) ← CIA + 4.

bcctr

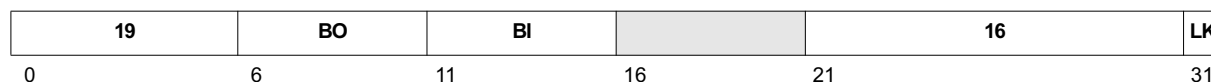
Branch Conditional to Count Register

Preliminary User's Manual

Table 31-9. Extended Mnemonics for bcctr, bcctrl (Continued)

Mnemonic	Operands	Function	Other Registers Altered
bnlctr	[cr_field]	Branch, if not less than, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+0	
bnlctrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+0	(LR) ← CIA + 4.
bnsctr	[cr_field]	Branch, if not summary overflow, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+3	
bnsctrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+3	(LR) ← CIA + 4.
bnuctr	[cr_field]	Branch, if not unordered, to address in CTR; use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+3	
bnuctrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+3	(LR) ← CIA + 4.
bsocctr	[cr_field]	Branch, if summary overflow, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+3	
bsocctrl		<i>Extended mnemonic for</i> bcctrl 12,4*cr_field+3	(LR) ← CIA + 4.
btctr	cr_bit	Branch if CR _{cr_bit} = 1 to address in CTR. <i>Extended mnemonic for</i> bcctr 12,cr_bit	
btctrl		<i>Extended mnemonic for</i> bcctrl 12,cr_bit	(LR) ← CIA + 4.
bunctr	[cr_field]	Branch if unordered to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+3	
bunctrl		<i>Extended mnemonic for</i> bcctrl 12,4*cr_field+3	(LR) ← CIA + 4.

bclr BO, BI LK=0
bclrl BO, BI LK=1



```

if BO2 = 0 then
    CTR ← CTR – 1
if (BO2 = 1 ∨ ((CTR = 0) = BO3)) ∧ (BO0 = 1 ∨ (CRBI = BO1)) then
    NIA ← LR0:29 || 20
else
    NIA ← CIA + 4
if LK = 1 then
    (LR) ← CIA + 4
PC ← NIA

```

If BO₂ contains 0, the CTR decrements, and the decremented value is tested for 0 as part of the branch condition. In this case, BO₃ indicates whether the test for 0 must be true or false in order for the branch to be taken. If BO₂ contains 1, then the CTR is neither decremented nor tested as part of the branch condition.

If BO₀ contains 0, then the CR bit specified by the BI field is compared to BO₁ as part of the branch condition. If BO₀ contains 1, then the CR is not tested as part of the branch condition, and the BI field is ignored.

The next instruction address (NIA) is either the effective address of the branch target, or the address of the instruction after the branch, depending on whether the branch is taken or not. The branch target address is formed by concatenating two 0-bits to the right of the 30 most significant bits of the LR.

BO₄ affects branch prediction, a performance-improvement feature. See *Branch Prediction* on page 181 for a complete discussion.

Instruction execution resumes with the instruction at the NIA.

If the LK field contains 1, then (CIA + 4) is placed into the LR.

Registers Altered

- CTR if BO₂ contains 0
- LR if LK contains 1

Invalid Instruction Forms

- Reserved fields

Table 31-10. Extended Mnemonics for *bclr*, *bclrl*

Mnemonic	Operands	Function	Other Registers Altered
blr		Branch unconditionally to address in LR. <i>Extended mnemonic for</i> bclr 20,0	
bclrl		<i>Extended mnemonic for</i> bclrl 20,0	(LR) ← CIA + 4.

bclr

Branch Conditional to Link Register

Preliminary User's Manual

Table 31-10. Extended Mnemonics for bclr, bclrl (Continued)

Mnemonic	Operands	Function	Other Registers Altered
bdnzlrl		Decrement CTR. Branch if CTR $\neq 0$ to address in LR. <i>Extended mnemonic for</i> bclr 16,0	
bdnzlrl		<i>Extended mnemonic for</i> bclrl 16,0	(LR) \leftarrow CIA + 4.
bdnzflr	cr_bit	Decrement CTR. Branch if CTR $\neq 0$ AND CR _{cr_bit} = 0 to address in LR. <i>Extended mnemonic for</i> bclr 0,cr_bit	
bdnzflrl		<i>Extended mnemonic for</i> bclrl 0,cr_bit	(LR) \leftarrow CIA + 4.
bdnztlr	cr_bit	Decrement CTR. Branch if CTR $\neq 0$ AND CR _{cr_bit} = 1 to address in LR. <i>Extended mnemonic for</i> bclr 8,cr_bit	
bdnztlrl		<i>Extended mnemonic for</i> bclrl 8,cr_bit	(LR) \leftarrow CIA + 4.
bdzlr		Decrement CTR. Branch if CTR = 0 to address in LR. <i>Extended mnemonic for</i> bclr 18,0	
bdzlr		<i>Extended mnemonic for</i> bclrl 18,0	(LR) \leftarrow CIA + 4.
bdzflr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR _{cr_bit} = 0 to address in LR. <i>Extended mnemonic for</i> bclr 2,cr_bit	
bdzflrl		<i>Extended mnemonic for</i> bclrl 2,cr_bit	(LR) \leftarrow CIA + 4.
bdztlr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR _{cr_bit} = 1 to address in LR. <i>Extended mnemonic for</i> bclr 10,cr_bit	
bdztlrl		<i>Extended mnemonic for</i> bclrl 10,cr_bit	(LR) \leftarrow CIA + 4.
beqlr	[cr_field]	Branch if equal to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+2	
beqlrl		<i>Extended mnemonic for</i> bclrl 12,4*cr_field+2	(LR) \leftarrow CIA + 4.
bflr	cr_bit	Branch if CR _{cr_bit} = 0 to address in LR. <i>Extended mnemonic for</i> bclr 4,cr_bit	
bflrl		<i>Extended mnemonic for</i> bclrl 4,cr_bit	(LR) \leftarrow CIA + 4.

Table 31-10. Extended Mnemonics for bclr, bclrl (Continued)

Mnemonic	Operands	Function	Other Registers Altered
bgebr	[cr_field]	Branch, if greater than or equal, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+0	
bgebrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+0	(LR) ← CIA + 4.
bgtbr	[cr_field]	Branch, if greater than, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+1	
bgtbrl		<i>Extended mnemonic for</i> bclrl 12,4*cr_field+1	(LR) ← CIA + 4.
blebr	[cr_field]	Branch, if less than or equal, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+1	
blebrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+1	(LR) ← CIA + 4.
bltbr	[cr_field]	Branch, if less than, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+0	
bltbrl		<i>Extended mnemonic for</i> bclrl 12,4*cr_field+0	(LR) ← CIA + 4.
bnlebr	[cr_field]	Branch, if not equal, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+2	
bnlebrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+2	(LR) ← CIA + 4.
bngebr	[cr_field]	Branch, if not greater than, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+1	
bngebrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+1	(LR) ← CIA + 4.
bnltbr	[cr_field]	Branch, if not less than, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+0	
bnltbrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+0	(LR) ← CIA + 4.
bnslr	[cr_field]	Branch if not summary overflow to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+3	
bnslrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+3	(LR) ← CIA + 4.

bclr

Branch Conditional to Link Register

Preliminary User's Manual

Table 31-10. Extended Mnemonics for bclr, bclrl (Continued)

Mnemonic	Operands	Function	Other Registers Altered
bnulr	[cr_field]	Branch if not unordered to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+3	
bnulrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+3	(LR) ← CIA + 4.
bsolr	[cr_field]	Branch if summary overflow to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+3	
bsolrl		<i>Extended mnemonic for</i> bclrl 12,4*cr_field+3	(LR) ← CIA + 4.
btlr	cr_bit	Branch if CR _{cr_bit} = 1 to address in LR. <i>Extended mnemonic for</i> bclr 12,cr_bit	
btlrl		<i>Extended mnemonic for</i> bclrl 12,cr_bit	(LR) ← CIA + 4.
bunlr	[cr_field]	Branch if unordered to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+3	
bunlrl		<i>Extended mnemonic for</i> bclrl 12,4*cr_field+3	(LR) ← CIA + 4.

Preliminary User's Manual

cmp BF, 0, RA, RB



```
c0:3 ← 40
if (RA) < (RB) then c0 ← 1
if (RA) > (RB) then c1 ← 1
if (RA) = (RB) then c2 ← 1
c3 ← XER[SO]
n ← BF
CR[CRn] ← c0:3
```

The contents of register RA are compared with the contents of register RB using a 32-bit signed compare.

The CR field specified by the BF field is updated to reflect the results of the compare and the value of XER[SO] is placed into the same CR field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR[CRn] where n is specified by the BF field

Invalid Instruction Forms

- Reserved fields

Programming Note

PowerPC Book-E architecture defines this instruction as **cmp BF,L,RA,RB**, where L selects operand size for 64-bit implementations. For all 32-bit implementations, L = 0 is required (L = 1 is an invalid form); hence for the PPC440GX, use of the extended mnemonic **cmpw BF,RA,RB** is recommended.

Table 31-11. Extended Mnemonics for cmp

Mnemonic	Operands	Function	Other Registers Altered
cmpw	[BF,] RA, RB	Compare Word; use CR0 if BF is omitted. <i>Extended mnemonic for</i> cmp BF,0,RA,RB	

cmpi

Compare Immediate

Preliminary User's Manual**cmpi**

BF, 0, RA, IM



```

c0:3 ← 40
if (RA) < EXTS(IM) then c0 ← 1
if (RA) > EXTS(IM) then c1 ← 1
if (RA) = EXTS(IM) then c2 ← 1
c3 ← XER[SO]
n ← BF
CR[CRn] ← c0:3

```

The IM field is sign-extended to 32 bits. The contents of register RA are compared with the extended IM field, using a 32-bit signed compare.

The CR field specified by the BF field is updated to reflect the results of the compare and the value of XER[SO] is placed into the same CR field.

Registers Altered

- CR[CR_n] where *n* is specified by the BF field

Invalid Instruction Forms

- Reserved fields

Programming Note

PowerPC Book-E Architecture defines this instruction as **cmpi BF,L,RA,IM**, where L selects operand size for 64-bit implementations. For all 32-bit implementations, L = 0 is required (L = 1 is an invalid form); hence for the PPC440GX, use of the extended mnemonic **cmpwi BF,RA,IM** is recommended.

Table 31-12. Extended Mnemonics for cmpi

Mnemonic	Operands	Function	Other Registers Altered
cmpwi	[BF,] RA, IM	Compare Word Immediate. Use CR0 if BF is omitted. <i>Extended mnemonic for cmpi BF,0,RA,IM</i>	

cmpl BF, 0, RA, RB



```
c0:3 ← 40
if (RA) <u (RB) then c0 ← 1
if (RA) >u (RB) then c1 ← 1
if (RA) = (RB) then c2 ← 1
c3 ← XER[SO]
n ← BF
CR[CRn] ← c0:3
```

The contents of register RA are compared with the contents of register RB, using a 32-bit unsigned compare.

The CR field specified by the BF field is updated to reflect the results of the compare and the value of XER[SO] is placed into the same CR field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR[CRn] where n is specified by the BF field

Invalid Instruction Forms

- Reserved fields

Programming Notes

PowerPC Book-E Architecture defines this instruction as **cmpl BF,L,RA,RB**, where L selects operand size for 64-bit implementations. For all 32-bit implementations, L = 0 is required (L = 1 is an invalid form); hence for PPC440GX, use of the extended mnemonic **cmplw BF,RA,RB** is recommended.

Table 31-13. Extended Mnemonics for *cmpl*

Mnemonic	Operands	Function	Other Registers Altered
cmplw	[BF,] RA, RB	Compare Logical Word. Use CR0 if BF is omitted. <i>Extended mnemonic for cmpl BF,0,RA,RB</i>	

cmpli

Compare Logical Immediate

Preliminary User's Manual**cmpli** BF, 0, RA, IM

```

c0:3 ← 40
if (RA) <u (160 || IM) then c0 ← 1
if (RA) >u (160 || IM) then c1 ← 1
if (RA) = (160 || IM) then c2 ← 1
c3 ← XER[SO]
n ← BF
CR[CRn] ← c0:3

```

The IM field is extended to 32 bits by concatenating 16 0-bits to its left. The contents of register RA are compared with IM using a 32-bit unsigned compare.

The CR field specified by the BF field is updated to reflect the results of the compare and the value of XER[SO] is placed into the same CR field.

Registers Altered

- CR[CR_n] where *n* is specified by the BF field

Invalid Instruction Forms

- Reserved fields

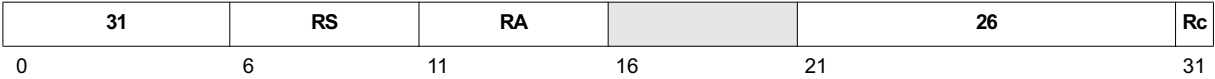
Programming Note

PowerPC Book-E Architecture defines this instruction as **cmpli BF,L,RA,IM**, where L selects operand size for 64-bit implementations. For all 32-bit implementations, L = 0 is required (L = 1 is an invalid form); hence for the PPC440GX, use of the extended mnemonic **cmplwi BF,RA,IM** is recommended.

Table 31-14. Extended Mnemonics for *cmpli*

Mnemonic	Operands	Function	Other Registers Changed
cmplwi	[BF,] RA, IM	Compare Logical Word Immediate. Use CR0 if BF is omitted. <i>Extended mnemonic for</i> cmpli BF,0,RA,IM	

cntlzw	RA, RS	Rc=0
cntlzw.	RA, RS	Rc=1



```
n ← 0
do while n < 32
  if (RS)n = 1 then leave
  n ← n + 1
(RA) ← n
```

The consecutive leading 0 bits in register RS are counted; the count is placed into register RA.

The count ranges from 0 through 32, inclusive.

Registers Altered

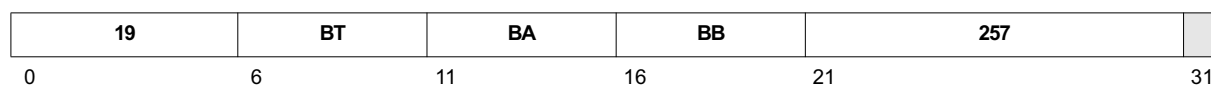
- RA
- CR[CR0] if Rc contains 1

Invalid Instruction Forms

- Reserved fields

crand

Condition Register AND

Preliminary User's Manual**crand** BT, BA, BB

$$CR_{BT} \leftarrow CR_{BA} \wedge CR_{BB}$$

The CR bit specified by the BA field is ANDed with the CR bit specified by the BB field; the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

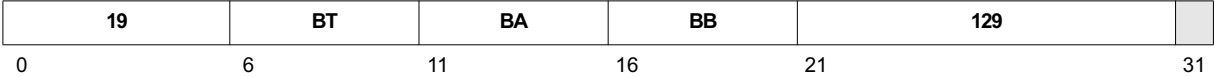
Registers Altered

- CR_{BT}

Invalid Instruction Forms

- Reserved fields

crandc BT, BA, BB



$CR_{BT} \leftarrow CR_{BA} \wedge \neg CR_{BB}$

The CR bit specified by the BA field is ANDed with the ones complement of the CR bit specified by the BB field; the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR_{BT}

Invalid Instruction Forms

- Reserved fields

creqv BT, BA, BB



$CR_{BT} \leftarrow \neg(CR_{BA} \oplus CR_{BB})$

The CR bit specified by the BA field is XORed with the CR bit specified by the BB field; the ones complement of the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR_{BT}

Invalid Instruction Forms

- Reserved fields

Table 31-15. Extended Mnemonics for creqv

Mnemonic	Operands	Function	Other Registers Altered
crset	bx	CR set. Extended mnemonic for creqv bx,bx,bx	

Preliminary User's Manual

crnand BT, BA, BB



$CR_{BT} \leftarrow \neg(CR_{BA} \wedge CR_{BB})$

The CR bit specified by the BA field is ANDed with the CR bit specified by the BB field; the ones complement of the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR_{BT}

Invalid Instruction Forms

- Reserved fields

crror BT, BA, BB



$CR_{BT} \leftarrow \neg(CR_{BA} \vee CR_{BB})$

The CR bit specified by the BA field is ORed with the CR bit specified by the BB field; the ones complement of the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR_{BT}

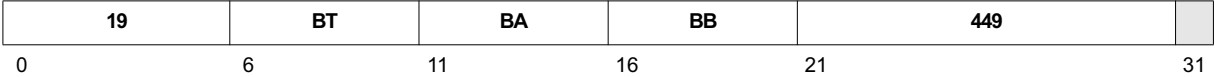
Invalid Instruction Forms

- Reserved fields

Table 31-16. Extended Mnemonics for crror

Mnemonic	Operands	Function	Other Registers Altered
crror	bx, by	CR not. Extended mnemonic for crror bx,by,by	

crr BT, BA, BB



$CR_{BT} \leftarrow CR_{BA} \vee CR_{BB}$

The CR bit specified by the BA field is ORed with the CR bit specified by the BB field; the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR_{BT}

Invalid Instruction Forms

- Reserved fields

Table 31-17. Extended Mnemonics for crr

Mnemonic	Operands	Function	Other Registers Altered
crrmove	bx, by	CR move. Extended mnemonic for crr bx,by,by	

crorc

Condition Register OR with Complement

Preliminary User's Manual**crorc** BT, BA, BB

$$CR_{BT} \leftarrow CR_{BA} \vee \neg CR_{BB}$$

The condition register (CR) bit specified by the BA field is ORed with the ones complement of the CR bit specified by the BB field; the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR_{BT}

Invalid Instruction Forms

- Reserved fields

crxorBT, BA, BB



$CR_{BT} \leftarrow CR_{BA} \oplus CR_{BB}$

The CR bit specified by the BA field is XORed with the CR bit specified by the BB field; the result is placed into the CR bit specified by the BT field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR_{BT}

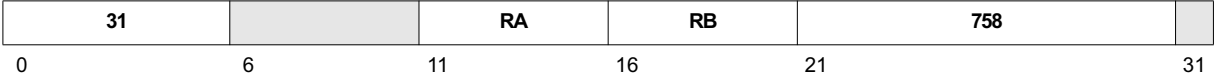
Invalid Instruction Forms

- Reserved fields

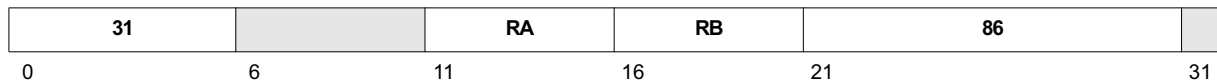
Table 31-18. Extended Mnemonics for crxor

Mnemonic	Operands	Function	Other Registers Altered
crclr	bx	Condition register clear. Extended mnemonic for crxor bx,bx,bx	

dcba RA, RB



dcba is treated as a no-op by the PPC440GX.

dcbf RA, RB

$$EA \leftarrow (RA|0) + (RB)$$

$$DCBF(EA)$$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the data block corresponding to the EA is in the data cache and marked as modified (stored into), the data block is copied back to main storage and then marked invalid in the data cache. If the data block is not marked as modified, it is simply marked invalid in the data cache. The operation is performed whether or not the memory page referenced by the EA is marked as cacheable.

If the data block at the EA is not in the data cache, no operation is performed.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Exceptions

This instruction is considered a “load” with respect to Data Storage exceptions. See *Data Storage Interrupt* on page 439 for more information.

This instruction is considered a “store” with respect to data address compare (DAC) Debug exceptions. See *Debug Interrupt* on page 448 for more information.

This instruction may cause a Cache Locking type of Data Storage exception. See *Data Storage Interrupt* on page 439 for more information.

dcbi

Data Cache Block Invalidate

Preliminary User's Manual**dcbi** RA, RB

EA ← (RA|0) + (RB)
 DCBI(EA)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the data block at the EA is in the data cache, the data block is marked invalid, regardless of whether or not the memory page referenced by the EA is marked as cacheable. If modified data existed in the data block prior to the operation of this instruction, that data is lost.

If the data block at the EA is not in the data cache, no operation is performed.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Notes

Execution of this instruction is privileged.

Exceptions

This instruction is considered a “store” with respect to Data Storage exceptions. See *Data Storage Interrupt* on page 439 for more information.

This instruction is considered a “store” with respect to data address compare (DAC) Debug exceptions. See *Debug Interrupt* on page 448 for more information.

Preliminary User's Manual

dcbst RA, RB



$EA \leftarrow (RA|0) + (RB)$
DCBST(EA)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0, and is the contents of register RA otherwise.

If the data block at the EA is in the data cache and marked as modified, the data block is copied back to main storage and marked as unmodified in the data cache.

If the data block at the EA is in the data cache, and is not marked as modified, or if the data block at the EA is not in the data cache, no operation is performed.

The operation specified by this instruction is performed whether or not the memory page referenced by the EA is marked as cacheable.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Exceptions

This instruction is considered a “load” with respect to Data Storage exceptions. See *Data Storage Interrupt* on page 439 for more information.

This instruction is considered a “store” with respect to data address compare (DAC) Debug exceptions. See *Debug Interrupt* on page 448 for more information.

dcbt

Data Cache Block Touch

Preliminary User's Manual**dcbt** RA, RB

EA ← (RA|0) + (RB)
 DCBT(EA)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

If the data block at the EA is not in the data cache and the memory page referenced by the EA is marked as cacheable, the block is read from main storage into the data cache.

If the data block at the EA is in the data cache, or if the memory page referenced by the EA is marked as caching inhibited, no operation is performed.

This instruction is not allowed to cause Data Storage interrupts nor Data TLB Error interrupts. If execution of the instruction causes either of these types of exception, then no operation is performed, and no interrupt occurs.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

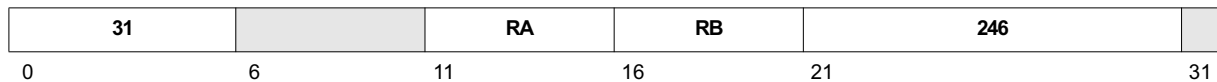
Programming Notes

The **dcbt** instruction allows a program to begin a cache block fetch from main storage before the program needs the data. The program can later load data from the cache into registers without incurring the latency of a cache miss.

Exceptions

This instruction is considered a “load” with respect to Data Storage exceptions. See *Data Storage Interrupt* on page 439 for more information.

This instruction is considered a “load” with respect to data address compare (DAC) Debug exceptions. See *Debug Interrupt* on page 448 for more information.

dcbtst RA, RB

$$EA \leftarrow (RA|0) + (RB)$$

$$DCBTST(EA)$$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the data block at the EA is not in the data cache and the memory page referenced by the EA address is marked as cacheable, the data block is loaded into the data cache.

If the data block at the EA is in the data cache, or if the memory page referenced by the EA is marked as caching inhibited, no operation is performed.

This instruction is not allowed to cause Data Storage interrupts nor Data TLB Error interrupts. If execution of the instruction causes either of these types of exception, then no operation is performed, and no interrupt occurs.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Notes

The **dcbtst** instruction allows a program to begin a cache block fetch from main storage before the program needs the data. The program can later store data from GPRs into the cache block, without incurring the latency of a cache miss.

Architecturally, **dcbtst** is intended to bring a cache block into the data cache in a manner which will permit future instructions to store to that block efficiently. For example, in an implementation which supports the “MESI” cache coherency protocol, the block would be brought into the cache in “Exclusive” mode, allowing the block to be stored to without having to broadcast any coherency operations on the system bus. However, since the PPC440GX does not support hardware-enforcement of multiprocessor coherency, there is no distinction between a block being brought in for a read or a write, and hence the implementation of the **dcbtst** instruction is identical to the implementation of the **dcbt** instruction.

Exceptions

This instruction is considered a “load” with respect to Data Storage exceptions. See *Data Storage Interrupt* on page 439 for more information.

This instruction is considered a “load” with respect to data address compare (DAC) Debug exceptions. See *Debug Interrupt* on page 448 for more information.

dcbz

Data Cache Block Set to Zero

Preliminary User's Manual**dcbz** RA, RB

$$EA \leftarrow (RA|0) + (RB)$$

$$DCBZ(EA)$$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the data block at the EA is in the data cache and the memory page referenced by the EA is marked as cacheable and non-write-through, the data in the cache block is set to 0 and marked as *dirty* (modified).

If the data block at the EA is not in the data cache and the memory page referenced by the EA is marked as cacheable and non-write-through, a cache block is established and set to 0 and marked as dirty. Note that nothing is read from main storage, as described in the programming note.

If the memory page referenced by the EA is marked as either write-through or as caching inhibited, an Alignment exception occurs.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Notes

Because **dcbz** can establish an address in the data cache without copying the contents of that address from main storage, the address established may be invalid with respect to the storage subsystem. A subsequent operation may cause the address to be copied back to main storage, for example, to make room for a new cache block; a Data Machine Check exception could occur under these circumstances.

If **dcbz** is attempted to an EA in a memory page which is marked as caching inhibited or as write-through, the software alignment exception handler should emulate the instruction by storing zeros to the block referenced by the EA. The store instructions in the emulation software will cause main storage to be updated (and possibly the cache, if the EA is in a page marked as write-through).

Exceptions

An alignment exception occurs if the EA is marked as caching inhibited or as write-through.

This instruction is considered a “store” with respect to Data Storage exceptions. See *Data Storage Interrupt* on page 439 for more information.

This instruction is considered a “store” with respect to data address compare (DAC) Debug exceptions. See *Debug Interrupt* on page 448 for more information.

dccci RA, RB**DCCCI**

This instruction flash invalidates the entire data cache array. The RA and RB operands are not used; previous implementations used these operands to calculate an effective address (EA) which specified the particular block or blocks to be invalidated. The instruction form (including the specification of RA and RB operands) is maintained for software and tool compatibility.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Notes

Execution of this instruction is privileged.

This instruction is intended for use in the power-on reset routine to invalidate the entire data cache array before caching is enabled.

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

dcread

Data Cache Read

Preliminary User's Manual**dcread** RT, RA, RB

$$EA \leftarrow (RA[0] + (RB))$$

$$INDEX \leftarrow EA_{17:26}$$

$$WORD \leftarrow EA_{27:29}$$

$$(RT) \leftarrow (\text{data cache data})[INDEX, WORD]$$

$$DCDBTRH \leftarrow (\text{data cache tag high})[INDEX]$$

$$DCDBTRL \leftarrow (\text{data cache tag low})[INDEX]$$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

EA_{17:26} selects a line of tag and data from the data cache. EA_{27:29} selects a word from the 8-word data portion of the selected cache line, and this word is read into register RT. EA_{30:31} must be 0b00; if not, the value placed in register RT is undefined.

The tag portion of the selected cache line is read into the DCDBTRH and DCDBTRL registers, as follows:

Register[bit(s)]	Tag Field	Name	
DCDBTRH[0:23]	TRA	Tag Real Address	Bits 0:23 of the lower 32 bits of the 36-bit real address associated with this cache line
DCDBTRH[24]	V	Valid	The valid indicator for the cache line (1 indicates valid)
DCDBTRH[25:27]		reserved	Reserved fields are read as 0s
DCDBTRH[28:31]	TERA	Tag Extended Real Address	Upper 4 bits of the 36-bit real address associated with this cache line
DCDBTRL[0:23]		reserved	Reserved fields are read as 0s
DCDBTRL[24:27]	D	Dirty Indicators	The "dirty" (modified) indicators for each of the four doublewords in the cache line
DCDBTRL[28]	U0	U0 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line
DCDBTRL[29]	U1	U1 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line
DCDBTRL[30]	U2	U2 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line
DCDBTRL[31]	U3	U3 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line

This instruction can be used by a debug tool to determine the contents of the data cache, without knowing the specific addresses of the lines which are currently contained within the cache.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT
- DCDBTRH
- DCDBTRL

Invalid Instruction Forms

- Reserved fields

Programming Note

Execution of this instruction is privileged.

The PPC440GX does not support the use of the **dcread** instruction when the data cache controller is still in the process of performing cache operations associated with previously executed instructions (such as line fills and line flushes). Also, the PPC440GX does not automatically synchronize context between a **dcread** instruction and the subsequent **mfsprr** instructions that read the results of the **dcread** instruction into GPRs. In order to guarantee that the **dcread** instruction operates correctly, and that the **mfsprr** instructions obtain the results of the **dcread** instruction, a sequence such as the following must be used:

```
msync                # ensure that all previous cache operations have completed
dcread    regT,regA,regB # read cache information; the contents of GPR A and GPR B are
                        # added and the result used to specify a cache line index to be read;
                        # the data word is moved into GPR T and the tag information is read
                        # into DCDBTRH and DCDBTRL
isync                # ensure dcread completes before attempting to read results
mfdcdbtrh    regD      # move high portion of tag into GPR D
mfdcdbtrl    regE      # move low portion of tag into GPR E
```

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

divw

Divide Word

Preliminary User's Manual

divw	RT, RA, RB	OE=0, Rc=0
divw.	RT, RA, RB	OE=0, Rc=1
divwo	RT, RA, RB	OE=1, Rc=0
divwo.	RT, RA, RB	OE=1, Rc=1



$$(RT) \leftarrow (RA) \div (RB)$$

The contents of register RA are divided by the contents of register RB. The quotient is placed into register RT.

Both the dividend and the divisor are interpreted as signed integers. The quotient is the unique signed integer that satisfies:

$$\text{dividend} = (\text{quotient} \times \text{divisor}) + \text{remainder}$$

where the remainder has the same sign as the dividend and its magnitude is less than that of the divisor.

If an attempt is made to perform $(0x8000\ 0000 \div -1)$ or $(n \div 0)$, the contents of register RT are undefined; if the Rc field also contains 1, the contents of CR[CR0]_{0:2} are undefined. Either invalid division operation sets XER[OV, SO] (and CR[CR0]₃ if Rc contains 1) to 1 if the OE field contains 1.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[OV, SO] if OE contains 1

Programming Note

The 32-bit remainder can be calculated using the following sequence of instructions:

divw	RT,RA,RB	# RT = quotient
mullw	RT,RT,RB	# RT = quotient × divisor
subf	RT,RT,RA	# RT = remainder

The sequence does not calculate correct results for the invalid divide operations.

divwu	RT, RA, RB	OE=0, Rc=0
divwu.	RT, RA, RB	OE=0, Rc=1
divwuo	RT, RA, RB	OE=1, Rc=0
divwuo.	RT, RA, RB	OE=1, Rc=1

31	RT	RA	RB	OE	459	Rc
0	6	11	16	21 22		31

$$(RT) \leftarrow (RA) \div (RB)$$

The contents of register RA are divided by the contents of register RB. The quotient is placed into register RT.

The dividend and the divisor are interpreted as unsigned integers. The quotient is the unique unsigned integer that satisfies:

$$\text{dividend} = (\text{quotient} \times \text{divisor}) + \text{remainder}$$

If an attempt is made to perform $(n \div 0)$, the contents of register RT are undefined; if the Rc also contains 1, the contents of CR[CR0]_{0:2} are also undefined. The invalid division operation also sets XER[OV, SO] (and CR[CR0]₃ if Rc contains 1) to 1 if the OE field contains 1.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[OV, SO] if OE contains 1

Programming Note

The 32-bit remainder can be calculated using the following sequence of instructions

divwu	RT,RA,RB	# RT = quotient
mullw	RT,RT,RB	# RT = quotient × divisor
subf	RT,RT,RA	# RT = remainder

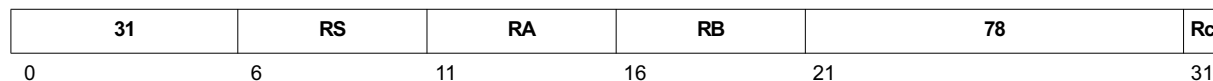
This sequence does not calculate the correct result if the divisor is 0.

dlimzb

Determine Leftmost Zero Byte

Preliminary User's Manual

dlimzb RA, RS, RB Rc=0
dlimzb. RA, RS, RB Rc=1



```

d ← (RS) || (RB)
i, x, y ← 0
do while (x < 8) ∧ (y = 0)
  x ← x + 1
  if di:i+7 = 0 then
    y ← 1
  else
    i ← i + 8
(RA) ← x
XER[TBC] ← x
if Rc = 1 then
  CR[CR0]3 ← XER[SO]
  if y = 1 then
    if x < 5 then
      CR[CR0]0:2 ← 0b010
    else
      CR[CR0]0:2 ← 0b100
  else
    CR[CR0]0:2 ← 0b001

```

The contents of registers RS and RB are concatenated to form an 8-byte operand. The operand is searched for the leftmost byte in which each bit is 0 (a 0-byte).

Bytes in the operand are numbered from left to right starting with 1. If a 0-byte is found, its byte number is placed into XER[TBC] and register RA. Otherwise, the number 8 is placed into XER[TBC] and register RA.

If the Rc field contains 1, XER[SO] is copied to CR[CR0]₃ and CR[CR0]_{0:2} are updated as follows:

- If no 0-byte is found, CR[CR0]_{0:2} is set to 0b001.
- If the leftmost 0-byte is in the first 4 bytes (in the RS register), CR[CR0]_{0:2} is set to 0b010.
- If the leftmost 0-byte is in the last 4 bytes (in the RB register), CR[CR0]_{0:2} is set to 0b100.

Registers Altered

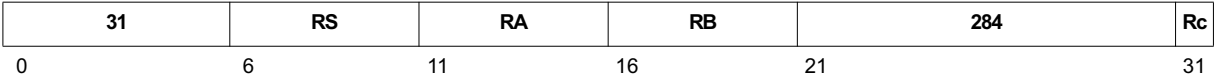
- XER[TBC]
- RA
- CR[CR0] if Rc contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

Preliminary User's Manual

eqv RA, RS, RB Rc=0
eqv. RA, RS, RB Rc=1



$(RA) \leftarrow \neg((RS) \oplus (RB))$

The contents of register RS are XORed with the contents of register RB; the ones complement of the result is placed into register RA.

Registers Altered

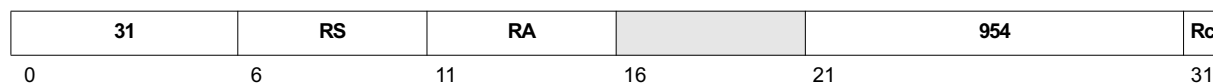
- RA
- CR[CR0] if Rc contains 1

extsb

Extend Sign Byte

Preliminary User's Manual

extsb	RA, RS	Rc=0
extsb.	RA, RS	Rc=1



$$(RA) \leftarrow \text{EXTS}(RS)_{24:31}$$

The least significant byte of register RS is sign-extended to 32 bits by replicating bit 24 of the register into bits 0 through 23 of the result. The result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

Invalid Instruction Forms

- Reserved fields

Preliminary User's Manual

extsh RA, RS Rc=0
extsh. RA, RS Rc=1



$(RA) \leftarrow \text{EXTS}(RS)_{16:31}$

The least significant halfword of register RS is sign-extended to 32 bits by replicating bit 16 of the register into bits 0 through 15 of the result. The result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

Invalid Instruction Forms

- Reserved fields

icbi

Instruction Cache Block Invalidate

Preliminary User's Manual**icbi** RA, RB

EA ← (RA[0] + (RB))
ICBI(EA)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the instruction block at the EA is in the instruction cache, the cache block is marked invalid.

If the instruction block at the EA is not in the instruction cache, no additional operation is performed.

The operation specified by this instruction is performed whether or not the EA is marked as cacheable.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Note

Instruction cache management instructions use MSR[DS], not MSR[IS], as part of the virtual address. Also, the instruction cache on the PPC440GX Embedded Processor is “virtually-tagged”, which means that the EA is converted to a virtual address (VA), and the VA is compared against the cache tag field. See *Instruction Cache Synonyms* on page 210 for more information on the ramifications of virtual tagging on software.

Exceptions

Instruction Storage interrupts and Instruction TLB Error interrupts are associated with exceptions which occur during instruction *fetching*, not during instruction *execution*. *Execution* of instruction cache management instructions may cause Data Storage or Data TLB Error exceptions.

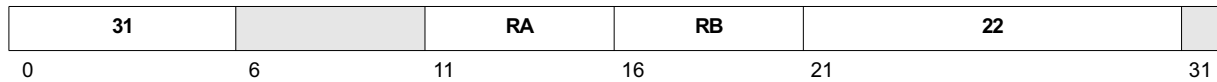
This instruction is considered a “load” with respect to Data Storage exceptions. See *Data Storage Interrupt* on page 439 for more information.

This instruction is considered a “load” with respect to data address compare (DAC) Debug exceptions. See *Debug Interrupt* on page 448 for more information.

This instruction may cause a Cache Locking type of Data Storage exception. See *Data Storage Interrupt* on page 439 for more information.

icbt

RA, RB



EA ← (RA|0) + (RB)
ICBT(EA)

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the instruction block at the EA is not in the instruction cache and the memory page referenced by the EA is marked as cacheable, the instruction block is fetched into the instruction cache.

If the instruction block at the EA is in the instruction cache, or if the memory page referenced by the EA is marked as caching inhibited, no operation is performed.

If the memory page referenced by the EA is marked as “no-execute” for the current operating mode (user mode or supervisor mode, as specified by MSR[PR]), no operation is performed.

This instruction is not allowed to cause Data Storage interrupts nor Data TLB Error interrupts. If execution of the instruction causes either of these types of exception, then no operation is performed, and no interrupt occurs.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Notes

This instruction allows a program to begin a cache block fetch from main storage before the program needs the instruction. The program can later branch to the instruction address and fetch the instruction from the cache without incurring the latency of a cache miss.

Instruction cache management instructions use MSR[DS], not MSR[IS], as part of the virtual address. Also, the instruction cache on the PPC440GX Embedded Processor is “virtually-tagged”, which means that the EA is converted to a virtual address (VA), and the VA is compared against the cache tag field. See *Instruction Cache Synonyms* on page 210 for more information on the ramifications of virtual tagging on software.

Exceptions

Instruction Storage interrupts and Instruction TLB Error interrupts are associated with exceptions which occur during instruction *fetching*, not during instruction *execution*. *Execution* of instruction cache management instructions may cause Data Storage or Data TLB Error exceptions, but are not allowed to cause the associated interrupt. Instead, if such an exception occurs, then no operation is performed.

This instruction is considered a “load” with respect to Data Storage exceptions. See *Data Storage Interrupt* on page 439 for more information.

icbt

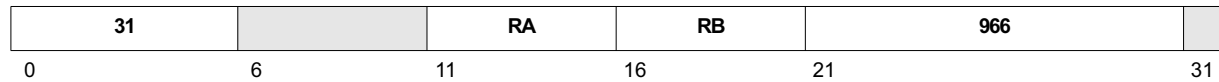
Instruction Cache Block Touch

Preliminary User's Manual

This instruction is considered a “load” with respect to data address compare (DAC) Debug exceptions. See *Debug Interrupt* on page 448 for more information.

iccci

RA, RB

**ICCCI**

This instruction flash invalidates the entire instruction cache array. The RA and RB operands are not used; previous implementations used these operands to calculate an effective address (EA) which specified the particular block or blocks to be invalidated. The instruction form (including the specification of RA and RB operands) is maintained for software and tool compatibility.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Notes

Execution of this instruction is privileged.

This instruction is intended for use in the power-on reset routine to invalidate the entire instruction cache array before caching is enabled.

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

icread

Instruction Cache Read

Preliminary User's Manual**icread** RA, RB
 $EA \leftarrow (RA[0] + (RB))$
 $INDEX \leftarrow EA_{17:26}$
 $WORD \leftarrow EA_{27:29}$
 $ICDBDR \leftarrow (\text{instruction cache data})[INDEX, WORD]$
 $ICDBTRH \leftarrow (\text{instruction cache tag high})[INDEX]$
 $ICDBTRL \leftarrow (\text{instruction cache tag low})[INDEX]$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

$EA_{17:26}$ selects a line of tag and data (instructions) from the instruction cache. $EA_{27:29}$ selects a 32-bit instruction from the 8-instruction data portion of the selected cache line, and this instruction is read into the ICDBDR. $EA_{30:31}$ are ignored, as are $EA_{0:16}$.

The tag portion of the selected cache line is read into the ICDBTRH and ICDBTRL registers, as follows:

Register[bit(s)]	Tag Field	Name	
ICDBTRH[0:23]	TEA	Tag Effective Address	Bits 0:23 of the 32-bit effective address associated with this cache line
ICDBTRH[24]	V	Valid	The valid indicator for the cache line (1 indicates valid)
ICDBTRH[25:31]		reserved	Reserved fields are read as 0s
ICDBTRL[0:21]		reserved	Reserved fields are read as 0s
ICDBTRL[22]	TS	Translation Space	The address space portion of the virtual address associated with this cache line.
ICDBTRL[23]	TD	Translation ID (TID) Disable	TID Disable field for the memory page associated with this cache line
ICDBTRL[24:31]	TID	Translation ID	TID field portion of the virtual address associated with this cache line

The instruction cache on PPC440GX Embedded Processor is “virtually-tagged”, which means that the tag field contains the virtual address, which consists of the TEA, TS, and TID fields. See *Memory Management* on page 235 for more information on the function of the TS, TD, and TID fields.

This instruction can be used by a debug tool to determine the contents of the instruction cache, without knowing the specific addresses of the lines which are currently contained within the cache.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- ICDBDR
- ICDBTRH
- ICDBTRL

Invalid Instruction Forms

- Reserved fields

Programming Note

Execution of this instruction is privileged.

The PPC440GX Embedded Processor does not automatically synchronize context between an **icread** instruction and the subsequent **mfsprr** instructions which read the results of the **icread** instruction into GPRs. In order to guarantee that the **mfsprr** instructions obtain the results of the **icread** instruction, a sequence such as the following must be used:

icread	regA,regB	# read cache information (the contents of GPR A and GPR B are
		# added and the result used to specify a cache line index to be read)
isync		# ensure icread completes before attempting to read results
mficbdr	regC	# move instruction information into GPR C
mficbtrh	regD	# move high portion of tag into GPR D
mficbtrl	regE	# move low portion of tag into GPR E

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

isel

Integer Select

Preliminary User's Manual**isel** RT, RA, RB, CRb

31	RT	RA	RB	CRb	15	
0	6	11	16	21	26	31

if CR[CRb] = 1 then

(RT) \leftarrow (RA[0])

else

(RT) \leftarrow (RB)

If CR[CRb] = 0, register RT is written with the contents of register RB.

If CR[CRb] = 1 and RA \neq 0, register RT is written with the contents of register RA.

If CR[CRb] = 1 and RA = 0, register RT is written with 0.

Registers Altered

- RT

isync

The **isync** instruction is a context synchronizing instruction.

isync provides an ordering function for the effects of all instructions executed by the processor. Executing **isync** insures that all instructions preceding the **isync** instruction execute before **isync** completes, except that storage accesses caused by those instructions need not have completed. Furthermore, all instructions preceding the **isync** are guaranteed to be unaffected by any context changes initiated by instructions after the **isync**.

No subsequent instructions are initiated by the processor until **isync** completes. Finally, execution of **isync** causes the processor to discard any prefetched instructions (prefetched from the cache, not instructions that are in the cache or on their way into the cache), with the effect that subsequent instructions are fetched and executed in the context established by the instructions preceding **isync**.

isync causes any caching inhibited instruction fetches from memory to be aborted and any data associated with them to be discarded. Cacheable instruction fetches from memory are not aborted however, as these should be handled by the **icbi** instructions which must precede the **isync** if software wishes to invalidate any cached instructions.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Note

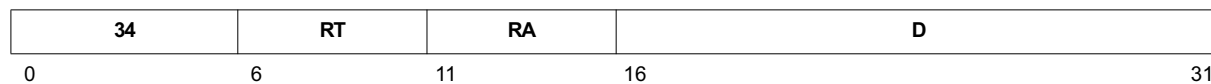
See the discussion of context synchronizing instructions in *Synchronization* on page 196.

The following code example illustrates the necessary steps for self-modifying code. This example assumes that `addr1` is both data and instruction cacheable.

<pre> stw regN, addr1 dcbst addr1 msync icbi addr1 of being fetched into the cache) msync isync </pre>	<pre> # data in regN is to become an instruction at addr1 # forces data from the data cache to memory # wait until the data actually reaches the memory # invalidate the instruction if it is in the cache (or in the # process # wait until the icbi completes # discard and refetch any instructions (including # possibly the instruction at addr1) which may have # already been fetched from the cache and be in the # pipeline after the isync </pre>
--	---

lbz

Load Byte and Zero

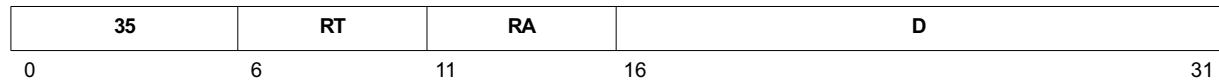
Preliminary User's Manual**lbz** RT, D(RA) $EA \leftarrow (RA|0) + \text{EXTS}(D)$ $(RT) \leftarrow {}^{24}_0 \parallel \text{MS}(EA, 1)$

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The byte at the EA is extended to 32 bits by concatenating 24 0-bits to its left. The result is placed into register RT.

Registers Altered

- RT

lbzu RT, D(RA)

$$EA \leftarrow (RA|0) + \text{EXTS}(D)$$

$$(RA) \leftarrow EA$$

$$(RT) \leftarrow {}^{24}0 \parallel \text{MS}(EA,1)$$

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The byte at the EA is extended to 32 bits by concatenating 24 0-bits to its left. The result is placed into register RT.

Registers Altered

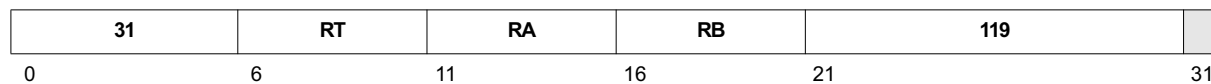
- RA
- RT

Invalid Instruction Forms

- RA = RT
- RA = 0

lbzux

Load Byte and Zero with Update Indexed

Preliminary User's Manual**lbzux** RT, RA, RB

$$EA \leftarrow (RA|0) + (RB)$$

$$(RA) \leftarrow EA$$

$$(RT) \leftarrow {}^{24}0 \parallel MS(EA,1)$$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The byte at the EA is extended to 32 bits by concatenating 24 0-bits to its left. The result is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RA
- RT

Invalid Instruction Forms

- Reserved fields
- RA = RT
- RA = 0

lbzx RT,RA, RB



$EA \leftarrow (RA|0) + (RB)$
 $(RT) \leftarrow {}^{24}_0 || MS(EA,1)$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The byte at the EA is extended to 32 bits by concatenating 24 0-bits to its left. The result is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

lha

Load Halfword Algebraic

Preliminary User's Manual**lha** RT, D(RA) $EA \leftarrow (RA|0) + \text{EXTS}(D)$ $(RT) \leftarrow \text{EXTS}(\text{MS}(EA, 2))$

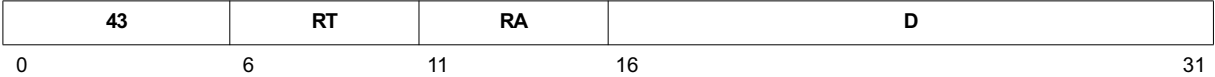
An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The halfword at the EA is sign-extended to 32 bits and placed into register RT.

Registers Altered

- RT

lhau RT, D(RA)



EA ← (RA|0) + EXTS(D)
(RA) ← EA
(RT) ← EXTS(MS(EA,2))

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The halfword at the EA is sign-extended to 32 bits and placed into register RT.

Registers Altered

- RA
- RT

Invalid Instruction Forms

- RA = RT
- RA = 0

lhaux

Load Halfword Algebraic with Update Indexed

lhaux RT, RA, RB



$EA \leftarrow (RA|0) + (RB)$
 $(RA) \leftarrow EA$
 $(RT) \leftarrow EXTS(MS(EA,2))$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The halfword at the EA is sign-extended to 32 bits and placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RA
- RT

Invalid Instruction Forms

- Reserved fields
- RA = RT
- RA = 0

lhax RT, RA, RB



$EA \leftarrow (RA|0) + (RB)$
 $(RT) \leftarrow EXTS(MS(EA,2))$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The halfword at the EA is sign-extended to 32 bits and placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

lhbrx

Load Halfword Byte-Reverse Indexed

Preliminary User's Manual**lhbrx** RT, RA, RB

$$EA \leftarrow (RA \ll 0) + (RB)$$

$$(RT) \leftarrow {}^{16}0 \parallel \text{BYTE_REVERSE}(\text{MS}(EA, 2))$$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The halfword at the EA is byte-reversed from the default byte ordering for the memory page referenced by the EA. The resulting halfword is extended to 32 bits by concatenating 16 0-bits to its left. The result is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

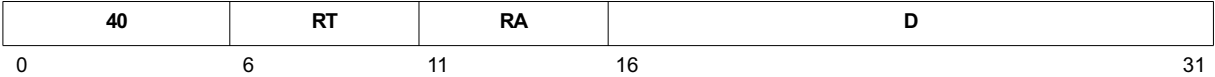
Invalid Instruction Forms

- Reserved fields

Programming Note

Byte ordering is generally controlled by the Endian (E) storage attribute (see *Memory Management* on page 235). The load byte reverse instructions provide a mechanism for data to be loaded from a memory page using the opposite byte ordering from that specified by the Endian storage attribute.

lhz RT, D(RA)



$EA \leftarrow (RA|0) + EXTS(D)$
 $(RT) \leftarrow {}^{16}_0 \parallel MS(EA,2)$

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The halfword at the EA is extended to 32 bits by concatenating 16 0-bits to its left. The result is placed into register RT.

Registers Altered

- RT

lhzu

Load Halfword and Zero with Update

Preliminary User's Manual**lhzu** RT, D(RA)

$$EA \leftarrow (RA|0) + EXTS(D)$$

$$(RA) \leftarrow EA$$

$$(RT) \leftarrow {}^{16}0 \parallel MS(EA,2)$$

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

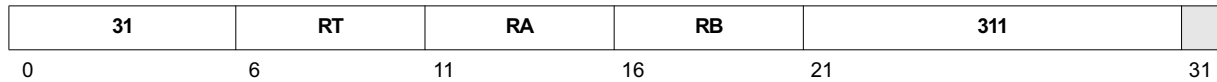
The halfword at the EA is extended to 32 bits by concatenating 16 0-bits to its left. The result is placed into register RT.

Registers Altered

- RA
- RT

Invalid Instruction Forms

- RA = RT
- RA = 0

lhzux **RT, RA, RB**

$EA \leftarrow (RA|0) + (RB)$
 $(RA) \leftarrow EA$
 $(RT) \leftarrow {}^{16}0 \parallel MS(EA,2)$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The halfword at the EA is extended to 32 bits by concatenating 16 0-bits to its left. The result is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RA
- RT

Invalid Instruction Forms

- Reserved fields
- RA = RT
- RA = 0

lhzx

Load Halfword and Zero Indexed

Preliminary User's Manual**lhzx** RT, RA, RB

$$EA \leftarrow (RA|0) + (RB)$$

$$(RT) \leftarrow {}^{16}_0 || MS(EA,2)$$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The halfword at the EA is extended to 32 bits by concatenating 16 0-bits to its left. The result is placed into register RT.

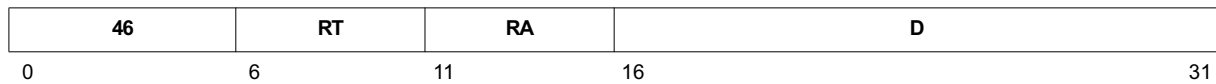
If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

Imw RT, D(RA)

```

EA ← (RA|0) + EXTS(D)
r ← RT
do while r ≤ 31
    GPR(r) ← MS(EA,4)
    r ← r + 1
    EA ← EA + 4

```

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field in the instruction to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

A series of consecutive words starting at the EA are loaded into a set of consecutive GPRs, starting with register RT and continuing to and including GPR(31).

Registers Altered

•RT through GPR(31).

Invalid Instruction Forms

•RA is in the range of registers to be loaded, including the case RA = RT = 0.

Programming Note

This instruction can be restarted, meaning that it could be interrupted after having already updated some of the target registers, and then re-executed from the beginning (after returning from the interrupt), in which case the registers which had already been loaded prior to the interrupt will be loaded a second time. Note that if RA is in the range of registers to be loaded (an invalid form; see above) and is also one of the registers which is loaded prior to the interrupt, then when the instruction is restarted the re-calculated EA will be incorrect, since RA will no longer contain the original base address. Hence the definition of this as an invalid form which software must avoid.

lswi

Load String Word Immediate

Preliminary User's Manual**lswi** RT, RA, NB

```

EA ← (RA|0)
if NB = 0 then
    CNT ← 32
else
    CNT ← NB
n ← CNT
RFINAL ← ((RT + CEIL(CNT/4) – 1) % 32)
r ← RT – 1
i ← 0
do while n > 0
    if i = 0 then
        r ← r + 1
        if r = 32 then
            r ← 0
        GPR(r) ← 0
        GPR(r)i:i+7 ← MS(EA,1)
        i ← i + 8
        if i = 32 then
            i ← 0
        EA ← EA + 1
        n ← n – 1

```

An effective address (EA) is determined by the RA field. If the RA field contains 0, the EA is 0. Otherwise, the EA is the contents of register RA.

The NB field specifies the byte count CNT. If the NB field contains 0, the byte count is CNT = 32. Otherwise, the byte count is CNT = NB.

A series of CNT consecutive bytes in main storage, starting at the EA, are loaded into CEIL(CNT/4) consecutive GPRs, four bytes per GPR, until the byte count is exhausted. Bytes are loaded into GPRs; the byte at the lowest address is loaded into the most significant byte. Bits to the right of the last byte loaded into the last GPR are set to 0.

The set of loaded GPRs starts at register RT, continues consecutively through GPR(31), and wraps to register 0, loading until the byte count is exhausted, which occurs in register R_{FINAL}.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

•RT and subsequent GPRs as described above.

Invalid Instruction Forms

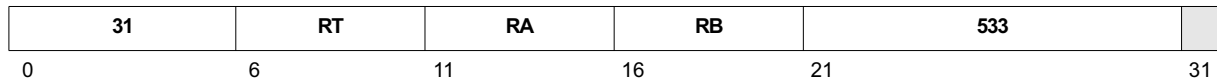
- Reserved fields
- RA is in the range of registers to be loaded
- RA = RT = 0

Programming Note

This instruction can be restarted, meaning that it could be interrupted after having already updated some of the target registers, and then re-executed from the beginning (after returning from the interrupt), in which case the registers which had already been loaded prior to the interrupt will be loaded a second time. Note that if RA is in the range of registers to be loaded (an invalid form; see above) and is also one of the registers which is loaded prior to the interrupt, then when the instruction is restarted the re-calculated EA will be incorrect, since RA will no longer contain the original base address. Hence the definition of this as an invalid form which software must avoid.

lswx

Load String Word Indexed

Preliminary User's Manual**lswx** RT, RA, RB

```

EA ← (RA[0] + (RB))
CNT ← XER[TBC]
n ← CNT
RFINAL ← ((RT + CEIL(CNT/4) – 1) % 32)
r ← RT – 1
i ← 0
do while n > 0
    if i = 0 then
        r ← r + 1
        if r = 32 then
            r ← 0
        GPR(r) ← 0
        GPR(r);i+7) ← MS(EA,1)
        i ← i + 8
        if i = 32 then
            i ← 0
        EA ← EA + 1
        n ← n – 1

```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

A byte count CNT is obtained from XER[TBC].

A series of CNT consecutive bytes in main storage, starting at the EA, are loaded into CEIL(CNT/4) consecutive GPRs, four bytes per GPR, until the byte count is exhausted. Bytes are loaded into GPRs; the byte having the lowest address is loaded into the most significant byte. Bits to the right of the last byte loaded in the last GPR used are set to 0.

The set of consecutive GPRs loaded starts at register RT, continues through GPR(31), and wraps to register 0, loading until the byte count is exhausted, which occurs in register R_{FINAL}.

If XER[TBC] is 0, the byte count is 0 and the contents of register RT are undefined.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT and subsequent GPRs as described above.

Invalid Instruction Forms

- Reserved fields
- RA or RB is in the range of registers to be loaded.
- RA = RT = 0

Programming Note

This instruction can be restarted, meaning that it could be interrupted after having already updated some of the target registers, and then re-executed from the beginning (after returning from the interrupt), in which case the registers which had already been loaded prior to the interrupt will be loaded a second time. Note that if RA or RB is in the range of registers to be loaded (an invalid form; see above) and is also one of the registers which is loaded prior to the interrupt, then when the instruction is restarted the re-calculated EA will be incorrect, since the affected register will no longer contain the original base address or index. Hence the definition of these as invalid forms which software must avoid.

If XER[TBC] = 0, the contents of register RT are undefined and **lswx** is treated as a no-op. Furthermore, if the EA is such that a Data Storage, Data TLB Error, or Data Address Compare Debug exception occurs, **lswx** is treated as a no-op and no interrupt occurs as a result of the exception.

lwarx

Load Word and Reserve Indexed

Preliminary User's Manual**lwarx** RT, RA, RB

$$EA \leftarrow (RA|0) + (RB)$$

$$RESERVE \leftarrow 1$$

$$(RT) \leftarrow MS(EA,4)$$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The word at the EA is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Execution of the **lwarx** instruction sets the reservation bit.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

Programming Note

The **lwarx** and **stwcx.** instructions are typically paired in a loop, as shown in the following example, to create the effect of an atomic operation to a memory area used as a semaphore between multiple processes. Only **lwarx** can set the reservation bit to 1. **stwcx.** sets the reservation bit to 0 upon its completion, whether or not **stwcx.** actually stored (RS) to memory. CR[CR0]₂ must be examined to determine whether (RS) was sent to memory.

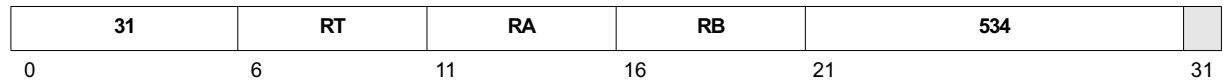
```

loop: lwarx      # read the semaphore from memory; set reservation
      "alter"    # change the semaphore bits in the register as required
      stwcx.     # attempt to store the semaphore; reset reservation
      bne loop   # some other process intervened and cleared the reservation prior to the above
                  # stwcx.; try again

```

The PowerPC Book-E architecture specifies that the EA for the **lwarx** instruction must be word-aligned (that is, a multiple of 4 bytes); otherwise, the result is undefined. Although the PPC440GX Embedded Processor will execute **lwarx** regardless of the EA alignment, in order for the operation of the pairing of **lwarx** and **stwcx.** to produce the desired result, software must ensure that the EA for both instructions is word-aligned. This requirement is due to the manner in which misaligned storage accesses may be broken up into separate, aligned accesses by the PPC440GX Embedded Processor.

lwbrx RT, RA, RB



$EA \leftarrow (RA|0) + (RB)$
 $(RT) \leftarrow \text{BYTE_REVERSE}(\text{MS}(EA,4))$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The word at the EA is byte-reversed from the default byte ordering for the memory page referenced by the EA. The result is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

Invalid Instruction Forms

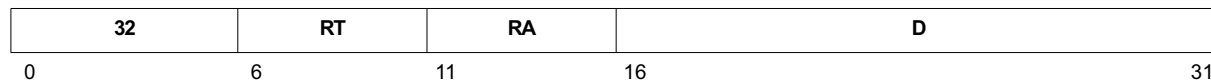
- Reserved fields

Programming Note

Byte ordering is generally controlled by the Endian (E) storage attribute (see *Memory Management* on page 235). The load byte reverse instructions provide a mechanism for data to be loaded from a memory page using the opposite byte ordering from that specified by the Endian storage attribute.

lwz

Load Word and Zero

Preliminary User's Manual**lwz** RT, D(RA) $EA \leftarrow (RA|0) + EXTS(D)$ $(RT) \leftarrow MS(EA,4)$

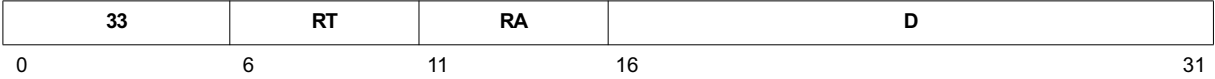
An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The word at the EA is placed into register RT.

Registers Altered

•RT

lwzu RT, D(RA)



EA ← (RA|0) + EXTS(D)
(RA) ← EA
(RT) ← MS(EA,4)

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The word at the EA is placed into register RT.

Registers Altered

- RA
- RT

Invalid Instruction Forms

- RA = RT
- RA = 0

lwzux

Load Word and Zero with Update Indexed

Preliminary User's Manual**lwzux** RT, RA, RB

$$EA \leftarrow (RA|0) + (RB)$$

$$(RA) \leftarrow EA$$

$$(RT) \leftarrow MS(EA,4)$$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The word at the EA is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RA
- RT

Invalid Instruction Forms

- Reserved fields
- RA = RT
- RA = 0

lwzx RT, RA, RB



$EA \leftarrow (RA|0) + (RB)$
 $(RT) \leftarrow MS(EA,4)$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The word at the EA is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

macchw

Multiply Accumulate Cross Halfword to Word Modulo Signed

Preliminary User's Manual

macchw	RT, RA, RB	OE=0, Rc=0
macchw.	RT, RA, RB	OE=0, Rc=1
macchwo	RT, RA, RB	OE=1, Rc=0
macchwo.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	172	Rc
0	6	11	16	21 22		31

$$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15} \text{ signed}$$

$$\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$$

$$(\text{RT}) \leftarrow \text{temp}_{1:32}$$

The low-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is summed with the contents of RT and RT is updated with the low-order 32 bits of the signed sum.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

macchws	RT, RA, RB	OE=0, Rc=0
macchws.	RT, RA, RB	OE=0, Rc=1
macchwso	RT, RA, RB	OE=1, Rc=0
macchwso.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	236	Rc
0	6	11	16	21 22		31

$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ signed

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \parallel {}^{31}(\neg RT_0))$

else $(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is summed with the contents of RT.

If the signed sum can be represented in 32 bits, then RT is updated with the low-order 32 bits of the signed sum.

If the signed sum cannot be represented in 32 bits, then RT is updated with a value which is “saturated” to the nearest representable value. That is, if the signed sum is less than -2^{31} , then RT is updated with -2^{31} . Likewise, if the signed sum is greater than $2^{31} - 1$, then RT is updated with $2^{31} - 1$.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

macchwsu

Multiply Accumulate Cross Halfword to Word Saturate Unsigned

Preliminary User's Manual

macchwsu	RT, RA, RB	OE=0, Rc=0
macchwsu.	RT, RA, RB	OE=0, Rc=1
macchwsuo	RT, RA, RB	OE=1, Rc=0
macchwsuo.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	204	Rc
0	6	11	16	21 22		31

$$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15} \text{ unsigned}$$

$$\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$$

$$(\text{RT}) \leftarrow (\text{temp}_{1:32} \vee {}^{32}\text{temp}_0)$$

The low-order halfword of RA is multiplied by the high-order halfword of RB. The unsigned product is summed with the contents of RT.

If the unsigned sum can be represented in 32 bits, then RT is updated with the low-order 32 bits of the unsigned sum.

If the unsigned sum cannot be represented in 32 bits, then RT is updated with a value which is “saturated” to the maximum representable value of $2^{32} - 1$.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

macchwu	RT, RA, RB	OE=0, Rc=0
macchwu.	RT, RA, RB	OE=0, Rc=1
macchwuo	RT, RA, RB	OE=1, Rc=0
macchwuo.	RT, RA, RB	OE=1, Rc=1



```
prod0:31 ← (RA)16:31 × (RB)0:15 unsigned
temp0:32 ← prod0:31 + (RT)
(RT) ← temp1:32
```

The low-order halfword of RA is multiplied by the high-order halfword of RB. The unsigned product is summed with the contents of RT and RT is updated with the low-order 32 bits of the unsigned sum.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

machhw

Multiply Accumulate High Halfword to Word Modulo Signed

Preliminary User's Manual

machhw	RT, RA, RB	OE=0, Rc=0
machhw.	RT, RA, RB	OE=0, Rc=1
machhwo	RT, RA, RB	OE=1, Rc=0
machhwo.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	44	Rc
0	6	11	16	21 22		31

$$\text{prod}_{0:31} \leftarrow (\text{RA})_{0:15} \times (\text{RB})_{0:15} \text{ signed}$$

$$\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$$

$$(\text{RT}) \leftarrow \text{temp}_{1:32}$$

The high-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is summed with the contents of RT and RT is updated with the low-order 32 bits of the signed sum.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

machhws	RT, RA, RB	OE=0, Rc=0
machhws.	RT, RA, RB	OE=0, Rc=1
machhwso	RT, RA, RB	OE=1, Rc=0
machhwso.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	108	Rc
0	6	11	16	21 22		31

$\text{prod}_{0:31} \leftarrow (\text{RA})_{0:15} \times (\text{RB})_{0:15} \text{ signed}$
 $\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$
 if $((\text{prod}_0 = \text{RT}_0) \wedge (\text{RT}_0 \neq \text{temp}_1))$ then $(\text{RT}) \leftarrow (\text{RT}_0 \parallel {}^{31}(\neg \text{RT}_0))$
 else $(\text{RT}) \leftarrow \text{temp}_{1:32}$

The high-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is summed with the contents of RT.

If the signed sum can be represented in 32 bits, then RT is updated with the low-order 32 bits of the signed sum.

If the signed sum cannot be represented in 32 bits, then RT is updated with a value which is “saturated” to the nearest representable value. That is, if the signed sum is less than -2^{31} , then RT is updated with -2^{31} . Likewise, if the signed sum is greater than $2^{31} - 1$, then RT is updated with $2^{31} - 1$.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

machhwsu

Multiply Accumulate High Halfword to Word Saturate Unsigned

Preliminary User's Manual

machhwsu	RT, RA, RB	OE=0, Rc=0
machhwsu.	RT, RA, RB	OE=0, Rc=1
machhwsuo	RT, RA, RB	OE=1, Rc=0
machhwsuo.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	76	Rc
0	6	11	16	21 22		31

$$\text{prod}_{0:31} \leftarrow (\text{RA})_{0:15} \times (\text{RB})_{0:15} \text{ unsigned}$$

$$\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$$

$$(\text{RT}) \leftarrow (\text{temp}_{1:32} \vee {}^{32}\text{temp}_0)$$

The high-order halfword of RA is multiplied by the high-order halfword of RB. The unsigned product is summed with the contents of RT.

If the unsigned sum can be represented in 32 bits, then RT is updated with the low-order 32 bits of the unsigned sum.

If the unsigned sum cannot be represented in 32 bits, then RT is updated with a value which is “saturated” to the maximum representable value of $2^{32} - 1$.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

machhwu	RT, RA, RB	OE=0, Rc=0
machhwu.	RT, RA, RB	OE=0, Rc=1
machhwuo	RT, RA, RB	OE=1, Rc=0
machhwuo.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	12	Rc
0	6	11	16	21 22		31

$\text{prod}_{0:31} \leftarrow (\text{RA})_{0:15} \times (\text{RB})_{0:15} \text{ unsigned}$

$\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$

$(\text{RT}) \leftarrow \text{temp}_{1:32}$

The high-order halfword of RA is multiplied by the high-order halfword of RB. The unsigned product is summed with the contents of RT and RT is updated with the low-order 32 bits of the unsigned sum.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

macldw

Multiply Accumulate Low Halfword to Word Modulo Signed

Preliminary User's Manual

macldw	RT, RA, RB	OE=0, Rc=0
macldw.	RT, RA, RB	OE=0, Rc=1
macldwo	RT, RA, RB	OE=1, Rc=0
macldwo.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	428	Rc
0	6	11	16	21 22		31

$$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{16:31} \text{ signed}$$

$$\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$$

$$(\text{RT}) \leftarrow \text{temp}_{1:32}$$

The low-order halfword of RA is multiplied by the low-order halfword of RB. The signed product is summed with the contents of RT and RT is updated with the low-order 32 bits of the signed sum.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

Preliminary User's Manual

maclhws	RT, RA, RB	OE=0, Rc=0
maclhws.	RT, RA, RB	OE=0, Rc=1
maclhws0	RT, RA, RB	OE=1, Rc=0
maclhws0.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	492	Rc
0	6	11	16	21 22		31

$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{16:31} \text{ signed}$
 $\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$
 if $((\text{prod}_0 = \text{RT}_0) \wedge (\text{RT}_0 \neq \text{temp}_1))$ then $(\text{RT}) \leftarrow (\text{RT}_0 \parallel {}^{31}(\neg \text{RT}_0))$
 else $(\text{RT}) \leftarrow \text{temp}_{1:32}$

The low-order halfword of RA is multiplied by the low-order halfword of RB. The signed product is summed with the contents of RT.

If the signed sum can be represented in 32 bits, then RT is updated with the low-order 32 bits of the signed sum.

If the signed sum cannot be represented in 32 bits, then RT is updated with a value which is “saturated” to the nearest representable value. That is, if the signed sum is less than -2^{31} , then RT is updated with -2^{31} . Likewise, if the signed sum is greater than $2^{31} - 1$, then RT is updated with $2^{31} - 1$.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

maclhwsu

Multiply Accumulate Low Halfword to Word Saturate Unsigned

Preliminary User's Manual

maclhwsu	RT, RA, RB	OE=0, Rc=0
maclhwsu.	RT, RA, RB	OE=0, Rc=1
maclhwsuo	RT, RA, RB	OE=1, Rc=0
maclhwsuo.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	460	Rc
0	6	11	16	21 22		31

$$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{16:31} \text{ unsigned}$$

$$\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$$

$$(\text{RT}) \leftarrow (\text{temp}_{1:32} \vee {}^{32}\text{temp}_0)$$

The low-order halfword of RA is multiplied by the low-order halfword of RB. The unsigned product is summed with the contents of RT.

If the unsigned sum can be represented in 32 bits, then RT is updated with the low-order 32 bits of the unsigned sum.

If the unsigned sum cannot be represented in 32 bits, then RT is updated with a value which is “saturated” to the maximum representable value of $2^{32} - 1$.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

macldwu	RT, RA, RB	OE=0, Rc=0
macldwu.	RT, RA, RB	OE=0, Rc=1
macldwuo	RT, RA, RB	OE=1, Rc=0
macldwuo.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	396	Rc
0	6	11	16	21 22		31

$$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{16:31} \text{ unsigned}$$

$$\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$$

$$(\text{RT}) \leftarrow \text{temp}_{1:32}$$

The low-order halfword of RA is multiplied by the low-order halfword of RB. The unsigned product is summed with the contents of RT and RT is updated with the low-order 32 bits of the unsigned sum.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

mbar



The **mbar** instruction ensures that all loads and stores preceding **mbar** complete with respect to main storage before any loads and stores following **mbar** access main storage. As implemented in the PPC440GX, the MO field of **mbar** is ignored and treated as 0, providing a storage ordering function for all storage access instructions executed by the processor. Other processors implementing the **mbar** instruction may support one or more non-zero MO settings, specifying different subsets of storage accesses to be ordered by the **mbar** instruction in those implementations.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Note

Architecturally, **mbar** merely orders storage accesses, and does not perform execution nor context synchronization (see *Synchronization* on page 196). Therefore, non-storage access instructions after **mbar** could complete before the storage access instructions which were executed prior to **mbar** have actually completed their storage accesses. The **msync** instruction, on the other hand, *is* execution synchronizing, and *does* guarantee that all storage accesses initiated by instructions executed prior to the **msync** have completed before any instructions after the **msync** begin execution. However, the PPC440GX implements the **mbar** instruction identically to the **msync** instruction, and thus both are execution synchronizing.

Software should nevertheless use the correct instruction (**mbar** or **msync**) as called for by the specific ordering and synchronizing requirements of the application, in order to guarantee portability to other implementations.

See *Storage Ordering and Synchronization* on page 198 for additional information on the use of the **msync** and **mbar** instructions.

Table 31-19. Extended Mnemonics for mbar

Mnemonic	Operands	Function	Other Registers Altered
mbar	None	Memory Barrier. <i>Extended mnemonic for mbar 0</i>	

Architecture Note

mbar replaces the PowerPC **eieio** instruction. **mbar** uses the same opcode as **eieio**; PowerPC applications which used **eieio** will get the function of **mbar** when executed on a PowerPC Book-E implementation. **mbar** is architecturally “stronger” than **eieio**, in that **eieio** forced separate ordering amongst different *categories* of storage accesses, while **mbar** forces such ordering amongst *all* storage accesses as a single category.

mcrf BF, BFA



m ← BFA
n ← BF
(CR[CRn]) ← (CR[CRm])

The contents of the CR field specified by the BFA field are placed into the CR field specified by the BF field.

Registers Altered

- CR[CRn] where n is specified by the BF field.

Invalid Instruction Forms

- Reserved fields

mcrxr

Move to Condition Register from XER

Preliminary User's Manual**mcrxr** BF

$$n \leftarrow \text{BF}$$

$$(\text{CR}[\text{CR}_n]) \leftarrow \text{XER}_{0:3}$$

$$\text{XER}_{0:3} \leftarrow 40$$

The contents of $\text{XER}_{0:3}$ are placed into the CR field specified by the BF field. $\text{XER}_{0:3}$ are then set to 0.

If instruction bit 31 contains 1, the contents of $\text{CR}[\text{CR}_0]$ are undefined.

Registers Altered

- $\text{CR}[\text{CR}_n]$ where n is specified by the BF field.
- $\text{XER}[\text{SO}, \text{OV}, \text{CA}]$

Invalid Instruction Forms

- Reserved fields

mfcrr RT



$(RT) \leftarrow (CR)$

The contents of the CR are placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

mfdcr

Move from Device Control Register

Preliminary User's Manual**mfdcr** RT, DCRN

$$\text{DCRN} \leftarrow \text{DCRF}_{5:9} \parallel \text{DCRF}_{0:4}$$

$$(\text{RT}) \leftarrow (\text{DCR}(\text{DCRN}))$$

The contents of the DCR specified by the DCRF field are placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

Programming Notes

Execution of this instruction is privileged.

The DCR number (DCRN) specified in the assembler language coding of the **mfdcr** instruction refers to a DCR number. The assembler handles the unusual register number encoding to generate the DCRF field.

Architecture Note

The specific numbers and definitions of any DCRs are outside the scope of both the PowerPC Book-E architecture and the PPC440GX. Any DCRs are defined as part of the chip-level product incorporating the PPC440GX.

mfmsr RT



$(RT) \leftarrow (MSR)$

The contents of the MSR are placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

Programming Note

Execution of this instruction is privileged.

mf spr

Move From Special Purpose Register

Preliminary User's Manual**mf spr**

RT, SPRN



$$\text{SPRN} \leftarrow \text{SPRF}_{5:9} \parallel \text{SPRF}_{0:4}$$

$$(\text{RT}) \leftarrow (\text{SPR}(\text{SPRN}))$$

The contents of the SPR specified by the SPRF field are placed into register RT. See *Special Purpose Registers Sorted by SPR Number* on page 1217 for a listing of SPR mnemonics and corresponding SPRN values.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields
- Invalid SPRF values

Programming Note

Execution of this instruction is privileged if instruction bit 11 contains 1. See *Privileged SPRs* on page 196 for a list of privileged SPRs.

The SPR number (SPRN) specified in the assembler language coding of the **mf spr** instruction refers to an SPR number. The assembler handles the unusual register number encoding to generate the SPRF field.

mfspr

Move From Special Purpose Register

Preliminary User's ManualTable 31-20. Extended Mnemonics for *mfspir*

Mnemonic	Operands	Function
mfccr0 mfccr1 mfcsrr0 mfcsrr1 mfctr mfdac1 mfdac2 mfd bcr0 mfd bcr1 mfd bcr2 mfd bdr mfd bsr mfd cdbtrh mfd cdbtrl mfdear mfdec mfdnv0 mfdnv1 mfdnv2 mfdnv3 mfdtv0 mfdtv1 mfdtv2 mfdtv3 mfdvc1 mfdvc2 mfdvlim mfesr mfiac1 mfiac2 mfiac3 mfiac4 mficdbdr mficdbtrh mficdbtrl mfinv0 mfinv1 mfinv2 mfinv3 mfitv0 mfitv1 mfitv2 mfitv3 mfivlim mfivor0 mfivor1 mfivor2 mfivor3 mfivor4 mfivor5 mfivor6 mfivor7 mfivor8 mfivor9 mfivor10 mfivor11 mfivor12 mfivor13 mfivor14 mfivor15 mfivpr mflr mfmsr mfmsrr0 mfmsrr1 mfmmucr	RT	<p>Move from special purpose register SPRN. <i>Extended mnemonic for</i> mfspir RT,SPRN</p> <p>See <i>Special Purpose Registers Sorted by SPR Number</i> on page 1217 for a list of valid SPRN values.</p>

Table 31-20. Extended Mnemonics for mfspir (Continued)

Mnemonic	Operands	Function
mfpid mfpir mfpvr mfsprg0 mfsprg1 mfsprg2 mfsprg3 mfsprg4 mfsprg5 mfsprg6 mfsprg7 mfsrr0 mfsrr1 mftbl mftbu mftcr mftsr mfusprg0 mfxer		

msync

Memory Synchronize

Preliminary User's Manual**msync**

The **msync** instruction guarantees that all instructions initiated by the processor preceding **msync** will complete before **msync** completes, and that no subsequent instructions will be initiated by the processor until after **msync** completes. **msync** also will not complete until all storage accesses associated with instructions preceding **msync** have completed.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None.

Invalid Instruction Forms

- Reserved fields

Programming Notes

The **msync** instruction is execution synchronizing (see *Execution Synchronization* on page 198), and guarantees that all storage accesses initiated by instructions executed prior to the **msync** have completed before any instructions after the **msync** begin execution. On the other hand, architecturally the **mbar** instruction merely *orders* storage accesses, and does *not* perform execution synchronization. Therefore, non-storage access instructions after **mbar** *could* complete before the storage access instructions which were executed prior to **mbar** have actually completed their storage accesses. However, the PPC440GX implements the **mbar** instruction identically to the **msync** instruction, and thus both are execution synchronizing.

Software should nevertheless use the correct instruction (**mbar** or **msync**) as called for by the specific ordering and synchronizing requirements of the application, in order to guarantee portability to other implementations.

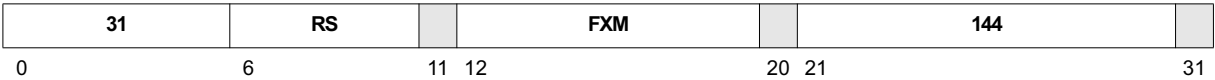
See *Storage Ordering and Synchronization* on page 198 for additional information on the use of the **msync** and **mbar** instructions.

Architecture Note

mbar replaces the PowerPC **eieio** instruction. **mbar** uses the same opcode as **eieio**; PowerPC applications which used **eieio** will get the function of **mbar** when executed on a PowerPC Book-E implementation. **mbar** is architecturally “stronger” than **eieio**, in that **eieio** forced separate ordering amongst different *categories* of storage accesses, while **mbar** forces such ordering amongst *all* storage accesses as a single category.

msync replaces the PowerPC **sync** instruction. **msync** uses the same opcode as **sync**; PowerPC applications which used **sync** get the function of **msync** when executed on a PowerPC Book-E implementation. **msync** is architecturally identical to the version of **sync** specified by an earlier version of the PowerPC architecture.

mtcrf FXM, RS



$$\text{mask} \leftarrow {}^4(\text{FXM}_0) \parallel {}^4(\text{FXM}_1) \parallel \dots \parallel {}^4(\text{FXM}_6) \parallel {}^4(\text{FXM}_7)$$
$$(\text{CR}) \leftarrow ((\text{RS}) \wedge \text{mask}) \vee ((\text{CR}) \wedge \neg \text{mask})$$

Some or all of the contents of register RS are placed into the CR as specified by the FXM field.

Each bit in the FXM field controls the copying of 4 bits in register RS into the corresponding bits in the CR. The correspondence between the bits in the FXM field and the bit copying operation is shown in the following table:

Table 31-21. FXM Bit Field Correspondence

FXM Bit Number	CR Bits Affected
0	0:3
1	4:7
2	8:11
3	12:15
4	16:19
5	20:23
6	24:27
7	28:31

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- CR

Invalid Instruction Forms

- Reserved fields

Table 31-22. Extended Mnemonics for mtcrrf

Mnemonic	Operands	Function
mtcr	RS	Move to CR. Extended mnemonic for mtcrf 0xFF,RS

mtdcr

Move To Device Control Register

Preliminary User's Manual**mtdcr** DCRN, RS

$$\text{DCRN} \leftarrow \text{DCRF}_{5:9} \parallel \text{DCRF}_{0:4}$$

$$(\text{DCR}(\text{DCRN})) \leftarrow (\text{RS})$$

The contents of register RS are placed into the DCR specified by the DCRF field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- DCR(DCRN)

Invalid Instruction Forms

- Reserved fields

Programming Note

Execution of this instruction is privileged.

The DCR number (DCRN) specified in the assembler language coding of the **mtdcr** instruction refers to a DCR number. The assembler handles the unusual register number encoding to generate the DCRF field.

Architecture Note

The specific numbers and definitions of any DCRs are outside the scope of both the PowerPC Book-E architecture and the PPC440GX. Any DCRs are defined as part of the chip-level product incorporating the PPC440GX.

mtmsr RS



(MSR) ← (RS)

The contents of register RS are placed into the MSR.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- MSR

Invalid Instruction Forms

- Reserved fields

Programming Note

The **mtmsr** instruction is privileged and execution synchronizing (see *Execution Synchronization* on page 198).

mtspr SPRN, RS



$$\text{SPRN} \leftarrow \text{SPRF}_{5:9} \parallel \text{SPRF}_{0:4}$$
$$(\text{SPR}(\text{SPRN})) \leftarrow (\text{RS})$$

The contents of register RS are placed into register RT. See *Special Purpose Registers Sorted by SPR Number* on page 1217 for a listing of SPR mnemonics and corresponding SPRN values.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- SPR (SPRN)

Invalid Instruction Forms

- Reserved fields
- Invalid SPRF values

Programming Note

Execution of this instruction is privileged if instruction bit 11 contains 1. See *Privileged SPRs* on page 196 for a list of privileged SPRs.

The SPR number (SPRN) specified in the assembler language coding of the **mtspr** instruction refers to an SPR number. The assembler handles the unusual register number encoding to generate the SPRF field.

Table 31-23. Extended Mnemonics for *mtspr*

Mnemonic	Operands	Function
mtccr0 mtccr1 mtcsrr0 mtcsrr1 mtctr mtdac1 mtdac2 mtdbcr0 mtdbcr1 mtdbcr2 mtdbdr mtdbsr mtdear mtdec mtdecar mtdnv0 mtdnv1 mtdnv2 mtdnv3 mtdtv0 mtdtv1 mtdtv2 mtdtv3 mtdvc1 mtdvc2 mtdvlim mtesr mtiac1 mtiac2 mtiac3 mtiac4 mtinv0 mtinv1 mtinv2 mtinv3 mtitv0 mtitv1 mtitv2 mtitv3 mtivlim mtivor0 mtivor1 mtivor2 mtivor3 mtivor4 mtivor5 mtivor6 mtivor7 mtivor8 mtivor9 mtivor10 mtivor11 mtivor12 mtivor13 mtivor14 mtivor15 mtivpr mtlr mtmcsr mtmcsrr0 mtmcsrr1 mtmmucr mtpid	RT	<p>Move to special purpose register SPRN. <i>Extended mnemonic for</i> mtspr RT,SPRN</p> <p>See <i>Special Purpose Registers Sorted by SPR Number</i> on page 1217 for a list of valid SPRN values.</p>

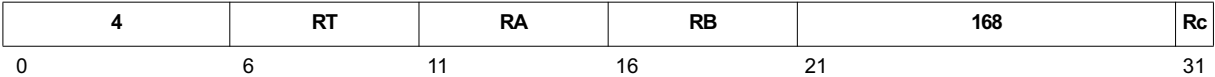
mtspr

Move To Special Purpose Register

Preliminary User's ManualTable 31-23. Extended Mnemonics for *mtspr* (Continued)

Mnemonic	Operands	Function
mtsprg0 mtsprg1 mtsprg2 mtsprg3 mtsprg4 mtsprg5 mtsprg6 mtsprg7 mtsrr0 mtsrr1 mttbl mttbu mttcr mttsr mtusprg0 mtxer		

mulchw RT, RA, RB Rc=0
mulchw. RT, RA, RB Rc=1



$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15} \text{ signed}$

The low-order halfword of RA is multiplied by the high-order halfword of RB, considering both source operands as signed integers. The 32-bit result is placed into register RT.

Registers Altered

- RT
- CR[CR0] if Rc contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

mulchwu

Multiply Cross Halfword to Word Unsigned

Preliminary User's Manual

mulchwu RT, RA, RB Rc=0
mulchwu. RT, RA, RB Rc=1

4	RT	RA	RB	136	Rc
0	6	11	16	21	31

$$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15} \text{ unsigned}$$

The low-order halfword of RA is multiplied by the high-order halfword of RB, considering both source operands as unsigned integers. The 32-bit result is placed into register RT.

Registers Altered

- RT
- CR[CR0] if Rc contains 1

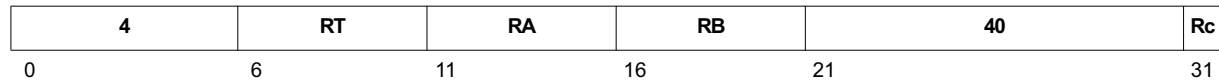
Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

Preliminary User's Manual**mulhww**

Multiply High Halfword to Word Signed

mulhww RT, RA, RB Rc=0
mulhww. RT, RA, RB Rc=1



$$(RT)_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15} \text{ signed}$$

The high-order halfword of RA is multiplied by the high-order halfword of RB, considering both source operands as signed integers. The 32-bit result is placed into register RT.

Registers Altered

- RT
- CR[CR0] if Rc contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

mulhhuw

Multiply High Halfword to Word Unsigned

Preliminary User's Manual

mulhhuw RT, RA, RB Rc=0
mulhhuw. RT, RA, RB Rc=1



$$(RT)_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15} \text{ unsigned}$$

The high-order halfword of RA is multiplied by the high-order halfword of RB, considering both source operands as unsigned integers. The 32-bit result is placed into register RT.

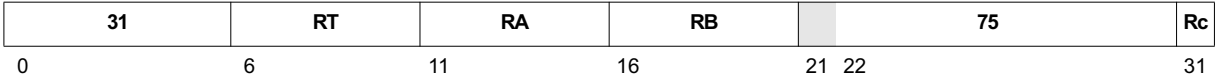
Registers Altered

- RT
- CR[CR0] if Rc contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

mulhwRT, RA, RB
Rc=0
mulhw. RT, RA, RB
Rc=1



$$\text{prod}_{0:63} \leftarrow (\text{RA}) \times (\text{RB}) \text{ signed}$$

$$(\text{RT}) \leftarrow \text{prod}_{0:31}$$

The 64-bit signed product of registers RA and RB is formed. The most significant 32 bits of the result is placed into register RT.

Registers Altered

- RT
- CR[CR0] if Rc contains 1

Programming Note

The most significant 32 bits of the product, unlike the least significant 32 bits, may differ depending on whether the registers RA and RB are interpreted as signed or unsigned quantities. **mulhw** generates the correct result when these operands are interpreted as signed quantities. **mulhwu** generates the correct result when these operands are interpreted as unsigned quantities.

Invalid Instruction Forms

- Reserved fields

mulhwu.



$$R_c = 0$$
$$R_c = 1$$

$$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31} \text{ signed}$$

The low-order halfword of RA is multiplied by the low-order halfword of RB, considering both source operands as signed integers. The 32-bit result is placed into register RT.

Registers Altered

- RT
- CR[CR0] if Rc contains 1

Architecture Note

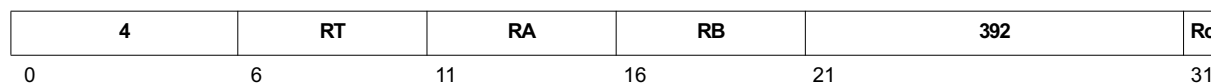
This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

mullhww

Multiply Low Halfword to Word Unsigned

Preliminary User's Manual

mullhww RT, RA, RB Rc=0
mullhww. RT, RA, RB Rc=1



$$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31} \text{ unsigned}$$

The low-order halfword of RA is multiplied by the low-order halfword of RB, considering both source operands as unsigned integers. The 32-bit result is placed into register RT.

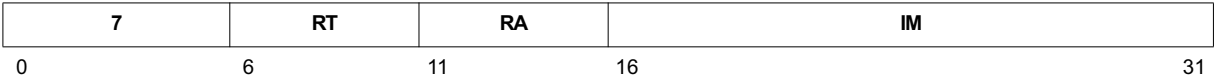
Registers Altered

- RT
- CR[CR0] if Rc contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

mulli RT, RA, IM



$$\text{prod}_{0:47} \leftarrow (\text{RA}) \times \text{IM}$$
$$(\text{RT}) \leftarrow \text{prod}_{16:47}$$

The 48-bit product of register RA and the 16-bit IM field is formed. The least significant 32 bits of the product are placed into register RT.

Registers Altered

- RT

Programming Note

The least significant 32 bits of the product are correct, regardless of whether register RA and field IM are interpreted as signed or unsigned numbers.

mullw

Multiply Low Word

Preliminary User's Manual

mullw	RT, RA, RB	OE=0, Rc=0
mullw.	RT, RA, RB	OE=0, Rc=1
mullwo	RT, RA, RB	OE=1, Rc=0
mullwo.	RT, RA, RB	OE=1, Rc=1



$\text{prod}_{0:63} \leftarrow (\text{RA}) \times (\text{RB}) \text{ signed}$
 $(\text{RT}) \leftarrow \text{prod}_{32:63}$

The 64-bit signed product of register RA and register RB is formed. The least significant 32 bits of the result is placed into register RT.

If the signed product cannot be represented in 32 bits and OE=1, XER[SO, OV] are set to 1.

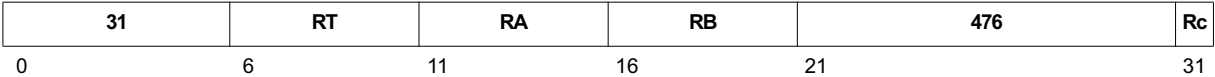
Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE=1

Programming Note

The least significant 32 bits of the product are correct, regardless of whether register RA and register RB are interpreted as signed or unsigned numbers. The overflow indication, however, is calculated specifically for a 64-bit signed product, and is dependent upon interpretation of the source operands as signed numbers.

nand RA, RS, RB Rc=0
nand. RA, RS, RB Rc=1



$(RA) \leftarrow \neg((RS) \wedge (RB))$

The contents of register RS is ANDed with the contents of register RB; the ones complement of the result is placed into register RA.

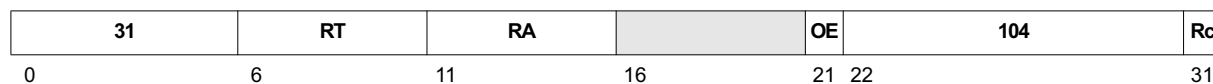
Registers Altered

- RA
- CR[CR0] if Rc contains 1

neg
Negate

Preliminary User's Manual

neg	RT, RA	OE=0, Rc=0
neg.	RT, RA	OE=0, Rc=1
nego	RT, RA	OE=1, Rc=0
nego.	RT, RA	OE=1, Rc=1



$$(RT) \leftarrow \neg(RA) + 1$$

The two's complement of the contents of register RA are placed into register RT.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE=1

Invalid Instruction Forms

- Reserved fields

nmacchw	RT, RA, RB	OE=0, Rc=0
nmacchw.	RT, RA, RB	OE=0, Rc=1
nmacchwo	RT, RA, RB	OE=1, Rc=0
nmacchwo.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	174	Rc
0	6	11	16	21 22		31

$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{0:15})$ signed

$temp_{0:32} \leftarrow nprod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is subtracted from the contents of RT and RT is updated with the low-order 32 bits of the result.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

nmacchws

Negative Multiply Accumulate Cross Halfword to Word Saturate

Preliminary User's Manual

nmacchws	RT, RA, RB	OE=0, Rc=0
nmacchws.	RT, RA, RB	OE=0, Rc=1
nmacchwso	RT, RA, RB	OE=1, Rc=0
nmacchwso.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	238	Rc
0	6	11	16	21 22		31

$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{0:15} \text{ signed})$
 $temp_{0:32} \leftarrow nprod_{0:31} + (RT)$
 if $((nprod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \parallel {}^{31}(\neg RT_0))$
 else $(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is subtracted from the contents of RT.

If the result of the subtraction can be represented in 32 bits, then RT is updated with the low-order 32 bits of the result.

If the result of the subtraction cannot be represented in 32 bits, then RT is updated with a value which is “saturated” to the nearest representable value. That is, if the result is less than -2^{31} , then RT is updated with -2^{31} . Likewise, if the result is greater than $2^{31} - 1$, then RT is updated with $2^{31} - 1$.

Registers Altered

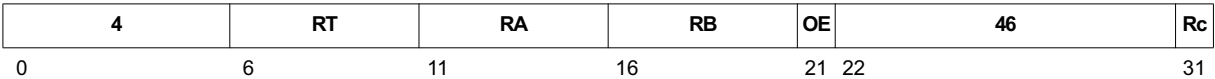
- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

Preliminary User's Manual

nmachhw	RT, RA, RB	OE=0, Rc=0
nmachhw.	RT, RA, RB	OE=0, Rc=1
nmachhwo	RT, RA, RB	OE=1, Rc=0
nmachhwo.	RT, RA, RB	OE=1, Rc=1



```
nprod0:31 ← −((RA)0:15 × (RB)0:15) signed
temp0:32 ← nprod0:31 + (RT)
(RT) ← temp1:32
```

The high-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is subtracted from the contents of RT and RT is updated with the low-order 32 bits of the result.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

nmachhws

Negative Multiply Accumulate High Halfword to Word Saturate

Preliminary User's Manual

nmachhws	RT, RA, RB	OE=0, Rc=0
nmachhws.	RT, RA, RB	OE=0, Rc=1
nmachhwso	RT, RA, RB	OE=1, Rc=0
nmachhwso.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	110	Rc
0	6	11	16	21 22		31

$nprod_{0:31} \leftarrow -((RA)_{0:15} \times (RB)_{0:15})$ signed
 $temp_{0:32} \leftarrow nprod_{0:31} + (RT)$
 if $((nprod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \parallel {}^{31}(\neg RT_0))$
 else $(RT) \leftarrow temp_{1:32}$

The high-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is subtracted from the contents of RT.

If the result of the subtraction can be represented in 32 bits, then RT is updated with the low-order 32 bits of the result.

If the result of the subtraction cannot be represented in 32 bits, then RT is updated with a value which is “saturated” to the nearest representable value. That is, if the result is less than -2^{31} , then RT is updated with -2^{31} . Likewise, if the result is greater than $2^{31} - 1$, then RT is updated with $2^{31} - 1$.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

nmacldw	RT, RA, RB	OE=0, Rc=0
nmacldw.	RT, RA, RB	OE=0, Rc=1
nmacldwo	RT, RA, RB	OE=1, Rc=0
nmacldwo.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	430	Rc
0	6	11	16	21 22		31

$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{16:31})$ signed

$temp_{0:32} \leftarrow nprod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the low-order halfword of RB. The signed product is subtracted from the contents of RT and RT is updated with the low-order 32 bits of the result.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

nmaclhws

Negative Multiply Accumulate High Halfword to Word Saturate

Preliminary User's Manual

nmaclhws	RT, RA, RB	OE=0, Rc=0
nmaclhws.	RT, RA, RB	OE=0, Rc=1
nmaclhws0	RT, RA, RB	OE=1, Rc=0
nmaclhws0.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	494	Rc
0	6	11	16	21 22		31

$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{16:31})$ signed
 $temp_{0:32} \leftarrow nprod_{0:31} + (RT)$
 if $((nprod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \parallel {}^{31}(\neg RT_0))$
 else $(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the low-order halfword of RB. The signed product is subtracted from the contents of RT.

If the result of the subtraction can be represented in 32 bits, then RT is updated with the low-order 32 bits of the result.

If the result of the subtraction cannot be represented in 32 bits, then RT is updated with a value which is “saturated” to the nearest representable value. That is, if the result is less than -2^{31} , then RT is updated with -2^{31} . Likewise, if the result is greater than $2^{31} - 1$, then RT is updated with $2^{31} - 1$.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

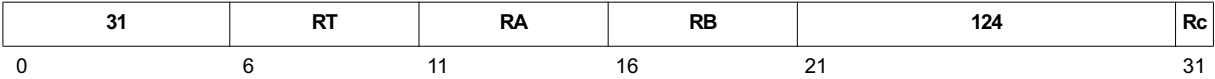
Architecture Note

This instruction is implementation-specific and programs which use this instruction may not be portable to other PowerPC Book-E implementations. See *Instruction Set Portability* on page 1020.

Preliminary User’s Manual

nor
NOR

nor RA, RS, RB Rc=0
nor. RA, RS, RB Rc=1



$(RA) \leftarrow \neg((RS) \vee (RB))$

The contents of register RS is ORed with the contents of register RB; the ones complement of the result is placed into register RA.

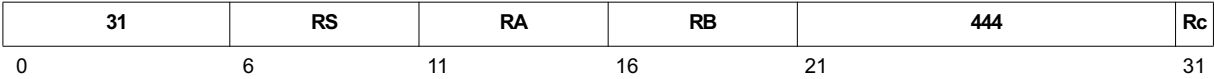
Registers Altered

- RA
- CR[CR0] if Rc contains 1

Table 31-24. Extended Mnemonics for nor, nor.

Mnemonic	Operands	Function	Other Registers Altered
not	RA, RS	Complement register. $(RA) \leftarrow \neg(RS)$ Extended mnemonic for nor RA,RS,RS	
not.		Extended mnemonic for nor. RA,RS,RS	CR[CR0]

or RA, RS, RB Rc=0
or. RA, RS, RB Rc=1



(RA) ← (RS) ∨ (RB)

The contents of register RS is ORed with the contents of register RB; the result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

Table 31-25. Extended Mnemonics for or, or.

Mnemonic	Operands	Function	Other Registers Altered
mr	RT, RS	Move register. (RT) ← (RS) <i>Extended mnemonic for or RT,RS,RS</i>	
mr.		<i>Extended mnemonic for or. RT,RS,RS</i>	CR[CR0]

orc	RA, RS, RB	Rc=0
orc.	RA, RS, RB	Rc=1



$(RA) \leftarrow (RS) \vee \neg(RB)$

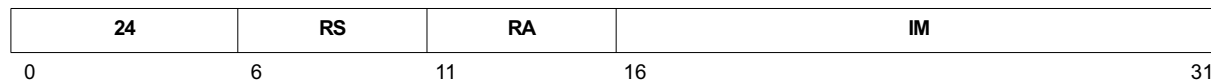
The contents of register RS is ORed with the ones complement of the contents of register RB; the result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

ori

OR Immediate

ori RA, RS, IM

$$(RA) \leftarrow (RS) \vee (^{16}0 \parallel IM)$$

The IM field is extended to 32 bits by concatenating 16 0-bits on the left. Register RS is ORed with the extended IM field; the result is placed into register RA.

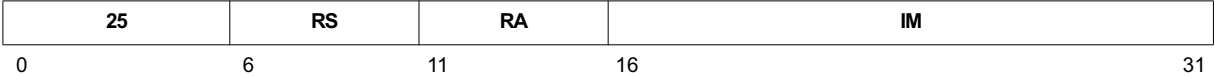
Registers Altered

•RA

Table 31-26. Extended Mnemonics for ori

Mnemonic	Operands	Function	Other Registers Changed
nop		Preferred no-op; triggers optimizations based on no-ops. <i>Extended mnemonic for ori 0,0,0</i>	

oris RA, RS, IM



$(RA) \leftarrow (RS) \vee (IM \parallel 16_0)$

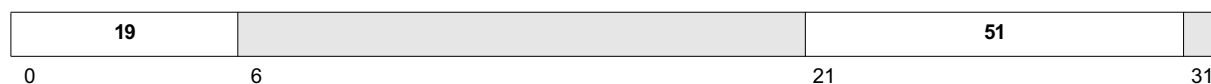
The IM Field is extended to 32 bits by concatenating 16 0-bits on the right. Register RS is ORed with the extended IM field and the result is placed into register RA.

Registers Altered

•RA

rfci

Return From Critical Interrupt

rfci $(PC) \leftarrow (CSRR0)$ $(MSR) \leftarrow (CSRR1)$

This instruction is used to return from a critical interrupt.

The program counter (PC) is restored with the contents of CSRR0 and the MSR is restored with the contents of CSRR1.

Instruction execution returns to the address contained in the PC.

Registers Altered

•MSR

Programming Note

Execution of this instruction is privileged and context-synchronizing (see *Context Synchronization* on page 196).

rfi



(PC) ← (SRR0)
(MSR) ← (SRR1)

This instruction is used to return from a non-critical interrupt.

The program counter (PC) is restored with the contents of SRR0 and the MSR is restored with the contents of SRR1.

Instruction execution returns to the address contained in the PC.

Registers Altered

- MSR

Invalid Instruction Forms

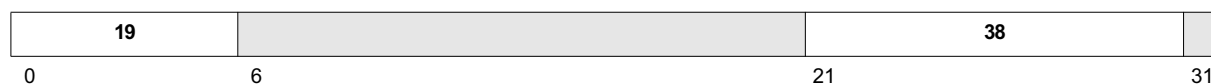
- Reserved fields

Programming Note

Execution of this instruction is privileged and context-synchronizing (see *Context Synchronization* on page 196).

rfmci

Return From Machine Check Interrupt

Preliminary User's Manual**rfmci** $(PC) \leftarrow (MCSRR0)$ $(MSR) \leftarrow (MCSRR1)$

This instruction is used to return from a machine check interrupt.

The program counter (PC) is restored with the contents of MCSRR0 and the MSR is restored with the contents of MCSRR1.

Instruction execution returns to the address contained in the PC.

Registers Altered

•MSR

Programming Note

Execution of this instruction is privileged and context-synchronizing (see “Context Synchronization” on page 196).

Preliminary User's Manual**rlwimi**

Rotate Left Word Immediate then Mask Insert

rlwimi RA, RS, SH, MB, ME Rc=0
rlwimi. RA, RS, SH, MB, ME Rc=1

20	RS	RA	SH	MB	ME	Rc
0	6	11	16	21	26	31

$r \leftarrow \text{ROTL}((RS), SH)$
 $m \leftarrow \text{MASK}(MB, ME)$
 $(RA) \leftarrow (r \wedge m) \vee ((RA) \wedge \neg m)$

The contents of register RS are rotated left by the number of bit positions specified in the SH field. A mask is generated, having 1-bits starting at the bit position specified in the MB field and ending in the bit position specified by the ME field, with 0-bits elsewhere.

If the starting point of the mask is at a higher bit position than the ending point, the 1-bits portion of the mask wraps from the highest bit position back around to the lowest. The rotated data is inserted into register RA, in positions corresponding to the bit positions in the mask that contain a 1-bit.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

Table 31-27. Extended Mnemonics for *rlwimi*, *rlwimi.*

Mnemonic	Operands	Function	Other Registers Altered
inslwi	RA, RS, n, b	Insert from left immediate (n > 0). $(RA)_{b:b+n-1} \leftarrow (RS)_{0:n-1}$ <i>Extended mnemonic for</i> rlwimi RA,RS,32-b,b,b+n-1	
inslwi.		<i>Extended mnemonic for</i> rlwimi. RA,RS,32-b,b,b+n-1	CR[CR0]
insrwi	RA, RS, n, b	Insert from right immediate. (n > 0) $(RA)_{b:b+n-1} \leftarrow (RS)_{32-n:31}$ <i>Extended mnemonic for</i> rlwimi RA,RS,32-b-n,b,b+n-1	
insrwi.		<i>Extended mnemonic for</i> rlwimi. RA,RS,32-b-n,b,b+n-1	CR[CR0]

rlwinm

Rotate Left Word Immediate then AND with Mask

Preliminary User's Manual

rlwinm RA, RS, SH, MB, ME Rc=0
rlwinm. RA, RS, SH, MB, ME Rc=1

21	RS	RA	SH	MB	ME	Rc
0	6	11	16	21	26	31

$r \leftarrow \text{ROTL}((RS), SH)$
 $m \leftarrow \text{MASK}(MB, ME)$
 $(RA) \leftarrow r \wedge m$

The contents of register RS are rotated left by the number of bit positions specified in the SH field. A mask is generated, having 1-bits starting at the bit position specified in the MB field and ending in the bit position specified by the ME field with 0-bits elsewhere.

If the starting point of the mask is at a higher bit position than the ending point, the 1-bits portion of the mask wraps from the highest bit position back around to the lowest. The rotated data is ANDed with the generated mask; the result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

Table 31-28. Extended Mnemonics for *rlwinm*, *rlwinm.*

Mnemonic	Operands	Function	Other Registers Altered
clrlwi	RA, RS, n	Clear left immediate. ($n < 32$) $(RA)_{0:n-1} \leftarrow n_0$ <i>Extended mnemonic for</i> rlwinm RA,RS,0,n,31	
clrlwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,0,n,31	CR[CR0]
clrlslwi	RA, RS, b, n	Clear left and shift left immediate. $(n \leq b < 32)$ $(RA)_{b-n:31-n} \leftarrow (RS)_{b:31}$ $(RA)_{32-n:31} \leftarrow n_0$ $(RA)_{0:b-n-1} \leftarrow b-n_0$ <i>Extended mnemonic for</i> rlwinm RA,RS,n,b-n,31-n	
clrlslwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,n,b-n,31-n	CR[CR0]
clrrwi	RA, RS, n	Clear right immediate. ($n < 32$) $(RA)_{32-n:31} \leftarrow n_0$ <i>Extended mnemonic for</i> rlwinm RA,RS,0,0,31-n	
clrrwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,0,0,31-n	CR[CR0]

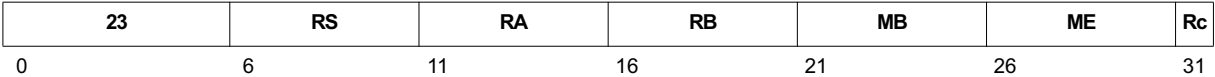
Table 31-28. Extended Mnemonics for rlwinm, rlwinm. (Continued)

Mnemonic	Operands	Function	Other Registers Altered
extlwi	RA, RS, n, b	Extract and left justify immediate. ($n > 0$) $(RA)_{0:n-1} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{n:31} \leftarrow 32-n_0$ <i>Extended mnemonic for</i> rlwinm RA,RS,b,0,n-1	
extlwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,b,0,n-1	CR[CR0]
extrwi	RA, RS, n, b	Extract and right justify immediate. ($n > 0$) $(RA)_{32-n:31} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{0:31-n} \leftarrow 32-n_0$ <i>Extended mnemonic for</i> rlwinm RA,RS,b+n,32-n,31	
extrwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,b+n,32-n,31	CR[CR0]
rotlwi	RA, RS, n	Rotate left immediate. $(RA) \leftarrow \text{ROTL}((RS), n)$ <i>Extended mnemonic for</i> rlwinm RA,RS,n,0,31	
rotlwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,n,0,31	CR[CR0]
rotrwi	RA, RS, n	Rotate right immediate. $(RA) \leftarrow \text{ROTL}((RS), 32-n)$ <i>Extended mnemonic for</i> rlwinm RA,RS,32-n,0,31	
rotrwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,32-n,0,31	CR[CR0]
slwi	RA, RS, n	Shift left immediate. ($n < 32$) $(RA)_{0:31-n} \leftarrow (RS)_{n:31}$ $(RA)_{32-n:31} \leftarrow n_0$ <i>Extended mnemonic for</i> rlwinm RA,RS,n,0,31-n	
slwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,n,0,31-n	CR[CR0]
srwi	RA, RS, n	Shift right immediate. ($n < 32$) $(RA)_{n:31} \leftarrow (RS)_{0:31-n}$ $(RA)_{0:n-1} \leftarrow n_0$ <i>Extended mnemonic for</i> rlwinm RA,RS,32-n,n,31	
srwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,32-n,n,31	CR[CR0]

rlwnm

Rotate Left Word then AND with Mask

rlwnm RA, RS, RB, MB, ME Rc=0
rlwnm. RA, RS, RB, MB, ME Rc=1



$r \leftarrow \text{ROTL}((RS), (RB)_{27:31})$
 $m \leftarrow \text{MASK}(MB, ME)$
 $(RA) \leftarrow r \wedge m$

The contents of register RS are rotated left by the number of bit positions specified by the contents of register RB_{27:31}. A mask is generated, having 1-bits starting at the bit position specified in the MB field and ending in the bit position specified by the ME field with 0-bits elsewhere.

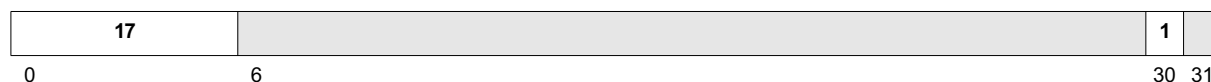
If the starting point of the mask is at a higher bit position than the ending point, the ones portion of the mask wraps from the highest bit position back to the lowest. The rotated data is ANDed with the generated mask and the result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

Table 31-29. Extended Mnemonics for rlwnm, rlwnm.

Mnemonic	Operands	Function	Other Registers Altered
rotlw	RA, RS, RB	Rotate left. (RA) ← ROTL((RS), (RB) _{27:31}) Extended mnemonic for rlwnm RA,RS,RB,0,31	
rotlw.		Extended mnemonic for rlwnm. RA,RS,RB,0,31	CR[CR0]

sc

$$\text{SRR1} \leftarrow \text{MSR}$$

$$\text{SRR0} \leftarrow 4 + \text{address of } \text{sc} \text{ instruction}$$

$$\text{PC} \leftarrow \text{IVPR}_{0:15} \parallel \text{IVOR8}_{16:27} \parallel 40$$

$$\text{MSR}[\text{WE}, \text{EE}, \text{PR}, \text{FP}, \text{FE0}, \text{FE1}, \text{DWE}, \text{DS}, \text{IS}] \leftarrow 0$$

A System Call exception is generated, and a System Call interrupt occurs (see *System Call Interrupt* on page 444 for more information on System Call interrupts). The contents of the MSR are copied into SRR1 and (4 + address of **sc** instruction) is placed into SRR0.

The program counter (PC) is then loaded with the interrupt vector address. The interrupt vector address is formed by concatenating the high halfword of the Interrupt Vector Prefix Register (IVPR), bits 16:27 of the Interrupt Vector Offset Register 8 (IVOR8), and 0b0000.

The MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS] bits are set to 0.

Program execution continues at the new address in the PC.

Registers Altered

- SRR0
- SRR1
- MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS]

Invalid Instruction Forms

- Reserved fields

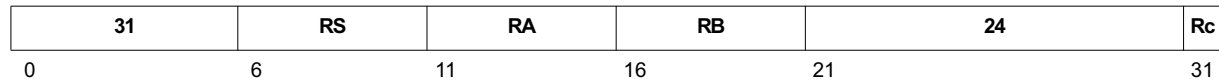
Programming Note

Execution of this instruction is context-synchronizing (see *Context Synchronization* on page 196).

slw
Shift Left Word

Preliminary User's Manual

slw RA, RS, RB Rc=0
slw. RA, RS, RB Rc=1



```

n ← (RB)26:31
r ← ROTL((RS), n)
if n < 32 then
    m ← MASK(0, 31 – n)
else
    m ← 320
(RA) ← r ∧ m

```

The contents of register RS are shifted left by the number of bits specified by the contents of register RB_{26:31}. Bits shifted left out of the most significant bit are lost, and 0-bits fill vacated bit positions on the right. The result is placed into register RA.

Note that if RB₂₆ = 1, then the shift amount is 32 bits or more, and thus all bits are shifted out such that register RA is set to zero.

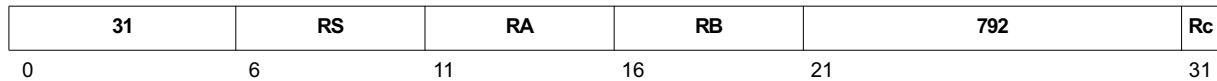
Registers Altered

- RA
- CR[CR0] if Rc contains 1

Preliminary User's Manual**sraw**

Shift Right Algebraic Word

sraw	RA, RS, RB	Rc=0
sraw.	RA, RS, RB	Rc=1



```

n ← (RB)26:31
r ← ROTL((RS), 32 – n)
if n < 32 then
    m ← MASK(n, 31)
else
    m ← 320
s ← (RS)0
(RA) ← (r ∧ m) ∨ (32s ∧ ¬m)
XER[CA] ← s ∧ ((r ∧ ¬m) ≠ 0)

```

The contents of register RS are shifted right by the number of bits specified the contents of register RB_{26:31}. Bits shifted out of the least significant bit are lost. Bit 0 of Register RS is replicated to fill the vacated positions on the left. The result is placed into register RA.

If register RS contains a negative number and any 1-bits were shifted out of the least significant bit position, XER[CA] is set to 1; otherwise, it is set to 0.

Note that if RB₂₆ = 1, then the shift amount is 32 bits or more, and thus all bits are shifted out such that register RA and XER[CA] are set to bit 0 of register RS.

Registers Altered

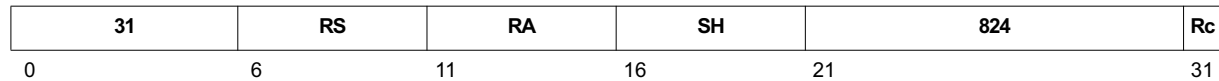
- RA
- XER[CA]
- CR[CR0] if Rc contains 1

srawi

Shift Right Algebraic Word Immediate

Preliminary User's Manual

srawi	RA, RS, SH	Rc=0
srawi.	RA, RS, SH	Rc=1



```

n ← SH
r ← ROTL((RS), 32 – n)
m ← MASK(n, 31)
s ← (RS)0
(RA) ← (r ∧ m) ∨ (32s ∧ ¬m)
XER[CA] ← s ∧ ((r ∧ ¬m) ≠ 0)

```

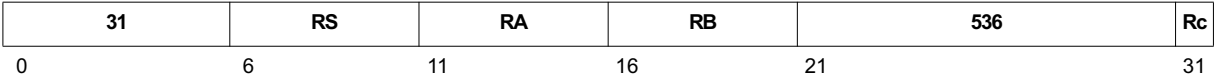
The contents of register RS are shifted right by the number of bits specified in the SH field. Bits shifted out of the least significant bit are lost. Bit RS₀ is replicated to fill the vacated positions on the left. The result is placed into register RA.

If register RS contains a negative number and any 1-bits were shifted out of the least significant bit position, XER[CA] is set to 1; otherwise, it is set to 0.

Registers Altered

- RA
- XER[CA]
- CR[CR0] if Rc contains 1

srw	RA, RS, RB	Rc=0
srw.	RA, RS, RB	Rc=1



```
n ← (RB)26:31
r ← ROTL((RS), 32 – n)
if n < 32 then
    m ← MASK(n, 31)
else
    m ← 320
(RA) ← r ∧ m
```

The contents of register RS are shifted right by the number of bits specified the contents of register RB_{26:31}. Bits shifted right out of the least significant bit are lost, and 0-bits fill the vacated bit positions on the left. The result is placed into register RA.

Note that if RB₂₆ = 1, then the shift amount is 32 bits or more, and thus all bits are shifted out such that register RA is set to zero.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

stb

Store Byte

Preliminary User's Manual**stb** RS, D(RA) $EA \leftarrow (RA|0) + EXTS(D)$ $MS(EA, 1) \leftarrow (RS)_{24:31}$

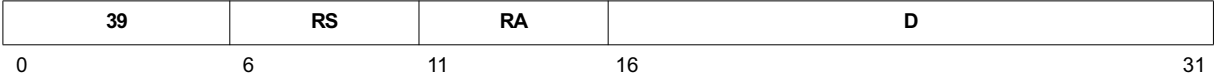
An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The least significant byte of register RS is stored into the byte at the EA.

Registers Altered

•None

stbu RS, D(RA)



$EA \leftarrow (RA|0) + EXTS(D)$
 $MS(EA, 1) \leftarrow (RS)_{24:31}$
 $(RA) \leftarrow EA$

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The least significant byte of register RS is stored into the byte at the EA.

The EA is placed into register RA.

Registers Altered

•RA

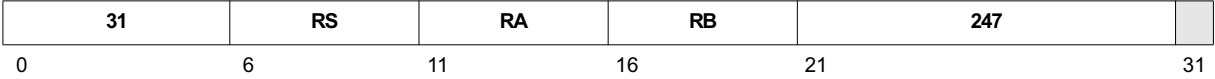
Invalid Instruction Forms

RA = 0

stbux

Store Byte with Update Indexed

stbux RS, RA, RB



$EA \leftarrow (RA|0) + (RB)$
 $MS(EA, 1) \leftarrow (RS)_{24:31}$
 $(RA) \leftarrow EA$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The least significant byte of register RS is stored into the byte at the EA.

The EA is placed into register RA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RA

Invalid Instruction Forms

- Reserved fields

RA = 0

stbx RS, RA, RB



$EA \leftarrow (RA|0) + (RB)$
 $MS(EA, 1) \leftarrow (RS)_{24:31}$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The least significant byte of register RS is stored into the byte at the EA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

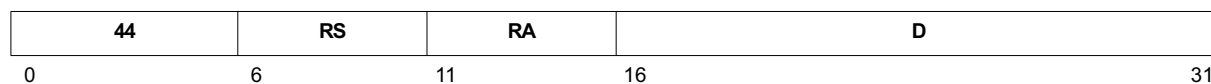
- None

Invalid Instruction Forms

- Reserved fields

sth

Store Halfword

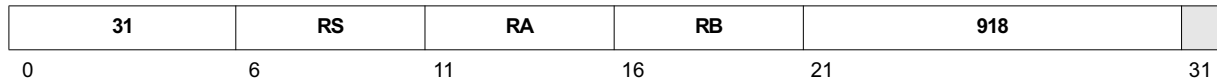
Preliminary User's Manual**sth** RS, D(RA) $EA \leftarrow (RA|0) + EXTS(D)$ $MS(EA, 2) \leftarrow (RS)_{16:31}$

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0 and is the contents of register RA otherwise.

The least significant halfword of register RS is stored into the halfword at the EA in main storage.

Registers Altered

•None

sthbrx RS, RA, RB

$$EA \leftarrow (RA|0) + (RB)$$

$$MS(EA, 2) \leftarrow \text{BYTE_REVERSE}((RS)_{16:31})$$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0 and is the contents of register RA otherwise.

The least significant halfword of register RS is byte-reversed from the default byte ordering for the memory page referenced by the EA. The resulting halfword is stored at the EA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Note

Byte ordering is generally controlled by the Endian (E) storage attribute (see *Memory Management* on page 235). The store byte reverse instructions provide a mechanism for data to be stored to a memory page using the opposite byte ordering from that specified by the Endian storage attribute.

sth

Store Halfword with Update

Preliminary User's Manual**sth** RS, D(RA) $EA \leftarrow (RA|0) + EXTS(D)$ $MS(EA, 2) \leftarrow (RS)_{16:31}$ $(RA) \leftarrow EA$

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The least significant halfword of register RS is stored into the halfword at the EA.

The EA is placed into register RA.

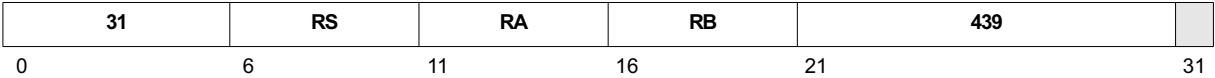
Registers Altered

•RA

Invalid Instruction Forms

RA = 0

sthux RS, RA, RB



$EA \leftarrow (RA|0) + (RB)$
 $MS(EA, 2) \leftarrow (RS)_{16:31}$
 $(RA) \leftarrow EA$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The least significant halfword of register RS is stored into the halfword at the EA.

The EA is placed into register RA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RA

Invalid Instruction Forms

- Reserved fields
- RA = 0

sthx

Store Halfword Indexed

Preliminary User's Manual**sthx** RS, RA, RB

$$EA \leftarrow (RA|0) + (RB)$$

$$MS(EA, 2) \leftarrow (RS)_{16:31}$$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The least significant halfword of register RS is stored into the halfword at the EA.

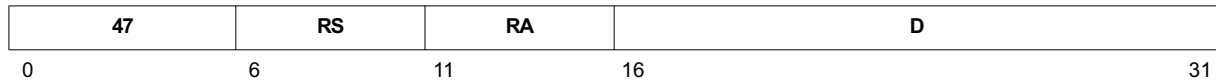
If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

stmw RS, D(RA)

```

EA ← (RA|0) + EXTS(D)
r ← RS
do while r ≤ 31
    MS(EA, 4) ← (GPR(r))
    r ← r + 1
    EA ← EA + 4

```

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The contents of a series of consecutive registers, starting with register RS and continuing through GPR(31), are stored into consecutive words starting at the EA.

Registers Altered

•None

Programming Note

This instruction can be restarted, meaning that it could be interrupted after having already stored some of the register values to memory, and then re-executed from the beginning (after returning from the interrupt), in which case the registers which had already been stored prior to the interrupt will be stored a second time.

stswi

Store String Word Immediate

Preliminary User's Manual

Store String Word Immediate

stswi RS, RA, NB

```

EA ← (RA|0)
if NB = 0 then
    n ← 32
else
    n ← NB
r ← RS - 1
i ← 0
do while n > 0
    if i = 0 then
        r ← r + 1
    if r = 32 then
        r ← 0
    MS(EA,1) ← (GPR(r))i:i+7
    i ← i + 8
    if i = 32 then
        i ← 0
    EA ← EA + 1
    n ← n - 1

```

An effective address (EA) is determined by the RA field. If the RA field contains 0, the EA is 0; otherwise, the EA is the contents of register RA.

A byte count is determined by the NB field. If the NB field contains 0, the byte count is 32; otherwise, the byte count is the contents of the NB field.

The contents of a series of consecutive GPRs (starting with register RS, continuing through GPR(31) and wrapping to GPR(0) as necessary, and continuing to the final byte count) are stored, starting at the EA. The bytes in each GPR are accessed starting with the most significant byte. The byte count determines the number of transferred bytes.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

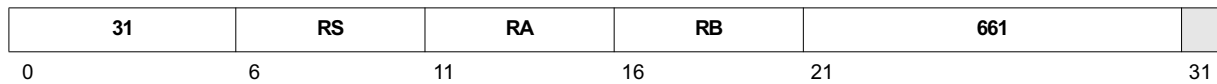
- None

Invalid Instruction Forms

- Reserved fields

Programming Note

This instruction can be restarted, meaning that it could be interrupted after having already stored some of the register values to memory, and then re-executed from the beginning (after returning from the interrupt), in which case the registers which had already been stored prior to the interrupt will be stored a second time.

stswx RS, RA, RB

```

EA ← (RA|0) + (RB)
n ← XER[TBC]
r ← RS - 1
i ← 0
do while n > 0
  if i = 0 then
    r ← r + 1
  if r = 32 then
    r ← 0
  MS(EA, 1) ← (GPR(r))i:i+7
  i ← i + 8
  if i = 32 then
    i ← 0
  EA ← EA + 1
  n ← n - 1

```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

A byte count is contained in XER[TBC].

The contents of a series of consecutive GPRs (starting with register RS, continuing through GPR(31) and wrapping to GPR(0) as necessary, and continuing to the final byte count) are stored, starting at the EA. The bytes in each GPR are accessed starting with the most significant byte. The byte count determines the number of transferred bytes.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Note

This instruction can be restarted, meaning that it could be interrupted after having already stored some of the register values to memory, and then re-executed from the beginning (after returning from the interrupt), in which case the registers which had already been stored prior to the interrupt will be stored a second time.

If XER[TBC] = 0, no GPRs are stored to memory, and **stswx** is treated as a no-op. Furthermore, if the EA is such that a Data Storage, Data TLB Error, or Data Address Compare Debug exception occurs, **stswx** is treated as a no-op and no interrupt occurs as a result of the exception.

stw

Store Word

Preliminary User's Manual**stw** RS, D(RA) $EA \leftarrow (RA|0) + EXTS(D)$ $MS(EA, 4) \leftarrow (RS)$

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The contents of register RS are stored at the EA.

Registers Altered

•None

stwbrx RS, RA, RB



$EA \leftarrow (RA|0) + (RB)$
 $MS(EA, 4) \leftarrow \text{BYTE_REVERSE}((RS)_{0:31})$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0 and is the contents of register RA otherwise.

The word in register RS is byte-reversed from the default byte ordering for the memory page referenced by the EA. The resulting word is stored at the EA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

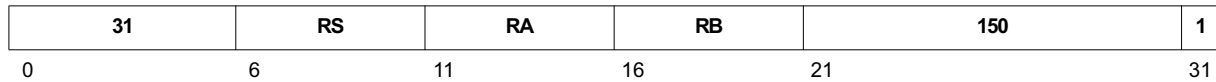
- Reserved fields

Programming Note

Byte ordering is generally controlled by the Endian (E) storage attribute (see *Memory Management* on page 235). The store byte reverse instructions provide a mechanism for data to be stored to a memory page using the opposite byte ordering from that specified by the Endian storage attribute.

stwcx.

Store Word Conditional Indexed

Preliminary User's Manual**stwcx.** RS, RA, RB

```

EA ← (RA|0) + (RB)
if RESERVE = 1 then
    MS(EA, 4) ← (RS)
    RESERVE ← 0
    (CR[CR0]) ← 200 || 1 || XER[SO]
else
    (CR[CR0]) ← 200 || 0 || XER[SO]

```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

If the reservation bit contains 1 when the instruction is executed, the contents of register RS are stored into the word at the EA and the reservation bit is cleared. If the reservation bit contains 0 when the instruction is executed, no store operation is performed.

CR[CR0] is set as follows:

- CR[CR0]_{0:1} are cleared
- CR[CR0]₂ is set to indicate whether or not the store was performed (1 indicates that it was)
- CR[CR0]₃ is set to the contents of the XER[SO] bit

Registers Altered

- CR[CR0]

Programming Notes

The **lwarx** and **stwcx.** instructions are typically paired in a loop, as shown in the following example, to create the effect of an atomic operation to a memory area used as a semaphore between multiple processes. Only **lwarx** can set the reservation bit to 1. **stwcx.** sets the reservation bit to 0 upon its completion, whether or not **stwcx.** actually stored (RS) to memory. CR[CR0]₂ must be examined to determine whether (RS) was sent to memory.

```

loop: lwarx      # read the semaphore from memory; set reservation
      "alter"   # change the semaphore bits in the register as required
      stwcx.    # attempt to store the semaphore; reset reservation
      bne loop  # some other process intervened and cleared the reservation prior to the above
              # stwcx.; try again

```

The PowerPC Book-E architecture specifies that the EA for the **lwarx** instruction must be word-aligned (that is, a multiple of 4 bytes); otherwise, the result is undefined. Although the PPC440GX Embedded Processor will execute **stwcx.** regardless of the EA alignment, in order for the operation of the pairing of **lwarx** and **stwcx.** to produce the desired result, software must ensure that the EA for both instructions is word-aligned. This requirement is due to the manner in which misaligned storage accesses may be broken up into separate, aligned accesses by the PPC440GX Embedded Processor.

The PowerPC Book-E architecture also specifies that it is implementation-dependent as to whether a Data Storage, Data TLB Error, Alignment, or Debug interrupt occurs when the reservation bit is off at the time of execution of an **stwcx.** instruction, and when the conditions are such that a non-**stwcx.** store-type storage access instruction would have resulted in such an interrupt. The PPC440GX Embedded Processor implements **stwcx.**

such that Data Storage and Debug (DAC and/or DVC exception type) interrupts do not occur when the reservation bit is off at the time of execution of the **stwcx.** Instead, the **stwcx.** instruction completes without causing the interrupt and without storing to memory, and CR[CR0] is updated to indicate the failure of the **stwcx.**

On the other hand, the PPC440GX Embedded Processor causes a Data TLB Error interrupt if a Data TLB Miss exception occurs during due to the execution of a **stwcx.** instruction, regardless of the state of the reservation. Similarly, the PPC440GX Embedded Processor causes an Alignment interrupt if the EA of the **stwcx.** operand is not word-aligned when CCR0[FLSTA] is 1, regardless of the state of the reservation (see *Core Configuration Register 0 (CCR0)* on page 212 for more information on the Force Load/Store Alignment function).

stwu

Store Word with Update

Preliminary User's Manual**stwu** RS, D(RA) $EA \leftarrow (RA|0) + EXTS(D)$ $MS(EA, 4) \leftarrow (RS)$ $(RA) \leftarrow EA$

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The contents of register RS are stored into the word at the EA.

The EA is placed into register RA.

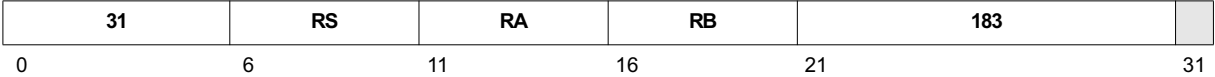
Registers Altered

•RA

Invalid Instruction Forms

RA = 0

stwux RS, RA, RB



$EA \leftarrow (RA|0) + (RB)$
 $MS(EA, 4) \leftarrow (RS)$
 $(RA) \leftarrow EA$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The contents of register RS are stored into the word at the EA.

The EA is placed into register RA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RA

Invalid Instruction Forms

- Reserved fields
- RA = 0

stwx

Store Word Indexed

Preliminary User's Manual**stwx** RS, RA, RB

$$EA \leftarrow (RA|0) + (RB)$$

$$MS(EA,4) \leftarrow (RS)$$

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

The contents of register RS are stored into the word at the EA.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Preliminary User's Manual

subf	RT, RA, RB	OE=0, Rc=0
subf.	RT, RA, RB	OE=0, Rc=1
subfo	RT, RA, RB	OE=1, Rc=0
subfo.	RT, RA, RB	OE=1, Rc=1



$(RT) \leftarrow \neg(RA) + (RB) + 1$

The sum of the ones complement of register RA, register RB, and 1 is stored into register RT.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Table 31-30. Extended Mnemonics for subf, subf., subfo, subfo.

Mnemonic	Operands	Function	Other Registers Altered
sub	RT, RA, RB	Subtract (RB) from (RA). $(RT) \leftarrow \neg(RB) + (RA) + 1$. <i>Extended mnemonic for subf RT,RB,RA</i>	
sub.		<i>Extended mnemonic for subf. RT,RB,RA</i>	CR[CR0]
subo		<i>Extended mnemonic for subfo RT,RB,RA</i>	XER[SO, OV]
subo.		<i>Extended mnemonic for subfo. RT,RB,RA</i>	CR[CR0] XER[SO, OV]

subfc

Subtract From Carrying

subfc	RT, RA, RB	OE=0, Rc=0
subfc.	RT, RA, RB	OE=0, Rc=1
subfco	RT, RA, RB	OE=1, Rc=0
subfco.	RT, RA, RB	OE=1, Rc=1



```
(RT) ← ¬(RA) + (RB) + 1
if ¬(RA) + (RB) + 1 > 232 - 1 then
    XER[CA] ← 1
else
    XER[CA] ← 0
```

The sum of the ones complement of register RA, register RB, and 1 is stored into register RT.
XER[CA] is set to a value determined by the unsigned magnitude of the result of the subtract operation.

Registers Altered

- RT
- XER[CA]
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Table 31-31. Extended Mnemonics for subfc, subfc., subfco, subfco.

Mnemonic	Operands	Function	Other Registers Altered
subc	RT, RA, RB	Subtract (RB) from (RA). (RT) ← ¬(RB) + (RA) + 1. Place carry-out in XER[CA]. <i>Extended mnemonic for subfc RT,RB,RA</i>	
subc.		<i>Extended mnemonic for subfc. RT,RB,RA</i>	CR[CR0]
subco		<i>Extended mnemonic for subfco RT,RB,RA</i>	XER[SO, OV]
subco.		<i>Extended mnemonic for subfco. RT,RB,RA</i>	CR[CR0] XER[SO, OV]

Preliminary User's Manual

subfe	RT, RA, RB	OE=0, Rc=0
subfe.	RT, RA, RB	OE=0, Rc=1
subfeo	RT, RA, RB	OE=1, Rc=0
subfeo.	RT, RA, RB	OE=1, Rc=1



```
(RT) ← ¬(RA) + (RB) + XER[CA]
if ¬(RA) + (RB) + XER[CA] > 232 - 1 then
    XER[CA] ← 1
else
    XER[CA] ← 0
```

The sum of the ones complement of register RA, register RB, and XER[CA] is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the subtract operation.

Registers Altered

- RT
- XER[CA]
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

subfic

Subtract From Immediate Carrying

Preliminary User's Manual**subfic** RT, RA, IM

```

(RT) ← ¬(RA) + EXTS(IM) + 1
if ¬(RA) + EXTS(IM) + 1 ≥ 232 - 1 then
    XER[CA] ← 1
else
    XER[CA] ← 0

```

The sum of the ones complement of RA, the IM field sign-extended to 32 bits, and 1 is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the subtract operation.

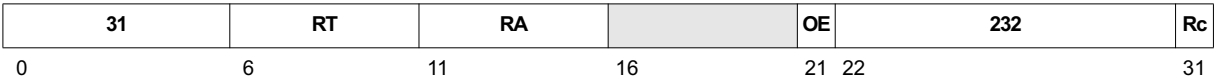
Registers Altered

- RT
- XER[CA]

Preliminary User's Manual

Subtract from Minus One Extended

subfme	RT, RA	OE=0, Rc=0
subfme.	RT, RA	OE=0, Rc=1
subfmeo	RT, RA	OE=1, Rc=0
subfmeo.	RT, RA	OE=1, Rc=1



```
(RT) ← ¬(RA) – 1 + XER[CA]
if ¬(RA) + 0xFFFF FFFF + XER[CA] ≥ 232 – 1 then
    XER[CA] ← 1
else
    XER[CA] ← 0
```

The sum of the ones complement of register RA, –1, and XER[CA] is placed into register RT.
XER[CA] is set to a value determined by the unsigned magnitude of the result of the subtract operation.

Registers Altered

- RT
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1
- XER[CA]

Invalid Instruction Forms

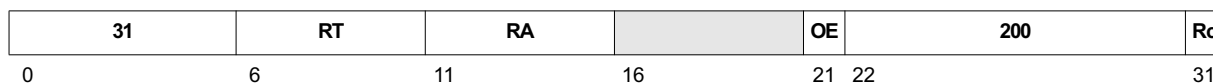
- Reserved fields

subfze

Subtract from Zero Extended

Preliminary User's Manual

subfze	RT, RA	OE=0, Rc=0
subfze.	RT, RA	OE=0, Rc=1
subfzeo	RT, RA	OE=1, Rc=0
subfzeo.	RT, RA	OE=1, Rc=1



```

(RT) ← ¬(RA) + XER[CA]
if ¬(RA) + XER[CA]  $\geq 2^{32} - 1$  then
    XER[CA] ← 1
else
    XER[CA] ← 0

```

The sum of the ones complement of register RA and XER[CA] is stored into register RT.

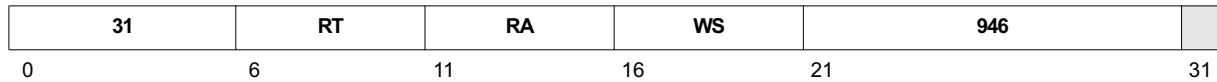
XER[CA] is set to a value determined by the unsigned magnitude of the result of the subtract operation.

Registers Altered

- RT
- XER[CA]
- CR[CR0] if Rc contains 1
- XER[SO, OV] if OE contains 1

Invalid Instruction Forms

- Reserved fields

tlbre RT, RA, WS

```

tlbentry ← TLB[(RA)26:31]
if WS = 0
    (RT)0:27 ← tlbentry[EPN,V,TS,SIZE]
    if CCR0[CRPE] = 0
        (RT)28:31 ← 40
    else
        (RT)28:31 ← TPAR
    MMUCR[STID] ← tlbentry[TID]
else if WS = 1
    (RT)0:21 ← tlbentry[RPN]
    if CCR0[CRPE] = 0
        (RT)22:23 ← 20
    else
        (RT)22:23 ← PAR1
        (RT)24:27 ← 40
        (RT)28:31 ← tlbentry[ERPN]
else if WS = 2
    if CCR0[CRPE] = 0
        (RT)0:1 ← 20
    else
        (RT)0:1 ← PAR2
        (RT)2:15 ← 140
        (RT)16:24 ← tlbentry[U0,U1,U2,U3,W,I,M,G,E]
        (RT)25 ← 0
        (RT)26:31 ← tlbentry[UX,UW,UR,SX,SW,SR]
else (RT), MMUCR[STID] ← undefined

```

The contents of the specified portion of the selected TLB entry are placed into register RT (and also MMUCR[STID] if WS = 0).

The parity bits in the TLB entry (TPAR, PAR1, and PAR2) are placed into the register RT if and only if the Cache Read Parity Enable bit, CCR0[CRPE], is set to 1.

The contents of RA are used as an index into the TLB. If this value is greater than the index of the highest numbered TLB entry (63), the results are undefined.

The WS field specifies which portion of the TLB entry is placed into RT. If WS = 0, the TID field of the selected TLB entry is read into MMUCR[STID]. See *Memory Management* on page 235 for descriptions of the TLB entry fields.

If the value of the WS field is greater than 2, the instruction form is invalid and the result is undefined.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT
- MMUCR[STID] (if WS = 0)

Invalid Instruction Forms

- Reserved fields
- Invalid WS value

Programming Notes

Execution of this instruction is privileged.

The PPC440GX does not automatically synchronize the context of the MMUCR[STID] field between a **tlbre** instruction which updates the field, and a **tlbsx[.]** instruction which uses it as a source operand. Therefore, software must execute an **isync** instruction between the execution of a **tlbre** instruction and a subsequent **tlbsx[.]** instruction to ensure that the **tlbsx[.]** instruction will use the new value of MMUCR[STID]. On the other hand, the PPC440GX *does* automatically synchronize the context of MMUCR[STID] between **tlbre** and **tlbwe**, as well as between **tlbre** and **mfspr** which specifies the MMUCR as the source SPR, so no **isync** is required in these cases.

tlbsx	RT, RA, RB	Rc=0
tlbsx.	RT, RA, RB	Rc=1

31	RT	RA	RB	914	Rc
0	6	11	16	21	31

```

EA ← (RA[0] + (RB))
if Rc = 1
    CR[CR0]₀ ← 0
    CR[CR0]₁ ← 0
    CR[CR0]₃ ← XER[SO]
if Valid TLB entry matching EA and MMUCR[STID,STS] is in the TLB then
    (RT) ← Index of matching TLB Entry
    if Rc = 1
        CR[CR0]₂ ← 1
else
    (RT) ← Undefined
    if Rc = 1
        CR[CR0]₂ ← 0

```

An effective address is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The TLB is searched for a valid entry which translates EA and MMUCR[STID,STS]. See *Memory Management* on page 235 for descriptions of the TLB fields and how they participate in the determination of a match. If a matching entry is found, its index (0 - 63) is placed into bits 26:31 of RT, and bits 0:25 are set to 0. If no match is found, the contents of RT are undefined.

The record bit (Rc) specifies whether the results of the search will affect CR[CR0] as shown above, such that CR[CR0]₂ can be tested if there is a possibility that the search may fail.

Registers Altered

- CR[CR0] if Rc contains 1

Invalid Instruction Forms

- None

Programming Notes

Execution of this instruction is privileged.

The PPC440GX does not automatically synchronize the context of the MMUCR[STID] field between a **tlbre** instruction which updates the field, and a **tlbsx[.]** instruction which uses it as a source operand. Therefore, software must execute an **isync** instruction between the execution of a **tlbre** instruction and a subsequent **tlbsx[.]** instruction to ensure that the **tlbsx[.]** instruction will use the new value of MMUCR[STID]. On the other hand, the PPC440GX *does* automatically synchronize the context of MMUCR[STID] between **tlbre** and **tlbwe**, as well as between **tlbre** and **mfspr** which specifies the MMUCR as the source SPR, so no **isync** is required in these cases.

tlbsync

The **tlbsync** instruction is provided by the PowerPC Book-E architecture to support synchronization of TLB operations between processors in a coherent multi-processor system. Since the PPC440GX does not support coherent multi-processing, this instruction performs no operation, and is provided only to facilitate code portability.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Note

This instruction is privileged. Translation is not required to be active during the execution of this instruction.

Since the PPC440GX does not support tightly-coupled multiprocessor systems, **tlbsync** performs no operation.

tlbwe RS, RA, WS

31	RS	RA	WS	978	
0	6	11	16	21	31

```

tlbentry ← TLB[(RA)26:31]
if WS = 0
    tlbentry[EPN,V,TS,SIZE] ← (RS)0:27
    tlbentry[TID] ← MMUCR[STID]
else if WS = 1
    tlbentry[RPN] ← (RS)0:21
    tlbentry[ERP] ← (RS)28:31
else if WS = 2
    tlbentry[U0,U1,U2,U3,W,I,M,G,E] ← (RS)16:24
    tlbentry[U,X,U,W,U,R,S,X,S,W,S,R] ← (RS)26:31
else tlbentry ← undefined

```

The contents of the specified portion of the selected TLB entry are replaced with the contents of register RS (and also MMUCR[STID] if WS = 0).

Parity check bits are automatically calculated and stored in the TLB entry as the tlbwe is executed. The contents of the RS register in the TPAR, PAR1, and PAR2 fields (for WS=0,1,or 2, respectively) is ignored by tlbwe; the parity is calculated from the other data bits being written to the TLB entry.

The contents of RA are used as an index into the TLB. If this value is greater than the index of the highest numbered TLB entry (63), the results are undefined.

The WS field specifies which portion of the TLB entry is replaced by the contents of RS. If WS = 0, the TID field of the selected TLB entry is replaced by the value in MMUCR[STID]. See *Memory Management* on page 235 for descriptions of the TLB entry fields.

If the value of the WS field is greater than 2, the instruction form is invalid and the result is undefined.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

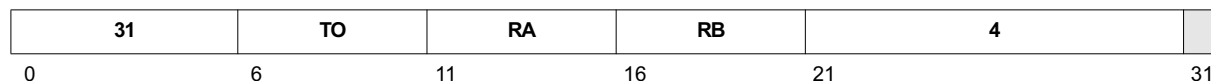
- Reserved fields
- Invalid WS value

Programming Note

Execution of this instruction is privileged.

tw

Trap Word

tw TO, RA, RB

```

if ( ((RA) < (RB) ∧ TO0 = 1) ∨
      ((RA) > (RB) ∧ TO1 = 1) ∨
      ((RA) = (RB) ∧ TO2 = 1) ∨
      ((RA) <sub>u (RB) ∧ TO3 = 1) ∨
      ((RA) >sub>u (RB) ∧ TO4 = 1) )
  SRR0 ← address of tw instruction
  SRR1 ← MSR
  ESR[PTR] ← 1 (other bits cleared)
  MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS] ← 90
  PC ← IVPR0:15 || IVOR616:27 || 40
else no operation

```

Register RA is compared with register RB. If any comparison condition enabled by the TO field is true, a Trap exception type Program interrupt occurs as follows (see *Program Interrupt* on page 443 for more information on Program interrupts). The contents of the MSR are copied into SRR1 and the address of the **tw** instruction is placed into SRR0. ESR[PTR] is set to 1 and the other bits ESR bits cleared to indicate the type of exception causing the Program interrupt.

The program counter (PC) is then loaded with the interrupt vector address. The interrupt vector address is formed by concatenating the high halfword of the Interrupt Vector Prefix Register (IVPR), bits 16:27 of the Interrupt Vector Offset Register 6 (IVOR6), and 0b0000.

MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS] are set to 0.

Program execution continues at the new address in the PC.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- SRR0 (if trap condition is met)
- SRR1 (if trap condition is met)
- MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS] (if trap condition is met)
- ESR (if trap condition is met)

Invalid Instruction Forms

- Reserved fields

Programming Notes

This instruction can be inserted into the execution stream by a debugger to implement breakpoints, and is not typically used by application code.

The enabling of trap debug events may affect the interrupt type caused by the execution of **tw** instruction. Specifically, trap instructions may be enabled to cause Debug interrupts instead of Program interrupts. See *Trap (TRAP) Debug Event* on page 520 for more details.

Preliminary User's Manual**tw**
Trap Word

Table 31-32. Extended Mnemonics for tw

Mnemonic	Operands	Function	Other Registers Altered
trap		Trap unconditionally. <i>Extended mnemonic for</i> tw 31,0,0	
tweq	RA, RB	Trap if (RA) equal to (RB). <i>Extended mnemonic for</i> tw 4,RA,RB	
twge	RA, RB	Trap if (RA) greater than or equal to (RB). <i>Extended mnemonic for</i> tw 12,RA,RB	
twgt	RA, RB	Trap if (RA) greater than (RB). <i>Extended mnemonic for</i> tw 8,RA,RB	
twle	RA, RB	Trap if (RA) less than or equal to (RB). <i>Extended mnemonic for</i> tw 20,RA,RB	
twlge	RA, RB	Trap if (RA) logically greater than or equal to (RB). <i>Extended mnemonic for</i> tw 5,RA,RB	
twlgt	RA, RB	Trap if (RA) logically greater than (RB). <i>Extended mnemonic for</i> tw 1,RA,RB	
twlle	RA, RB	Trap if (RA) logically less than or equal to (RB). <i>Extended mnemonic for</i> tw 6,RA,RB	
twllt	RA, RB	Trap if (RA) logically less than (RB). <i>Extended mnemonic for</i> tw 2,RA,RB	
twlng	RA, RB	Trap if (RA) logically not greater than (RB). <i>Extended mnemonic for</i> tw 6,RA,RB	
twlnl	RA, RB	Trap if (RA) logically not less than (RB). <i>Extended mnemonic for</i> tw 5,RA,RB	
twlt	RA, RB	Trap if (RA) less than (RB). <i>Extended mnemonic for</i> tw 16,RA,RB	
twne	RA, RB	Trap if (RA) not equal to (RB). <i>Extended mnemonic for</i> tw 24,RA,RB	
twng	RA, RB	Trap if (RA) not greater than (RB). <i>Extended mnemonic for</i> tw 20,RA,RB	
twnl	RA, RB	Trap if (RA) not less than (RB). <i>Extended mnemonic for</i> tw 12,RA,RB	

twi

Trap Word Immediate

twi

TO, RA, IM



```

if ( ((RA) < EXTS(IM) ∧ TO0 = 1) ∨
      ((RA) > EXTS(IM) ∧ TO1 = 1) ∨
      ((RA) = EXTS(IM) ∧ TO2 = 1) ∨
      ((RA) ≤ EXTS(IM) ∧ TO3 = 1) ∨
      ((RA) ≥ EXTS(IM) ∧ TO4 = 1) )
  SRR0 ← address of twi instruction
  SRR1 ← MSR
  ESR[PTR] ← 1 (other bits cleared)
  MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS] ← 0
  PC ← IVPR0:15 || IVOR616:27 || 0
else no operation

```

Register RA is compared with the sign-extended IM field. If any comparison condition selected by the TO field is true, a Trap exception type Program interrupt occurs as follows (see *Program Interrupt* on page 443 for more information on Program interrupts). The contents of the MSR are copied into SRR1 and the address of the **twi** instruction is placed into SRR0. ESR[PTR] is set to 1 and the other bits ESR bits cleared to indicate the type of exception causing the Program interrupt.

The program counter (PC) is then loaded with the interrupt vector address. The interrupt vector address is formed by concatenating the high halfword of the Interrupt Vector Prefix Register (IVPR), bits 16:27 of the Interrupt Vector Offset Register 6 (IVOR6), and 0b0000.

MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS] are set to 0.

Program execution continues at the new address in the PC.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- SRR0 (if trap condition is met)
- SRR1 (if trap condition is met)
- MSR[WE, EE, PR, FP, FE0, FE1, DWE, DS, IS] (if trap condition is met)
- ESR (if trap condition is met)

Invalid Instruction Forms

- Reserved fields

Programming Notes

This instruction can be inserted into the execution stream by a debugger to implement breakpoints, and is not typically used by application code.

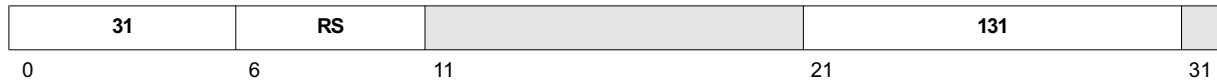
The enabling of trap debug events may affect the interrupt type caused by the execution of **tw** instruction. Specifically, trap instructions may be enabled to cause Debug interrupts instead of Program interrupts. See *Trap (TRAP) Debug Event* on page 520 for more details.

Table 31-33. Extended Mnemonics for twi

Mnemonic	Operands	Function	Other Registers Altered
tweqi	RA, IM	Trap if (RA) equal to EXTS(IM). <i>Extended mnemonic for</i> twi 4,RA,IM	
twgei	RA, IM	Trap if (RA) greater than or equal to EXTS(IM). <i>Extended mnemonic for</i> twi 12,RA,IM	
twgti	RA, IM	Trap if (RA) greater than EXTS(IM). <i>Extended mnemonic for</i> twi 8,RA,IM	
twlei	RA, IM	Trap if (RA) less than or equal to EXTS(IM). <i>Extended mnemonic for</i> twi 20,RA,IM	
twlgei	RA, IM	Trap if (RA) logically greater than or equal to EXTS(IM). <i>Extended mnemonic for</i> twi 5,RA,IM	
twlgti	RA, IM	Trap if (RA) logically greater than EXTS(IM). <i>Extended mnemonic for</i> twi 1,RA,IM	
twllei	RA, IM	Trap if (RA) logically less than or equal to EXTS(IM). <i>Extended mnemonic for</i> twi 6,RA,IM	
twlIti	RA, IM	Trap if (RA) logically less than EXTS(IM). <i>Extended mnemonic for</i> twi 2,RA,IM	
twlngi	RA, IM	Trap if (RA) logically not greater than EXTS(IM). <i>Extended mnemonic for</i> twi 6,RA,IM	
twlnli	RA, IM	Trap if (RA) logically not less than EXTS(IM). <i>Extended mnemonic for</i> twi 5,RA,IM	
twlti	RA, IM	Trap if (RA) less than EXTS(IM). <i>Extended mnemonic for</i> twi 16,RA,IM	
twnei	RA, IM	Trap if (RA) not equal to EXTS(IM). <i>Extended mnemonic for</i> twi 24,RA,IM	
twngi	RA, IM	Trap if (RA) not greater than EXTS(IM). <i>Extended mnemonic for</i> twi 20,RA,IM	
twnli	RA, IM	Trap if (RA) not less than EXTS(IM). <i>Extended mnemonic for</i> twi 12,RA,IM	

wrtee

Write External Enable

Preliminary User's Manual**wrtee** RS

$$\text{MSR}[\text{EE}] \leftarrow (\text{RS})_{16}$$

MSR[EE] is set to the value specified by bit 16 of register RS.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- MSR[EE]

Invalid Instruction Forms:

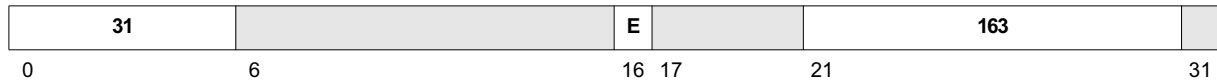
- Reserved fields

Programming Notes

Execution of this instruction is privileged.

This instruction is typically used as part of a code sequence which can provide the equivalent of an atomic read-modify-write of the MSR, as follows:

```
mfmsr Rn    #save EE in Rn[16]
wrteei 0     #Turn off EE (leaving other bits unchanged)
•           #Code with EE disabled
•
•
wrtee Rn     #restore EE without affecting any MSR changes that occurred in the disabled code
```

wrteei E

MSR[EE] ← E

MSR[EE] is set to the value specified by the E field.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

•MSR[EE]

Invalid Instruction Forms:

•Reserved fields

Programming Notes

Execution of this instruction is privileged.

This instruction is typically used as part of a code sequence which can provide the equivalent of an atomic read-modify-write of the MSR, as follows:

```

mfmsr Rn    #save EE in Rn[16]
wrteei 0     #Turn off EE (leaving other bits unchanged)
•           #Code with EE disabled
•
•
wrtee Rn     #restore EE without affecting any MSR changes that occurred in the disabled code

```

xor
XOR

Preliminary User's Manual

xor RA, RS, RB Rc=0
xor. RA, RS, RB Rc=1



$$(RA) \leftarrow (RS) \oplus (RB)$$

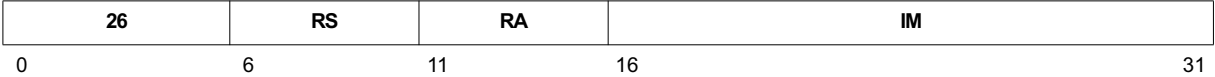
The contents of register RS are XORed with the contents of register RB; the result is placed into register RA.

Registers Altered

- RA
- CR[CR0] if Rc contains 1

Preliminary User's Manual

xori RA, RS, IM



$(RA) \leftarrow (RS) \oplus (^{16}0 \parallel IM)$

The IM field is extended to 32 bits by concatenating 16 0-bits on the left. The contents of register RS are XORed with the extended IM field; the result is placed into register RA.

Registers Altered

•RA

xoris

XOR Immediate Shifted

Preliminary User's Manual**xoris** RA, RS, IM

$$(RA) \leftarrow (RS) \oplus (IM \parallel 16_0)$$

The IM field is extended to 32 bits by concatenating 16 0-bits on the right. The contents of register RS are XORed with the extended IM field; the result is placed into register RA.

Registers Altered

•RA

Preliminary User's Manual

32. Register Summary

This chapter provides an alphabetical listing of and bit definitions for the registers contained in the PPC440GX.

The registers, of five types, are grouped into several functional categories according to the processor functions with which they are associated. More information about the registers and register categories is provided in *Section 4.2 Registers* on page 139, and in the chapters describing the processor functions with which each register category is associated.

32.1 Register Categories

Table 32-1 summarizes the register categories and the registers contained in each category. Italicized register names are implementation-specific. All other registers are defined by the Book-E Enhanced PowerPC Architecture.

Table 32-1. Register Categories

Register Category	Register(s)	Model and Access	Type	Page
Branch Control	CR	User	CR	183
	CTR	User	SPR	183
	LR	User	SPR	182
Cache Control	<i>DNV0–DNV3</i>	Supervisor	SPR	202
	<i>DTV0–DTV3</i>	Supervisor	SPR	202
	<i>DVLIM</i>	Supervisor	SPR	204
	<i>INV0–INV3</i>	Supervisor	SPR	202
	<i>ITV0–ITV3</i>	Supervisor	SPR	202
	<i>IVLIM</i>	Supervisor	SPR	204
Cache Debug	<i>DCDBTRH, DCDBTRL</i>	Supervisor, read-only	SPR	228
	<i>ICDBDR, ICDBTRH, ICDBTRL</i>	Supervisor, read-only	SPR	215
Debug	DAC1–DAC2	Supervisor	SPR	530
	DBCR0–DBCR2	Supervisor	SPR	524
	DBDR	Supervisor	SPR	531
	DBSR	Supervisor	SPR	528
	DVC1–DVC2	Supervisor	SPR	531
	IAC1–IAC4	Supervisor	SPR	530
Device Control	Implemented outside core	Supervisor	DCR	145
Integer Processing	GPR0–GPR31	User	GPR	186
	XER	User	SPR	187

Table 32-1. Register Categories

Register Category	Register(s)	Model and Access	Type	Page
Interrupt Processing	CSRR0–CSRR1	Supervisor	SPR	427
	DEAR	Supervisor	SPR	429
	ESR	Supervisor	SPR	431
	IVOR0–IVOR15	Supervisor	SPR	430
	IVPR	Supervisor	SPR	431
	MCSR	Supervisor	SPR	433
	MCSRR0–MCSRR1	Supervisor	SPR	421
	SRR0–SRR1	Supervisor	SPR	426
Processor Control	CCR0	Supervisor	SPR	212
	CCR1	Supervisor	SPR	212
	MSR	Supervisor	MSR	424
	PIR, PVR	Supervisor, read-only	SPR	190
	RSTCFG	Supervisor, read-only	SPR	194
	SPRG0–SPRG3	Supervisor	SPR	190
	SPRG4–SPRG7	User, read-only; Supervisor	SPR	190
	USPRG0	User	SPR	190
Storage Control	MMUCR	Supervisor	SPR	249
	PID	Supervisor	SPR	252
Timer	DEC	Supervisor	SPR	461
	DECAR	Supervisor, write-only	SPR	461
	TBL, TBU	User read, Supervisor write	SPR	460
	TCR	Supervisor	SPR	466
	TSR	Supervisor	SPR	467

Table 32-2 *Special Purpose Registers Sorted by SPR Number* on page 1217, lists the Special Purpose Registers (SPRs) in order by SPR number (SPRN). The table provides mnemonics, names, SPRN, model (user or supervisor), and access. All SPR numbers not listed are reserved, and should be neither read nor written.

Note that three registers, DBSR, MCSR, and TSR, are indicated as having the access type of *read/clear*. These three registers are *status* registers, and as such behave differently than other SPRs when written. The term “read/clear” does not mean that these registers are automatically cleared upon being read. Rather, the “clear” refers to their behavior when being written. Instead of simply overwriting the SPR with the data in the source GPR, the status SPR is updated by zeroing those bit positions corresponding to 1 values in the source GPR, with those bit positions corresponding to 0 values in the source GPR being left unchanged. In this fashion, it is possible for software to read one of these status SPRs, and then write to it using the same data which was read. Any bits which were read as 1 will then be cleared, and any bits which were not yet set at the time the SPR was read will be left unchanged. If any of these previously clear bits happen to be set between the time the SPR is read and when it is written, then when the SPR is later read again, software will observe any newly set bit[s]. If it were not for this behavior, then software could erroneously clear bits which it had not yet observed as having been set, and overlook the occurrence of certain exceptions.

Preliminary User's Manual*Table 32-2. Special Purpose Registers Sorted by SPR Number*

Mnemonic	Register Name	SPRN	Model	Access
XER	Integer Exception Register	0x001	User	Read/Write
LR	Link Register	0x008	User	Read/Write
CTR	Count Register	0x009	User	Read/Write
DEC	Decrementer	0x016	Supervisor	Read/Write
SRR0	Save/Restore Register 0	0x01A	Supervisor	Read/Write
SRR1	Save/Restore Register 1	0x01B	Supervisor	Read/Write
PID	Process ID	0x030	Supervisor	Read/Write
DECAR	Decrementer Auto-Reload	0x036	Supervisor	Write-only
CSRR0	Critical Save/Restore Register 0	0x03A	Supervisor	Read/Write
CSRR1	Critical Save/Restore Register 1	0x03B	Supervisor	Read/Write
DEAR	Data Exception Address Register	0x03D	Supervisor	Read/Write
ESR	Exception Syndrome Register	0x03E	Supervisor	Read/Write
IVPR	Interrupt Vector Prefix Register	0x03F	Supervisor	Read/Write
USPRG0	User Special Purpose Register General 0	0x100	User	Read/Write
SPRG4	Special Purpose Register General 4	0x104	User	Read-only
SPRG5	Special Purpose Register General 5	0x105	User	Read-only
SPRG6	Special Purpose Register General 6	0x106	User	Read-only
SPRG7	Special Purpose Register General 7	0x107	User	Read-only
TBL	Time Base Lower	0x10C	User	Read-only
TBU	Time Base Upper	0x10D	User	Read-only
SPRG0	Special Purpose Register General 0	0x110	Supervisor	Read/Write
SPRG1	Special Purpose Register General 1	0x111	Supervisor	Read/Write
SPRG2	Special Purpose Register General 2	0x112	Supervisor	Read/Write
SPRG3	Special Purpose Register General 3	0x113	Supervisor	Read/Write
SPRG4	Special Purpose Register General 4	0x114	Supervisor	Write-only
SPRG5	Special Purpose Register General 5	0x115	Supervisor	Write-only
SPRG6	Special Purpose Register General 6	0x116	Supervisor	Write-only
SPRG7	Special Purpose Register General 7	0x117	Supervisor	Write-only
TBL	Time Base Lower	0x11C	Supervisor	Write-only
TBU	Time Base Upper	0x11D	Supervisor	Write-only
PIR	Processor ID Register	0x11E	Supervisor	Read-only
PVR	Processor Version Register	0x11F	Supervisor	Read-only
DBSR	Debug Status Register	0x130	Supervisor	Read/Clear
DBCR0	Debug Control Register 0	0x134	Supervisor	Read/Write
DBCR1	Debug Control Register 1	0x135	Supervisor	Read/Write
DBCR2	Debug Control Register 2	0x136	Supervisor	Read/Write
IAC1	Instruction Address Compare 1	0x138	Supervisor	Read/Write
IAC2	Instruction Address Compare 2	0x139	Supervisor	Read/Write
IAC3	Instruction Address Compare 3	0x13A	Supervisor	Read/Write

Table 32-2. Special Purpose Registers Sorted by SPR Number (Continued)

Mnemonic	Register Name	SPRN	Model	Access
IAC4	Instruction Address Compare 4	0x13B	Supervisor	Read/Write
DAC1	Data Address Compare 1	0x13C	Supervisor	Read/Write
DAC2	Data Address Compare 2	0x13D	Supervisor	Read/Write
DVC1	Data Value Compare 1	0x13E	Supervisor	Read/Write
DVC2	Data Value Compare 2	0x13F	Supervisor	Read/Write
TSR	Timer Status Register	0x150	Supervisor	Read/Clear
TCR	Timer Control Register	0x154	Supervisor	Read/Write
IVOR0	Interrupt Vector Offset Register 0	0x190	Supervisor	Read/Write
IVOR1	Interrupt Vector Offset Register 1	0x191	Supervisor	Read/Write
IVOR2	Interrupt Vector Offset Register 2	0x192	Supervisor	Read/Write
IVOR3	Interrupt Vector Offset Register 3	0x193	Supervisor	Read/Write
IVOR4	Interrupt Vector Offset Register 4	0x194	Supervisor	Read/Write
IVOR5	Interrupt Vector Offset Register 5	0x195	Supervisor	Read/Write
IVOR6	Interrupt Vector Offset Register 6	0x196	Supervisor	Read/Write
IVOR7	Interrupt Vector Offset Register 7	0x197	Supervisor	Read/Write
IVOR8	Interrupt Vector Offset Register 8	0x198	Supervisor	Read/Write
IVOR9	Interrupt Vector Offset Register 9	0x199	Supervisor	Read/Write
IVOR10	Interrupt Vector Offset Register 10	0x19A	Supervisor	Read/Write
IVOR11	Interrupt Vector Offset Register 11	0x19B	Supervisor	Read/Write
IVOR12	Interrupt Vector Offset Register 12	0x19C	Supervisor	Read/Write
IVOR13	Interrupt Vector Offset Register 13	0x19D	Supervisor	Read/Write
IVOR14	Interrupt Vector Offset Register 14	0x19E	Supervisor	Read/Write
IVOR15	Interrupt Vector Offset Register 15	0x19F	Supervisor	Read/Write
MCSRR0	Machine Check Save Restore Register 0	0x23A	Supervisor	Read/Write
MCSRR1	Machine Check Save Restore Register 1	0x23B	Supervisor	Read/Write
MCSR	Machine Check Status Register	0x23C	Supervisor	Read/Clear
INV0	Instruction Cache Normal Victim 0	0x370	Supervisor	Read/Write
INV1	Instruction Cache Normal Victim 1	0x371	Supervisor	Read/Write
INV2	Instruction Cache Normal Victim 2	0x372	Supervisor	Read/Write
INV3	Instruction Cache Normal Victim 3	0x373	Supervisor	Read/Write
ITV0	Instruction Cache Transient Victim 0	0x374	Supervisor	Read/Write
ITV1	Instruction Cache Transient Victim 1	0x375	Supervisor	Read/Write
ITV2	Instruction Cache Transient Victim 2	0x376	Supervisor	Read/Write
ITV3	Instruction Cache Transient Victim 3	0x377	Supervisor	Read/Write
CCR1	Core Configuration Register 1	0x378	Supervisor	Read/Write
DNV0	Data Cache Normal Victim 0	0x390	Supervisor	Read/Write
DNV1	Data Cache Normal Victim 1	0x391	Supervisor	Read/Write
DNV2	Data Cache Normal Victim 2	0x392	Supervisor	Read/Write
DNV3	Data Cache Normal Victim 3	0x393	Supervisor	Read/Write

Preliminary User's Manual

Table 32-2. Special Purpose Registers Sorted by SPR Number (Continued)

Mnemonic	Register Name	SPRN	Model	Access
DTV0	Data Cache Transient Victim 0	0x394	Supervisor	Read/Write
DTV1	Data Cache Transient Victim 1	0x395	Supervisor	Read/Write
DTV2	Data Cache Transient Victim 2	0x396	Supervisor	Read/Write
DTV3	Data Cache Transient Victim 3	0x397	Supervisor	Read/Write
DVLIM	Data Cache Victim Limit	0x398	Supervisor	Read/Write
IVLIM	Instruction Cache Victim Limit	0x399	Supervisor	Read/Write
RSTCFG	Reset Configuration	0x39B	Supervisor	Read-only
DCDBTRL	Data Cache Debug Tag Register Low	0x39C	Supervisor	Read-only
DCDBTRH	Data Cache Debug Tag Register High	0x39D	Supervisor	Read-only
ICDBTRL	Instruction Cache Debug Tag Register Low	0x39E	Supervisor	Read-only
ICDBTRH	Instruction Cache Debug Tag Register High	0x39F	Supervisor	Read-only
MMUCR	Memory Management Unit Control Register	0x3B2	Supervisor	Read/Write
CCR0	Core Configuration Register 0	0x3B3	Supervisor	Read/Write
ICDBDR	Instruction Cache Debug Data Register	0x3D3	Supervisor	Read-only
DBDR	Debug Data Register	0x3F3	Supervisor	Read/Write

32.2 Reserved Fields

For all registers with fields marked as reserved, the reserved fields should be written as *zero* and read as *undefined*. That is, when writing to a reserved field, write a zero to that field. When reading from a reserved field, ignore that field.

The recommended coding practice is to perform the initial write to a register with reserved fields as described in the preceding paragraph, and to perform all subsequent writes to the register using a read-modify-write strategy: read the register, alter desired fields with logical instructions, and then write the register.

32.3 Device Control Registers

Device Control Registers (DCRs) are on-chip registers used to control, configure, and hold status for various functional units. DCRs are accessed using the **mfdcr** and **mtdcr** instructions.

The **mfdcr** and **mtdcr** instructions are privileged, for all DCR numbers. Therefore, all DCR accesses are privileged. All DCR numbers are reserved, and should be neither read nor written.

Some DCRs are directly accessed, that is, they are accessed using their DCR numbers. Other DCRs are indirectly accessed. Such DCRs are accessed by writing an offset to a directly accessed DCR and then reading the data at the offset in another directly accessed DCR. Table 32-3 lists the directly accessed DCRs. The indirectly accessed DCRs are described immediately following Table 32-3.

Table 32-3. Directly Accessed DCRs

Register	DCR Number	Access	Description
DCRs Used for Indirect Access			
CPR0_CFGADDR	0x00C	R/W	CPR Address Register
CPR0_CFGDATA	0x00D	R/W	CPR Data Register
SDR0_CFGADDR	0x00E	R/W	SDR Address Register
SDR0_CFGDATA	0x00F	R/W	SDR Data Register
SDRAM0_CFGADDR	0x010	R/W	DDR-SDRAM Address Register
SDRAM0_CFGDATA	0x011	R/W	DDR-SDRAM Data Register
EBC0_CFGADDR	0x012	R/W	EBCO Address Register
EBC0_CFGDATA	0x013	R/W	EBCO Data Register
EBM0_CFGADDR	0x014	R/W	EBMI Address Register
EBM0_CFGDATA	0x015	R/W	EBMI Data Register
PPM0_CFGADDR	0x016	R/W	PPM Address Register
PPM0_CFGDATA	0x017	R/W	PPM Data Register
Internal SRAM Controller			
SRAM0_SB0CR	0x020	R/W	SRAM Bank Configuration Register 0
SRAM0_SB1CR	0x021	R/W	SRAM Bank Configuration Register 1
SRAM0_SB2CR	0x022	R/W	SRAM Bank Configuration Register 2
SRAM0_SB3CR	0x023	R/W	SRAM Bank Configuration Register 3
SRAM0_BEAR	0x024	R/W	SRAM Bus Error Address Register
SRAM0_BESR0	0x025	R/W	SRAM Bus Error Status Register 0
SRAM0_BESR1	0x026	R/W	SRAM Bus Error Status Register 1
SRAM0_PMEG	0x027	R/W	SRAM Power Management
SRAM0_CID	0x028	R/W	SRAM Bus Core ID Register 1
SRAM0_REVID	0x029	R/W	SRAM Bus Revision ID Register
SRAM0_DPC	0x02A	R/W	SRAM Data Parity Check Register
L2 Cache Controller			
L2C0_CFG	0x030	R/W	L2 Cache Configuration Register
L2C0_CMD	0x031	R/W	L2 Cache Command Register
L2C0_ADDR	0x032	R/W	L2 Cache Address Register
L2C0_DATA	0x033	R	L2 Cache Data Register
L2C0_SR	0x034	R	L2 Cache Status Register
L2C0_REVID	0x035	R	L2 Cache Revision ID Register
L2C0_SNP0	0x036	R/W	L2 Cache Snoop Register 0
L2C0_SNP1	0x037	R/W	L2 Cache Snoop Register 1
On-Chip Buses			
PLB0_REVID	0x082	R	PLB Arbiter Revision ID
PLB0_ACR	0x083	R/W	PLB Arbiter Control Register
PLB0_BESR	0x084	R/Clear	PLB Bus Error Status Register

Preliminary User's Manual

Table 32-3. Directly Accessed DCRs (Continued)

Register	DCR Number	Access	Description
PLB0_BEARL	0x086	R	PLB Bus Error Address Register
PLB0_BEARH	0x087	R	PLB Bus Error Address Register
POB0_BESR0	0x090	R/Clear	PLB to OPB Bridge Error Status Register 0
POB0_BEARL	0x092	R	PLB to OPB Bridge Error Address Register
POB0_BEARH	0x093	R	PLB to OPB Bridge Error Address Register
POB0_BESR1	0x094	R/Clear	PLB to OPB Bridge Error Status Register 1
POB0_CONFIG	0x096	R/Clear	PLB to OPB Bridge Error Configuration Register
POB0_LATENCY	0x098	R/Clear	PLB to OPB Bridge Burst Latency Timer
POB0_REVID	0x09A	R	PLB to OPB Bridge Revision ID Register
OPB0_BCTRL	0x0A8	R/W	OPB to PLB Bridge Control Register
OPB0_BSTAT	0x0A9	R/C	OPB to PLB Bridge Status Register
OPB0_BEARL	0x0AA	R/C	OPB to PLB Bridge Error Address Register Low
OPB0_BEARH	0x0AB	R/C	OPB to PLB Bridge Error Address Register High
OPB0_REVID	0x0AC	R/C	OPB to PLB Bridge Revision ID Register
Clocking and Power Management			
CPM0_ER	0x0B0	R/W	CPM Enable Register
CPM0_FR	0x0B1	R/W	CPM Force Register
CPM0_SR	0x0B2	R/W	CPM Status Register
Universal Interrupt Controllers			
UIC0_SR	0x0C0	R/Clear	UIC 0 Status Register
UIC0_ER	0x0C2	R/W	UIC 0 Enable Register
UIC0_CR	0x0C3	R/W	UIC 0 Critical Register
UIC0_PR	0x0C4	R/W	UIC 0 Polarity Register
UIC0_TR	0x0C5	R/W	UIC 0 Triggering Register
UIC0_MSR	0x0C6	R	UIC 0 Masked Status Register
UIC0_VCR	0x0C7	R	UIC 0 Vector Register
UIC0_VR	0x0C8	W	UIC 0 Vector Configuration Register
UIC1_SR	0x0D0	R/Clear	UIC 1 Status Register
UIC1_ER	0x0D2	R/W	UIC 1 Enable Register
UIC1_CR	0x0D3	R/W	UIC 1 Critical Register
UIC1_PR	0x0D4	R/W	UIC 1 Polarity Register
UIC1_TR	0x0D5	R/W	UIC 1 Triggering Register
UIC1_MSR	0x0D6	R	UIC 1 Masked Status Register
UIC1_VCR	0x0D7	R	UIC 1 Vector Register
UIC1_VR	0x0D8	W	UIC 1 Vector Configuration Register
Direct Memory Access			
DMA0_CR0	0x100	R/W	DMA Channel Control Register 0
DMA0_CT0	0x101	R/W	DMA Count Register 0

Table 32-3. Directly Accessed DCRs (Continued)

Register	DCR Number	Access	Description
DMA0_SAH0	0x102	R/W	DMA Source Address High Register 0
DMA0_SAL0	0x103	R/W	DMA Source Address Low Register 0
DMA0_DAH0	0x104	R/W	DMA Destination Address High Register 0
DMA0_DAL0	0x105	R/W	DMA Destination Address Low Register 0
DMA0_SGH0	0x106	R/W	DMA Scatter/Gather Descriptor Address High Register 0
DMA0_SGL0	0x107	R/W	DMA Scatter/Gather Descriptor Address Low Register 0
DMA0_CR1	0x108	R/W	DMA Channel Control Register 1
DMA0_CT1	0x109	R/W	DMA Count Register 1
DMA0_SAH1	0x10A	R/W	DMA Source Address High Register 1
DMA0_SAL1	0x10B	R/W	DMA Source Address Low Register 1
DMA0_DAH1	0x10C	R/W	DMA Destination Address High Register 1
DMA0_DAL1	0x10D	R/W	DMA Destination Address Low Register 1
DMA0_SGH1	0x10E	R/W	DMA Scatter/Gather Descriptor Address High Register 1
DMA0_SGL1	0x10F	R/W	DMA Scatter/Gather Descriptor Address Low Register 1
DMA0_CR2	0x110	R/W	DMA Channel Control Register 2
DMA0_CT2	0x111	R/W	DMA Count Register 2
DMA0_SAH2	0x112	R/W	DMA Source Address High Register 2
DMA0_SAL2	0x113	R/W	DMA Source Address Low Register 2
DMA0_DAH2	0x114	R/W	DMA Destination Address High Register 2
DMA0_DAL2	0x115	R/W	DMA Destination Address Low Register 2
DMA0_SGH2	0x116	R/W	DMA Scatter/Gather Descriptor Address High Register 2
DMA0_SGL2	0x117	R/W	DMA Scatter/Gather Descriptor Address Low Register 2
DMA0_CR3	0x118	R/W	DMA Channel Control Register 3
DMA0_CT3	0x119	R/W	DMA Count Register 3
DMA0_SAH3	0x11A	R/W	DMA Source Address High Register 3
DMA0_SAL3	0x11B	R/W	DMA Source Address Low Register 3
DMA0_DAH3	0x11C	R/W	DMA Destination Address High Register 3
DMA0_DAL3	0x11D	R/W	DMA Destination Address Low Register 3
DMA0_SGH3	0x11E	R/W	DMA Scatter/Gather Descriptor Address High Register 3
DMA0_SGL3	0x11F	R/W	DMA Scatter/Gather Descriptor Address Low Register 3
DMA0_SR	0x120	R/Clear	DMA Status Register
DMA0_SGC	0x123	R/W	DMA Scatter/Gather Command Register
DMA0_SLP	0x125	R/W	DMA Sleep Mode Register
DMA0_POL	0x126	R/W	DMA Polarity Configuration Register
Memory Access Layer			
MAL0_CFG	0x180	R/W	MAL Configuration Register
MAL0_ESR	0x181	R/Clear	MAL Error Status Register
MAL0_IER	0x182	R/W	MAL Interrupt Enable Register

Preliminary User's Manual

Table 32-3. Directly Accessed DCRs (Continued)

Register	DCR Number	Access	Description
MAL0_TXCASR	0x184	R/W	MAL Tx Channel Active Register (Set)
MAL0_TXCARR	0x185	R/W	MAL Tx Channel Active Register (Reset)
MAL0_TXEOBISR	0x186	R/Clear	MAL Tx End of Buffer Interrupt Status Register
MAL0_TXDEIR	0x187	R/Clear	MAL Tx Descriptor Error Interrupt Register
MAL0_TXTATTRR	0x188	R/W	MAL Tx PLB Attribute Register
MAL0_TXBADDR	0x189	R/W	MAL Tx Descriptor Base Address Register
MAL0_RXCASR	0x190	R/W	MAL Rx Channel Active Register (Set)
MAL0_RXCARR	0x191	R/W	MAL Rx Channel Active Register (Reset)
MAL0_RXEOBISR	0x192	R/Clear	MAL Rx End of Buffer Interrupt Status Register
MAL0_RXDEIR	0x193	R/Clear	MAL Rx Descriptor Error Interrupt Register
MAL0_RXTATTRR	0x194	R/W	MAL Rx PLB Attribute Register
MAL0_RXBADDR	0x195	R/W	MAL Rx Descriptor Base Address Register
MAL0_TXCTP0R	0x1A0	R/W	MAL TX Channel 0 Table Pointer Register
MAL0_TXCTP1R	0x1A1	R/W	MAL TX Channel 1 Table Pointer Register
MAL0_TXCTP2R	0x1A2	R/W	MAL TX Channel 2 Table Pointer Register
MAL0_TXCTP3R	0x1A3	R/W	MAL TX Channel 3 Table Pointer Register
MAL0_RXCTP0R	0x1C0	R/W	MAL RX Channel 0 Table Pointer Register
MAL0_RXCTP1R	0x1C1	R/W	MAL RX Channel 1 Table Pointer Register
MAL0_RCBS0	0x1E0	R/W	MAL RX Channel 0 Buffer Size Register
MAL0_RCBS1	0x1E1	R/W	MAL RX Channel 1 Buffer Size Register
Universal Interrupt Controllers			
UICB0_SR	0x200	R/Clear	UIC Base Status Register
UICB0_ER	0x202	R/W	UIC Base Enable Register
UICB0_CR	0x203	R/W	UIC Base Critical Register
UICB0_PR	0x204	R/W	UIC Base Polarity Register
UICB0_TR	0x205	R/W	UIC Base Triggering Register
UICB0_MSR	0x206	R	UIC Base Masked Status Register
UICB0_VCR	0x207	R	UIC Base Vector Register
UICB0_VR	0x208	W	UIC Base Vector Configuration Register
UIC2_SR	0x210	R/Clear	UIC 2 Status Register
UIC2_ER	0x212	R/W	UIC 2 Enable Register
UIC2_CR	0x213	R/W	UIC 2 Critical Register
UIC2_PR	0x214	R/W	UIC 2 Polarity Register
UIC2_TR	0x215	R/W	UIC 2 Triggering Register
UIC2_MSR	0x216	R	UIC 2 Masked Status Register
UIC2_VCR	0x217	R	UIC 2 Vector Register
UIC2_VR	0x218	W	UIC 2 Vector Configuration Register

Indirectly Accessed DCRs

The DCRs for the Clocking and PowerOn Reset, System, DDR-SDRAM controller, external bus controller (EBCO), and external bus master interface (EBMI) are indirectly accessed.

Indirect Access of Clocking and PowerOn Reset DCRs

The following procedure accesses the clocking and power-on reset DCRs listed in *Table 32-4*.

1. Write the offset from *Table 32-5* to the Clocking and Power-on Reset Address Register (CPR0_CFGADDR).
2. Read data from or write data to the Clocking and Power-on Reset Data Register (CPR0_CFGDATA).

Table 32-4. Clocking and PowerOn Reset DCR Usage

Register	DCR Number	Access	Description
CPR0_CFGADDR	0x00C	R/W	Clocking and Power-On Reset Address Register
CPR0_CFGDATA	0x00D	R/W	Clocking and Power-On Reset Data Register

Table 32-5. Offsets for Clocking and PowerOn Reset Registers

Register	Offset	R/W	Description
CPR0_CLKUPD	0x0020	R/W	Clocking Update Register
CPR0_PLLC	0x0040	R/W	PLL Control Register
CPR0_PLLD	0x0060	R/W	PLL Divisor Register
CPR0_PRIMAD	0x0080	R/W	Primary Divisor Register A
CPR0_PRIMBD	0x00A0	R/W	Primary Divisor Register B
CPR0_OPBD	0x00C0	R/W	OPB Clock Divisor Register
CPR0_PERD	0x00E0	R/W	Peripheral Clock Divisor Register
CPR0_MALD	0x0100	R/W	MAL Clock Divisor Register
CPR0_ICFG	0x0140	R/W	Initial Configuration Register

Indirect Access of System DCRs

The following procedure accesses the system DCRs listed in *Table 32-6*.

1. Write the offset from *Table 32-7* to the System DCR Address Register (SDR0_CFGADDR).
2. Read data from or write data to the System DCR Data Register (SDR0_CFGDATA).

Table 32-6. System DCR Usage

Register	DCR Number	Access	Description
SDR0_CFGADDR	0x00E	R/W	System DCR Address Register
SDR0_CFGDATA	0x00F	R/W	System DCR Data Register

Preliminary User's Manual

Table 32-7. Offsets for System Device Control Registers

Register	Offset	R/W	Description
SDR0_SDSTP0	0x0020	R	Serial Device Strap Register 0
SDR0_SDSTP1	0x0021	R	Serial Device Strap Register 1
SDR0_PINSTP	0x0040	R	Pin Strapping Register
SDR0_SDCS	0x0060	R/W	Serial Device Controller Settings Register
SDR0_ECID0	0x0080	R/W	Electronic Chip ID Register 0
SDR0_ECID1	0x0081	R/W	Electronic Chip ID Register 1
SDR0_ECID2	0x0082	R/W	Electronic Chip ID Register 2
SDR0_JTAG	0x00C0	R/W	JTAG ID Register
SDR0_DDRDL	0x00E0	R/W	DDR Delay Line Register
SDR0_EBC	0x0100	R/W	EBC Configuration Register
SDR0_UART0	0x0120	R/W	UART Configuration Register 0
SDR0_UART1	0x0121	R/W	UART Configuration Register 1
SDR0_CP440	0x0180	R/W	CPU Control Register
SDR0_XCR	0x01C0	R/W	PCIX Control Register
SDR0_XPLLC	0x01C1	R/W	PLL Control Register
SDR0_XPLLD	0x01C2	R/W	PLL Divisor Register
SDR0_SRST	0x0200	R/W	Soft Reset Register
SDR0_SLPIPE	0x0220	R/W	Slave Address Pipeline Register
SDR0_AMP	0x0240	R/W	Alternate PLB Master Priority Register
SDR0_MIRQ0	0x0260	R/W	Master Interrupt Request Register 0
SDR0_MIRQ1	0x0261	R/W	Master Interrupt Request Register 1
SDR0_MALTBL	0x0280	R/W	MAL Transmit Burst Length Register
SDR0_MALRBL	0x02A0	R/W	MAL Receive Burst Length Register
SDR0_MALTBS	0x02C0	R/W	MAL Transmit Bus Size Register
SDR0_MALRBS	0x02E0	R/W	MAL Receive Bus Size Register
SDR0_CUST0	0x4000	R/W	Customer Configuration Register 0
SDR0_SDSTP2	0x4001	R	Serial Device Strap Register 2
SDR0_CUST1	0x4002	R/W	Customer Configuration Register 1
SDR0_SDSTP3	0x4003	R	Serial Device Strap Register 3
SDR0_PFC0	0x4100	R/W	Pin Function Control Register 0
SDR0_PFC1	0x4101	R/W	Pin Function Control Register 1
SDR0_PLBTR	0x4200	R/W	PLB Trace Register
SDR0_MFR	0x4300	R/W	Miscellaneous Function Register

Indirect Access of DDR-SDRAM Controller DCRs

The following procedure accesses the DDR-SDRAM controller DCRs listed in *Table 32-8*.

1. Write the offset from *Table 32-9* to the DDR-SDRAM Address Register (SDRAM0_CFGADDR).
2. Read data from or write data to the DDR-SDRAM Data Register (SDRAM0_CFGDATA).

Table 32-8. SDRAM Controller DCR Usage

Register	DCR Number	Access	Description
SDRAM0_CFGADDR	0x010	R/W	DDR-SDRAM Address Register
SDRAM0_CFGDATA	0x011	R/W	DDR-SDRAM Data Register

Table 32-9. Offsets for SDRAM Controller Registers

Register	Offset	R/W	Description
SDRAM0_BESR0	0x00	R/W	DDR-SDRAM Bus Error Syndrome Register 0
SDRAM0_BESR1	0x08	R/W	DDR-SDRAM Bus Error Syndrome Register 1
SDRAM0_BEAR	0x10	R	DDR-SDRAM Bus Error Address Register
SDRAM0_MIRQ	0x11	R/W	DDR-SDRAM Master Write Interrupt
SDRAM0_SLIO	0x18	R/W	DDR-SDRAM Slave Interface Options
SDRAM0_CFG0	0x20	R/W	DDR-SDRAM Options 0
SDRAM0_CFG1	0x21	R/W	DDR-SDRAM Options 1
SDRAM0_DEVOPT	0x22	R/W	DDR-SDRAM Device Options
SDRAM0_MCSTS	0x24	R	DDR-SDRAM Memory Controller Status
SDRAM0_RTR	0x30	R/W	DDR-SDRAM Refresh Timer Register
SDRAM0_PMIT	0x34	R/W	DDR-SDRAM Power Management Idle Timer
SDRAM0_UABBA	0x38	R/W	DDR-SDRAM PLB UA Bus Base Address
SDRAM0_B0CR	0x40	R/W	DDR-SDRAM Bank 0 Configuration Register
SDRAM0_B1CR	0x44	R/W	DDR-SDRAM Bank 1 Configuration Register
SDRAM0_B2CR	0x48	R/W	DDR-SDRAM Bank 2 Configuration Register
SDRAM0_B3CR	0x4C	R/W	DDR-SDRAM Bank 3 Configuration Register
SDRAM0_TR0	0x80	R/W	DDR-SDRAM Timing Register 0
SDRAM0_TR1	0x81	R/W	DDR-SDRAM Timing Register 1
SDRAM0_CLKTR	0x82	R/W	DDR-SDRAM Clock Timing Register
SDRAM0_WDDCTR	0x83	R/W	DDR-SDRAM Write Data, DQS, DM Clock Timing Register
SDRAM0_DLYCAL	0x84	R/W	DDR-SDRAM Delay Line Calibration Register
SDRAM0_ECCESR	0x98	R/W	DDR-SDRAM ECC Error Status Register
SDRAM0_CID	0xA4	R	DDR-SDRAM Core ID Register
SDRAM0_RID	0xA8	R	DDR-SDRAM Revision ID Register

Preliminary User's Manual**Indirect Access of External Bus Controller DCRs**

The following procedure accesses the EBCO DCRs listed in *Table 32-10*.

1. Write the offset from *Table 32-11* to the EBCO Address Register (EBC0_CFGADDR).
2. Read data from or write data to the EBCO Data Register (EBC0_CFGDATA).

Table 32-10. External Bus Controller DCR Usage

Register	DCR Number	Access	Description
EBC0_CFGADDR	0x012	R/W	EBCO Controller Address Register
EBC0_CFGDATA	0x013	R/W	EBCO Controller Data Register

Indirect Access of External Bus Controller DCRs

Table 32-11. Offsets for External Bus Controller Registers

Register	Offset	Access	Description
EBC0_B0CR	0x00	R/W	EBCO Bank 0 Configuration Register
EBC0_B1CR	0x01	R/W	EBCO Bank 1 Configuration Register
EBC0_B2CR	0x02	R/W	EBCO Bank 2 Configuration Register
EBC0_B3CR	0x03	R/W	EBCO Bank 3 Configuration Register
EBC0_B4CR	0x04	R/W	EBCO Bank 4 Configuration Register
EBC0_B5CR	0x05	R/W	EBCO Bank 5 Configuration Register
EBC0_B6CR	0x06	R/W	EBCO Bank 6 Configuration Register
EBC0_B7CR	0x07	R/W	EBCO Bank 7 Configuration Register
EBC0_B0AP	0x10	R/W	EBCO Bank 0 Access Parameters
EBC0_B1AP	0x11	R/W	EBCO Bank 1 Access Parameters
EBC0_B2AP	0x12	R/W	EBCO Bank 2 Access Parameters
EBC0_B3AP	0x13	R/W	EBCO Bank 3 Access Parameters
EBC0_B4AP	0x14	R/W	EBCO Bank 4 Access Parameters
EBC0_B5AP	0x15	R/W	EBCO Bank 5 Access Parameters
EBC0_B6AP	0x16	R/W	EBCO Bank 6 Access Parameters
EBC0_B7AP	0x17	R/W	EBCO Bank 7 Access Parameters
EBC0_BEAR	0x20	R/W	EBCO Bus Error Address Register
EBC0_BESR	0x21	R/W	EBCO Bus Error Status Register 0
EBC0_CFG	0x23	R/W	EBCO Peripheral Control Register
EBC0_CID	0x24	R/W	EBCO Peripheral Core ID Register

Indirect Access of External Bus Master DCRs

The following procedure accesses the EBMI DCRs listed in *Table 32-12*.

1. Write the offset from *Table 32-13* to the EBMI Address Register (EBM0_CFGADDR).
2. Read data from or write data to the EBMI Data Register (EBM0_CFGDATA).

Table 32-12. External Bus Master DCR Usage

Register	DCR Number	Access	Description
EBM0_CFGADDR	0x014	R/W	EBMI Controller Address Register
EBM0_CFGDATA	0x015	R/W	EBMI Controller Data Register

Table 32-13. Offsets for External Bus Master Registers

Register	Offset	Access	Description
EBM0_CTL	0x00	R/W	EBMI Control Register
EBM0_LCNT	0x01	R/W	EBMI OPB Latency Count Register
EBM0_BEAR	0x02	R/W	EBMI Bus Error Address Register
EBM0_BESR	0x03	R/W	EBMI Bus Error Status Register
EBM0_BEMR	0x04	R/W	EBMI Bus Error Mask Register
EBM0_UAR	0x05	R/W	EBMI OPB Upper Address Register
EBM0_UAM	0x06	R/W	EBMI OPB Upper Address Mask
EBM0_SLPMD	0x07	R/W	EBMI Sleep Mode Register
EBM0_FAIR	0x08	R/W	EBMI Fairness Control Register
EBM0_CID	0x11	R	EBMI Core ID Register

Indirect Access of PLB Performance Monitor DCRs

The following procedure accesses the PPM DCRs listed in *Table 32-14*.

1. Write the offset from *Table 32-15* to the PPM Address Register (PPM0_CFGADDR).
2. Read data from or write data to the PPM Data Register (PPM0_CFGDATA).

Table 32-14. PLB Performance Monitor DCR Usage

Register	DCR Number	Access	Description
PPM0_CFGADDR	0x016	R/W	PPM Address Register
PPM0_CFGDATA	0x017	R/W	PPM Data Register

Preliminary User's Manual

Table 32-15. Offsets for PLB Performance Monitor Registers

Register	Offset	Access	Description
PPM0_ISR	0x0	Read	Interrupt Status Register
PPM0_CR	0x2	R/W	Control Register
PPM0_CCR	0x3	R/W	Cycle Control Register
PPM0_UAR	0x4	R/W	Upper Address Register
PPM0_LAR	0x5	R/W	Lower Address Register
PPM0_UAMR	0x6	R/W	Upper Address Mask Register
PPM0_LAMR	0x7	R/W	Lower Address Mask Register
PPM0_RIDR	0x8	R	Revision ID Register
PPM0_MCSR0	0x9	R/W	Master Event Counter Selection Register 0
PPM0_MCSR1	0xA	R/W	Master Event Counter Selection Register 1
PPM0_MCSR2	0xB	R/W	Master Event Counter Selection Register 2
PPM0_MCSR3	0xC	R/W	Master Event Counter Selection Register 3
PPM0_SCSR0	0x11	R/W	Slave Event Counter Selection Register 0
PPM0_SCSR1	0x12	R/W	Slave Event Counter Selection Register 1
PPM0_SCSR2	0x13	R/W	Slave Event Counter Selection Register 2
PPM0_SCSR3	0x14	R/W	Slave Event Counter Selection Register 3
PPM0_GCSR0	0x19	R/W	Generic Event Counter Selection Register 0
PPM0_GCSR1	0x1A	R/W	Generic Event Counter Selection Register 1
PPM0_GCSR2	0x1B	R/W	Generic Event Counter Selection Register 2
PPM0_GCSR3	0x1C	R/W	Generic Event Counter Selection Register 3
PPM0_MCR0	0x1D	R/W	Master Event Counter Register 0
PPM0_MCR1	0x1E	R/W	Master Event Counter Register 1
PPM0_MCR2	0x1F	R/W	Master Event Counter Register 2
PPM0_MCR3	0x20	R/W	Master Event Counter Register 3
PPM0_SCR0	0x25	R/W	Slave Event Counter Register 0
PPM0_SCR1	0x26	R/W	Slave Event Counter Register 1
PPM0_SCR2	0x27	R/W	Slave Event Counter Register 2
PPM0_SCR3	0x28	R/W	Slave Event Counter Register 3
PPM0_GCR0	0x2D	R/W	Generic Pipeline Event Counter Register 0
PPM0_GCR1	0x2E	R/W	Generic Pipeline Event Counter Register 1
PPM0_GCR2	0x2F	R/W	Generic Pipeline Event Counter Register 2
PPM0_GCR3	0x30	R/W	Generic Pipeline Event Counter Register 3
PPM0_DCSR0	0x31	R/W	Duration Counter Selection Register 0
PPM0_DCSR1	0x32	R/W	Duration Counter Selection Register 1
PPM0_DCMXR0	0x33	R/W	Duration Counter Max Register 0
PPM0_DCMXR1	0x34	R/W	Duration Counter Max Register 1
PPM0_DCMNR0	0x35	R/W	Duration Counter Min Register 0
PPM0_DCMNR1	0x36	R/W	Duration Counter Minimum Register 1
PPM0_DCTVR0	0x37	R/W	Duration Counter Total Value Register 0

Table 32-15. Offsets for PLB Performance Monitor Registers (Continued)

Register	Offset	Access	Description
PPM0_DCTVR1	0x38	R/W	Duration Counter Total Value Register 1
PPM0_DCOTR0	0x39	R/W	Duration Counter Occurrence Total Register 0
PPM0_DCOTR1	0x3A	R/W	Duration Counter Occurrence Total Register 1

Indirect Access of Gigabit Mode Physical Coding Sublayer (GPCS) DCRs

All GPCS 16 bit registers listed in *Table 32-16* are accessed via EMACx_STACR[PHYD] bits 0:15 which reflect the data to be sent to the PHY if the command is a write, or data is read from the PHY if the command is a read.

Table 32-16. Offsets for Gigabit Mode Physical Coding Sublayer (GPCS) Registers

Register	Offset	Access	Description
GPCSx_CR	0x00	R/W	GPCS Control Register
GPCSx_SR	0x01	R/W	GPCS Status Register
GPCSx_ID0	0x02	R	GPCS ID0 Register
GPCSx_ID1	0x03	R	GPCS ID1 Register
GPCSx_ANAR	0x04	R/W	GPCS Auto Negotiation Advertisement Register
GPCSx_ANLR	0x05	R	GPCS Auto Negotiation Link Partner Base Page Ability Register
GPCSx_ANER	0x06	R	GPCS Auto Negotiation Expansion Register
GPCSx_ANPTR	0x07	R	GPCS Auto Negotiation Next Page Transmit Register
GPCSx_ANLNPR	0x08	R	GPCS Auto Negotiation Link Partner Ability Next Page Register
GPCSx_ESR	0x0F	R	GPCS Extended Status Register
GPCSx_RAR	0x10	R	GPCS Resolved Ability Register
GPCSx_ISR	0x11	R/W	GPCS Interrupt Status Register
GPCSx_ISER	0x12	R/W	GPCS Interrupt Status Enable Register
GPCSx_CFG	0x13	R/W	GPCS Configuration Register

Preliminary User's Manual**32.4 Memory-Mapped Input/Output Registers**

Some registers associated with on-chip peripherals are memory-mapped input/output (MMIO) registers. Such registers are mapped into the system memory space and are accessed using load/store instructions that contain the register addresses. The tables that follow list all MMIO registers.

Table 32-17. MMIO Registers

Register	Address	Access	Description
I2O Messaging Unit (IMU)			
IMU0_IMR0	0x0 FFFF0090	R/W	IMU Inbound Message Register 0
IMU0_IMR1	0x0 FFFF0094	R/W	IMU Inbound Message Register 1
IMU0_OMR0	0x0 FFFF0098	R/W	IMU Outbound Message Register 0
IMU0_OMR1	0x0 FFFF009C	R/W	IMU Outbound Message Register 1
IMU0_IDR	0x0 FFFF00A0	R/W	IMU Inbound Doorbell Register
IMU0_IISR	0x0 FFFF00A4	R/W	IMU Inbound Interrupt Status Register
IMU0_IIMR	0x0 FFFF00A8	R/W	IMU Inbound Interrupt Mask Register
IMU0_ODR	0x0 FFFF00AC	R/W	IMU Outbound Doorbell Register
IMU0_OISR	0x0 FFFF00B0	R/W	IMU Outbound Interrupt Status Register
IMU0_OIMR	0x0 FFFF00B4	R/W	IMU Outbound Interrupt Mask Register
IMU0_IQPR	0x0 FFFF00B8	R/W	IMU Inbound Queue Port Register
IMU0_OQPR	0x0 FFFF00BC	R/W	IMU Outbound Queue Port Register
IMU0_CFG	0x0 FFFF00D0	R/W	IMU Configuration Register
IMU0_QBAHR	0x0 FFFF00D4	R/W	IMU Queue Base Address Register High
IMU0_QBALR	0x0 FFFF00D8	R/W	IMU Queue Base Address Register Low
IMU0_IFHPR	0x0 FFFF00E0	R/W	IMU Inbound Free Head Pointer Register
IMU0_IFTP	0x0 FFFF00E4	R/W	IMU Inbound Free Tail Pointer Register
IMU0_IPHPR	0x0 FFFF00E8	R/W	IMU Inbound Post Head Pointer Register
IMU0_IPTPR	0x0 FFFF00EC	R/W	IMU Inbound Post Tail Pointer Register
IMU0_OFHPR	0x0 FFFF00F0	R/W	IMU Outbound Free Head Pointer Register
IMU0_OTPR	0x0 FFFF00F4	R/W	IMU Outbound Free Tail Pointer Register
IMU0_OPHPR	0x0 FFFF00F8	R/W	IMU Outbound Post Head Pointer Register
IMU0_OTPR	0x0 FFFF00FC	R/W	IMU Outbound Post Tail Pointer Register
IMU0_BESR	0x0 FFFF0100	R/W	IMU Bus Error Status Register
IMU0_BEMR	0x0 FFFF0104	R/W	IMU Bus Error Mask Register
IMU0_BEAR	0x0 FFFF0108	R/W	IMU Bus Error Address register
IMU0_MIRQ	0x0 FFFF010C	R/W	IMU Master Interrupt Request Register
IMU0_PPR	0x0 FFFF0110	R/W	IMU PLB Programmable Register
IMU0_SLP	0x0 FFFF0130	R/W	IMU Sleep Register
IMU0_REVID	0x0 FFFF0134	R/W	IMU Revision ID Register
IMU0_SR	0x0 FFFF0138	R/W	IMU Status Register

Table 32-17. MMIO Registers (Continued)

Register	Address	Access	Description
Serial Port 0			
UART0_RBR	0x1 40000200	R	UART 0 Receiver Buffer Register Note: Set UART0_LCR[DLAB] = 0 to access.
UART0_THR		W	UART 0 Transmitter Holding Register Note: Set UART0_LCR[DLAB] = 0 to access.
UART0_DLL		R/W	UART 0 Baud-rate Divisor Latch LSB Note: Set UART0_LCR[DLAB] = 1 to access.
UART0_IER	0x1 40000201	R/W	UART 0 Interrupt Enable Register Note: Set UART0_LCR[DLAB] = 0 to access.
UART0_DLM		R/W	UART 0 Baud-rate Divisor Latch MSB Note: Set UART0_LCR[DLAB] = 1 to access.
UART0_IIR	0x1 40000202	R	UART 0 Interrupt Identification Register
UART0_FCR	0x1 40000202	W	UART 0 FIFO Control Register
UART0_LCR	0x1 40000203	R/W	UART 0 Line Control Register
UART0_MCR	0x1 40000204	R/W	UART 0 Modem Control Register
UART0_LSR	0x1 40000205	R/W	UART 0 Line Status Register
UART0_MSR	0x1 40000206	R/W	UART 0 Modem Status Register
UART0_SCR	0x1 40000207	R/W	UART 0 Scratch Register
Serial Port 1			
UART1_RBR	0x1 40000300	R	UART 1 Receiver Buffer Register Note: Set UART1_LCR[DLAB] = 0 to access.
UART1_THR		W	UART 1 Transmitter Holding Register Note: Set UART1_LCR[DLAB] = 0 to access.
UART1_DLL		R/W	UART 1 Baud-rate Divisor Latch LSB Note: Set UART1_LCR[DLAB] = 1 to access.
UART1_IER	0x1 40000301	R/W	UART 1 Interrupt Enable Register Note: Set UART1_LCR[DLAB] = 0 to access.
UART1_DLM		R/W	UART 1 Baud-rate Divisor Latch MSB Note: Set UART1_LCR[DLAB] = 1 to access.
UART1_IIR	0x1 40000302	R	UART 1 Interrupt Identification Register
UART1_FCR	0x1 40000302	W	UART 1 FIFO Control Register
UART1_LCR	0x1 40000303	R/W	UART 1 Line Control Register
UART1_MCR	0x1 40000304	R/W	UART 1 Modem Control Register
UART1_LSR	0x1 40000305	R/W	UART 1 Line Status Register
UART1_MSR	0x1 40000306	R/W	UART 1 Modem Status Register
UART1_SCR	0x1 40000307	R/W	UART 1 Scratch Register
Inter-Integrated Circuit 0			
IIC0_MDBUF	0x1 40000400	R/W	IIC 0 Master Data Buffer
IIC0_SDBUF	0x1 40000402	R/W	IIC 0 Slave Data Buffer
IIC0_LMADR	0x1 40000404	R/W	IIC 0 Low Master Address
IIC0_HMADR	0x1 40000405	R/W	IIC 0 High Master Address
IIC0_CNTL	0x1 40000406	R/W	IIC 0 Control

Preliminary User's Manual

Table 32-17. MMIO Registers (Continued)

Register	Address	Access	Description
IIC0_MDCNTL	0x1 40000407	R/W	IIC 0 Mode Control
IIC0_STS	0x1 40000408	R/W	IIC 0 Status
IIC0_EXTSTS	0x1 40000409	R/W	IIC 0 Extended Status
IIC0_LSADR	0x1 4000040A	R/W	IIC 0 Low Slave Address
IIC0_HSADR	0x1 4000040B	R/W	IIC 0 High Slave Address
IIC0_CLKDIV	0x1 4000040C	R/W	IIC 0 Clock Divide
IIC0_INTRMSK	0x1 4000040D	R/W	IIC 0 Interrupt Mask
IIC0_XFRCNT	0x1 4000040E	R/W	IIC 0 Transfer Count
IIC0_XTCNTLSS	0x1 4000040F	R/W	IIC 0 Extended Control and Slave Status
IIC0_DIRECTCNTL	0x1 40000410	R/W	IIC 0 Direct Control
Inter-Integrated Circuit 1			
IIC1_MDBUF	0x1 40000500	R/W	IIC 1 Master Data Buffer
IIC1_SDBUF	0x1 40000502	R/W	IIC 1 Slave Data Buffer
IIC1_LMADR	0x1 40000504	R/W	IIC 1 Low Master Address
IIC1_HMADR	0x1 40000505	R/W	IIC 1 High Master Address
IIC1_CNTL	0x1 40000506	R/W	IIC 1 Control
IIC1_MDCNTL	0x1 40000507	R/W	IIC 1 Mode Control
IIC1_STS	0x1 40000508	R/W	IIC 1 Status
IIC1_EXTSTS	0x1 40000509	R/W	IIC 1 Extended Status
IIC1_LSADR	0x1 4000050A	R/W	IIC 1 Low Slave Address
IIC1_HSADR	0x1 4000050B	R/W	IIC 1 High Slave Address
IIC1_CLKDIV	0x1 4000050C	R/W	IIC 1 Clock Divide
IIC1_INTRMSK	0x1 4000050D	R/W	IIC 1 Interrupt Mask
IIC1_XFRCNT	0x1 4000050E	R/W	IIC 1 Transfer Count
IIC1_XTCNTLSS	0x1 4000050F	R/W	IIC 1 Extended Control and Slave Status
IIC1_DIRECTCNTL	0x1 40000510	R/W	IIC 1 Direct Control
OPB Arbiter			
OPBA0_PR	0x1 40000600	R/W	OPB Arbiter Priority Register
OPBA0_CR	0x1 40000601	R/W	OPB Arbiter Control Register
General-Purpose I/O			
GPIO0_OR	0x1 40000700	R/W	GPIO Output Register
GPIO0_TCR	0x1 40000704	R/W	GPIO Three-State Control Register
GPIO0_ODR	0x1 40000718	R/W	GPIO Open Drain Register
GPIO0_IR	0x1 4000071C	R	GPIO Input Register
Ethernet to PHY Bridge (ZMII)			
ZMII0_FER	0x1 40000780	R/W	ZMII Function Enable Register
ZMII0_SSR	0x1 40000784	R/W	ZMII Speed Select Register
ZMII0_SMIISR	0x1 40000788	R/W	ZMII SMII Status Register
Ethernet to PHY Bridge (RGMII)			

Table 32-17. MMIO Registers (Continued)

Register	Address	Access	Description
RGMII0_FER	0x1 40000790	R/W	RGMII Function Enable Register
RGMII0_SSR	0x1 40000794	R/W	RGMII Speed Select Register
Ethernet MAC 0			
EMAC0_MR0	0x1 40000800	R/W	EMAC 0 Mode Register 0
EMAC0_MR1	0x1 40000804	R/W	EMAC 0 Mode Register 1
EMAC0_TMR0	0x1 40000808	R/W	EMAC 0 Transmit Mode Register 0
EMAC0_TMR1	0x1 4000080C	R/W	EMAC 0 Transmit Mode Register 1
EMAC0_RMR	0x1 40000810	R/W	EMAC 0 Receive Mode Register
EMAC0_ISR	0x1 40000814	R/W	EMAC 0 Interrupt Status Register
EMAC0_ISER	0x1 40000818	R/W	EMAC 0 Interrupt Status Enable Register
EMAC0_IAHR	0x1 4000081C	R/W	EMAC 0 Individual Address High
EMAC0_IALR	0x1 40000820	R/W	EMAC 0 Individual Address Low
EMAC0_VTPID	0x1 40000824	R/W	EMAC 0 VLAN TPID Register
EMAC0_VTCI	0x1 40000828	R/W	EMAC 0 VLAN TCI Register
EMAC0_PTR	0x1 4000082C	R/W	EMAC 0 Pause Timer Register
EMAC0_IAHT1	0x1 40000830	R/W	EMAC 0 Individual Address Hash Table 1
EMAC0_IAHT2	0x1 40000834	R/W	EMAC 0 Individual Address Hash Table 2
EMAC0_IAHT3	0x1 40000838	R/W	EMAC 0 Individual Address Hash Table 3
EMAC0_IAHT4	0x1 4000083C	R/W	EMAC 0 Individual Address Hash Table 4
EMAC0_GAHT1	0x1 40000840	R/W	EMAC 0 Group Address Hash Table 1
EMAC0_GAHT2	0x1 40000844	R/W	EMAC 0 Group Address Hash Table 2
EMAC0_GAHT3	0x1 40000848	R/W	EMAC 0 Group Address Hash Table 3
EMAC0_GAHT4	0x1 4000084C	R/W	EMAC 0 Group Address Hash Table 4
EMAC0_LSAH	0x1 40000850	R	EMAC 0 Last Source Address Low
EMAC0_LSAH	0x1 40000854	R	EMAC 0 Last Source Address High
EMAC0_IPGVR	0x1 40000858	R/W	EMAC 0 Inter-Packet Gap Value Register
EMAC0_STACR	0x1 4000085C	R/W	EMAC 0 STA Control Register
EMAC0_TRTR	0x1 40000860	R/W	EMAC 0 Transmit Request Threshold Register
EMAC0_RWMR	0x1 40000864	R/W	EMAC 0 Receive Low/High Water Mark Register
EMAC0_OCTX	0x1 40000868	R	EMAC 0 Number of Octets Transmitted
EMAC0_OCRX	0x1 4000086C	R	EMAC 0 Number of Octets Received
Ethernet MAC 1			
EMAC1_MR0	0x1 40000900	R/W	EMAC 1 Mode Register 0
EMAC1_MR1	0x1 40000904	R/W	EMAC 1 Mode Register 1
EMAC1_TMR0	0x1 40000908	R/W	EMAC 1 Transmit Mode Register 0
EMAC1_TMR1	0x1 4000090C	R/W	EMAC 1 Transmit Mode Register 1
EMAC1_RMR	0x1 40000910	R/W	EMAC 1 Receive Mode Register
EMAC1_ISR	0x1 40000914	R/W	EMAC 1 Interrupt Status Register
EMAC1_ISER	0x1 40000918	R/W	EMAC 1 Interrupt Status Enable Register

Preliminary User's Manual

Table 32-17. MMIO Registers (Continued)

Register	Address	Access	Description
EMAC1_IAHR	0x1 4000091C	R/W	EMAC 1 Individual Address High
EMAC1_IALR	0x1 40000920	R/W	EMAC 1 Individual Address Low
EMAC1_VTPID	0x1 40000924	R/W	EMAC 1 VLAN TPID Register
EMAC1_VTCI	0x1 40000928	R/W	EMAC 1 VLAN TCI Register
EMAC1_PTR	0x1 4000092C	R/W	EMAC 1 Pause Timer Register
EMAC1_IAHT1	0x1 40000930	R/W	EMAC 1 Individual Address Hash Table 1
EMAC1_IAHT2	0x1 40000934	R/W	EMAC 1 Individual Address Hash Table 2
EMAC1_IAHT3	0x1 40000938	R/W	EMAC 1 Individual Address Hash Table 3
EMAC1_IAHT4	0x1 4000093C	R/W	EMAC 1 Individual Address Hash Table 4
EMAC1_GAHT1	0x1 40000940	R/W	EMAC 1 Group Address Hash Table 1
EMAC1_GAHT2	0x1 40000944	R/W	EMAC 1 Group Address Hash Table 2
EMAC1_GAHT3	0x1 40000948	R/W	EMAC 1 Group Address Hash Table 3
EMAC1_GAHT4	0x1 4000094C	R/W	EMAC 1 Group Address Hash Table 4
EMAC1_LSAH	0x1 40000950	R	EMAC 1 Last Source Address Low
EMAC1_LSAH	0x1 40000950	R	EMAC 1 Last Source Address High
EMAC1_IPGVR	0x1 40000958	R/W	EMAC 1 Inter-Packet Gap Value Register
EMAC1_STACR	0x1 4000095C	R/W	EMAC 1 STA Control Register
EMAC1_TRTR	0x1 40000960	R/W	EMAC 1 Transmit Request Threshold Register
EMAC1_RWMR	0x1 40000964	R/W	EMAC 1 Receive Low/High Water Mark Register
EMAC1_OCTX	0x1 40000968	R	EMAC 1 Number of Octets Transmitted
EMAC1_OCRX	0x1 4000096C	R	EMAC 1 Number of Octets Received
General Purpose Timer			
GPT0_TBC	0x1 40000A00	R/W	GPT Time Base Counter
GPT0_IM	0x1 40000A18	R/W	GPT Interrupt Mask
GPT0_ISS	0x1 40000A1C	R/W	GPT Interrupt Status (Set bits if write 1)
GPT0_ISC	0x1 40000A20	R/W	GPT Interrupt Status (Clear bits if write 1)
GPT0_IE	0x1 40000A24	R/W	GPT Interrupt Enable
GPT0_COMP0	0x1 40000A80	R/W	GPT Compare Timer 0
GPT0_COMP1	0x1 40000A84	R/W	GPT Compare Timer 1
GPT0_COMP2	0x1 40000A88	R/W	GPT Compare Timer 2
GPT0_COMP3	0x1 40000A8C	R/W	GPT Compare Timer 3
GPT0_COMP4	0x1 40000A90	R/W	GPT Compare Timer 4
GPT0_COMP5	0x1 40000A94	R/W	GPT Compare Timer 5
GPT0_COMP6	0x1 40000A98	R/W	GPT Compare Timer 6
GPT0_MASK0	0x1 40000AC0	R/W	GPT Compare Mask 0
GPT0_MASK1	0x1 40000AC4	R/W	GPT Compare Mask 1
GPT0_MASK2	0x1 40000AC8	R/W	GPT Compare Mask 2
GPT0_MASK3	0x1 40000ACC	R/W	GPT Compare Mask 3
GPT0_MASK4	0x1 40000AD0	R/W	GPT Compare Mask 4

Table 32-17. MMIO Registers (Continued)

Register	Address	Access	Description
GPT0_MASK5	0x1 40000AD4	R/W	GPT Compare Mask 5
GPT0_MASK6	0x1 40000AD8	R/W	GPT Compare Mask 6
TCP/IP Accelerator on Hardware 0			
TAH0_MR0	0x1 40000B00	R/W	TAH 0 Mode Register 0
TAH0_SSR0	0x1 40000B04	R/W	TAH 0 Segment Size Register 0
TAH0_SSR1	0x1 40000B08	R/W	TAH 0 Segment Size Register 1
TAH0_SSR2	0x1 40000B0C	R/W	TAH 0 Segment Size Register 2
TAH0_SSR3	0x1 40000B10	R/W	TAH 0 Segment Size Register 3
TAH0_SSR4	0x1 40000B14	R/W	TAH 0 Segment Size Register 4
TAH0_SSR5	0x1 40000B18	R/W	TAH 0 Segment Size Register 5
TAH0_SR	0x1 40000B1C	R/W	TAH 0 Status Register
Ethernet MAC 2			
EMAC2_MR0	0x1 40000C00	R/W	EMAC 2 Mode Register 0
EMAC2_MR1	0x1 40000C04	R/W	EMAC 2 Mode Register 1
EMAC2_TMR0	0x1 40000C08	R/W	EMAC 2 Transmit Mode Register 0
EMAC2_TMR1	0x1 40000C0C	R/W	EMAC 2 Transmit Mode Register 1
EMAC2_RMR	0x1 40000C10	R/W	EMAC 2 Receive Mode Register
EMAC2_ISR	0x1 40000C14	R/W	EMAC 2 Interrupt Status Register
EMAC2_ISER	0x1 40000C18	R/W	EMAC 2 Interrupt Status Enable Register
EMAC2_IAHR	0x1 40000C1C	R/W	EMAC 2 Individual Address High
EMAC2_IALR	0x1 40000C20	R/W	EMAC 2 Individual Address Low
EMAC2_VTPID	0x1 40000C24	R/W	EMAC 2 VLAN TPID Register
EMAC2_VTCI	0x1 40000C28	R/W	EMAC 2 VLAN TCI Register
EMAC2_PTR	0x1 40000C2C	R/W	EMAC 2 Pause Timer Register
EMAC2_IAHT1	0x1 40000C30	R/W	EMAC 2 Individual Address Hash Table 1
EMAC2_IAHT2	0x1 40000C34	R/W	EMAC 2 Individual Address Hash Table 2
EMAC2_IAHT3	0x1 40000C38	R/W	EMAC 2 Individual Address Hash Table 3
EMAC2_IAHT4	0x1 40000C3C	R/W	EMAC 2 Individual Address Hash Table 4
EMAC2_GAHT1	0x1 40000C40	R/W	EMAC 2 Group Address Hash Table 1
EMAC2_GAHT2	0x1 40000C44	R/W	EMAC 2 Group Address Hash Table 2
EMAC2_GAHT3	0x1 40000C48	R/W	EMAC 2 Group Address Hash Table 3
EMAC2_GAHT4	0x1 40000C4C	R/W	EMAC 2 Group Address Hash Table 4
EMAC2_LSAH	0x1 40000C50	R	EMAC 2 Last Source Address Low
EMAC2_LSAL	0x1 40000C54	R	EMAC 2 Last Source Address High
EMAC2_IPGVR	0x1 40000C58	R/W	EMAC 2 Inter-Packet Gap Value Register
EMAC2_STACR	0x1 40000C5C	R/W	EMAC 2 STA Control Register
EMAC2_TRTR	0x1 40000C60	R/W	EMAC 2 Transmit Request Threshold Register
EMAC2_RWMR	0x1 40000C64	R/W	EMAC 2 Receive Low/High Water Mark Register
EMAC2_OCTX	0x1 40000C68	R	EMAC 2 Number of Octets Transmitted

Preliminary User's Manual

Table 32-17. MMIO Registers (Continued)

Register	Address	Access	Description
EMAC2_OCRX	0x1 40000C6C	R	EMAC 2 Number of Octets Received
TCP/IP Accelerator on Hardware 1			
TAH1_MR0	0x1 40000D00	R/W	TAH 1 Mode Register 0
TAH1_SSR0	0x1 40000D04	R/W	TAH 1 Segment Size Register 0
TAH1_SSR1	0x1 40000D08	R/W	TAH 1 Segment Size Register 1
TAH1_SSR2	0x1 40000D0C	R/W	TAH 1 Segment Size Register 2
TAH1_SSR3	0x1 40000D10	R/W	TAH 1 Segment Size Register 3
TAH1_SSR4	0x1 40000D14	R/W	TAH 1 Segment Size Register 4
TAH1_SSR5	0x1 40000D18	R/W	TAH 1 Segment Size Register 5
TAH1_SR	0x1 40000D1C	R/W	TAH 1 Status Register
Ethernet MAC 3			
EMAC3_MR0	0x1 40000E00	R/W	EMAC 3 Mode Register 0
EMAC3_MR1	0x1 40000E04	R/W	EMAC 3 Mode Register 1
EMAC3_TMR0	0x1 40000E08	R/W	EMAC 3 Transmit Mode Register 0
EMAC3_TMR1	0x1 40000E0C	R/W	EMAC 3 Transmit Mode Register 1
EMAC3_RMR	0x1 40000E10	R/W	EMAC 3 Receive Mode Register
EMAC3_ISR	0x1 40000E14	R/W	EMAC 3 Interrupt Status Register
EMAC3_ISER	0x1 40000E18	R/W	EMAC 3 Interrupt Status Enable Register
EMAC3_IAHR	0x1 40000E1C	R/W	EMAC 3 Individual Address High
EMAC3_IALR	0x1 40000E20	R/W	EMAC 3 Individual Address Low
EMAC3_VTPID	0x1 40000E24	R/W	EMAC 3 VLAN TPID Register
EMAC3_VTCI	0x1 40000E28	R/W	EMAC 3 VLAN TCI Register
EMAC3_PTR	0x1 40000E2C	R/W	EMAC 3 Pause Timer Register
EMAC3_IAHT1	0x1 40000E30	R/W	EMAC 3 Individual Address Hash Table 1
EMAC3_IAHT2	0x1 40000E34	R/W	EMAC 3 Individual Address Hash Table 2
EMAC3_IAHT3	0x1 40000E38	R/W	EMAC 3 Individual Address Hash Table 3
EMAC3_IAHT4	0x1 40000E3C	R/W	EMAC 3 Individual Address Hash Table 4
EMAC3_GAHT1	0x1 40000E40	R/W	EMAC 3 Group Address Hash Table 1
EMAC3_GAHT2	0x1 40000E44	R/W	EMAC 3 Group Address Hash Table 2
EMAC3_GAHT3	0x1 40000E48	R/W	EMAC 3 Group Address Hash Table 3
EMAC3_GAHT4	0x1 40000E4C	R/W	EMAC 3 Group Address Hash Table 4
EMAC3_LSAH	0x1 40000E50	R	EMAC 3 Last Source Address Low
EMAC3_LSAL	0x1 40000E54	R	EMAC 3 Last Source Address High
EMAC3_IPGVR	0x1 40000E58	R/W	EMAC 3 Inter-Packet Gap Value Register
EMAC3_STACR	0x1 40000E5C	R/W	EMAC 3 STA Control Register
EMAC3_TRTR	0x1 40000E60	R/W	EMAC 3 Transmit Request Threshold Register
EMAC3_RWMR	0x1 40000E64	R/W	EMAC 3 Receive Low/High Water Mark Register
EMAC3_OCTX	0x1 40000E68	R	EMAC 3 Number of Octets Transmitted
EMAC3_OCRX	0x1 40000E6C	R	EMAC 3 Number of Octets Received

The two registers listed in Table 32-18 are used to access the configuration registers of external PCI-X devices only.

Table 32-18. External PCI-X Device Configuration Register Access

Register	Address	Access	Description
PCIX0_CFGADDR	0x2 0EC00000	R/W	PCI-X Configuration Address Register
PCIX0_CFGDATA	0x2 0EC00004	R/W	PCI-X Configuration Data Register

Registers listed in Table 32-19 can be accessed in two ways:

1. By the processor using the PLB address
2. By an external PCI-X device using configuration cycles to the proper offset.

Preliminary User's Manual

Table 32-19. Internal PCI-X Configuration Registers

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_VENDID	0x2 0EC80000	R/W	0x01–0x00	R	PCI-X Vendor ID
PCIX0_DEVID	0x2 0EC80002	R/W	0x03–0x02	R	PCI-X Device ID
PCIX0_CMD	0x2 0EC80004	R/W	0x05–0x04	R/W	PCI-X Command Register
PCIX0_STATUS	0x2 0EC80006	R/W	0x07–0x06	R/W	PCI-X Status Register
PCIX0_REVID	0x2 0EC80008	R/W	0x08	R	PCI-X Revision ID
PCIX0_CLS	0x2 0EC80009	R/W	0x0B–0x09	R	PCI-X Class Register
PCIX0_CACHELS	0x2 0EC8000C	R/W	0x0C	R/W	PCI-X Cache Line Size
PCIX0_LATTIM	0x2 0EC8000D	R/W	0x0D	R/W	PCI-X Latency Timer
PCIX0_HDTYPE	0x2 0EC8000E	R	0x0E	R	PCI-X Header Type
PCIX0_BIST	0x2 0EC8000F	R	0x0F	R	PCI-X Built In Self Test Control
PCIX0_BAR0L	0x2 0EC80010	R/W	0x13–0x10	R/W	PCI-X BAR 0 Low
PCIX0_BAR0H	0x2 0EC80014	R/W	0x17–0x14	R/W	PCI-X BAR 0 High
PCIX0_BAR1	0x2 0EC80018	R/W	0x1B–0x18	R/W	PCI-X BAR 1
PCIX0_BAR2L	0x2 0EC8001C	R/W	0x1F–0x1C	R/W	PCI-X BAR 2 Low
PCIX0_BAR2H	0x2 0EC80020	R/W	0x23–0x20	R/W	PCI-X BAR 2 High
PCIX0_BAR3	0x2 0EC80024	R	0x27–0x24	R	Unused BAR 3
PCIX0_CISPTR	0x2 0EC80028	R	0x2B–0x28	R	Unused Cardbus CIS Pointer
PCIX0_SBSYSVID	0x2 0EC8002C	R/W	0x2D–0x2C	R	PCI-X Subsystem Vendor ID
PCIX0_SBSYSID	0x2 0EC8002E	R/W	0x2F–0x2E	R	PCI-X Subsystem ID
PCIX0_EROMBA	0x2 0EC80030	R/W	0x33–0x30	R/W	PCI-X Expansion ROM Base Address
PCIX0_CAP	0x2 0EC80034	R	0x34	R	PCI-X Capabilities Pointer
PCIX0_RES0	0x2 0EC80035	R	0x35	R	Reserved
PCIX0_RES1	0x2 0EC80036	R	0x37–0x36	R	Reserved
PCIX0_RES2	0x2 0EC80038	R	0x3B–0x38	R	Reserved
PCIX0_INTLN	0x2 0EC8003C	R/W	0x3C	R/W	PCI-X Interrupt Line
PCIX0_INTPN	0x2 0EC8003D	R	0x3D	R	PCI-X Interrupt Pin
PCIX0_MINGNT	0x2 0EC8003E	R	0x3E	R	PCI-X Minimum Grant
PCIX0_MAXLTNCY	0x2 0EC8003F	R	0x3F	R	PCI-X Maximum Latency
PCIX0_BRDGOPT1	0x2 0EC80040	R/W	0x43–0x40	R/W	PCI-X Bridge Options 1
PCIX0_BRDGOPT2	0x2 0EC80044	R/W	0x47–0x44	R/W	PCI-X Bridge Options 2
PCIX0_ERREN	0x2 0EC80050	R/W	0x53–0x50	R/W	PCI-X Error Enable
PCIX0_ERRSTS	0x2 0EC80054	R/W	0x57–0x54	R/W	PCI-X Error Status
PCIX0_PLBBESR	0x2 0EC80058	R	0x5B–0x58	R	PCI-X PLB Slave Error Syndrome Register
PCIX0_PLBBEARL	0x2 0EC8005C	R	0x5F–0x5C	R	PCI-X PLB Slave Error Address Register Low
PCIX0_PLBBEARH	0x2 0EC80060	R	0x63–0x60	R	PCI-X PLB Slave Error Address Register High
PCIX0_POM0LAL	0x2 0EC80068	R/W	0x6B–0x68	R/W	PCI-X POM0 Local Address Low
PCIX0_POM0LAH	0x2 0EC8006C	R/W	0x6F–0x6C	R/W	PCI-X POM0 Local Address High

Table 32-19. Internal PCI-X Configuration Registers (Continued)

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_POM0SA	0x2 0EC80070	R/W	0x73–0x70	R/W	PCI-X POM0 Size Attribute
PCIX0_POM0PCIAL	0x2 0EC80074	R/W	0x77–0x74	R/W	PCI-X POM0 PCI Address Low
PCIX0_POM0PCIAH	0x2 0EC80078	R/W	0x7B–0x78	R/W	PCI-X POM0 PCI Address High
PCIX0_POM1LAL	0x2 0EC8007C	R/W	0x7F–0x7C	R/W	PCI-X POM1 Local Address Low
PCIX0_POM1LAH	0x2 0EC80080	R/W	0x83–0x80	R/W	PCI-X POM1 Local Address High
PCIX0_POM1SA	0x2 0EC80084	R/W	0x87–0x84	R/W	PCI-X POM1 Size Attribute
PCIX0_POM1PCIAL	0x2 0EC80088	R/W	0x8B–0x88	R/W	PCI-X POM1 PCI Address Low
PCIX0_POM1PCIAH	0x2 0EC8008C	R/W	0x8F–0x8C	R/W	PCI-X POM1 PCI Address High
PCIX0_POM2SA	0x2 0EC80090	R/W	0x93–0x90	R/W	PCI-X POM2 Size Attribute
PCIX0_PIM0SAL	0x2 0EC80098	R/W	0x9B–0x98	R/W	PCI-X PIM0 Size/Attribute Low
PCIX0_PIM0SAH	0x2 0EC800F8	R/W	0xFB–0xF8	R/W	PCI-X PIM0 Size/Attribute High
PCIX0_PIM0LAL	0x2 0EC8009C	R/W	0x9F–0x9C	R/W	PCI-X PIM0 Local Address Low
PCIX0_PIM0LAH	0x2 0EC800A0	R/W	0xA3–0xA0	R/W	PCI-X PIM0 Local Address High
PCIX0_PIM1SA	0x2 0EC800A4	R/W	0xA7–0xA4	R/W	PCI-X PIM1 Size/Attribute
PCIX0_PIM1LAL	0x2 0EC800A8	R/W	0xAB–0xA8	R/W	PCI-X PIM1 Local Address Low
PCIX0_PIM1LAH	0x2 0EC800AC	R/W	0xAF–0xAC	R/W	PCI-X PIM1 Local Address High
PCIX0_PIM2SAL	0x2 0EC800B0	R/W	0xB3–0xB0	R/W	PCI-X PIM2 Size/Attribute Low
PCIX0_PIM2SAH	0x2 0EC800FC	R/W	0xFF–0xFC	R/W	PCI-X PIM2 Size/Attribute High
PCIX0_PIM2LAL	0x2 0EC800B4	R/W	0xB7–0xB4	R/W	PCI-X PIM2 Local Address Low
PCIX0_PIM2LAH	0x2 0EC800B8	R/W	0xBB–0xB8	R/W	PCI-X PIM2 Local Address High
PCIX0_OMCAPID	0x2 0EC800C0	R	0xC0	R	PCI-X Outbound MSI Capability Identifier
PCIX0_OMNIPTR	0x2 0EC800C1	R/W	0xC1	R	PCI-X Outbound MSI Next Item Pointer
PCIX0_OMMC	0x2 0EC800C2	R/W	0xC3–0xC2	R/W	PCI-X Outbound MSI Message Control
PCIX0_OMMA	0x2 0EC800C4	R/W	0xC7–0xC4	R/W	PCI-X Outbound MSI Message Address
PCIX0_OMMUA	0x2 0EC800C8	R/W	0xCB–0xC8	R/W	PCI-X Outbound MSI Message Upper Address
PCIX0_OMMDATA	0x2 0EC800CC	R/W	0xCD–0xCC	R/W	PCI-X Outbound MSI Message Data
PCIX0_OMMEOI	0x2 0EC800CE	R/W	0xCE	R/W	PCI-X Outbound MSI Message End Of Interrupt
PCIX0_PMCAPID	0x2 0EC800D0	R/W	0xD0	R	PCI-X PMC Capability Identifier
PCIX0_PMNIPTR	0x2 0EC800D1	R/W	0xD1	R	PCI-X PMC Next Item Pointer
PCIX0_PMC	0x2 0EC800D2	R	0xD3–0xD2	R	PCI-X Power Management Capabilities
PCIX0_PMCSCR	0x2 0EC800D4	R/W	0xD5–0xD4	R/W	PCI-X Power Management Control Status
PCIX0_PMCSEBSE	0x2 0EC800D6	R	0xD6	R	PCI-X PMCSR PCI to PCI Bridge Support Extensions
PCIX0_PMDATA	0x2 0EC800D7	R	0xD7	R	PCI-X PMC Unused Data Register
PCIX0_PMSCRR	0x2 0EC800D8	R/W	0xD8	R/W	PCI-X Power Management State Change Request Register
PCIX0_PCIXCAPID	0x2 0EC800DC	R	0xDC	R	PCI-X Capability Identifier

Preliminary User's Manual

Table 32-19. Internal PCI-X Configuration Registers (Continued)

Register	PLB		PCI-X Configuration		Description
	Address	Access	Offset	Access	
PCIX0_PCIXNIPTR	0x2 0EC800DD	R	0xDD	R	PCI-X Next Item Pointer
PCIX0_PCIXCMD	0x2 0EC800DE	R/W	0xDF–0xDE	R/W	PCI-X Command
PCIX0_PCIXSTS	0x2 0EC800E0	R/W	0xE3–0xE0	R/W	PCI-X Status
PCIX0_PCIXIDR	0x2 0EC800E4	R/W	0xE7–0xE4	R/W	PCI-X Internal Debug Register
PCIX0_PCIXCID	0x2 0EC800E8	R	0xEB–0xE8	R	PCI-X Internal Core Device ID
PCIX0_PCIXRID	0x2 0EC800EC	R	0xEF–0xEC	R	PCI-X Internal Core Revision ID
PCIX0_VPDCAPID	0x2 0EC800F0	R	0xF0	R	PCI-X VPD Capability Identifier
PCIX0_VPDNIPTR	0x2 0EC800F1	R/W	0xF1	R	PCI-X VPD Next Item Pointer
PCIX0_VPDADR	0x2 0EC800F2	R/W	0xF2	R/W	PCI-X VPD Address
PCIX0_VPDDATA	0x2 0EC800F4	R/W	0xF4	R/W	PCI-X VPD Data

Registers listed in Table 32-20 can be accessed in two ways:

1. By the processor using the PLB address
2. By an external PCI-X device using memory cycles to the proper address.

Table 32-20. PCI-X Simple Message Passing and Inbound MSI Registers

Register	PLB		PCI-X Memory		Description
	Address	Access	Address	Access	
PCIX0_MSGIL	0x2 0EC80100	R/W	(PCIX0_BAR0H PCIX0_BAR0L) + (0x03–0x00)	R/W	PCI-X Message In Low
PCIX0_MSGIH	0x2 0EC80104	R/W	(PCIX0_BAR0H PCIX0_BAR0L) + (0x07–0x04)	R/W	PCI-X Message In High
PCIX0_MSGOL	0x2 0EC80108	R/W	(PCIX0_BAR0H PCIX0_BAR0L) + (0x0B–0x08)	R/W	PCI-X Message Out Low
PCIX0_MSGOH	0x2 0EC8010C	R/W	(PCIX0_BAR0H PCIX0_BAR0L) + (0x0F–0x0C)	R/W	PCI-X Message Out High
PCIX0_IM	0x2 0EC801F8	W	(PCIX0_BAR0H PCIX0_BAR0L) + (0xFB–0xF8)	W	PCI-X Inbound MSI

Preliminary User's Manual

32.5 Alphabetical Register Listing of Processor Core Registers

The following pages list the registers available in the PPC440GX. For each register, the following information is supplied:

- Register mnemonic and name
- Cross reference to detailed register information
- Register type (if SPR); the types of the other registers are the same as the register names (CR, GPR, MSR)
- Register number (address)
- Register programming model (user or supervisor) and access (read-clear, read-only, read/write (R/W), write-only)
- A diagram illustrating the register fields (all register fields have mnemonics, unless there is only one field)
- A table describing the register fields, giving field mnemonics, field bit locations, field names, and the functions associated with the various field values

SPR 0x3B3 Supervisor R/W

See *Core Configuration Register 0 (CCR0)* on page 212.

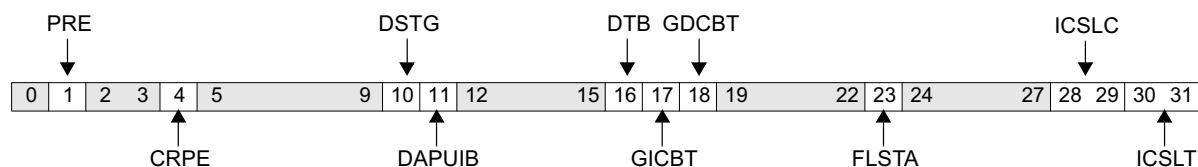


Figure 32-1. Core Configuration Register 0 (CCR0)

0		Reserved	
1	PRE	Parity Recoverability Enable 0 Semi-recoverable parity mode enabled for data cache 1 Fully recoverable parity mode enabled for data cache	Must be set to 1 to guarantee full recoverability from MMU and data cache parity errors.
2:3		Reserved	
4	CRPE	Cache Read Parity Enable 0 Disable parity information reads 1 Enable parity information reads	When enabled, execution of icread , dcread , or tlbre loads parity information into the ICDBTRH, DCDBTRL, or target GPR, respectively.
5:9		Reserved	
10	DSTG	Disable Store Gathering 0 Enabled; stores to contiguous addresses may be gathered into a single transfer 1 Disabled; all stores to memory will be performed independently	See <i>Store Gathering</i> on page 221.
11	DAPUIB	Disable APU Instruction Broadcast 0 Enabled. 1 Disabled; instructions not broadcast to APU for decoding	This mechanism is provided as a means of reducing power consumption when an auxiliary processor is not attached and/or is not being used. See <i>Reset and Initialization</i> on page 261.
12:15		Reserved	
16	DTB	Disable Trace Broadcast 0 Enabled. 1 Disabled; no trace information is broadcast.	This mechanism is provided as a means of reducing power consumption when instruction tracing is not needed. See <i>Reset and Initialization</i> on page 261.
17	GICBT	Guaranteed Instruction Cache Block Touch 0 icbt may be abandoned without having filled cache line if instruction pipeline stalls. 1 icbt is guaranteed to fill cache line even if instruction pipeline stalls.	See <i>icbt Operation</i> on page 214.
18	GDCBT	Guaranteed Data Cache Block Touch 0 dcbt/dcbtst may be abandoned without having filled cache line if load/store pipeline stalls. 1 dcbt/dcbtst are guaranteed to fill cache line even if load/store pipeline stalls.	See <i>Data Cache Control and Debug</i> on page 225.
19:22		Reserved	
23	FLSTA	Force Load/Store Alignment 0 No Alignment exception on integer storage access instructions, regardless of alignment 1 An alignment exception occurs on integer storage access instructions if data address is not on an operand boundary.	See <i>Load and Store Alignment</i> on page 220.
24:27		Reserved	

28:29	ICSLC	Instruction Cache Speculative Line Count	Number of additional lines (0–3) to fill on instruction fetch miss. See <i>Speculative Prefetch Mechanism</i> on page 209.
30:31		Reserved	

CCR1

Core Configuration Register 1

Preliminary User's Manual

SPR 0x378 Supervisor R/W

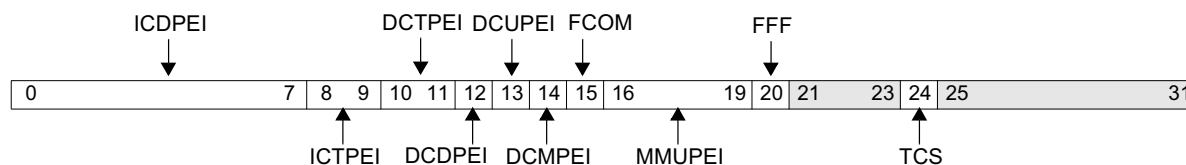
See *Core Configuration Register 1 (CCR1)* on page 213.

Figure 32-2. Core Configuration Register 1 (CCR1)

0:7	ICDPEI	Instruction Cache Data Parity Error Insert 0 record even parity (normal) 1 record odd parity (simulate parity error)	Controls inversion of parity bits recorded when the instruction cache is filled. Each of the 8 bits corresponds to one of the instruction words in the line.
8:9	ICTPEI	Instruction Cache Tag Parity Error Insert 0 record even parity (normal) 1 record odd parity (simulate parity error)	Controls inversion of parity bits recorded for the tag field in the instruction cache.
10:11	DCTPEI	Data Cache Tag Parity Error Insert 0 record even parity (normal) 1 record odd parity (simulate parity error)	Controls inversion of parity bits recorded for the tag field in the data cache.
12	DCDPEI	Data Cache Data Parity Error Insert 0 record even parity (normal) 1 record odd parity (simulate parity error)	Controls inversion of parity bits recorded for the data field in the data cache.
13	DCUPEI	Data Cache U-bit Parity Error Insert 0 record even parity (normal) 1 record odd parity (simulate parity error)	Controls inversion of parity bit recorded for the U fields in the data cache.
14	DCMPEI	Data Cache Modified-bit Parity Error Insert 0 record even parity (normal) 1 record odd parity (simulate parity error)	Controls inversion of parity bits recorded for the modified (dirty) field in the data cache.
15	FCOM	Force Cache Operation Miss 0 normal operation 1 cache ops appear to miss the cache	Force icbt , dcbt , dcbtst , dcbst , dcbf , dcbi , and dcbz to appear to miss the caches. The intended use is with icbt and dcbt only, which will fill a duplicate line and allow testing of multi-hit parity errors. See Section 5.2.4.7 <i>Simulating Instruction Cache Parity Errors for Software Testing</i> on page 217 and Figure 5.3.3.8 on page 232.
16:19	MMUPEI	Memory Management Unit Parity Error Insert 0 record even parity (normal) 1 record odd parity (simulate parity error)	Controls inversion of parity bits recorded for the tag field in the MMU.
20	FFF	Force Full-line Flush 0 flush only as much data as necessary. 1 always flush entire cache lines	When flushing 32-byte (8-word) lines from the data cache, normal operation is to write nothing, a double word, quad word, or the entire 8-word block to the memory as required by the dirty bits. This bit ensures that none or all dirty bits are set so that either nothing or the entire 8-word block is written to memory when flushing a line from the data cache. Refer to Section 5.3.1.4 <i>Line Flush Operations</i> on page 222.
21:23		Reserved	
24	TCS	Timer Clock Select 0 CPU timer advances by one at each rising edge of the CPU input clock (CPMC440CLOCK). 1 CPU timer advances by one for each rising edge of the CPU timer clock (CPMC440TIMERCLOCK).	When TCS = 1, CPU timer clock input can toggle at up to half of the CPU clock frequency.
25:31		Reserved	

User Read/Write

See *Condition Register (CR)* on page 183.

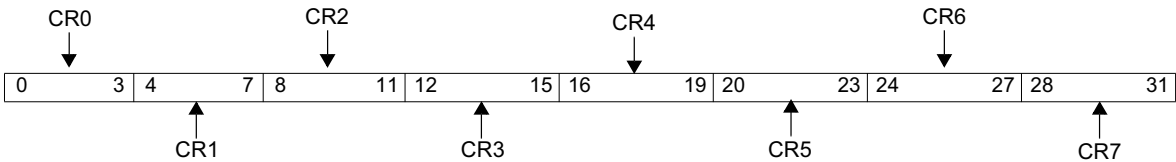


Figure 32-3. Condition Register (CR)

0:3	CR0	Condition Register Field 0
4:7	CR1	Condition Register Field 1
8:11	CR2	Condition Register Field 2
12:15	CR3	Condition Register Field 3
16:19	CR4	Condition Register Field 4
20:23	CR5	Condition Register Field 5
24:27	CR6	Condition Register Field 6
28:31	CR7	Condition Register Field 7

SPR 0x03A Supervisor R/W

See *Critical Save/Restore Register 0 (CSRR0)* on page 427.

0	29	30	31
---	----	----	----

Figure 32-4. Critical Save/Restore Register 0 (CSRR0)

0:29		Return address for critical interrupts
30:31		Reserved

SPR 0x03B Supervisor R/W

See *Critical Save/Restore Register 1 (CSRR1)* on page 428.

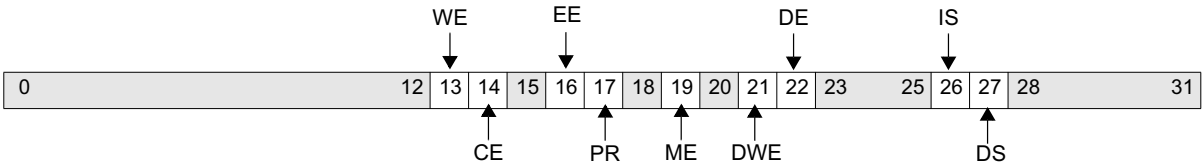


Figure 32-5. Critical Save/Restore Register 1 (CSRR1)

0:31	Copy of the MSR when a critical interrupt is taken
------	--

SPR 0x009 User R/W

See *Count Register (CTR)* on page 183.

0	31
---	----

Figure 32-6. Count Register (CTR)

0:31	Count	Used as count for branch conditional with decrement instructions, or as target address for bcctr instructions
------	-------	--

SPR 0x13C–0x13D Supervisor R/W

See *Data Address Compare Registers (DAC1–DAC2)* on page 530.

0	31
---	----

Figure 32-7. Data Address Compare Registers (DAC1–DAC2)

0:31		Data Address Compare (DAC) byte address
------	--	---

SPR 0x134 Supervisor R/W

See *Debug Control Register 0 (DBCR0)* on page 524.

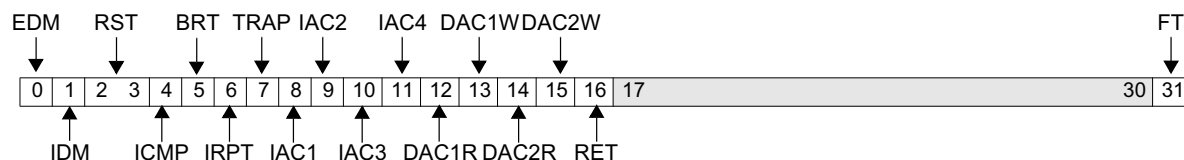


Figure 32-8. Debug Control Register 0 (DBCR0)

0	EDM	External Debug Mode 0 Disable external debug mode. 1 Enable external debug mode.	
1	IDM	Internal Debug Mode 0 Disable internal debug mode. 1 Enable internal debug mode.	
2:3	RST	Reset 00 No action 01 Core reset 10 Chip reset 11 System reset Attention: Writing 01, 10, or 11 to this field causes a processor reset to occur.	
4	ICMP	Instruction Completion Debug Event 0 Disable instruction completion debug event. 1 Enable instruction completion debug event.	Instruction completions do not cause instruction completion debug events if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.
5	BRT	Branch Taken Debug Event 0 Disable branch taken debug event. 1 Enable branch taken debug event.	Taken branches do not cause branch taken debug events if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.
6	IRPT	Interrupt Debug Event 0 Disable interrupt debug event. 1 Enable interrupt debug event.	Critical interrupts do not cause interrupt debug events in internal debug mode, unless also in external debug mode or debug wait mode.
7	TRAP	Trap Debug Event 0 Disable trap debug event. 1 Enable trap debug event.	
8	IAC1	Instruction Address Compare (IAC) 1 Debug Event 0 Disable IAC 1 debug event. 1 Enable IAC 1 debug event.	
9	IAC2	IAC 2 Debug Event 0 Disable IAC 2 debug event. 1 Enable IAC 2 debug event.	
10	IAC3	IAC 3 Debug Event 0 Disable IAC 3 debug event. 1 Enable IAC 3 debug event.	
11	IAC4	IAC 4 Debug Event 0 Disable IAC 4 debug event. 1 Enable IAC 4 debug event.	
12	DAC1R	Data Address Compare (DAC) 1 Read Debug Event 0 Disable DAC 1 read debug event. 1 Enable DAC 1 read debug event.	

13	DAC1W	DAC 1 Write Debug Event 0 Disable DAC 1 write debug event. 1 Enable DAC 1 write debug event.	
14	DAC2R	DAC 2 Read Debug Event 0 Disable DAC 2 read debug event. 1 Enable DAC 2 read debug event.	
15	DAC2W	DAC 2 Write Debug Event 0 Disable DAC 2 write debug event. 1 Enable DAC 2 write debug event.	
16	RET	Return Debug Event 0 Disable return (rfi/rfci/rfmc i) debug event. 1 Enable return (rfi/rfci/rfmc i) debug event.	rfci/rfmc i does not cause a return debug event if MSR[DE] = 0 in internal debug mode, unless also in external debug mode or debug wait mode.
17:30		Reserved	
31	FT	Freeze timers on debug event 0 Timers are not frozen. 1 Freeze timers if a DBSR field associated with a debug event is set.	

SPR 0x135 Supervisor R/W

See *Debug Control Register 1 (DBCR1)* on page 526.

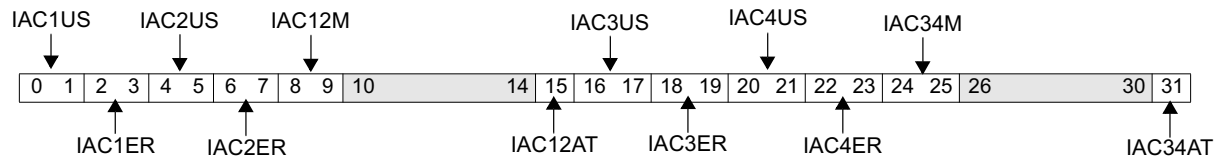


Figure 32-9. Debug Control Register 1 (DBCR1)

0:1	IAC1US	Instruction Address Compare (IAC) 1 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
2:3	IAC1ER	IAC 1 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0) 11 Virtual (MSR[IS] = 1)
4:5	IAC2US	IAC 2 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
6:7	IAC2ER	IAC 2 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0) 11 Virtual (MSR[IS] = 1)
8:9	IAC12M	IAC 1/2 Mode 00 Exact match 01 Reserved 10 Range inclusive 11 Range exclusive Match if address[0:29] = IAC 1/2[0:29]; two independent compares Match if IAC1 ≤ address < IAC2 Match if address < IAC1 OR address ≥ IAC2
10:14		Reserved
15	IAC12AT	IAC 1/2 Auto-Toggle Enable 0 Disable IAC 1/2 auto-toggle 1 Enable IAC 1/2 auto-toggle
16:17	IAC3US	IAC 3 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
18:19	IAC3ER	IAC 3 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0) 11 Virtual (MSR[IS] = 1)

20:21	IAC4US	IAC 4 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
22:23	IAC4ER	IAC 4 Effective/Real 00 Effective (MSR[IS] = don't care) 01 Reserved 10 Virtual (MSR[IS] = 0) 11 Virtual (MSR[IS] = 1)
24:25	IAC34M	IAC 3/4 Mode 00 Exact match 01 Reserved 10 Range inclusive 11 Range exclusive Match if address[0:29] = IAC 3/4[0:29]; two independent compares Match if $IAC3 \leq \text{address} < IAC4$ Match if $\text{address} < IAC3$ OR $\text{address} \geq IAC4$
26:30		Reserved
31	IAC34AT	IAC3/4 Auto-Toggle Enable 0 Disable IAC 3/4 auto-toggle 1 Enable IAC 3/4 auto-toggle

SPR 0x136 Supervisor R/W

See *Debug Control Register 2 (DBCR2)* on page 527.

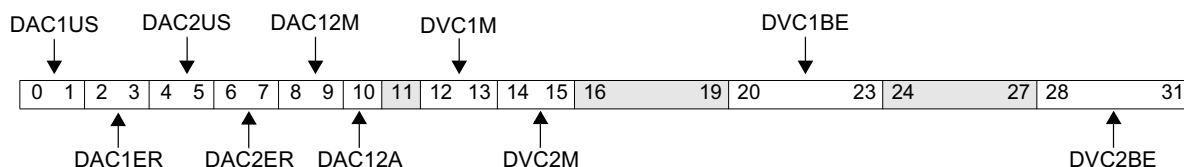


Figure 32-10. Debug Control Register 2 (DBCR2)

0:1	DAC1US	Data Address Compare (DAC) 1 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
2:3	DAC1ER	DAC 1 Effective/Real 00 Effective (MSR[DS] = don't care) 01 Reserved 10 Virtual (MSR[DS] = 0) 11 Virtual (MSR[DS] = 1)
4:5	DAC2US	DAC 2 User/Supervisor 00 Both 01 Reserved 10 Supervisor only (MSR[PR] = 0) 11 User only (MSR[PR] = 1)
6:7	DAC2ER	DAC 2 Effective/Real 00 Effective (MSR[DS] = don't care) 01 Reserved 10 Virtual (MSR[DS] = 0) 11 Virtual (MSR[DS] = 1)
8:9	DAC12M	DAC 1/2 Mode 00 Exact match 01 Address bit mask 10 Range inclusive 11 Range exclusive Match if address[0:31] = DAC 1/2[0:31]; two independent compares Match if address = DAC1; only compare bits corresponding to 1 bits in DAC2 Match if DAC1 ≤ address < DAC2 Match if address < DAC1 OR address ≥ DAC2
10	DAC12A	DAC 1/2 Asynchronous 0 Debug interrupt caused by DAC1/2 exception will be synchronous 1 Debug interrupt caused by DAC1/2 exception will be asynchronous
11		Reserved
12:13	DVC1M	Data Value Compare (DVC) 1 Mode 00 Reserved 01 AND all bytes enabled by DVC1BE 10 OR all bytes enabled by DVC1BE 11 AND-OR pairs of bytes enabled by DVC1BE (0 AND 1) OR (2 AND 3)
14:15	DVC2M	DVC 2 Mode 00 Reserved 01 AND all bytes enabled by DVC2BE 10 OR all bytes enabled by DVC2BE 11 AND-OR pairs of bytes enabled by DVC2BE (0 AND 1) OR (2 AND 3)
16:19		Reserved

20:23	DVC1BE	DVC 1 Byte Enables 0:3
24:27		Reserved
28:31	DVC2BE	DVC 2 Byte Enables 0:3

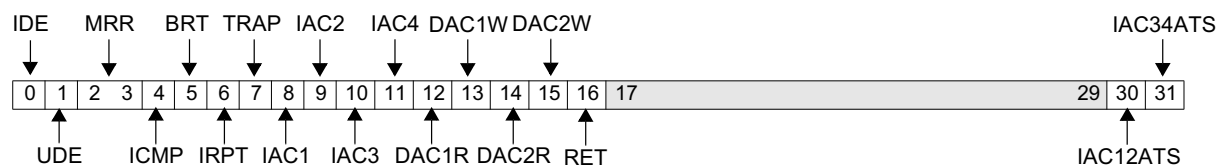
SPR 0x3F3 Supervisor R/W

See *Debug Data Register (DBDR)* on page 531.



Figure 32-11. Debug Data Register (DBDR)

0:31	Debug Data
------	------------

SPR 0x130 Supervisor Read/ClearSee *Debug Status Register (DBSR)* on page 528.**Figure 32-12. Debug Status Register (DBSR)**

0	IDE	Imprecise Debug Event 0 Debug event occurred while MSR[DE] = 1 1 Debug event occurred while MSR[DE] = 0	For synchronous debug events in internal debug mode, this field indicates whether the corresponding Debug interrupt occurs precisely or imprecisely
1	UDE	Unconditional Debug Event 0 Event didn't occur 1 Event occurred	
2:3	MRR	Most Recent Reset 00 No reset has occurred since this field was last cleared by software. 01 Core reset 10 Chip reset 11 System reset	This field is set upon any processor reset to a value indicating the type of reset.
4	ICMP	Instruction Completion Debug Event 0 Event didn't occur 1 Event occurred	
5	BRT	Branch Taken Debug Event 0 Event didn't occur 1 Event occurred	
6	IRPT	Interrupt Debug Event 0 Event didn't occur 1 Event occurred	
7	TRAP	Trap Debug Event 0 Event didn't occur 1 Event occurred	
8	IAC1	IAC 1 Debug Event 0 Event didn't occur 1 Event occurred	
9	IAC2	IAC 2 Debug Event 0 Event didn't occur 1 Event occurred	
10	IAC3	IAC 3 Debug Event 0 Event didn't occur 1 Event occurred	
11	IAC4	IAC 4 Debug Event 0 Event didn't occur 1 Event occurred	
12	DAC1R	DAC 1 Read Debug Event 0 Event didn't occur 1 Event occurred	
13	DAC1W	DAC 1 Write Debug Event 0 Event didn't occur 1 Event occurred	

DBSR (cont.)

Debug Status Register

Preliminary User's Manual

14	DAC2R	DAC 2 Read Debug Event 0 Event didn't occur 1 Event occurred
15	DAC2W	DAC 2 Write Debug Event 0 Event didn't occur 1 Event occurred
16	RET	Return Debug Event 0 Event didn't occur 1 Event occurred
17:29		Reserved
30	IAC12ATS	IAC 1/2 Auto-Toggle Status 0 Range is not reversed from value specified in DBCR1[IAC12M] 1 Range is reversed from value specified in DBCR1[IAC12M]
31	IAC34ATS	IAC 3/4 Auto-Toggle Status 0 Range is not reversed from value specified in DBCR1[IAC34M] 1 Range is reversed from value specified in DBCR1[IAC34M]

SPR 0x39D Supervisor Read-Only

See *dcread* Operation on page 228.

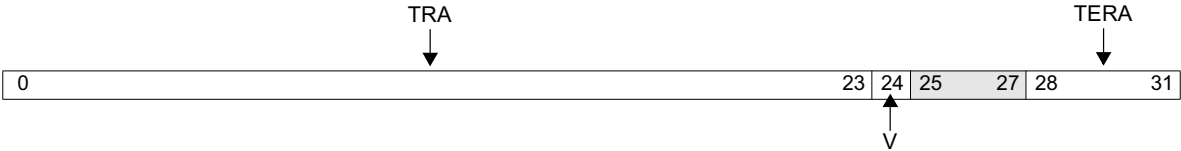


Figure 32-13. Data Cache Debug Tag Register High (DCDBTRH)

0:23	TRA	Tag Real Address	Bits 0:23 of the lower 32 bits of the 36-bit real address associated with the cache line read by dcread .
24	V	Cache Line Valid 0 Cache line is not valid. 1 Cache line is valid.	The valid indicator for the cache line read by dcread .
25:27		Reserved	
28:31	TERA	Tag Extended Real Address	Upper 4 bits of the 36-bit real address associated with the cache line read by dcread .

SPR 0x39C Supervisor Read-Only

See *dcread* Operation on page 228.

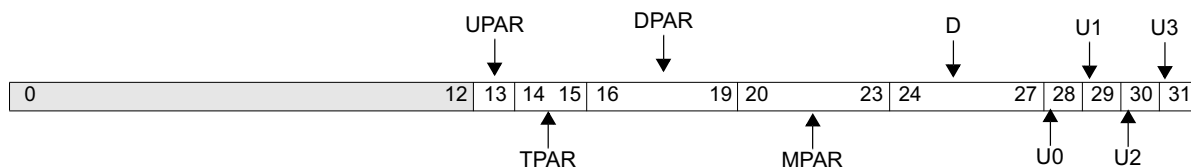


Figure 32-14. Data Cache Debug Tag Register Low (DCDBTRL)

0:12		Reserved	
13	UPAR	U bit parity UPAR = U0 XOR U1 XOR U2 XOR U3	The parity for the U0-U3 bits in the cache line read by dcread if CCR0[CRPE] = 1, otherwise 0.
14:15	TPAR	Tag parity bit 14 - XOR of odd address bits bit 15 - XOR of even address bits	The parity for the tag bits in the cache line read by dcread if CCR0[CRPE] = 1, otherwise 0.
16:19	DPAR	Data parity bit 16 - Data Parity of Double-Word 0 bit 17 - Data Parity of Double-Word 1 bit 18 - Data Parity of Double-Word 2 bit 19 - Data Parity of Double-Word 3	The parity <i>check values</i> for the data bytes in the word read by dcread if CCR0[CRPE] = 1, otherwise 0.
20:23	MPAR	Modified (dirty) parity bit 20 - Dirty bit Parity for Double-Word 0 bit 21 - Dirty bit Parity for Double-Word 1 bit 22 - Dirty bit Parity for Double-Word 2 bit 23 - Dirty bit Parity for Double-Word 3	The parity for the modified (dirty) indicators for each of the four doublewords in the cache line read by dcread if CCR0[CRPE] = 1, otherwise 0.
24:27	D	Dirty Indicators bit 24 - Dirty bit for Double-Word 0 bit 25 - Dirty bit for Double-Word 1 bit 26 - Dirty bit for Double-Word 2 bit 27 - Dirty bit for Double-Word 3	The “dirty” (modified) indicators for each of the four doublewords in the cache line read by dcread .
28	U0	U0 Storage Attribute	The U0 storage attribute for the memory page associated with this cache line read by dcread .
29	U1	U1 Storage Attribute	The U1 storage attribute for the memory page associated with this cache line read by dcread .
30	U2	U2 Storage Attribute	The U2 storage attribute for the memory page associated with this cache line read by dcread .
31	U3	U3 Storage Attribute	The U3 storage attribute for the memory page associated with this cache line read by dcread .

SPR 0x03D Supervisor R/W

See *Data Exception Address Register (DEAR)* on page 429.

0	31
---	----

Figure 32-15. Data Exception Address Register (DEAR)

0:31	Address of data exception for Data Storage, Alignment, and Data TLB Error interrupts
------	--

SPR 0x016 Supervisor R/W

See *Decrementer (DEC)* on page 461.

0	31
---	----

Figure 32-16. Decrementer (DEC)

0:31		Decrement value
------	--	-----------------

Preliminary User’s Manual

SPR 0x036 Supervisor Write-Only

See *Decrementer (DEC)* on page 461.



Figure 32-17. Decrementer Auto-Reload (DECAR)

0:31		Decrementer auto-reload value	Copied to DEC at next time base clock when DEC = 1 and auto-reload is enabled (TCR[ARE] = 1).
------	--	-------------------------------	---

SPR 0x390–0x393 Supervisor R/W

See *Cache Line Replacement Policy* on page 202.

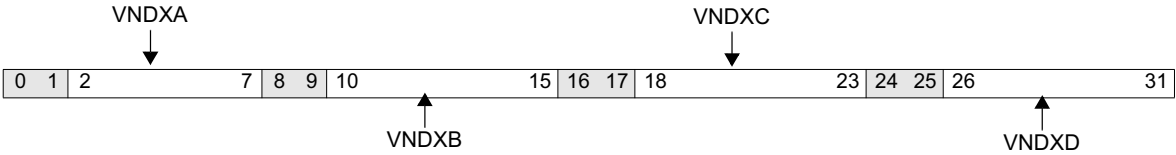


Figure 32-18. Data Cache Normal Victim Registers (DNV0–DNV3)

0:1		Reserved
2:7	VNDXA	Victim Index A (for cache lines with EA[25:26] = 0b00)
8:9		Reserved
10:15	VNDXB	Victim Index A (for cache lines with EA[25:26] = 0b01)
16:17		Reserved
18:23	VNDXC	Victim Index A (for cache lines with EA[25:26] = 0b10)
24:25		Reserved
26:31	VNDXD	Victim Index A (for cache lines with EA[25:26] = 0b11)

SPR 0x394–0x397 Supervisor R/W

See *Cache Line Replacement Policy* on page 202.

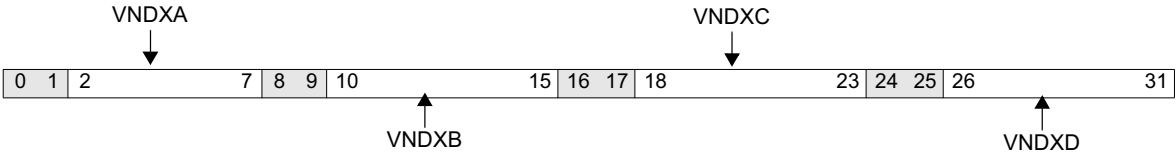


Figure 32-19. Data Cache Transient Victim Registers (DTV0–DTV3)

0:1		Reserved
2:7	VNDXA	Victim Index A (for cache lines with EA[25:26] = 0b00)
8:9		Reserved
10:15	VNDXB	Victim Index A (for cache lines with EA[25:26] = 0b01)
16:17		Reserved
18:23	VNDXC	Victim Index A (for cache lines with EA[25:26] = 0b10)
24:25		Reserved
26:31	VNDXD	Victim Index A (for cache lines with EA[25:26] = 0b11)

SPR 0x13E–0x13F Supervisor R/W

See *Data Value Compare Registers (DVC1–DVC2)* on page 531.

0	31
---	----

Figure 32-20. Data Value Compare Registers (DVC1–DVC2)

0:31	Data value to compare
------	-----------------------

Preliminary User's Manual

SPR 0x398 Supervisor R/W

See *Cache Locking and Transient Mechanism* on page 203.

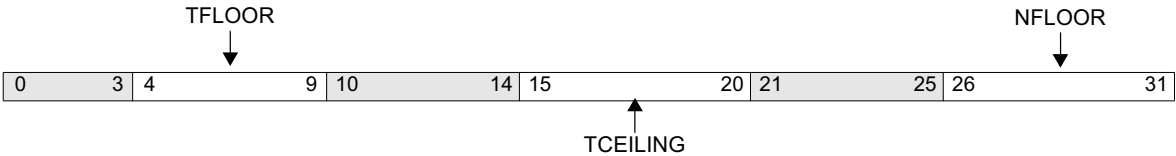


Figure 32-21. Data Cache Victim Limit (DVLIM)

0:3		Reserved
4:9	TFLOOR	Transient Floor
10:14		Reserved
15:20	TCEILING	Transient Ceiling
21:25		Reserved
26:31	NFLOOR	Normal Floor

SPR 0x03E Supervisor R/W

See *Exception Syndrome Register (ESR)* on page 431.

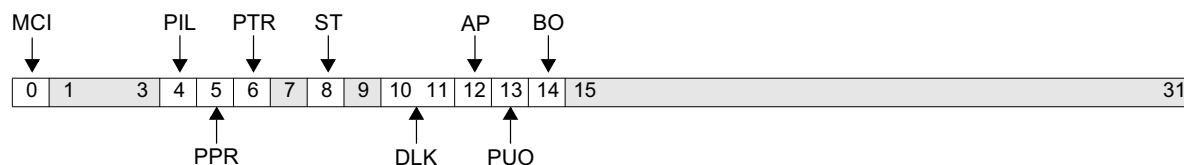


Figure 32-22. Exception Syndrome Register (ESR)

0	MCI	Machine Check—Instruction Fetch Exception 0 Instruction Machine Check exception did not occur. 1 Instruction Machine Check exception occurred.	This is an implementation-dependent field of the ESR and is not part of the PowerPC Book-E Architecture.
1:3		Reserved	
4	PIL	Program Interrupt—Illegal Instruction Exception 0 Illegal Instruction exception did not occur. 1 Illegal Instruction exception occurred.	
5	PPR	Program Interrupt—Privileged Instruction Exception 0 Privileged Instruction exception did not occur. 1 Privileged Instruction exception occurred.	
6	PTR	Program Interrupt—Trap Exception 0 Trap exception did not occur. 1 Trap exception occurred.	
7		Reserved	
8	ST	Store Operation 0 Exception was not caused by a store-type storage access or cache management instruction. 1 Exception was caused by a store-type storage access or cache management instruction.	
9		Reserved	
10:11	DLK	Data Storage Interrupt—Locking Exception 00 Locking exception did not occur. 01 Locking exception was caused by dcbf . 10 Locking exception was caused by icbi . 11 Reserved	
12	AP	AP Operation 0 Exception was not caused by an auxiliary processor instruction. 1 Exception was caused by an auxiliary processor instruction.	
13	PUO	Program Interrupt—Unimplemented Operation Exception 0 Unimplemented Operation exception did not occur. 1 Unimplemented Operation exception occurred.	
14	BO	Byte Ordering Exception 0 Byte Ordering exception did not occur. 1 Byte Ordering exception occurred.	
15:31		Reserved	

User R/W

See *General Purpose Registers (GPRs)* on page 186.



Figure 32-23. General Purpose Registers (R0-R31)

0:31	General Purpose Register data
------	-------------------------------

SPR 0x138–0x13B Supervisor R/W

See *Instruction Address Compare Registers (IAC1–IAC4)* on page 530.

0	29	30	31
---	----	----	----

Figure 32-24. Instruction Address Compare Registers (IAC1–IAC4)

0:29		Instruction Address Compare (IAC) word address
30:31		Reserved

SPR 0x3D3 Supervisor Read-Only

See *icread Operation* on page 215.

0	31
---	----

Figure 32-25. Instruction Cache Debug Data Register (ICDBDR)

0:31	Instruction machine code from instruction cache
------	---

SPR 0x39F Supervisor Read-Only

See *icread* Operation on page 215.

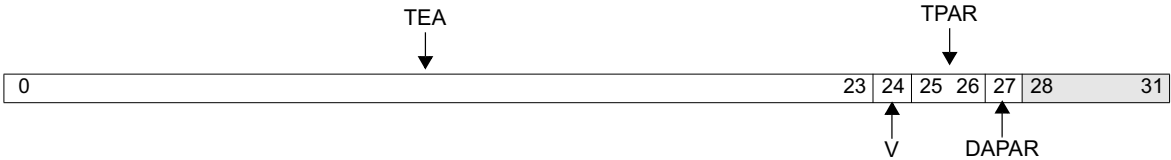


Figure 32-26. Instruction Cache Debug Tag Register High (ICDBTRH)

0:23		Tag Effective Address	Bits 0:23 of the 32-bit effective address associated with the cache line read by icread .
24	V	Cache Line Valid 0 Cache line is not valid. 1 Cache line is valid.	The valid indicator for the cache line read by icread .
25:26	TPAR	Tag Parity	The parity bits for the address tag for the cache line read by icread , if CCR0[CRPE] is set.
27	DAPAR	Instruction Data parity	The parity bit for the instruction word at the 32-bit effective address specified in the icread instruction, if CCR0[CRPE] is set.
28:31		Reserved	

SPR 0x39E Supervisor Read-Only

See *icread Operation* on page 215.

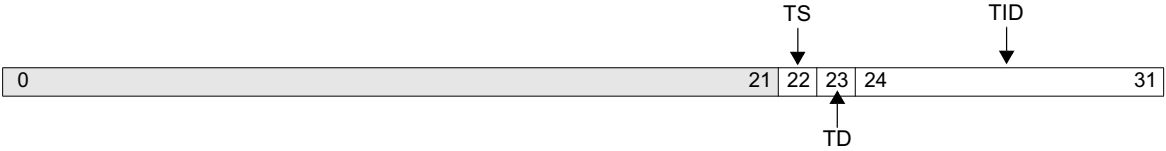


Figure 32-27. Instruction Cache Debug Tag Register Low (ICDBTRL)

0:21		Reserved	
22	TS	Translation Space	The address space portion of the virtual address associated with the cache line read by icread .
23	TD	Translation ID (TID) Disable 0 TID enable 1 TID disable	TID Disable field for the memory page associated with the cache line read by icread .
24:31	TID	Translation ID	TID field portion of the virtual address associated with the cache line read by icread .

SPR 0x370–0x373 Supervisor R/W

See *Cache Line Replacement Policy* on page 202.

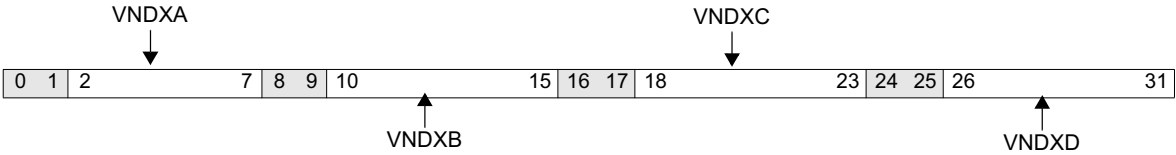


Figure 32-28. Instruction Cache Normal Victim Registers (INV0–INV3)

0:1		Reserved
2:7	VNDXA	Victim Index A (for cache lines with EA[25:26] = 0b00)
8:9		Reserved
10:15	VNDXB	Victim Index A (for cache lines with EA[25:26] = 0b01)
16:17		Reserved
18:23	VNDXC	Victim Index A (for cache lines with EA[25:26] = 0b10)
24:25		Reserved
26:31	VNDXD	Victim Index A (for cache lines with EA[25:26] = 0b11)

SPR 0x374–0x377 Supervisor R/W

See *Cache Line Replacement Policy* on page 202.

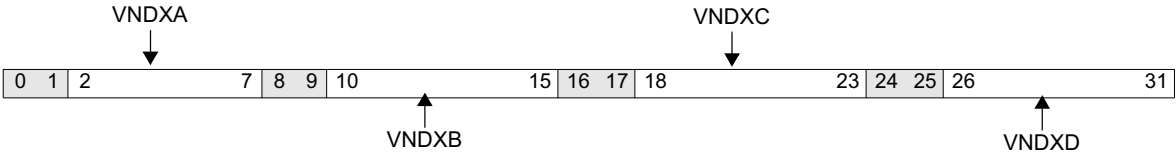


Figure 32-29. Instruction Cache Transient Victim Registers (ITV0–ITV3)

0:1		Reserved
2:7	VNDXA	Victim Index A (for cache lines with EA[25:26] = 0b00)
8:9		Reserved
10:15	VNDXB	Victim Index A (for cache lines with EA[25:26] = 0b01)
16:17		Reserved
18:23	VNDXC	Victim Index A (for cache lines with EA[25:26] = 0b10)
24:25		Reserved
26:31	VNDXD	Victim Index A (for cache lines with EA[25:26] = 0b11)

SPR 0x399 Supervisor R/W

See *Cache Locking and Transient Mechanism* on page 203.

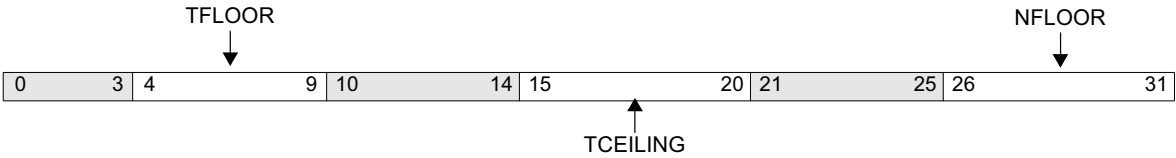


Figure 32-30. Instruction Cache Victim Limit (IVLIM)

0:3		Reserved
4:9	TFLOOR	Transient Floor
10:14		Reserved
15:20	TCEILING	Transient Ceiling
21:25		Reserved
26:31	NFLOOR	Normal Floor

SPR 0x190–0x19F Supervisor R/W

See *Interrupt Vector Offset Registers (IVOR0–IVOR15)* on page 430.

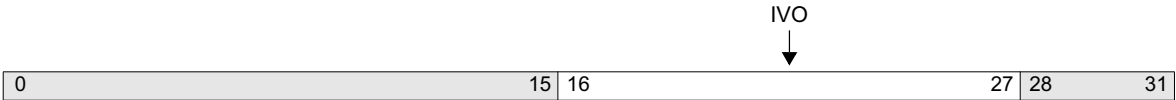


Figure 32-31. Interrupt Vector Offset Registers (IVOR0–IVOR15)

0:15		Reserved
16:27	IVO	Interrupt Vector Offset
28:31		Reserved

Table 32-21. Interrupt Types Associated with each IVOR

IVOR	Interrupt Type
IVOR0	Critical Input
IVOR1	Machine Check
IVOR2	Data Storage
IVOR3	Instruction Storage
IVOR4	External Input
IVOR5	Alignment
IVOR6	Program
IVOR7	Reserved
IVOR8	System Call
IVOR9	Reserved
IVOR10	Decrementer
IVOR11	Fixed Interval Timer
IVOR12	Watchdog Timer
IVOR13	Data TLB Error
IVOR14	Instruction TLB Error
IVOR15	Debug

SPR 0x03F Supervisor R/W

See *Interrupt Vector Prefix Register (IVPR)* on page 431.

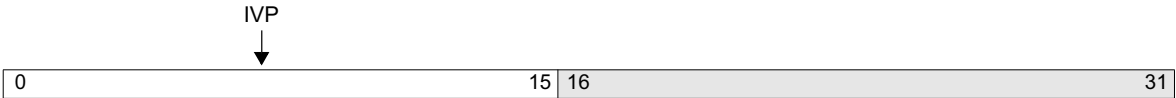


Figure 32-32. *Interrupt Vector Prefix Register (IVPR)*

0:15	IVP	Interrupt Vector Prefix
16:31		Reserved

Preliminary User’s Manual

SPR 0x008 User R/W

See *Link Register (LR)* on page 182.



Figure 32-33. Link Register (LR)

0:31	Link Register contents	Target address of bclr instruction
------	------------------------	---

MCSR

Machine Check Status Register

Preliminary User's Manual**SPR 0x23C Supervisor Read/Clear**

See *Machine Check Status Register (MCSR)* on page 433. Write a 1 to a bit to clear it.

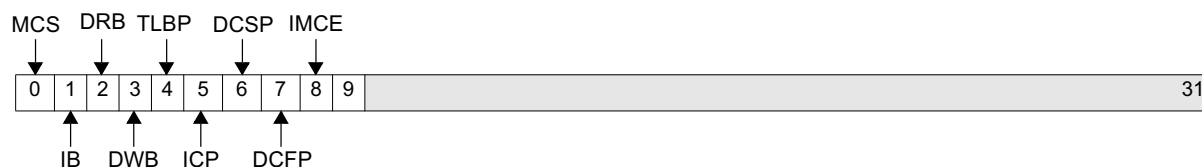


Figure 32-34. Machine Check Status Register (MCSR)

0	MCS	Machine Check Summary 0 No async machine check exception pending 1 Async machine check exception pending	Set when a machine check exception occurs that is handled in the imprecise fashion. One of MCSR bits 1:7 will be set simultaneously to indicate the exception type. When MSR[ME] and this bit are both set, Machine Check interrupt is taken.
1	IB	Instruction PLB Error 0 Exception not caused by Instruction Read PLB interrupt request (IRQ) 1 Exception caused by Instruction Read PLB interrupt request (IRQ)	
2	DRB	Data Read PLB Error 0 Exception not caused by Data Read PLB interrupt request (IRQ) 1 Exception caused by Data Read PLB interrupt request (IRQ)	
3	DWB	Data Write PLB Error 0 Exception not caused by Data Write PLB interrupt request (IRQ) 1 Exception caused by Data Write PLB interrupt request (IRQ)	
4	TLBP	Translation Lookaside Buffer Parity Error 0 Exception not caused by TLB parity error 1 Exception caused by TLB parity error	
5	ICP	Instruction Cache Parity Error 0 Exception not caused by I-cache parity error 1 Exception caused by I-cache parity error	
6	DCSP	Data Cache Search Parity Error 0 Exception not caused by DCU Search parity error 1 Exception caused by DCU Search parity error	Set if and only If the DCU parity error was discovered during a DCU Search operation. See <i>Data Cache Parity Operations</i> on page 230.
7	DCFP	Data Cache Flush Parity Error 0 Exception not caused by DCU Flush parity error 1 Exception caused by DCU Flush parity error	Set if and only If the DCU parity error was discovered during a DCU Flush operation. See <i>Data Cache Parity Operations</i> on page 230.
8	IMCE	Imprecise Machine Check Exception 0 No imprecise machine check exception occurred. 1 Imprecise machine check exception occurred.	Set if a machine check exception occurs that sets MCSR[MCS] (or would if it were not already set) and MSR[ME] = 0.
9:31		Reserved	

SPR 0x23A Supervisor R/W

See *Machine Check Save/Restore Register 0 (MCSRR0)* on page 428.

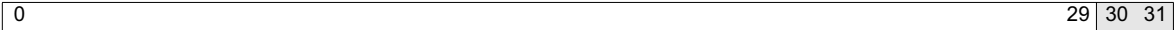


Figure 32-35. Machine Check Save/Restore Register 0 (MCSRR0)

0:29		Return address for machine check interrupts
30:31		Reserved

SPR 0x23B Supervisor R/W

See *Machine Check Save/Restore Register 1 (MCSRR1)* on page 429.

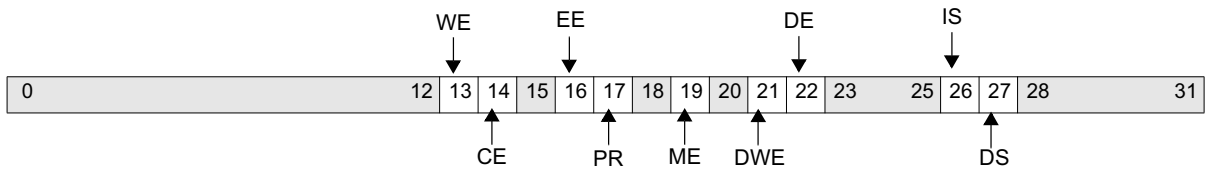
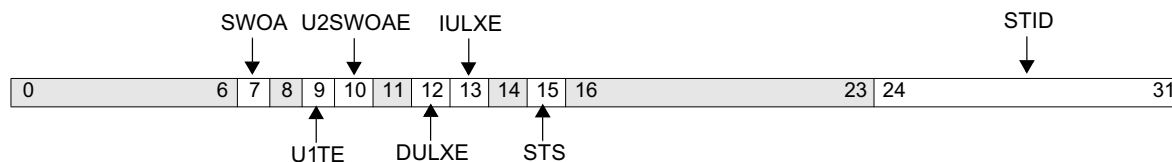


Figure 32-36. Machine Check Save/Restore Register 1 (MCSRR1)

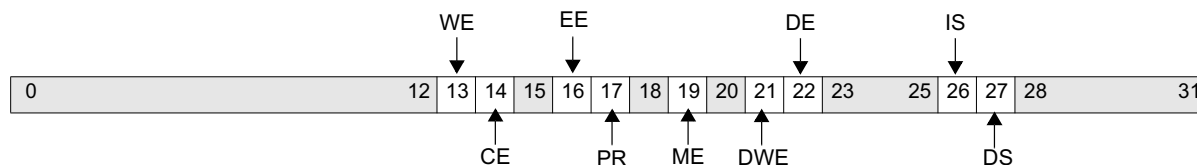
0:31	Copy of the MSR at the time of a machine check interrupt.
------	---

SPR 0x3B2 Supervisor R/WSee *Memory Management Unit Control Register (MMUCR)* on page 249.*Figure 32-37. Memory Management Unit Control Register (MMUCR)*

0:6		Reserved	
7	SWOA	Store Without Allocate 0 Cacheable store misses allocate a line in the data cache. 1 Cacheable store misses do not allocate a line in the data cache.	If MMUCR[U2SWOAE] = 1, this field is ignored.
8		Reserved	
9	U1TE	U1 Transient Enable 0 Disable U1 storage attribute as transient storage attribute. 1 Enable U1 storage attribute as transient storage attribute.	
10	U2SWOAE	U2 Store without Allocate Enable 0 Disable U2 storage attribute control of store without allocate. 1 Enable U2 storage attribute control of store without allocate.	If MMUCR[U2SWOAE] = 1, the U2 storage attribute overrides MMUCR[SWOA].
11		Reserved	
12	DULXE	Data Cache Unlock Exception Enable 0 Data cache unlock exception is disabled. 1 Data cache unlock exception is enabled.	dcbf in user mode causes a Cache Locking exception type Data Storage interrupt when MMUCR[DULXE] is 1.
13	IULXE	Instruction Cache Unlock Exception Enable 0 Instruction cache unlock exception is disabled. 1 Instruction cache unlock exception is enabled.	icbi in user mode causes a Cache Locking exception type Data Storage interrupt when MMUCR[IULXE] is 1.
14		Reserved	
15	STS	Search Translation Space	Specifies the value of the translation space (TS) field for tlbsx[.]
16:23		Reserved	
24:31	STID	Search Translation ID	Specifies the value of the TID field for the tlbsx[.] ; also used to transfer a TLB entry's TID value for tlbre and tlbwe .

MSR

Machine State Register

Preliminary User's Manual**Supervisor R/W**See *Machine State Register (MSR)* on page 424.*Figure 32-38. Machine State Register (MSR)*

0:12		Reserved	
13	WE	Wait State Enable 0 The processor is not in the wait state. 1 The processor is in the wait state.	If MSR[WE] = 1, the processor remains in the wait state until an interrupt is taken, a reset occurs, or an external debug tool clears WE.
14	CE	Critical Interrupt Enable 0 Critical Input and Watchdog Timer interrupts are disabled. 1 Critical Input and Watchdog Timer interrupts are enabled.	
15		Reserved	
16	EE	External Interrupt Enable 0 External Input, Decrementer, and Fixed Interval Timer interrupts are disabled. 1 External Input, Decrementer, and Fixed Interval Timer interrupts are enabled.	
17	PR	Problem State 0 Supervisor state (privileged instructions can be executed) 1 Problem state (privileged instructions can not be executed)	
18		Reserved	
19	ME	Machine Check Enable 0 Machine Check interrupts are disabled. 1 Machine Check interrupts are enabled.	
20		Reserved	
21	DWE	Debug Wait Enable 0 Disable debug wait mode. 1 Enable debug wait mode.	
22	DE	Debug interrupt Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.	
23:25		Reserved	
26	IS	Instruction Address Space 0 All instruction storage accesses are directed to address space 0 (TS = 0 in the relevant TLB entry). 1 All instruction storage accesses are directed to address space 1 (TS = 1 in the relevant TLB entry).	
27	DS	Data Address Space 0 All data storage accesses are directed to address space 0 (TS = 0 in the relevant TLB entry). 1 All data storage accesses are directed to address space 1 (TS = 1 in the relevant TLB entry).	
28:31		Reserved	

Preliminary User’s Manual

SPR 0x030 Supervisor R/W

See *Process ID (PID)* on page 252.



Figure 32-39. Process ID (PID)

0:23		Reserved
24:31	PID	Process ID

SPR 0x11E Supervisor Read-Only

See *Processor Identification Register (PIR)* on page 191.



Figure 32-40. Processor Identification Register (PIR)

0:27		Reserved
28:31	PIN	Processor Identification Number (PIN)

SPR 0x11F Supervisor Read-Only

See *Processor Version Register (PVR)* on page 190.

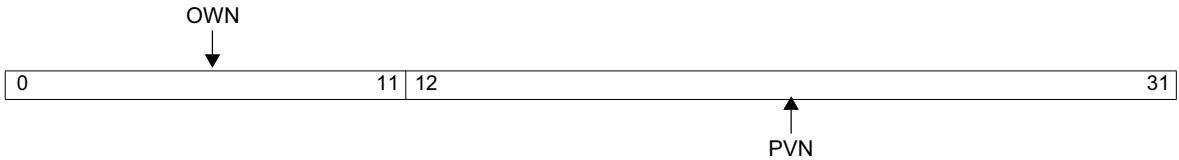


Figure 32-41. Processor Version Register (PVR)

0:31		Processor Version	Refer to <i>PowerPC 440GX Embedded Processor Data Sheet</i> for the PVR value.
------	--	-------------------	--

SPR 39B Supervisor Read-Only

See *Reset Configuration (RSTCFG)* on page 194.

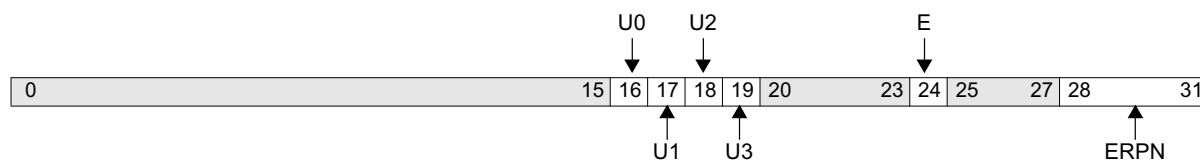


Figure 32-42. Reset Configuration

0:15		Reserved	
16	U0	U0 Storage Attribute 0 U0 storage attribute is disabled 1 U0 storage attribute is enabled	U0 has no effect in the PPC440GX Embedded Processor.
17	U1	U1 Storage Attribute 0 Memory page contains normal instructions and data 1 Memory page contains transient instructions or data	
18	U2	U2 Storage Attribute 0 A storage miss does not cause a line to be allocated in the data cache 1 A storage miss causes a line to be allocated in the data cache	
19	U3	U3 Storage Attribute 0 U3 storage attribute is disabled 1 U3 storage attribute is enabled	U3 has no effect in the PPC440GX Embedded Processor.
20:23		Reserved	
24	E	E Storage Attribute 0 Accesses to the page are big endian. 1 Accesses to the page are little endian.	
25:27		Reserved	
28:31	ERP	Extended Real Page Number	If ROM is connected to EBCO, ERP is 0b0001. If ROM is connected to PCI, ERP is 0b0010.

SPR 0x104–0x107 (User/Supervisor Read-Only); SPR 0x110–0x113 (Supervisor R/W);
SPR 0x114–0x117 (Supervisor Write-Only)

See *Special Purpose Registers General (USPRG0, SPRG0–SPRG7)* on page 190.

0	31
---	----

Figure 32-43. Special Purpose Registers General (SPRG0–SPRG7)

0:31	General data	Software value; no hardware usage.
------	--------------	------------------------------------

SPR 0x01A Supervisor R/W

See *Save/Restore Register 0 (SRR0)* on page 426.



Figure 32-44. *Save/Restore Register 0 (SRR0)*

0:29		Return address for non-critical interrupts
30:31		Reserved

SPR 0x01B Supervisor R/W

See *Save/Restore Register 1 (SRR1)* on page 427.

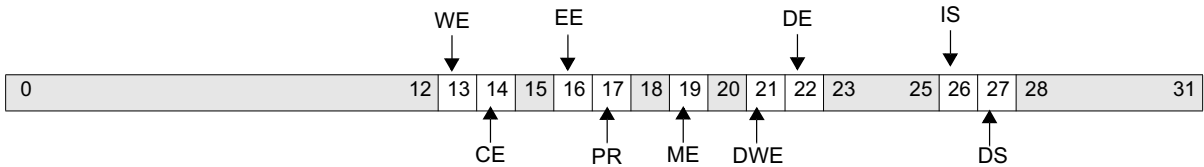


Figure 32-45. *Save/Restore Register 1 (SRR1)*

0:31		Copy of the MSR at the time of a non-critical interrupt.
------	--	--

SPR 0x10C (User/Supervisor Read-Only); SPR 0x11C (Supervisor Write-Only)

See *Time Base* on page 460.

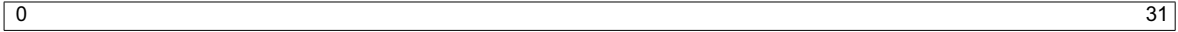


Figure 32-46. Time Base Lower (TBL)

0:31	Time Base Lower	Low-order 32 bits of time base.
------	-----------------	---------------------------------

SPR 0x10D (User/Supervisor Read-Only); SPR 0x11D (Supervisor Write-Only)

See *Time Base* on page 460.



Figure 32-47. Time Base Upper (TBU)

0:31	Time Base Upper	High-order 32 bits of time base.
------	-----------------	----------------------------------

SPR 0x154 Supervisor R/W

See *Timer Control Register (TCR)* on page 466.

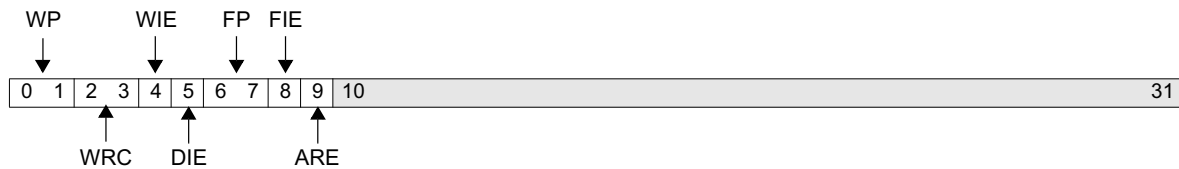


Figure 32-48. Timer Control Register (TCR)

0:1	WP	Watchdog Timer Period 00 2^{21} time base clocks 01 2^{25} time base clocks 10 2^{29} time base clocks 11 2^{33} time base clocks	
2:3	WRC	Watchdog Timer Reset Control 00 No Watchdog Timer reset will occur. 01 Core reset 10 Chip reset 11 System reset	TCR[WRC] resets to 0b00. Type of reset to cause upon Watchdog Timer exception with TSR[ENW,WIS]=0b11. This field can be set by software, but cannot be cleared by software, except by a software-induced reset.
4	WIE	Watchdog Timer Interrupt Enable 0 Disable Watchdog Timer interrupt. 1 Enable Watchdog Timer interrupt.	
5	DIE	Decrementer Interrupt Enable 0 Disable Decrementer interrupt. 1 Enable Decrementer interrupt.	
6:7	FP	Fixed Interval Timer (FIT) Period 00 2^{13} time base clocks 01 2^{17} time base clocks 10 2^{21} time base clocks 11 2^{25} time base clocks	
8	FIE	FIT Interrupt Enable 0 Disable Fixed Interval Timer interrupt. 1 Enable Fixed Interval Timer interrupt.	
9	ARE	Auto-Reload Enable 0 Disable auto reload. 1 Enable auto reload.	TCR[ARE] resets to 0b0.
10:31		Reserved	

SPR 0x150 Supervisor Read/Clear

See *Timer Status Register (TSR)* on page 467.

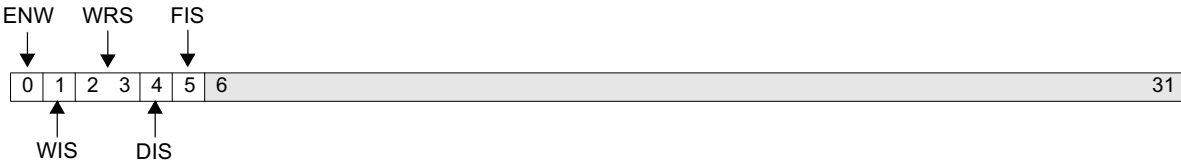


Figure 32-49. *Timer Status Register (TSR)*

0	ENW	Enable Next Watchdog Timer Exception 0 Action on next Watchdog Timer exception is to set TSR[ENW] = 1. 1 Action on next Watchdog Timer exception is governed by TSR[WIS].
1	WIS	Watchdog Timer Interrupt Status 0 Watchdog Timer exception has not occurred. 1 Watchdog Timer exception has occurred.
2:3	WRS	Watchdog Timer Reset Status 00 No Watchdog Timer reset has occurred. 01 Core reset was forced by Watchdog Timer. 10 Chip reset was forced by Watchdog Timer. 11 System reset was forced by Watchdog Timer.
4	DIS	Decrementer Interrupt Status 0 Decrementer exception has not occurred. 1 Decrementer exception has occurred.
5	FIS	Fixed Interval Timer (FIT) Interrupt Status 0 Fixed Interval Timer exception has not occurred. 1 Fixed Interval Timer exception has occurred.
6:31		Reserved

SPR 0x100 (User R/W)

See *Special Purpose Registers General (USPRG0, SPRG0–SPRG7)* on page 190.



Figure 32-50. User Special Purpose Register General (USPRG0)

0:31	General data	Software value; no hardware usage.
------	--------------	------------------------------------

SPR 0x001 User R/W

See *Integer Exception Register (XER)* on page 187.

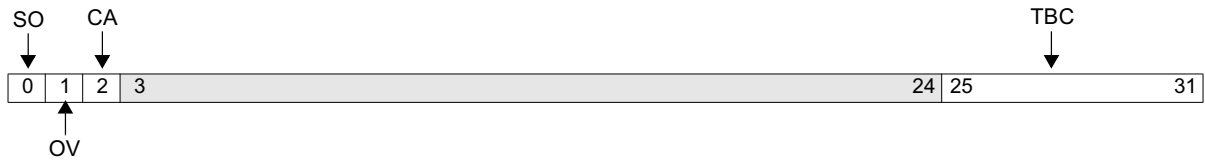


Figure 32-51. Integer Exception Register (XER)

0	SO	Summary Overflow 0 No overflow has occurred. 1 Overflow has occurred.	Can be <i>set</i> by mtspr or by integer or auxiliary processor instructions with the [o] option; can be <i>reset</i> by mtspr or by mcrxr .
1	OV	Overflow 0 No overflow has occurred. 1 Overflow has occurred.	Can be <i>set</i> by mtspr or by integer or allocated instructions with the [o] option; can be <i>reset</i> by mtspr , by mcrxr , or by integer or allocated instructions with the [o] option.
2	CA	Carry 0 Carry has not occurred. 1 Carry has occurred.	Can be <i>set</i> by mtspr or by certain integer arithmetic and shift instructions; can be <i>reset</i> by mtspr , by mcrxr , or by certain integer arithmetic and shift instructions.
3:24		Reserved	
25:31	TBC	Transfer Byte Count	Used as a byte count by lswx and stswx ; written by dlimzb[.] and by mtspr .

32.6 Alphabetical Chip Control and Peripheral Core Register Listing

The following pages list all the device control registers (DCRs) and memory mapped registers (MMIOs) implemented in the PPC440GX. For each register, the following information is supplied:

- Register mnemonic and name
- Cross reference to detailed register information
- Register type (DCR, MMIO)
- Register number (address)
- Register programming model (user or supervisor) and access (read-clear, read-only, read/write (R/W), write-only)
- A diagram illustrating the register fields (all register fields have mnemonics, unless there is only one field)
- A table describing the register fields, giving field mnemonics, field bit locations, field names, and the functions associated with the various field values

DCR accessed using CPR_CFGADDR and CPR_CFGDATA; Offset: 0x0020 R/W

Clocking Update Register (CPR0_CLKUPD) on page 490

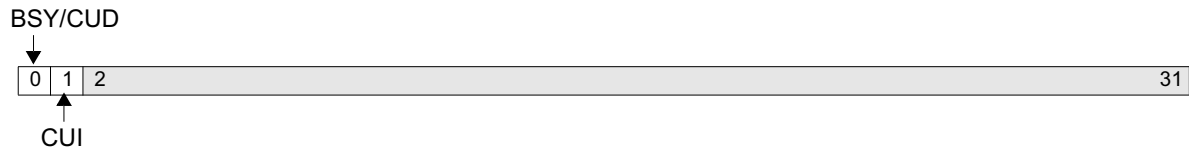


Figure 32-52. Clocking Update Register (CPR0_CLKUPD)

0	BSY	Clocking Subsystem Busy 0 Clocking subsystem is stable 1 Clocking subsystem is making changes.	In Read Access Mode. When CPR0_CLKUPD[BSY=1], software needs to wait until CPR0_CLKUPD[BSY]=0 before initiating any additional changes.
	CUD	Clocking Update Delay 0 Clocking update delay disabled 1 Clocking update delay enabled	In Write Access Mode. When CPR0_CLKUPD[CUD] is set to 1, then: Clocking update action is taken. Instruct the clocking logic to begin changing clocks to the newly programmed divider values. Result. System clocks will be momentarily stopped and then restarted using the currently programmed values in the CPR0 divider registers
1	CUI	Clocking Update Immediate 0 Clocking update immediate disabled 1 Clocking update immediate enabled	When you read it this bit is always 0. In Write Access Mode. When CPR0_CLKUPD[CUI]=1: Clocking update action taken. Allow the currently programmed values in all CPR0 registers to take effect immediately. Result. The clocking subsystem is now programmed according to the current settings of all CPR0 registers
2:31		Reserved	

Note: The CPR0_CLKUPD register is unique, in that writes will automatically result in an update to the clocking subsystem, while reads of the same bit(s) will report a status. Please note that this register is documented here for reference purposes only and should not be used to change clocking modes.

DCR accessed using CPR_CFGADDR and CPR_CFGDATA; Offset: 0x0140 R/W

Initial Configuration Register (CPR0_ICFG) on page 286

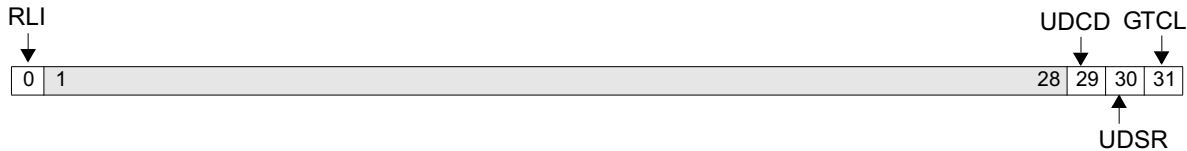


Figure 32-53. Initial Configuration Register (CPR0_ICFG)

0	RLI	Reload Inhibit 0 Reset CPR registers from configuration source defined by CPR0_ICFG[UDCD, UDSR, GTCL] 1 Ignore CPR0_ICFG[UDCD, UDSR, GTCL] configuration source and preserve current values in CPR0 registers (including CPR0_ICFG)	CPR0_ICFG[RLI] is used to preserve contents of CPR0 registers during a chip reset.
1:28		Reserved	
29	UDCD	UART0_DCD Strapping 0 UART0_DCD strapped low 1 UART0_DCD strapped high	Note: SDR0_PINSTP[UDCD] contains a copy of the pin strap status
30	UDSR	UART0_DSR Strapping 0 UART0_DSR strapped low 1 UART0_DSR strapped high	Note: SDR0_PINSTP[UDSR] contains a copy of the pin strap status
31	GTCL	GMC1TxCtl Strapping 0 GMC1TxCtl strapped low 1 GMC1TxCtl strapped high	Note: SDR0_PINSTP[GCTL] contains a copy of the pin strap status

Preliminary User's Manual

DCR accessed using CPR_CFGADDR and CPR_CFGDATA; Offset: 0x0100 R/W

MAL Clock Divider Register (CPR0_MALD) on page 495



Figure 32-54. MAL Clock Divider Register (CPR0_MALD)

0:5		Reserved
6:7	MALDV0	<div>MAL Clock Divisor 0</div> <div>00 MAL clock divisor 0 = 4</div> <div>01 MAL clock divisor 0 = 1</div> <div>10 MAL clock divisor 0 = 2</div> <div>11 MAL clock divisor 0 = 3</div> <div>Note: The minimum MAL clock frequency is 45MHz.</div>
8:31		Reserved

DCR accessed using CPR_CFGADDR and CPR_CFGDATA; Offset: 0x00C0 R/W

OPB Clock Divider Register (CPR0_OPBD) on page 494



Figure 32-55. OPB Clock Divisor Register (CPR0_OPBD)

0:5		Reserved
6:7	OPBDV0	OPB Clock Divisor 0 00 OPB clock divisor 0 = 4 01 OPB clock divisor 0 = 1 10 OPB clock divisor 0 = 2 11 OPB clock divisor 0 = 3
8:31		Reserved

DCR accessed using CPR_CFGADDR and CPR_CFGDATA; Offset: 0x00E0 R/W

Peripheral Clock Divisor Register (CPR0_PERD) on page 495



Figure 32-56. Peripheral Clock Divisor Register (CPR0_PERD)

0:5		Reserved	
6:7	PERDV0	Peripheral Clock Divisor 0 00 Peripheral clock divisor 0 = 4 01 Peripheral clock divisor 0 = 1 10 Peripheral clock divisor 0 = 2 11 Peripheral clock divisor 0 = 3	
8:31		Reserved	

CPR0_PLLC

PLL Control Register

Preliminary User's Manual

DCR accessed using CPR_CFGADDR and CPR_CFGDATA; Offset: 0x0040 R/W

PLL Control Register (CPR0_PLLC) on page 491

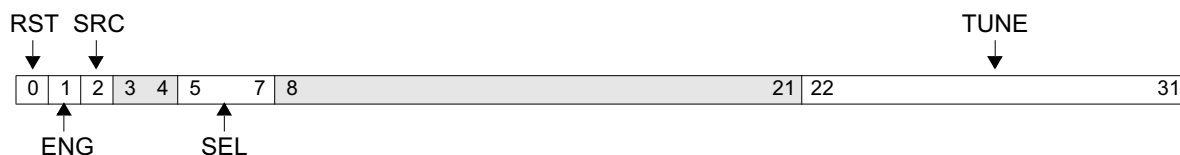


Figure 32-57. PLL Control Register (CPR0_PLLC)

0	RST	Reset 0 PLL allowed to lock 1 PLL is forced into reset.	For the CPR clocking logic, the reset value of the RST bit is the complement of the ENG bit
1	ENG	Engage 0 SysClk is the source for primary forward divisor. 1 PLL's VCO is the source for primary forward divisor.	
2	SRC	PLL Feedback Source 0 Feedback originates from PLLOUTA 1 Feedback originates from PLLOUTB	
3:4		Reserved	
5:7	SEL	Feedback Selection 000 PLL output (A or B) 001 CPU 101 PERClk	All other combinations are reserved Selection of PLL output A or B implies that the feedback clock is taken exactly at or close to the PLL output, and not at the end of a repowered clock tree. Using this feedback source implies that the PLL is not being used to adjust the generated clocks to be phase aligned with SysClk. The specific clock used for feedback is controlled by CPR0_PLLC[SRC] bit 2.
8:21		Reserved	
22:31	TUNE	TUNE bits	

DCR accessed using CPR_CFGADDR and CPR_CFGDATA; Offset: 0x0060 R/W

PLL Divider Register (CPR0_PLLD) on page 492

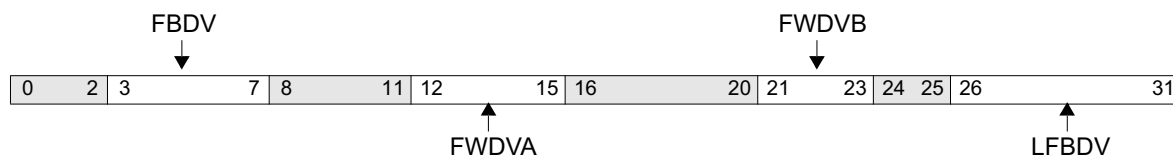


Figure 32-58. PLL Divisor Register (CPR0_PLLD)

0:2		Reserved
3:7	FBDV	PLL Feedback Divisor 00000 PLL Feedback Divisor = 32 00001 PLL Feedback Divisor = 1 00010 PLL Feedback Divisor = 2 00011 PLL Feedback Divisor = 3 11111 PLL Feedback Divisor = 31
8:11		Reserved
12:15	FWDVA	PLL Forward Divisor A 0000 PLL Forward Divisor A = 16 0001 PLL Forward Divisor A = 1 0010 PLL Forward Divisor A = 2 0011 PLL Forward Divisor A = 3 0100 PLL Forward Divisor A = 4 0101 PLL Forward Divisor A = 5 0110 PLL Forward Divisor A = 6 0111 PLL Forward Divisor A = 7 1000 PLL Forward Divisor A = 8 1001 Reserved 1010 PLL Forward Divisor A = 10 1011 Reserved 1100 PLL Forward Divisor A = 12 1101 Reserved 1110 PLL Forward Divisor A = 14 1111 Reserved
16:20		Reserved
21:23	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 PLL Forward Divisor B = 1 010 PLL Forward Divisor B = 2 011 PLL Forward Divisor B = 3 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 5 110 PLL Forward Divisor B = 6 111 PLL Forward Divisor B = 7
24:25		Reserved

26:31	LFBDV	<p>PLL Local Feedback Divisor</p> <p>00_0000 PLL local feedback divisor = 64</p> <p>00_0001 PLL local feedback divisor = 1</p> <p>00_0010 PLL local feedback divisor = 2</p> <p>.....</p> <p>.....</p> <p>.....</p> <p>11_1111 PLL local feedback divisor = 63</p>	<p>The LFBDV is outside of PLL and is used to allow the PLL to lock using feedback directly from a PLL output. Program this divider to a value equivalent to the divide from the PLL to some clock to allow the PLL to be switched to use feedback from that clock. This allows the PLL to compensate for the latency associated with that clock's tree.</p>
-------	-------	--	--

Preliminary User’s Manual

DCR accessed using CPR_CFGADDR and CPR_CFGDATA; Offset: 0x0080 R/W

Primary A Divider Register (CPR0_PRIMAD) on page 493



Figure 32-59. Primary A Divider Register (CPR0_PRIMAD)

0:4		Reserved
5:7	PRADV0	PLL Primary Divisor A 000 PLL Primary Divisor A = 8 001 PLL Primary Divisor A = 1 010 PLL Primary Divisor A = 2 011 PLL Primary Divisor A = 3 100 PLL Primary Divisor A = 4 101 PLL Primary Divisor A = 5 110 PLL Primary Divisor A = 6 111 PLL Primary Divisor A = 7
8:31		Reserved

DCR accessed using CPR_CFGADDR and CPR_CFGDATA; Offset: 0x00A0 R/W

Primary B Divider Register (CPR0_PRIMBD) on page 494



Figure 32-60. Primary B Divider Register (CPR0_PRIMBD)

0:4		Reserved
5:7	PRBDV0	PLL Primary Divisor B 000 PLL Primary Divisor B = 8 001 PLL Primary Divisor B = 1 010 PLL Primary Divisor B = 2 011 PLL Primary Divisor B = 3 100 PLL Primary Divisor B = 4 101 PLL Primary Divisor B = 5 110 PLL Primary Divisor B = 6 111 PLL Primary Divisor B = 7
8:31		Reserved

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0240 R/W

Alternate PLB Master Priority Register (SDR0_AMP) on page 78

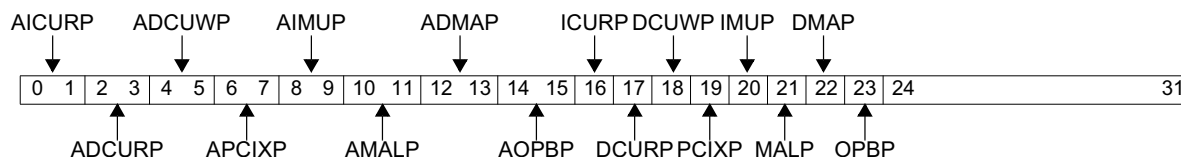


Figure 32-61. Alternate PLB Master Priority Register (SDR0_AMP)

0:1	AICURP	Alternate ICU Read Priority 00 Lowest 01 10 11 Highest Overrides priority specified by SDR0_CP440[IRF, IRT, IRS]
2:3	ADCURP	Alternate DCU Read Priority 00 Lowest 01 10 11 Highest Overrides priority specified by SDR0_CP440[DRU, DRT, DRNC, DRLC]
4:5	ADCUWP	Alternate DCU Write Priority 00 Lowest 01 10 11 Highest Overrides priority specified by SDR0_CP440[DWF, DWS, DWU]
6:7	APCIXP	Alternate PCIX Priority 00 Lowest 01 10 11 Highest Overrides priority specified by PCIX0_BROPT1[PLREQ]
8:9	AIMUP	Alternate IMU Priority 00 Lowest 01 10 11 Highest Overrides priority specified by IMU0_PPR[PR]
10:11	AMALP	Alternate MAL Priority 00 Lowest 01 10 11 Highest Overrides priority specified by MAL0_CFG[PLBP]
12:13	ADMAP	Alternate DMA Priority 00 Lowest 01 10 11 Highest Overrides priority specified by DMA0_CR[CP]
14:15	AOPBP	Alternate OPB to PLB Bridge Priority 00 Lowest 01 10 11 Highest Overrides priority specified by OPB0_BCTRL[PR]
16	ICURP	ICU Read Priority 0 ICU read controls own priority 1 ICU read uses alternate priority
17	DCURP	DCU Read Priority 0 DCU read controls own priority 1 DCU read uses alternate priority

18	DCUWP	DCU Write Priority 0 DCU write controls own priority 1 DCU write uses alternate priority
19	PCIXP	PCIX Priority 0 PCIX controls own priority 1 PCIX uses alternate priority
20	IMUP	IMU Priority 0 IMU controls own priority 1 IMU uses alternate priority
21	MALP	MAL Priority 0 MAL controls own priority 1 MAL uses alternate priority
22	DMAP	DMA Priority 0 DMA controls own priority 1 DMA uses alternate priority
23	OPBP	OPB to PLB Bridge Priority 0 OPB to PLB bridge controls own priority 1 OPB to PLB bridge uses alternate priority
24:31		Reserved

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0180 R/W

PPC440 CPU Control Register (SDR0_CP440) on page 80

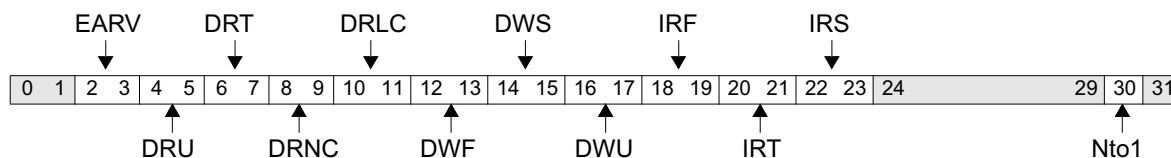


Figure 32-62. PPC440 CPU Register (SDR0_CP440)

0:1		Reserved	
2:3	EARV	ERPN Address Reset Vector 00 Reserved 01 Boot from EBC 10 Boot from PCI 11 Reserved	Specifies the upper four bits of the 36-bit reset address vector. If SDR0_CP440[EARV=01] then ERPN is 0b0001. If SDR0_CP440[EARV=10] then ERPN is 0b0010.
4:5	DRU	DcuRdUrgent 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with an urgent state in which two or more read data cache operations are pending, waiting for the previous request to be serviced. PLB master priority is updated to this value when in urgent state regardless of instruction type.
6:7	DRT	DcuRdTouch 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with dcbt instructions except when in urgent state.
8:9	DRNC	DcuRdNonCache 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with non-cacheable load instructions except when in urgent state.
10:11	DRLC	DcuRdLdCache 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with cacheable load instructions except when in urgent state.
12:13	DWF	DcuWrFlush 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with flush instructions except when in urgent state.
14:15	DWS	DcuWrStore 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with store instructions except when in urgent state.
16:17	DWU	DcuWrUrgent 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with an urgent state in which two or more write data cache operations are pending, waiting for the previous request to be serviced. PLB master priority is updated to this value when in urgent state regardless of instruction type.

18:19	IRF	IcuRdFetch 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with non-speculative ICU accesses.
20:21	IRT	IcuRdTouch 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with icbt instructions.
22:23	IRS	IcuRdSpec 00 Lowest 01 10 11 Highest	2-bit PLB priority level associated with speculative ICU accesses.
24:29		Reserved	
30	Nto1	CPU:PLB N to 1 clock ratio 0 CPU:PLB clock ratio is N:P where P is greater than 1 1 CPU:PLB clock ratio is N:1	
31		Reserved	

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x4000 R-only

Custom Configuration Register 0 (SDR0_CUST0) on page 356



Figure 32-63. Custom Configuration Register 0 (SDR0_CUST0)

0:31	Reserved	Reserved for user's data
------	----------	--------------------------

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x4002 Read-only
Custom Configuration Register 1 (SDR0_CUST1) on page 357



Figure 32-64. Custom Configuration Register 1 (SDR0_CUST1)

0:31	Reserved	Reserved for user's data
------	----------	--------------------------

Preliminary User’s Manual

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x00E0 R/W
DDR Delay Line Register (SDR0_DDRDL) on page 296



Figure 32-65. DDR Delay Line Register (SDR0_DDRDL)

0:25		Reserved	
26:31	TUNE	DDR Delay Line PLL TUNE	Reset value: 010010

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0100 R/W

EBC Configuration Register (SDR0_EBC) on page 296



Figure 32-66. EBC Configuration Register (SDR0_EBC)

0:1		Reserved
2:3	RW	ROM Width 00 8-bit ROM 01 16-bit ROM 10 32-bit ROM 11 reserved
4:31		Reserved

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0080 R/W

Electronic Chip ID Register 0 (SDR0_ECID0) on page 297



Figure 32-67. Electronic Chip ID Register 0 (SDR0_ECID0)

0:31	ECID0	Electronic chip ID 0
------	-------	----------------------

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0081 R/W

Electronic Chip ID Register 1 (SDR0_ECID1) on page 297



Figure 32-68. Electronic Chip ID Register 1 (SDR0_ECID1)

0:31	ECID1	Electronic chip ID 1
------	-------	----------------------

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0082 R/W

Electronic Chip ID Register 2 (SDR0_ECID2) on page 297

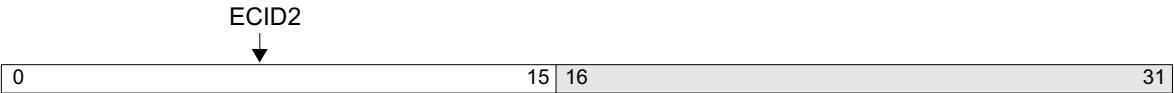


Figure 32-69. Electronic Chip ID Register 1 (SDR0_ECID2)

0:15	ECID2	Electronic chip ID 2
16:31		Reserved

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x00C0 R/W

JTAG ID Register (SDR0_JTAGID) on page 505

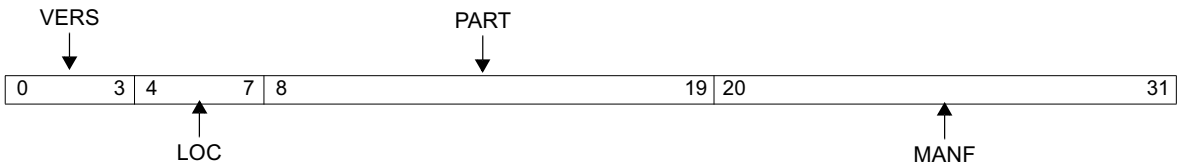


Figure 32-70. JTAG ID Register (SDR0_JTAG)

0:3	VERS	Version
4:7	LOC	Developer Location
8:19	PART	Part Number
20:31	MANF	Manufacturer Identifier

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x02A0 R/W

MAL Receive Burst Length Register (SDR0_MALRBL) on page 298

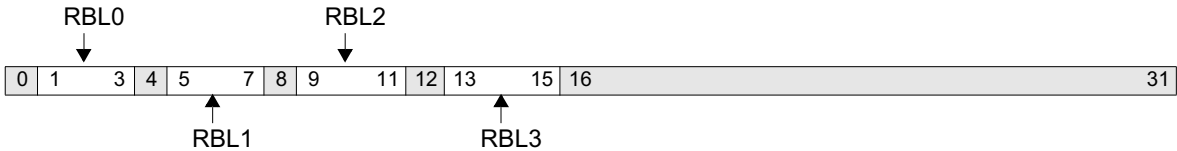


Figure 32-71. MAL Receive Burst Length Register (SDR0_MALRBL)

0		Reserved	
1:3	RBL0	MAL Receive Burst Length 0	Defines the maximum burst length transferred between EMAC0 and MAL. Reset default value = 0b101 (64 words).
4		Reserved	
5:7	RBL1	MAL Receive Burst Length 1	Defines the maximum burst length transferred between EMAC1 and MAL. Reset default value = 0b101 (64 words)
8		Reserved	
9:11	RBL2	MAL Receive Burst Length 2	Defines the maximum burst length transferred between EMAC2 and MAL. Reset default value = 0b101 (64 words)
12		Reserved	
13:15	RBL3	MAL Receive Burst Length 3	Defines the maximum burst length transferred between EMAC3 and MAL. Reset default value = 0b101 (64 words)
16:31		Reserved	
Note: Please use the default values for all the bits described in this register.			

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x02E0 R/W

MAL Receive Bus Size Register (SDR0_MALRBS) on page 299



Figure 32-72. MAL Receive Bus Size Register (SDR0_MALRBS)

0:3	RBS	MAL Receive Bus Size 0b0000 - reserved 0b1111 - 128-bit bus size	Indicates the OPB slave interface width for all four EMAC channels is 128 bits
4:31		Reserved	
Note: Please use the default values for all the bits described in this register.			

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0280 R/W

MAL Transmit Burst Length Register (SDR0_MALTBL) on page 299

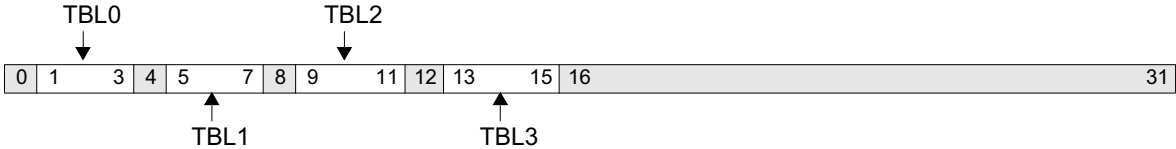


Figure 32-73. MAL Transmit Burst Length Register (SDR0_MALTBL)

0		Reserved	
1:3	TBL0	MAL Transfer Burst Length 0	Defines the maximum burst length transferred between MAL and EMAC0. Reset default value = 0b101 (64 words).
4		Reserved	
5:7	TBL1	MAL Transfer Burst Length 1	Defines the maximum burst length transferred between MAL and EMAC1. Reset default value = 0b101 (64 words).
8		Reserved	
9:11	TBL2	MAL Transfer Burst Length 2	Defines the maximum burst length transferred between MAL and EMAC2. Reset default value = 0b101 (64 words).
12		Reserved	
13:15	TBL3	MAL Transfer Burst Length 3	Defines the maximum burst length transferred between MAL and EMAC3. Reset default value = 0b101 (64 words).
16:31		Reserved	
Note: Please use the default values for all the bits described in this register.			

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x02C0 R/W

MAL Transmit Bus Size Register (SDR0_MALTBS) on page 300



Figure 32-74. MAL Transmit Bus Size Register (SDR0_MALTBS)

0:3	TBS	MAL Transfer Bus Size 0b0000 - reserved 0b1111 - 128-bit bus size	Indicates the OPB slave interface width for all four EMAC channels is 128 bits
4:31		Reserved	
Note: Please use the default values for all the bits described in this register.			

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x4300 R/W

Miscellaneous Function Register (SDR0_MFR) on page 300

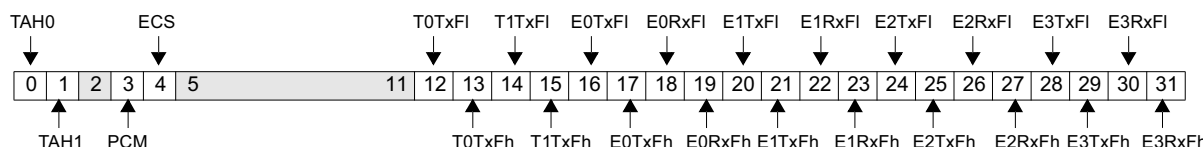


Figure 32-75. Miscellaneous Function Register (SDR0_MFR)

0	TAH0	TAH 0 Enable 0 TAH 0 enabled 1 TAH 0 disabled
1	TAH1	TAH 1 Enable 0 TAH 1 enabled 1 TAH 1 disabled
2		Reserved By default this bit is set to 0 and should not be changed.
3	PCM	PPC440GP Compatibility Mode 0 PPC440GX mode enabled 1 PPC440GP mode enabled
4	ECS	Ethernet Clock Select 0 Select external clock 1 Select internal clock
5:11		Reserved
12	T0TxFI	Force Parity Error TAH0 Tx Fifo Bits 0:63 0 No forced parity error 1 Force parity error into TAH0 Tx Fifo
13	T0TxFh	Force Parity Error TAH0 Tx Fifo Bits 64:127 0 No forced parity error 1 Force parity error into TAH0 Tx Fifo
14	T1TxFI	Force Parity Error TAH1 Tx Fifo Bits 0:63 0 No forced parity error 1 Force parity error into TAH1 Tx Fifo
15	T1TxFh	Force Parity Error TAH1 Tx Fifo Bits 64:127 0 No forced parity error 1 Force parity error into TAH1 Tx Fifo
16	E0TxFI	Force Parity Error EMAC0 Tx Fifo Bits 0:63 0 No forced parity error 1 Force parity error into EMAC0 Tx Fifo
17	E0TxFh	Force Parity Error EMAC0 Tx Fifo Bits 64:127 0 No forced parity error 1 Force parity error into EMAC0 Tx Fifo
18	E0RxFI	Force Parity Error EMAC0 Rx Fifo Bits 0:63 0 No forced parity error 1 Force parity error into EMAC0 Rx Fifo
19	E0RxFh	Force Parity Error EMAC0 Rx Fifo Bits 64:127 0 No forced parity error 1 Force parity error into EMAC0 Rx Fifo
20	E1TxFI	Force Parity Error EMAC1 Tx Fifo Bits 0:63 0 No forced parity error 1 Force parity error into EMAC1 Tx Fifo

21	E1TxFh	Force Parity Error EMAC1 Tx Fifo Bits 64:127 0 No forced parity error 1 Force parity error into EMAC1 Tx Fifo
22	E1RxFl	Force Parity Error EMAC1 Rx Fifo Bits 0:63 0 No forced parity error 1 Force parity error into EMAC1 Rx Fifo
23	E1RxFh	Force Parity Error EMAC1 Rx Fifo Bits 64:127 0 No forced parity error 1 Force parity error into EMAC1 Rx Fifo
24	E2TxFl	Force Parity Error EMAC2 Tx Fifo Bits 0:63 0 No forced parity error 1 Force parity error into EMAC2 Tx Fifo
25	E2TxFh	Force Parity Error EMAC2 Tx Fifo Bits 64:127 0 No forced parity error 1 Force parity error into EMAC2 Tx Fifo
26	E2RxFl	Force Parity Error EMAC2 Rx Fifo Bits 0:63 0 No forced parity error 1 Force parity error into EMAC2 Rx Fifo
27	E2RxFh	Force Parity Error EMAC2 Rx Fifo Bits 64:127 0 No forced parity error 1 Force parity error into EMAC2 Rx Fifo
28	E3TxFl	Force Parity Error EMAC3 Tx Fifo Bits 0:63 0 No forced parity error 1 Force parity error into EMAC3 Tx Fifo
29	E3TxFh	Force Parity Error EMAC3 Tx Fifo Bits 64:127 0 No forced parity error 1 Force parity error into EMAC3 Tx Fifo
30	E3RxFl	Force Parity Error EMAC3 Rx Fifo Bits 0:63 0 No forced parity error 1 Force parity error into EMAC3 Rx Fifo
31	E3RxFh	Force Parity Error EMAC3 Rx Fifo Bits 64:127 0 No forced parity error 1 Force parity error into EMAC3 Rx Fifo

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0260 R/W

Master Interrupt Request Register 0 (SDR0_MIRQ0) on page 81

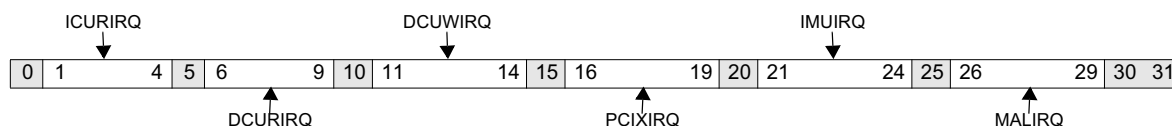


Figure 32-76. Master Interrupt Request Register 0 (SDR0_MIRQ0)

0		Reserved	
1:4	ICURIRQ	ICU Read Interrupt Request	1 - ICU read interrupt request from DDR SDRAM 2 - Reserved 3 - ICU read interrupt request from PLB to OPB bridge 4 - ICU read interrupt request from IMU controller.
5		Reserved	
6:9	DCURIRQ	DCU Read Interrupt Request	6 - DCU read interrupt request from DDR SDRAM 7 - Reserved 8 - DCU read interrupt request from PLB to OPB bridge 9 - DCU read interrupt request from IMU controller.
10		Reserved	
11:14	DCUWIRQ	DCU Write Interrupt Request	11 - DCU write interrupt request from DDR SDRAM 12 - Reserved 13 - DCU write interrupt request from PLB to OPB bridge 14 - DCU write interrupt request from IMU controller.
15		Reserved	
16:19	PCIXIRQ	PCIX Interrupt Request	16 - PCIX interrupt request from DDR SDRAM 17 - Reserved 18 - PCIX interrupt request from PLB to OPB bridge 19 - PCIX interrupt request from IMU controller.
20		Reserved	
21:24	IMUIRQ	IMU Interrupt Request	21 - IMU interrupt request from DDR SDRAM 22 - Reserved 23 - IMU interrupt request from PLB to OPB bridge 24 - IMU interrupt request from IMU controller.
25		Reserved	
26:29	MALIRQ	MAL Interrupt Request	26 - MAL interrupt request from DDR SDRAM 27 - Reserved 28 - MAL interrupt request from PLB to OPB bridge 29 - MAL interrupt request from IMU controller.
30:31		Reserved	

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0261 R/W

Master Interrupt Request Register 1 (SDR0_MIRQ1) on page 82

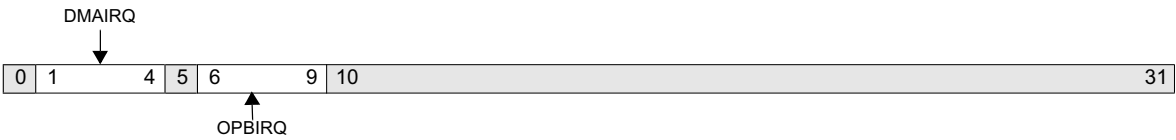


Figure 32-77. Master Interrupt Request Register 1 (SDR0_MIRQ1)

0		Reserved	
1:4	DMAIRQ	DMA Interrupt Request	1 - DMA interrupt request from DDR SDRAM 2 - Reserved 3 - DMA interrupt request from PLB to OPB bridge 4 - DMA interrupt request from IMU controller.
5		Reserved	
6:9	OPBIRQ	OPB to PLB Bridge interrupt Request	6 - OPB to PLB bridge interrupt request from DDR SDRAM 7 - Reserved 8 - OPB to PLB bridge interrupt request from PLB to OPB bridge 9 - OPB to PLB bridge interrupt request from IMU controller.
10:31		Reserved	

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x4100 R/W

Pin Function Control Register 0 (SDR0_PFC0) on page 303

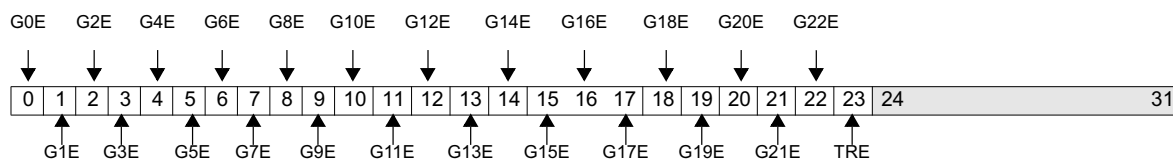


Figure 32-78. Pin Function Control Register 0 (SDR0_PFC0)

0	G0E	GPIO 0 Enable 0 Enable interrupt IRQ00 as an interrupt 1 Enable interrupt IRQ00 as GPIO0
1	G1E	GPIO 1 Enable 0 Enable interrupt IRQ01 as an interrupt 1 Enable interrupt IRQ01 as GPIO1
2	G2E	GPIO 2 Enable 0 Enable interrupt IRQ02 as an interrupt 1 Enable interrupt IRQ02 as GPIO2
3	G3E	GPIO 3 Enable 0 Enable interrupt IRQ03 as an interrupt 1 Enable interrupt IRQ03 as GPIO3
4	G4E	GPIO 4 Enable 0 Enable interrupt IRQ04 as an interrupt 1 Enable interrupt IRQ04 as GPIO4
5	G5E	GPIO 5 Enable 0 Enable interrupt IRQ05 as an interrupt 1 Enable interrupt IRQ05 as GPIO5
6	G6E	GPIO 6 Enable 0 Enable interrupt IRQ06 as an interrupt 1 Enable interrupt IRQ06 as GPIO6
7	G7E	GPIO 7 Enable 0 Enable interrupt IRQ07 as an interrupt 1 Enable interrupt IRQ07 as GPIO7
8	G8E	GPIO 8 Enable 0 Enable interrupt IRQ08 as an interrupt 1 Enable interrupt IRQ08 as GPIO8
9	G9E	GPIO 9 Enable 0 Enable interrupt IRQ09 as an interrupt 1 Enable interrupt IRQ09 as GPIO9
10	G10E	GPIO 10 Enable 0 Enable interrupt IRQ010 as an interrupt 1 Enable interrupt IRQ010 as GPIO10
11	G11E	GPIO 11 Enable 0 Enable GPIO11 as GNCTxCik or TBIRxCik1 1 Enable GPIO11 as GPIO11
12	G12E	GPIO 12 Enable 0 Enable UART1RX as UART1RX 1 Enable UART1RX as GPIO 12

13	G13E	GPIO 13 Enable 0 Enable UART1TX as UART1TX 1 Enable UART1TX as GPIO 13	
14	G14E	GPIO 14 Enable 0 Enable UART1DSR_CTS as UART1DSR_CTS 1 Enable UART1DSR_CTS as GPIO 14	
15	G15E	GPIO 15 Enable 0 Enable UART1RTS_DTR as UART1RTS_DTR 1 Enable UART1RTS_DTR as GPIO 15	
16	G16E	GPIO 16 Enable 0 Enable IIC1SCL as IIC1SCL 1 Enable IIC1SCL as GPIO 16	This is an open drain driver and therefore needs to be pulled up.
17	G17E	GPIO 17 Enable 0 Enable IIC1SDA as IIC1SDA 1 Enable IIC1SDA as GPIO 17	This is an open drain driver and therefore needs to be pulled up.
18	G18E	GPIO 18 Enable 0 Select TrcBS0 as external interrupt 13 1 Select TrcBS0 as GPIO 18 if SDR0_PFC0[TRE]=0. Select TrcBS0 as TrcBS0 (CPU trace) if SDR0_PFC0[TRE]=1	
19	G19E	GPIO 19 Enable 0 Select TrcBS1 as external interrupt 14 1 Select TrcBS1 as GPIO 19 if SDR0_PFC0[TRE]=0. Select TrcBS1 as TrcBS1 (CPU trace) if SDR0_PFC0[TRE]=1	
20	G20E	GPIO 20 Enable 0 Select TrcBS2 as external interrupt 15 1 Select TrcBS2 as GPIO 20 if SDR0_PFC0[TRE]=0. Select TrcBS2 as TrcBS2 (CPU trace) if SDR0_PFC0[TRE]=1	
21	G21E	GPIO 21 Enable 0 Select TrcES0 as external interrupt 16 1 Select TrcES0 as GPIO 21 if SDR0_PFC0[TRE]=0. Select TrcES0 as TrcES0 (CPU trace) if SDR0_PFC0[TRE]=1	
22	G22E	GPIO 22 Enable 0 Select TrcES1 as external interrupt 17 1 Select TrcES1 as GPIO 22 if SDR0_PFC0[TRE]=0. Select TrcES1 as TrcES1 (CPU trace) if SDR0_PFC0[TRE]=1	
23	TRE	GPIO Trace Enable 0 Enable GPIO's 18-26 1 Enable CPU trace (RISCTrace support)	<p>If SDR0_PFC0[TRE]=0 Set SDR0_PFC0[G18E]=1 to enable GPIO 18 Set SDR0_PFC0[G19E]=1 to enable GPIO 19 Set SDR0_PFC0[G20E]=1 to enable GPIO 20 Set SDR0_PFC0[G21E]=1 to enable GPIO 21 Set SDR0_PFC0[G22E]=1 to enable GPIO 22</p> <p>If SDR0_PFC0[TRE]=1 Set SDR0_PFC0[G18E]=1 Set SDR0_PFC0[G19E]=1 Set SDR0_PFC0[G20E]=1 Set SDR0_PFC0[G21E]=1 Set SDR0_PFC0[G22E]=1</p>
24:31		Reserved	

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x4101 R/W

Pin Function Control Register 1 (SDR0_PFC1) on page 305

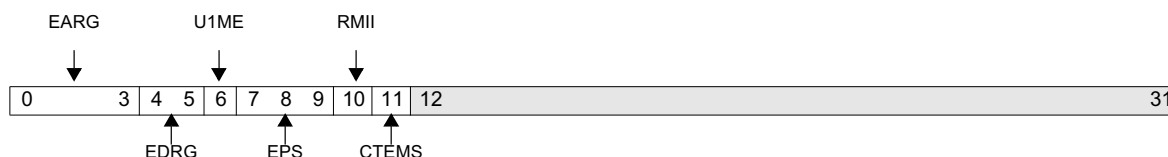


Figure 32-79. Pin Function Control Register 1 (SDR0_PFC1)

0:3	EARG	EBC Address Receiver Gating 0000 Receiver gating disabled 0001 Disable address bus inputs 0:1 0010 Disable address bus inputs 0:2 0011 Disable address bus inputs 0:3 1111 Disable address bus inputs 0:15	Encoded value for gating external bus controller address bus
4:5	EDRG	EBC Data Receiver Gating 00 Receiver gating disabled 01 Disable data bus input bytes 1, 2, 3 10 Disable data bus input bytes 2, 3 11 Disable data bus input byte 3	Encoded value for gating external bus controller data bus
6	U1ME	UART1 Mode Enable 0 Enable UART1DSR/CTS as DSR and UART1DTR/RTS as DTR 1 Enable UART1DSR/CTS as CTS and UART1DTR/RTS as RTS	
7:9	EPS	Ethernet Pin Selection 000 Group 0: MII 001 Group 1: RMII0, RMII1 010 Group 2: SMII0, SMII1, SMII2, SMII3 011 Group 3: RMII0, RGMII0/RTBIO 100 Group 4: SMII0, SMII1, RGMII0/RTBIO, RGMII1/RTBIO, GMII/TBI 101 Group 5: SMII0, SMII1, SMII2, RGMII1/RTBIO 110 Group 6: SMII0, SMII1, RGMII0/RTBIO 111 Group 7: Reserved	These groups provide several combinations of ethernet for the four gigabit ethernets implemented in PPC440GX. If selecting groups 4 or 5, SDR0_PFC0[TRE] must be set to 0. See Table 23-1 Ethernet Combinations on page 793 for addi- tional information.
10	RMII	RMII Mode 0 RMII mode 100 MBit 1 RMII mode 10 MBit	
11	CTEMS	CPU Trace/External Master Selection 0 Use trace A signal group 1 Use trace B signal group	Attention: Set SDR0_PFC0[TRE]=1 for CPU trace (RISCTrace) support Selecting SDR0_PFC1[CTEMS]=0 muxes the CPU trace functionality on the trace interface signals TrcTs1, TrcTS2, TrcTS3, TrcTS4 and TrcTS5. With this selection ethernet groups 3, 4, 5, 6 and GPIO's GPIO27, GPIO28, GPIO29, GPIO30 and GPIO31 cannot be used. Selecting SDR0_PFC1[CTEMS]=1 muxes the CPU trace functionality on the external master interface signals BusReq, ExtAck, ExtReq, HoldAck, HoldReq and PerErr. With this selection the external master interface cannot be used.
12:31		Reserved	

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0040 Read-only

Pin Strapping Register (SDR0_PINSTP) on page 351

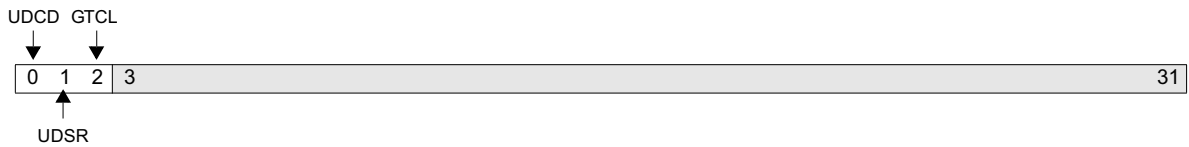


Figure 32-80. Pin Strapping Register (SDR0_PINSTP)

0	UDCD	UART0_DCD Strapping 0 UART0_DCD strapped low 1 UART0_DCD strapped high	Note: CPR0_ICFG[UDCD] contains a copy of the pin strap status
1	UDSR	UART0_DSR Strapping 0 UART0_DSR strapped low 1 UART0_DSR strapped high	Note: CPR0_ICFG[UDSR] contains a copy of the pin strap status
2	GTCL	GMC1TxCtl Strapping 0 GMC1TxCtl strapped low 1 GMC1TxCtl strapped high	Note: CPR0_ICFG[GTCL] contains a copy of the pin strap status
3:31		Reserved	

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0060 R/W

Serial Device Controller Settings Register (SDR0_SDCS) on page 351

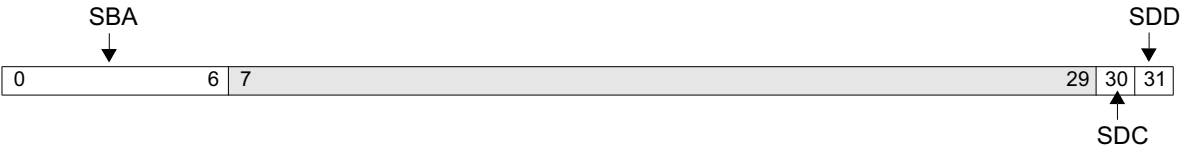


Figure 32-81. Serial Device Controller Settings Register (SDR0_SDCS)

0:6	SBA	SEEPROM Base Address	
7:29		Reserved	
30	SDC	Serial Device Control 0 Serial device control disabled 1 Serial device control enabled	
31	SDD	Serial Device Detection 0 Serial device detection disabled 1 Serial device detection enabled	SDR0_SDCR[SDD] = 1 if SDR0_SDCR[SDC] = 1

SDR0_SDSTP0

Serial Device Strap Register 0

Preliminary User's Manual

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0020 Read-only

Serial Device Strap Register 0 (SDR0_SDSTP0) on page 352

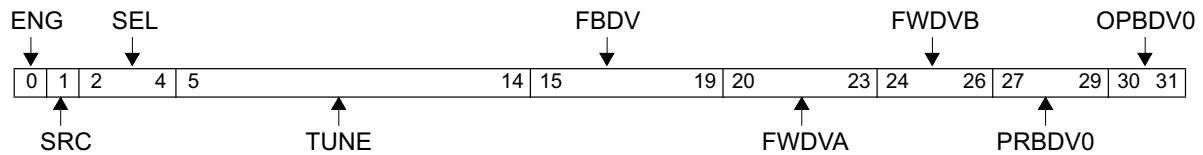


Figure 32-82. Serial Device Strap Register 0 (SDR0_SDSTP0)

0	ENG	Engage 0 SysClk is the source for PLL forward dividers. 1 PLL's VCO is the source for PLL forward dividers.
1	SRC	PLL Feedback Source 0 Feedback originates from PLLOUTA 1 Feedback originates from PLLOUTB
2:4	SEL	Feedback Selection 000 PLL output (A or B) 001 CPU 101 PERClk
5:14	TUNE	TUNE bits
15:19	FBDV	PLL Feedback Divisor 00000 PLL Feedback Divisor = 32 00001 PLL Feedback Divisor = 1 00010 PLL Feedback Divisor = 2 00011 PLL Feedback Divisor = 3 11111 PLL Feedback Divisor = 31
20:23	FWDVA	PLL Forward Divisor A 0000 PLL Forward Divisor A = 16 0001 PLL Forward Divisor A = 1 0010 PLL Forward Divisor A = 2 0011 PLL Forward Divisor A = 3 0100 PLL Forward Divisor A = 4 0101 PLL Forward Divisor A = 5 0110 PLL Forward Divisor A = 6 0111 PLL Forward Divisor A = 7 1000 PLL Forward Divisor A = 8 1001 Reserved 1010 PLL Forward Divisor A = 10 1011 Reserved 1100 PLL Forward Divisor A = 12 1101 Reserved 1110 PLL Forward Divisor A = 14 1111 Reserved
24:26	FWDVB	PLL Forward Divisor B 000 PLL Forward Divisor B = 8 001 PLL Forward Divisor B = 1 010 PLL Forward Divisor B = 2 011 PLL Forward Divisor B = 3 100 PLL Forward Divisor B = 4 101 PLL Forward Divisor B = 5 110 PLL Forward Divisor B = 6 111 PLL Forward Divisor B = 7

27:29	PRBDV0	PLL Primary Divisor B 000 PLL Primary Divisor B = 8 001 PLL Primary Divisor B = 1 010 PLL Primary Divisor B = 2 011 PLL Primary Divisor B = 3 100 PLL Primary Divisor B = 4 101 PLL Primary Divisor B = 5 110 PLL Primary Divisor B = 6 111 PLL Primary Divisor B = 7	Note: If CPR0_ICFG[RLI] = 0, then the reset value for PLL Primary Divisor A is 1 (CPR0_PRIMAD[PRADV0]=1).
30:31	OPBDV0	OPB Clock Divisor 0 00 OPB clock divisor 0 = 4 01 OPB clock divisor 0 = 1 10 OPB clock divisor 0 = 2 11 OPB clock divisor 0 = 3	

SDR0_SDSTP1

Serial Device Strap Register 1

Preliminary User's Manual

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0021 Read-only

Serial Device Strap Register 1 (SDR0_SDSTP1) on page 353

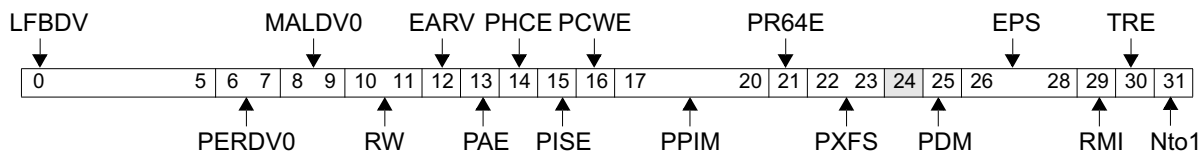


Figure 32-83. Serial Device Strap Register 1 (SDR0_SDSTP1)

0:5	LFBDV	PLL Local Feedback Divisor 00_0000 PLL local feedback divisor = 64 00_0001 PLL local feedback divisor = 1 00_0010 PLL local feedback divisor = 2 11_1111 PLL local feedback divisor = 63	
6:7	PERDV0	Peripheral Clock Divisor 0 00 Peripheral clock divisor 0 = 4 01 Peripheral clock divisor 0 = 1 10 Peripheral clock divisor 0 = 2 11 Peripheral clock divisor 0 = 3	
8:9	MALDV0	MAL Clock Divisor 0 00 MAL clock divisor 0 = 4 01 MAL clock divisor 0 = 1 10 MAL clock divisor 0 = 2 11 MAL clock divisor 0 = 3	
10:11	RW	EBC ROM Width 00 8-bit ROM 01 16-bit ROM 10 32-bit ROM 11 reserved	
12	EARV	ERPn Address Reset Vector 0 Boot from EBC 1 Boot from PCI	Specifies the upper four bits of the 36-bit reset address vector. If SDR0_SDSTP1[EARV=0] then ERPn is 0b0001. If SDR0_SDSTP1[EARV=1] then ERPn is 0b0010.
13	PAE	PCI arbiter enable 0 PCI arbiter disabled 1 PCI arbiter enabled	
14	PHCE	PCI host configuration enable 0 PCI host configuration disabled 1 PCI host configuration enabled	When PHCE=1, a PCI/PCIX host has access to PCI/PCIX registers. When PHCE=0, a PCI/PCIX host cannot access the PCI/PCIX registers and the PLB to PCIX bridge responds to type 0 configuration reads and writes with retries.
15	PISE	PCI initial sequence enable 0 PCI initial sequence disabled 1 PCI initial sequence enabled	Typically enabled when a PCI/PCIX host.
16	PCWE	PCI local CPU wait enable 0 PCI local CPU wait disabled 1 PCI local CPU wait enabled	Typically used when booting from PCI/PCIX attached memory. When PCWE=1, the 440CPU is halted and unable to boot until the PCI/PCIX host sets PCIX0_BRDOPT2[CPUW]=0.

17:20	PPIM	PCI Inbound Map (PIM) Settings 0000 PIM0 off, PIM1 off, PIM2 off 0001 PIM0 4K, PIM1 off, PIM2 off 0010 PIM0 1M, PIM1 off, PIM2 off 0011 PIM0 64M, PIM1 off, PIM2 off 0100 PIM0 4K prefetch enabled, PIM1 off, PIM2 off 0101 PIM0 1M prefetch enabled, PIM1 off, PIM2 off 0110 PIM0 64M prefetch enabled, PIM1 off, PIM2 off 0111 PIM0 64K, PIM1 off, PIM2 16k 1000 PIM0 1M, PIM1 off, PIM2 64k 1001 PIM0 64K prefetch enabled, PIM1 off, PIM2 16K 1010 PIM0 1M prefetch enabled, PIM1 off, PIM2 64K 1011 PIM0 64K, PIM1 off, PIM2 64K prefetch enabled 1100 PIM0 1M, PIM1 off, PIM2 1Mp 1101 PIM0 1M prefetch enabled, PIM1 off, PIM2 1M prefetch enabled 1110 PIM0 1M, PIM1 on, PIM2 off 1111 PIM0 1M, PIM1 on, PIM2 16K	These bits control the default settings for various PIM fields. See <i>Table 9-5</i> on page 348
21	PR64E	PCI initialize Req64 Enable	When PR64E=0, bits AD63:32 are driven by the PLB-PCIX Bridge controller, the Req64/Ack64 are not used and PCIX0_PCIXSTS[DEV]=0. Do not use pullups on AD63:32 in this configuration. When PR64E=1, bits AD63:32 float if not driving by a 64 bit address or 64bit data access. The content of PCIX0_PCIXSTS[DEV] is determined by Req64 and Ack64.
22:23	PXFS	PCIX Frequency Selection 00 100 - 133MHz frequency selection 01 66 - 100MHz frequency selection 10 50 - 66MHz frequency selection 11 reserved	Only applies to PCIX and not to conventional PCI.
24		Reserved	
25	PDM	PCIX Driver Mode 0 PCI compliant driver mode 1 PCIX compliant driver mode	
26:28	EPS	Ethernet Pin Selection (see table note) 000 Group 0: MII 001 Group 1: RMII0, RMII1 010 Group 2: SMII0, SMII1, SMII2, SMII3 011 Group 3: RMII0, RGMII0/RTBIO 100 Group 4: SMII0, SMII1, RGMII0/RTBIO, RGMII1/RTBI1, GMII/TBI 101 Group 5: SMII0, SMII1, SMII2, RGMII1/RTBI1 110 Group 6: SMII0, SMII1, RGMII0/RTBIO 111 Group 7: Reserved	These groups provide several combinations of ethernet for the four gigabit ethernets implemented in PPC440GX. See <i>Table 23-1 Ethernet Combinations</i> on page 793 for additional information.
29	RMII	RMII Mode 0 RMII mode 100 MBit 1 RMII mode 10 MBit	
30	TRE	GPIO Trace Enable 0 GPIO18-31 are enabled 1 GPIO18-31 are disabled	Trace interface cannot be used when GPIO is enabled. Trace interface can be used when GPIO is disabled.
31	Nto1	CPU:PLB N to 1 clock ratio 0 CPU:PLB clock ratio is N:P where P is greater than 1 1 CPU:PLB clock ratio is N:1	See <i>CPU / PLB Frequency N:1 Setting</i> on page 482

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x4001 Read-only
Serial Device Strap Register 2 (SDR0_SDSTP2) on page 355

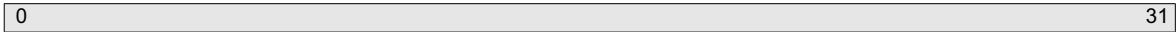


Figure 32-84. Serial Device Strap Register 2 (SDR0_SDSTP2)

0:31		Reserved	Reserved for user's data
------	--	----------	--------------------------

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x4003 Read-only

Serial Device Strap Register 3 (SDR0_SDSTP3) on page 356

0	31
---	----

Figure 32-85. Serial Device Strap Register 3 (SDR0_SDSTP3)

0:31	Reserved for user's data
------	--------------------------

SDR0_SLPIPE

PLB Slave Address Pipeline Register

Preliminary User's Manual

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0220 R/W

PLB Slave Address Pipeline Register (SDR0_SLPIPE) on page 83

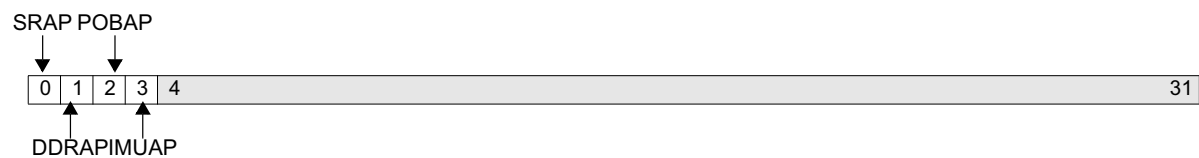


Figure 32-86. PLB Slave Address Pipeline Register (SDR0_SLPIPE)

0	SRAP	SRAM Address Pipeline 0 SRAM address pipeline disabled 1 SRAM address pipeline enabled	
1	DDRAP	DDR SDRAM Address Pipeline 0 DDR SDRAM address pipeline disabled 1 DDR SDRAM address pipeline enabled	
2	POBAP	PLB to OPB Bridge Address Pipeline 0 PLB to OPB bridge address pipeline disabled 1 PLB to OPB bridge address pipeline enabled	
3	IMUAP	IMU Address Pipeline 0 IMU address pipeline disabled 1 IMU address pipeline enabled	
4:31		Reserved	

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0200 R/W

Soft Reset Register (SDR0_SRST) on page 312

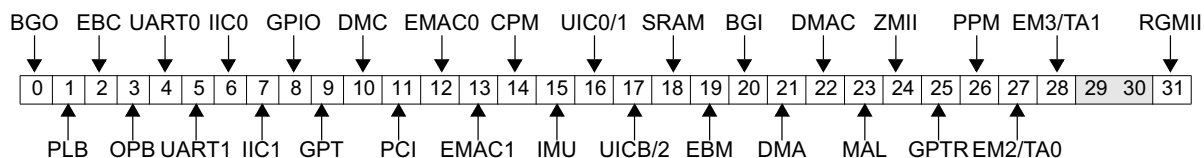


Figure 32-87. Soft Reset Register (SDR0_SRST)

0	BGO	PLB to OPB bridge
1	PLB	PLB arbiter
2	EBC	External bus controller
3	OPB	OPB arbiter
4	UART0	Universal asynchronous receiver/transmitter 0
5	UART1	Universal asynchronous receiver/transmitter 1
6	IIC0	Inter integrated circuit 0
7	IIC1	Inter integrated circuit 1
8	GPIO	General purpose I/O
9	GPT	General Purpose Timer
10	DMC	DDR SDRAM memory controller
11	PCI	PCI
12	EMAC0	Ethernet media access controller 0
13	EMAC1	Ethernet media access controller 1
14	CPM	Clock and power management
15	IMU	I2O messaging unit
16	UIC0 UIC1	Universal interrupt controller 0 Universal interrupt controller 1
17	UICB UIC2	Universal interrupt controller base Universal interrupt controller 2
18	SRAM	Internal SRAM controller
19	EBM	External bus master
20	BGI	OPB to PLB bridge
21	DMA	Direct memory access controller
22	DMAC	DMA channel
23	MAL	Media access layer
24	ZMII	Z media independent interface
25	GPTR	General purpose timer
26	PPM	PLB performance monitor
27	EMAC2 TAH0	Ethernet media access controller 2 TCP/IP accelerator hardware on ethernet 0
28	EMAC3 TAH1	Ethernet media access controller 3 TCP/IP accelerator hardware on ethernet 1

29:30		Reserved
31	RGMII	Reduced gigabit media independent interface

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0120 R/W

UART Configuration Registers (SDR0_UART0 and SDR0_UART1) on page 918

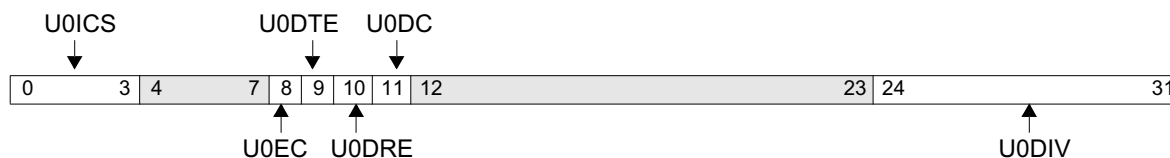


Figure 32-88. UART Configuration Register 0 (SDR0_UART0)

0:3	U0ICS	UART 0 Internal Clock Source 0000 Reserved 0001 Reserved 0010 UART 0 internal clock source = PLB 0011 Reserved
4:7		Reserved
8	U0EC	UART 0 EXT Clock Enable 0 UART0 uses the internal serial clock 1 UART0 uses the external serial clock
9	U0DTE	UART 0 DMA Transmit Enable 0 UART0 DMA transmit is disabled 1 UART0 DMA transmit is enabled
10	U0DRE	UART 0 DMA Receive Enable 0 UART0 DMA receive is disabled 1 UART0 DMA receive is enabled
11	U0DC	UART 0 DMA Clear 0 U0DTE and U0DRE are not cleared when UART receives a corresponding terminal count. 1 U0DTE and U0DRE are cleared when UART receives a corresponding terminal count.
12:23		Reserved
24:31	U0DIV	UART 0 Divider 0000_0000 - divider = 256 0000_0001 - divider = 1 0000_0010 - divider = 2 1111_1111 - divider = 255 The source for SDR0_UART0[U0DIV] is indicated by SDR0_UART0[U0ICS]

SDR0_UART1

UART Configuration Register 1

Preliminary User's Manual

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x0121 R/W

UART Configuration Registers (SDR0_UART0 and SDR0_UART1) on page 918

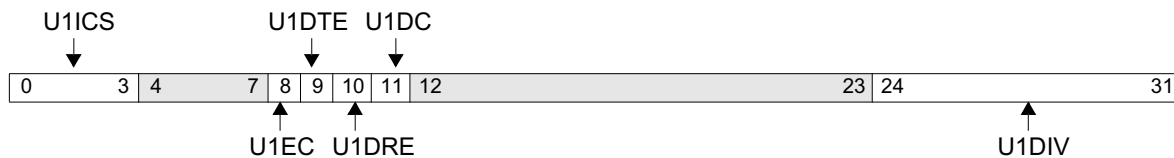


Figure 32-89. UART Configuration Register 1 (SDR0_UART1)

0:3	U1ICS	UART 1 Internal Clock Source 0000 Reserved 0001 Reserved 0010 UART 1 internal clock source = PLB 0011 Reserved
4:7		Reserved
8	U1EC	UART 1 EXT Clock Enable 0 UART1 uses the internal serial clock 1 UART1 uses the external serial clock
9	U1DTE	UART 1 DMA Transmit Enable 0 UART1 DMA transmit is disabled 1 UART1 DMA transmit is enabled
10	U1DRE	UART 1 DMA Receive Enable 0 UART1 DMA receive is disabled 1 UART1 DMA receive is enabled
11	U1DC	UART 1 DMA Clear 0 U1DTE and U1DRE are not cleared when UART receives a corresponding terminal count. 1 U1DTE and U1DRE are cleared when UART receives a corresponding terminal count.
12:23		Reserved
24:31	U1DIV	UART 1 Divider 0000_0000 - UART1 divisor = 256 0000_0001 - UART1 divisor = 1 0000_0010 - UART1 divisor = 2 1111_1111 - UART1 divisor = 255 The source for SDR0_UART1[U1DIV] is indicated by SDR0_UART1[U1ICS]

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x01C0 R/W

PCIX Control Register (SDR0_XCR) on page 357

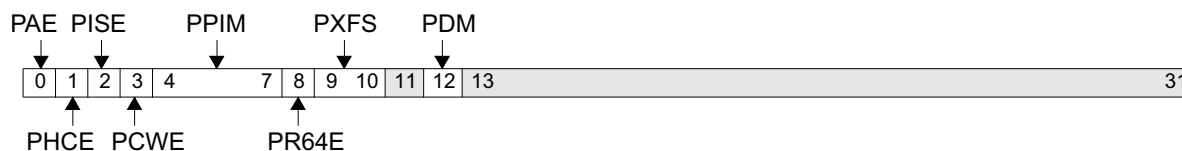


Figure 32-90. PCIX Control Register (SDR0_XCR)

0	PAE	PCI arbiter enable 0 PCI arbiter disabled 1 PCI arbiter enabled	
1	PHCE	PCI host configuration enable 0 PCI host configuration disabled 1 PCI host configuration enabled	When PHCE=1, a PCI/PCIX host has access to PCI/PCIX registers. When PHCE=0, a PCI/PCIX host cannot access the PCI/PCIX registers and the PLB to PCIX bridge responds to type 0 configuration reads and writes with retries.
2	PISE	PCI initial sequence enable 0 PCI initial sequence disabled 1 PCI initial sequence enabled	Typically enabled when a PCI/PCIX host.
3	PCWE	PCI local CPU wait enable 0 PCI local CPU wait disabled 1 PCI local CPU wait enabled	Typically used when booting from PCI/PCIX attached memory. When PCWE=1, the 440CPU is halted and unable to boot until the PCI/PCIX host sets PCIX0_BRDOPT2[CPUW]=0.
4:7	PPIM	PCI Inbound Map (PIM) Settings 0000 PIM0 off, PIM1 off, PIM2 off 0001 PIM0 4K, PIM1 off, PIM2 off 0010 PIM0 1M, PIM1 off, PIM2 off 0011 PIM0 64M, PIM1 off, PIM2 off 0100 PIM0 4K prefetch enabled, PIM1 off, PIM2 off 0101 PIM0 1M prefetch enabled, PIM1 off, PIM2 off 0110 PIM0 64M prefetch enabled, PIM1 off, PIM2 off 0111 PIM0 64K, PIM1 off, PIM2 16K 1000 PIM0 1M, PIM1 off, PIM2 64K 1001 PIM0 64K prefetch enabled, PIM1 off, PIM2 16K 1010 PIM0 1M prefetch enabled, PIM1 off, PIM2 64K 1011 PIM0 64K, PIM1 off, PIM2 64K prefetch enabled 1100 PIM0 1M, PIM1 off, PIM2 1M prefetch enabled 1101 PIM0 1M prefetch enabled, PIM1 off, PIM2 1M prefetch enabled 1110 PIM0 1M, PIM1 on, PIM2 off 1111 PIM0 1M, PIM1 on, PIM2 16K	These bits control the default settings for various PIM fields. See Table 9-5 on page 348
8	PR64E	PCI initialize Req64 Enable 0 PCI initialize Req64 disabled 1 PCI initialize Req64 enabled	When PR64E=0, bits AD63:32 are driven by the PCIX bridge, the Req64/Ack64 are not used and PCIX0_PCIXSTS[DEV]=0. Do not use pul-lups on AD63:32 in this configuration. When PR64E=1, bits AD63:32 float if not driving by a 64 bit address or 64bit data access. The content of PCIX0_PCIXSTS[DEV] is determined by Req64 and Ack64.

9:10	PXFS	PCIX Frequency Selection 00 100 - 133MHz frequency selection 01 66 - 100MHz frequency selection 10 50 - 66MHz frequency selection 11 reserved	Only applies to PCIX and not to conventional PCI.
11		Reserved	
12	PDM	PCIX Driver Mode 0 PCI compliant mode 1 PCIX compliant mode	The reset value will be dependent on the system configuration
13:31		Reserved	

Preliminary User’s Manual

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x01C1 R/W

PCIX PLL Control Register (SDR0_XPLLC) on page 316

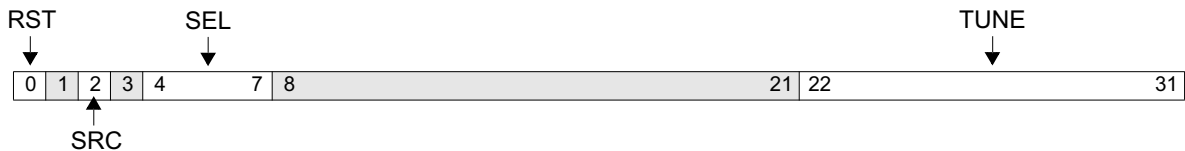


Figure 32-91. PCIX PLL Control Register (SDR0_XPLLC)

0	RST	Reset 0 PLL allowed to lock 1 PLL is forced into reset.
1		Reserved
2	SRC	PLL Feedback Source 0 Feedback originates from PLLOUTA 1 Feedback originates from PLLOUTB
3		Reserved
4:7	SEL	Feedback Selection 0000 PLL output (A or B) All other combinations are reserved Selection of PLL output A or B implies that the feedback clock is taken exactly at or close to the PLL output, and not at the end of a repowered clock tree. Using this feedback source implies that the PLL is not being used to adjust the generated clocks to be phase aligned with SysClk. The specific clock used for feedback is controlled by CPR0_PLLCN[<i>SRC</i>] bit 3
8:21		Reserved
22:31	TUNE	TUNE bits

DCR accessed using SDR0_CFGADDR and SDR0_CFGDATA; Offset: 0x01C2 R/W

PCIX PLL Divisor Register (SDR0_XPLLD) on page 317

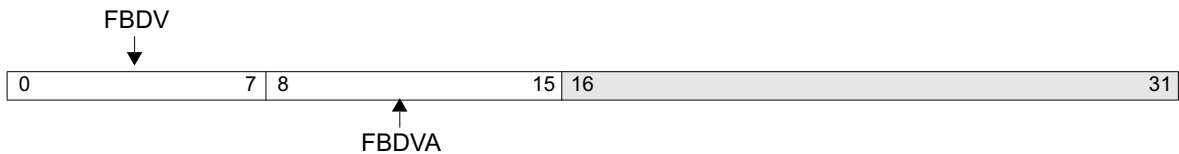


Figure 32-92. PCIX PLL Divisor Register (SDR0_XPLLD)

0:7	FBDV	PLL Feedback Divisor 0000 0001 PLL Feedback Divisor = 1
8:15	FWDVA	PLL Forward Divisor A 0b0000 PLL Forward Divisor A = max 0b0000 PLL Forward Divisor A = 1 0b0000 PLL Forward Divisor A = 2 0b1111 PLL Forward Divisor A = 255
16:31		Reserved

DCR 0x0B0 Read/Write

CPM Enable Register (CPM0_ER) on page 498

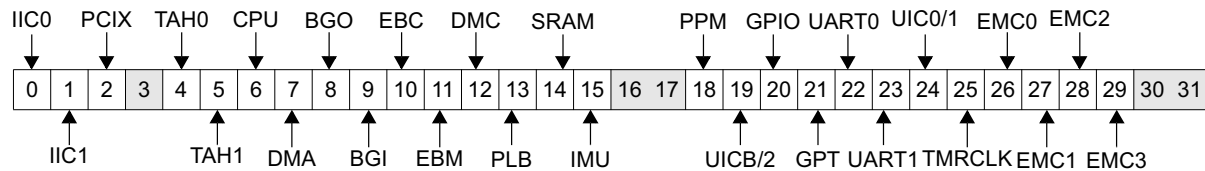


Figure 32-93. CPM Enable Register (CPM0_ER)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCIX	Peripheral Component Interconnect	Class 1
3		Reserved	
4	TAH0	TCP/IP Accelerator on Hardware 0	Class 1
5	TAH1	TCP/IP Accelerator on Hardware 1	Class 1
6	CPU	PPC440GX Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2
15	IMU	I2O Messaging Unit	Class 2
16:17		Reserved	
18	PPM	PLB Performance Monitor	Class 1
19	UICB UIC2	Base Universal Interrupt Controller Universal Interrupt Controller 2	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0 UIC1	Universal Interrupt Controller 0 Universal Interrupt Controller 1	Class 1
25	TMRCLK	CPU Timer	Class 1
26	EMC0	Ethernet 0	Class 1
27	EMC1	Ethernet 1	Class 1
28	EMC2	Ethernet 2	Class 1
29	EMC3	Ethernet 3	Class 1
30:31		Reserved	

CPM0_FR

CPM Force Register

Preliminary User's Manual

CPM0_FR

DCR 0x0B1 Read/Write

CPM Force Register (CPM0_FR) on page 499

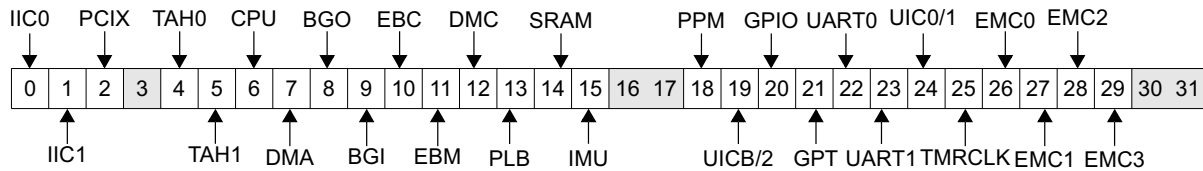


Figure 32-94. CPM Force Register (CPM0_FR)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCIX	Peripheral Component Interconnect	Class 1
3		Reserved	
4	TAH0	TCP/IP Accelerator on Hardware 0	Class 1
5	TAH1	TCP/IP Accelerator on Hardware 1	Class 1
6	CPU	PPC440GX Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2
15	IMU	I2O Messaging Unit	Class 2
16:17		Reserved	
18	PPM	PLB Performance Monitor	Class 1
19	UICB UIC2	Base Universal Interrupt Controller Universal Interrupt Controller 2	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0 UIC1	Universal Interrupt Controller 0 Universal Interrupt Controller 1	Class 1
25	TMRCLK	CPU Timer	Class 1
26	EMC0	Ethernet 0	Class 1
27	EMC1	Ethernet 1	Class 1
28	EMC2	Ethernet 2	Class 1
29	EMC3	Ethernet 3	Class 1
30:31		Reserved	

Preliminary User's Manual

CPM0_SR

CPM Status Register

CPM0_SR

DCR 0x0B2 Read/Only

CPM Status Register (CPM0_SR) on page 501

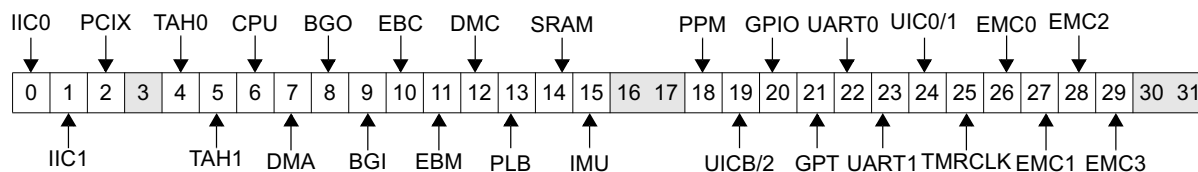


Figure 32-95. CPM Status Register (CPM0_SR)

0	IIC0	Inter-Integrated Circuit 0	Class 3
1	IIC1	Inter-Integrated Circuit 1	Class 3
2	PCIX	Peripheral Component Interconnect	Class 1
3		Reserved	
4	TAH0	TCP/IP Accelerator on Hardware 0	Class 1
5	TAH1	TCP/IP Accelerator on Hardware 1	Class 1
6	CPU	PPC440GX Processor Core	Class 2
7	DMA	Direct Memory Access Controller	Class 2
8	BGO	PLB to OPB Bridge	Class 2
9	BGI	OPB to PLB Bridge	Class 2
10	EBC	External Bus Controller	Class 2
11	EBM	External Bus Master Interface	Class 2
12	DMC	DDR SDRAM Controller	Class 2
13	PLB	PLB Arbiter	Class 2
14	SRAM	Internal SRAM Controller	Class 2
15	IMU	I2O Messaging Unit	Class 2
16:17		Reserved	
18	PPM	PLB Performance Monitor	Class 1
19	UICB UIC2	Base Universal Interrupt Controller Universal Interrupt Controller 2	Class 1
20	GPIO	General Purpose IO	Class 1
21	GPT	General Purpose Timer	Class 1
22	UART0	Universal Asynchronous Receiver/Transmitter 0	Class 1
23	UART1	Universal Asynchronous Receiver/Transmitter 1	Class 1
24	UIC0 UIC1	Universal Interrupt Controller 0 Universal Interrupt Controller 1	Class 1
25	TMRCLK	CPU Timer	Class 1
26	EMC0	Ethernet 0	Class 1
27	EMC1	Ethernet 1	Class 1
28	EMC2	Ethernet 2	Class 1
29	EMC3	Ethernet 3	Class 1
30:31		Reserved	

DMA0_CR0-DMA0_CR3

DMA Channel Control Registers 0–3

Preliminary User's Manual

DCR 0x100, 0x108, 0x110, 0x118 R/W

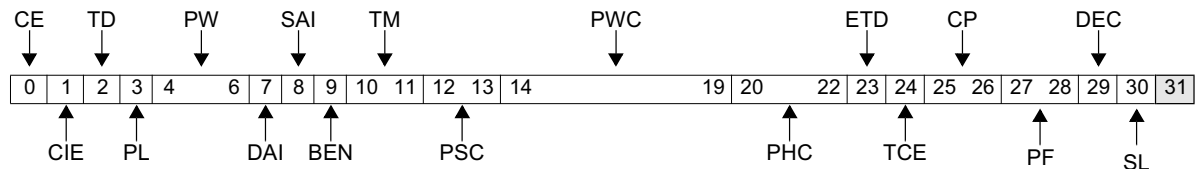
See *DMA Channel Control Registers (DMA0_CR0-DMA0_CR3)* on page 701

Figure 32-96. DMA Channel Control Registers (DMA0_CR0-DMA0_CR3)

0	CE	Channel Enable 0 Channel is disabled 1 Channel is enabled	This field is automatically cleared when the transfer completes or an error occurs.
1	CIE	Channel Interrupt Enable 0 Disable interrupts from this channel 1 Enable interrupts from this channel	When enabled, interrupts can be generated for terminal count, end of transfer, and errors conditions. Each of these interrupt types must be individually enabled in DMA0_CTn. See "Interrupts."
2	TD	In peripheral mode: 0 Transfers are from memory-to-peripheral 1 Transfers are from peripheral-to-memory In device-paced memory-to-memory mode: 0 Peripheral is at the destination address 1 Peripheral is at the source address	TD is not used (don't care) for software-initiated memory-to-memory transfers.
3	PL	Peripheral Location/Destination Memory Location For memory-to-memory transfers: 0 Destination address located in PLB memory space 1 Destination address located in OPB memory space For peripheral/DPM-to-memory or memory-to-DPM/peripheral transfers: 0 Device located on external bus 1 Device located on the OPB.	
4:6	PW	Peripheral Width/Transfer Width 000 Byte (8 bits) 001 Halfword (16 bits) 010 Word (32 bits) 011 Doubleword (64 bits) 100 Quadword (128 bits)	Transfers involving peripheral, device-paced-memory, and OPB FIFO devices can only be byte, halfword, or word. This limitation also applies to transfers involving EBC memory when SAI=0 or DAI=0. PLB FIFO devices can only be quadword size.
7	DAI	Destination Address Increment 0 Do not increment destination address 1 After each data transfer increment the destination address by: <ul style="list-style-type: none"> 1, for byte (8-bit) transfers 2, for halfword (16-bit) transfers 4, for word (32-bit) transfers 8, for doubleword (64-bit) transfers 16, for quadword (128-bit) transfers 	Valid only when DEC=0. Is required that DAI=1 for peripheral-to-memory and device-paced memory-to-memory transfers since peripheral-to-FIFO type devices or DPM-to-FIFO type devices are not supported.

8	SAI	Source Address Increment 0 Do not increment source address 1 After each data transfer increment the source address by: • 1, for byte (8-bit) transfers • 2, for halfword (16-bit) transfers • 4, for word (32-bit) transfers • 8, for doubleword (64-bit) transfers • 16, for quadword (128-bit) transfers	Valid only when DEC=0. Is required that SAI=1 for memory-to-peripheral and memory-to-device-paced memory transfers since FIFO-to-peripheral/DPM transfers are not supported.
9	BEN	Buffer Enable 0 Disable DMA 128-byte buffer 1 Enable DMA 128-byte buffer	If BEN=0 discrete read and write operations occur for each data transfer.
10:11	TM	Transfer mode 00 Peripheral 01 Reserved 10 Software-initiated memory-to-memory 11 Device-paced memory-to-memory	
12:13	PSC	Peripheral Setup Cycles 0-3	Number of PerClk cycles that the peripheral bus is idle from the last peripheral bus transaction to DMAAckn becoming active. Used only for the peripheral side of peripheral mode transfers.
14:19	PWC	Peripheral Wait Cycles 0-63	DMAAckn remains active for PWC+1 PerClk cycles. Used only for the peripheral side of peripheral mode transfers.
20:22	PHC	Peripheral Hold Cycles 0-7	The number of PerClk cycles between the time that DMAAckn becomes inactive until the peripheral bus is available for the next bus access. Used only during the peripheral side of peripheral mode transfers.
23	ETD	End-of-Transfer/Terminal Count (EOTn[TCn]) Pin Direction 0 EOTn[TCn] is an EOT input 1 EOTn[TCn] is a TC output	ETD must be set to 1 if the channel is configured for software-initiated memory-to-memory transfers.
24	TCE	Terminal Count (TC) Enable 0 Channel does not stop when TC is reached 1 Channel stops when TC is reached	If TCE=1, it is required that ETD=1.
25:26	CP	Channel Priority 00 Low priority 01 Medium low priority 10 Medium high priority 11 High priority	Actively requesting channels of the same priority are ranked in order by channel number, channel 0 having the highest priority. For more information, see "DMA Arbitration Transfer Priorities."
27:28	PF	Memory Read Prefetch Transfer 00 Prefetch 1 quadword (128-bits) 01 Prefetch 2 quadwords 10 Prefetch 4 quadwords 11 Prefetch 8 quadwords	Used only during memory-to-peripheral and memory to device-paced memory transfers. To enable prefetching it is required that BEN=1.
29	DEC	Address Decrement 0 SAI and DAI fields control memory address incrementing. 1 After each data transfer the memory address is decremented by the transfer width.	If DEC=1, it is required that BEN=0. This field is valid only for peripheral mode transfers (TM=00).
30	SL	Source Address Location/Memory Location 0 PLB 1 OPB	For memory-to-memory transfers (TM=1x) SL indicates whether the source memory controller is on the PLB or OPB. For peripheral mode transfers, this field denotes the location of the memory controller.
31		Reserved	

DCR 0x101, 0x109, 0x111, 0x119 R/W

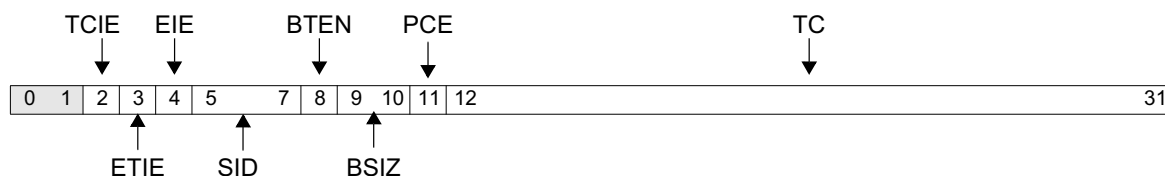
See *DMA Count and Control Registers (DMA0_CT0-DMA0_CT3)* on page 703

Figure 32-97. DMA Count and Control Registers (DMA0_CT0-DMA0_CT3)

0:1		Reserved	
2	TCIE	Terminal Count Interrupt Enable 0 Disable terminal count interrupts 1 Enable terminal count interrupts	Terminal Count interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
3	ETIE	EOT Interrupt Enable 0 Disable end of transfer interrupts 1 Enable end of transfer interrupts	EOT interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
4	EIE	Error Interrupt Enable 0 Disable error interrupts 1 Enable error interrupts	Error interrupts are further qualified by DMA0_CR0-DMA0_CR3[CIE] and UIC0_ER[DMAn].
5:7	SID	Subchannel ID	
8	BTEN	Burst Enable 0 Disable bursting 1 Enable bursting	Controls bursting to and from peripheral devices and EBC-attached device-paced memory. See "Peripheral and Device-Paced Memory Bursts."
9:10	BSIZ	Burst Size 00 Burst of 2 01 Burst of 4 10 Burst of 8 11 Burst of 16	Determines the burst length used when BEN=1. When BEN=1, TC must be programmed to a multiple of BRST-SIZ. For OPB memory burst transfers, these bits determine the maximum burst size.
11	PCE	Parity Check Enable 0 Disable parity checking 1 Enable parity checking	Enables parity for peripheral mode transfers. See "Data Parity During DMA Peripheral Transfers."
12:31	TC	Transfer Count 0 - 1024K-1	Programming TC=0 results in the maximum count of 1024K transfers.

DCR 0x104, 0x10C, 0x114, 0x11C R/W

See *DMA Destination Address Registers* on page 705

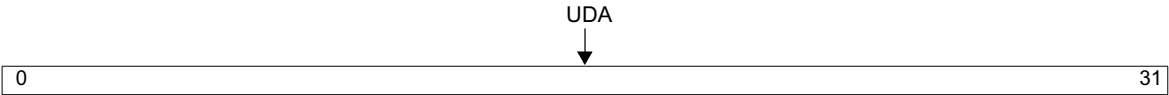


Figure 32-98. DMA Destination Address High Registers (DMA0_DAH0-DMA0_DAH3)

0:31	UDA	Upper 32-bits of destination address for memory-to-memory and peripheral-to-memory transfers.
------	-----	---

DCR 0x105, 0x10D, 0x115, 0x11D R/W

See DMA Destination Address Registers on page 705

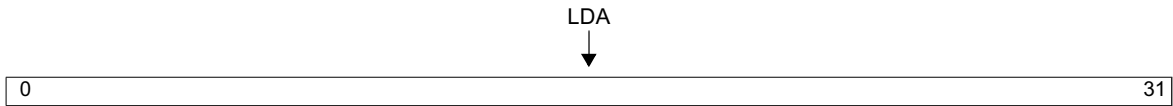
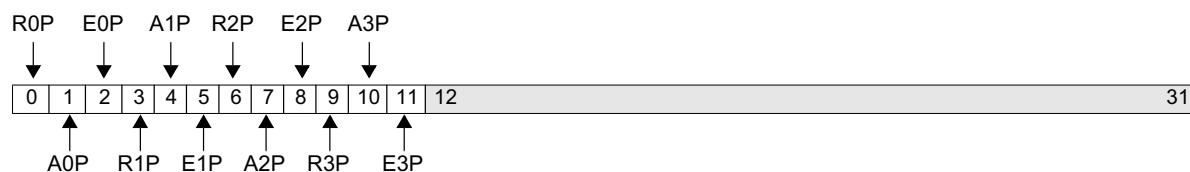


Figure 32-99. DMA Destination Address Low Registers (DMA0_DAL0-DMA0_DAL3)

0:31	LDA	Lower 32-bits of destination address for memory-to-memory and peripheral-to-memory transfers.
------	-----	---

DCR 0x126 R/WSee *DMA Polarity Configuration Register (DMA0_POL)* on page 711*Figure 32-100. DMA Polarity Configuration Register (DMA0_POL)*

0	R0P	DMAReq0 Polarity 0 Active high 1 Active low
1	A0P	DMAAck0 Polarity 0 Active high 1 Active low
2	E0P	EOT0[TC0] Polarity 0 Active high 1 Active low
3	R1P	DMAReq1 Polarity 0 Active high 1 Active low
4	A1P	DMAAck1 Polarity 0 Active high 1 Active low
5	E1P	EOT1[TC1] Polarity 0 Active high 1 Active low
6	R2P	DMAReq2 Polarity 0 Active high 1 Active low
7	A2P	DMAAck2 Polarity 0 Active high 1 Active low
8	E2P	EOT2[TC2] Polarity 0 Active high 1 Active low
9	R3P	DMAReq3 Polarity 0 Active high 1 Active low
10	A3P	DMAAck3 Polarity 0 Active high 1 Active low
11	E3P	EOT3[TC3] Polarity 0 Active high 1 Active low
12:31		Reserved

DCR 0x102, 0x10A, 0x112, 0x11A R/W

See DMA Source Address Registers) on page 704



Figure 32-101. DMA Source Address High Registers (DMA0-SAH0-DMA0-SAH3)

0:31	USA	Upper 32-bits of source address for memory-to-memory and memory-to-peripheral transfers
------	-----	---

DCR 0x103, 0x10B, 0x113, 0x11B R/W

See *DMA Source Address Registers*) on page 704

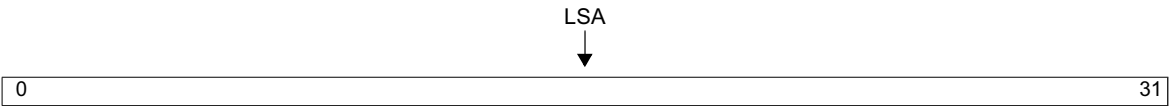


Figure 32-102. DMA Source Address Low Registers (DMA0-SAL0-DMA0-SAL3)

0:31	LSA	Lower 32-bits of source address for memory-to-memory and memory-to-peripheral transfers.
------	-----	--

DCR 0x123 R/W

See *DMA Scatter/Gather Command Register (DMA0_SGC)* on page 709

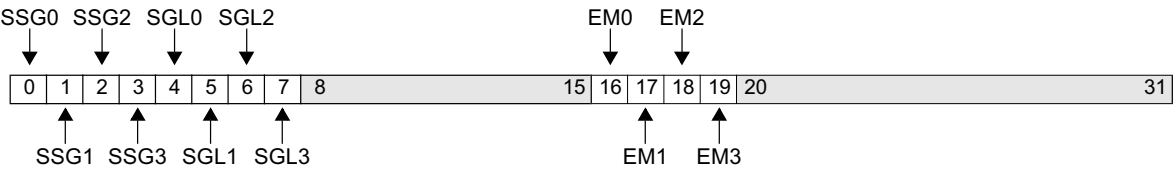


Figure 32-103. DMA Scatter/Gather Command Register (DMA0_SGC)

0:3	SSG[0:3]	Start Scatter/Gather for channels 0-3 0 Scatter/gather support is disabled 1 Scatter/gather support is enabled	To start a scatter/gather operation for channel n, EM[n] must also be set.
4:7	SGL[0:3]	Scatter/Gather Descriptor Table Location for channels 0-3 0 PLB memory space 1 OPB memory space	A channel's descriptor table may not cross between PLB and OPB address space. See <i>Scatter/Gather Transfers</i> on page 715.
8:15		Reserved	
16:19	EM[0:3]	Enable Mask for channels 0-3 0 Writes to SSG[n] and SGL[n] are ignored 1 Allow writing to SSG[n] and SGL[n]	To write SSG[n] or SGL[n], EM[n] must be set. Otherwise, writing SSG[n] or SGL[n] has no effect.
20:31		Reserved	

DCR 0x106, 0x10E, 0x116, 0x11E R/W

See DMA Scatter/Gather Descriptor Address Registers on page 707

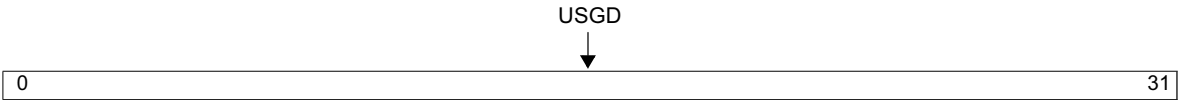


Figure 32-104. DMA Scatter/Gather Descriptor Address High Registers (DMA0_SGH0-DMA0_SGH3)

0:31	USGD	Upper 32-bits of address of next scatter/gather descriptor table.
------	------	---

DCR 0x107, 0x10F, 0x117, 0x11F R/W

See DMA Scatter/Gather Descriptor Address Registers on page 707

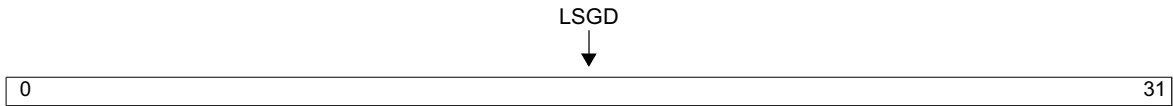


Figure 32-105. DMA Scatter/Gather Descriptor Address Low Registers (DMA0_SGL0-DMA0_SGL3)

0:31	LSGD	Lower 32-bits of address of next scatter/gather descriptor table.
------	------	---

Preliminary User’s Manual

DCR 0x125 R/W

See DMA Sleep Mode Register (DMA0_SLP) on page 710

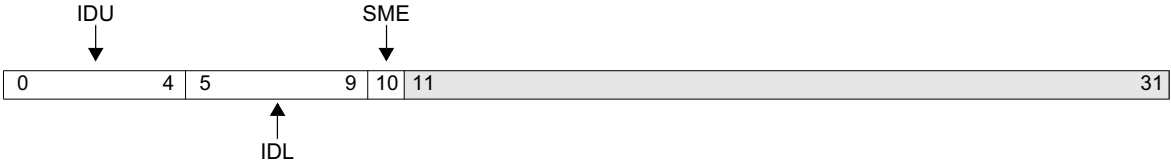
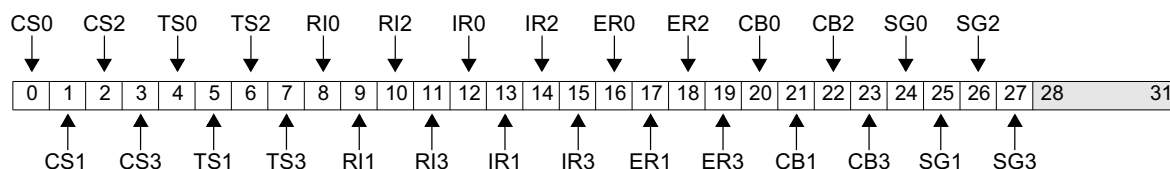


Figure 32-106. DMA Sleep Mode Register (DMA0_SLP)

0:4	IDU	Idle Timer Upper 0-31	Upper 5 bits of the idle timer.
5:9	IDL	Idle Timer Lower Hardcoded to 0b11111	Lower 5-bit portion of the idle timer. Writing this field has no effect.
10	SME	Sleep Mode Enable 0 Sleep disabled 1 Sleep enabled	If SME=1, also set CPM0_ER[DMA] to enable the Clock and Power Management macro to put the DMA controller to sleep.
11:31		Reserved	

DMA0_SR

DMA Status Register

Preliminary User's Manual**DCR 0x120 Read/Clear**See *DMA Status Register (DMA0_SR)* on page 708*Figure 32-107. DMA Status Register (DMA0_SR)*

0:3	CS[0:3]	Channel 0-3 Terminal Count Status 0 Terminal count has not occurred 1 Terminal count has been reached	Set when the transfer count reaches 0 and the DMA0_CRn(TCE) bit is set.
4:7	TS[0:3]	Channel 0-3 End of Transfer Status 0 End of transfer has not been requested 1 End of transfer has been requested	Only valid for channels with DMA0_CRn[ETD]=0 (no memory-to-memory transfers).
8:11	RI[0:3]	Channel 0-3 Error Status 0 No error 1 Error occurred	See Errors section for more information.
12:15	IR[0:3]	Internal DMA Request 0 No internal DMA request pending 1 Internal DMA request is pending	
16:19	ER[0:3]	External DMA Request 0 No external DMA request pending 1 External DMA request is pending	
20:23	CB[0:3]	Channel Busy 0 Channel is idle 1 Channel currently active	During scatter/gather fetches, these bits are active.
24:27	SG[0:3]	Scatter/Gather Status 0 No scatter/gather operation in progress 1 Scatter/gather operation in progress	For multiple scatter/gather links, the scatter/gather status bit is set when the first link is loaded and is kept active until the last link is completed.
28:31		Reserved	

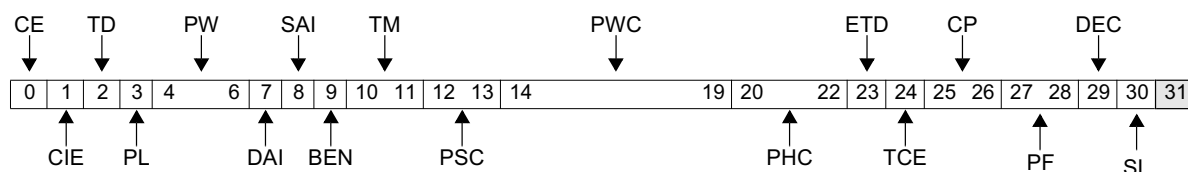
DCR 0x100, 0x108, 0x110, 0x118 R/WSee *DMA Channel Control Registers (DMA0_CR0–DMA0_CR3)* on page 701

Figure 32-108. DMA Channel Control Registers (DMA0_CR0–DMA0_CR3)

0	CE	Channel Enable 0 Channel is disabled 1 Channel is enabled	This field is automatically cleared when the transfer completes or an error occurs.
1	CIE	Channel Interrupt Enable 0 Disable interrupts from this channel 1 Enable interrupts from this channel	When enabled, interrupts can be generated for terminal count, end of transfer, and errors conditions. Each of these interrupt types must be individually enabled in DMA0_CTn. See "Interrupts."
2	TD	In peripheral mode: 0 Transfers are from memory-to-peripheral 1 Transfers are from peripheral-to-memory In device-paced memory-to-memory mode: 0 Peripheral is at the destination address 1 Peripheral is at the source address	TD is not used (don't care) for software-initiated memory-to-memory transfers.
3	PL	Peripheral Location/Destination Memory Location For memory-to-memory transfers: 0 Destination address located in PLB memory space 1 Destination address located in OPB memory space For peripheral/DPM-to-memory or memory-to-DPM/peripheral transfers: 0 Device located on external bus 1 Device located on the OPB.	
4:6	PW	Peripheral Width/Transfer Width 000 Byte (8 bits) 001 Halfword (16 bits) 010 Word (32 bits) 011 Doubleword (64 bits) 100 Quadword (128 bits)	Transfers involving peripheral, device-paced-memory, and OPB FIFO devices can only be byte, halfword, or word. This limitation also applies to transfers involving EBC memory when SAI=0 or DAI=0. PLB FIFO devices can only be quadword size.
7	DAI	Destination Address Increment 0 Do not increment destination address 1 After each data transfer increment the destination address by: <ul style="list-style-type: none"> 1, for byte (8-bit) transfers 2, for halfword (16-bit) transfers 4, for word (32-bit) transfers 8, for doubleword (64-bit) transfers 16, for quadword (128-bit) transfers 	Valid only when DEC=0. Is required that DAI=1 for peripheral-to-memory and device-paced memory-to-memory transfers since peripheral-to-FIFO type devices or DPM-to-FIFO type devices are not supported.

8	SAI	Source Address Increment 0 Do not increment source address 1 After each data transfer increment the source address by: <ul style="list-style-type: none"> 1, for byte (8-bit) transfers 2, for halfword (16-bit) transfers 4, for word (32-bit) transfers 8, for doubleword (64-bit) transfers 16, for quadword (128-bit) transfers 	Valid only when DEC=0. Is required that SAI=1 for memory-to-peripheral and memory-to-device-paced memory transfers since FIFO-to-peripheral/DPM transfers are not supported.
9	BEN	Buffer Enable 0 Disable DMA 128-byte buffer 1 Enable DMA 128-byte buffer	If BEN=0 discrete read and write operations occur for each data transfer.
10:11	TM	Transfer mode 00 Peripheral 01 Reserved 10 Software-initiated memory-to-memory 11 Device-paced memory-to-memory	
12:13	PSC	Peripheral Setup Cycles 0-3	Number of PerClk cycles that the peripheral bus is idle from the last peripheral bus transaction to DMAAckn becoming active. Used only for the peripheral side of peripheral mode transfers.
14:19	PWC	Peripheral Wait Cycles 0-63	DMAAckn remains active for PWC+1 PerClk cycles. Used only for the peripheral side of peripheral mode transfers.
20:22	PHC	Peripheral Hold Cycles 0-7	The number of PerClk cycles between the time that DMAAckn becomes inactive until the peripheral bus is available for the next bus access. Used only during the peripheral side of peripheral mode transfers.
23	ETD	End-of-Transfer/Terminal Count (EOTn[TCn]) Pin Direction 0 EOTn[TCn] is an EOT input 1 EOTn[TCn] is a TC output	ETD must be set to 1 if the channel is configured for software-initiated memory-to-memory transfers.
24	TCE	Terminal Count (TC) Enable 0 Channel does not stop when TC is reached 1 Channel stops when TC is reached	If TCE=1, it is required that ETD=1.
25:26	CP	Channel Priority 00 Low priority 01 Medium low priority 10 Medium high priority 11 High priority	Actively requesting channels of the same priority are ranked in order by channel number, channel 0 having the highest priority. For more information, see "DMA Arbitration Transfer Priorities."
27:28	PF	Memory Read Prefetch Transfer 00 Prefetch 1 quadword (128-bits) 01 Prefetch 2 quadwords 10 Prefetch 4 quadwords 11 Prefetch 8 quadwords	Used only during memory-to-peripheral and memory to device-paced memory transfers. To enable prefetching it is required that BEN=1.
29	DEC	Address Decrement 0 SAI and DAI fields control memory address incrementing. 1 After each data transfer the memory address is decremented by the transfer width.	If DEC=1, it is required that BEN=0. This field is valid only for peripheral mode transfers (TM=00).
30	SL	Source Address Location/Memory Location 0 PLB 1 OPB	For memory-to-memory transfers (TM=1x) SL indicates whether the source memory controller is on the PLB or OPB. For peripheral mode transfers, this field denotes the location of the memory controller.
31		Reserved	

DCR 0x101, 0x109, 0x111, 0x119 R/W

See *DMA Count and Control Registers (DMA0_CT0–DMA0_CT3)* on page 703

Figure 32-109. DMA Count and Control Registers (DMA0_CT0–DMA0_CT3)

0:1		Reserved	
2	TCIE	Terminal Count Interrupt Enable 0 Disable terminal count interrupts 1 Enable terminal count interrupts	Terminal Count interrupts are further qualified by DMA0_CR0–DMA0_CR3[CIE] and UIC0_ER[DMAn].
3	ETIE	EOT Interrupt Enable 0 Disable end of transfer interrupts 1 Enable end of transfer interrupts	EOT interrupts are further qualified by DMA0_CR0–DMA0_CR3[CIE] and UIC0_ER[DMAn].
4	EIE	Error Interrupt Enable 0 Disable error interrupts 1 Enable error interrupts	Error interrupts are further qualified by DMA0_CR0–DMA0_CR3[CIE] and UIC0_ER[DMAn].
5:7	SID	Subchannel ID	
8	BTEN	Burst Enable 0 Disable bursting 1 Enable bursting	Controls bursting to and from peripheral devices and EBC-attached device-paced memory. See “Peripheral and Device-Paced Memory Bursts.”
9:10	BSIZ	Burst Size 00 Burst of 2 01 Burst of 4 10 Burst of 8 11 Burst of 16	Determines the burst length used when BEN=1. When BEN=1, TC must be programmed to a multiple of BRST-SIZ. For OPB memory burst transfers, these bits determine the maximum burst size.
11	PCE	Parity Check Enable 0 Disable parity checking 1 Enable parity checking	Enables parity for peripheral mode transfers. See “Data Parity During DMA Peripheral Transfers.”
12:31	TC	Transfer Count 0 - 1024K-1	Programming TC=0 results in the maximum count of 1024K transfers.

DCR 0x104, 0x10C, 0x114, 0x11C R/W

See DMA Destination Address Registers on page 705

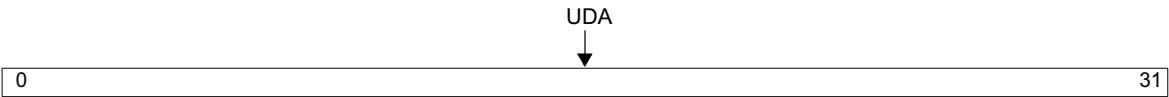


Figure 32-110. DMA Destination Address High Registers (DMA0_DAH0-DMA0_DAH3)

0:31	UDA	Upper 32-bits of destination address for memory-to-memory and peripheral-to-memory transfers.
------	-----	---

DCR 0x105, 0x10D, 0x115, 0x11D R/W

See *DMA Destination Address Registers* on page 705

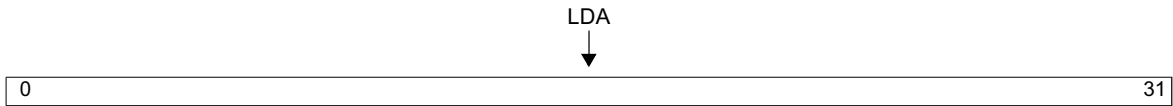


Figure 32-111. DMA Destination Address Low Registers (DMA0_DAL0-DMA0_DAL3)

0:31	LDA	Lower 32-bits of destination address for memory-to-memory and peripheral-to-memory transfers.
------	-----	---

DCR 0x126 R/W

See *DMA Polarity Configuration Register (DMA0_POL)* on page 711

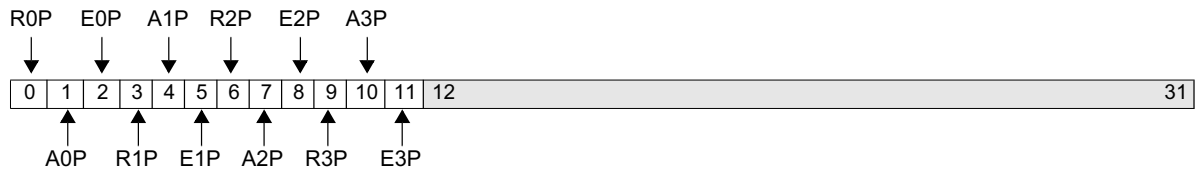


Figure 32-112. DMA Polarity Configuration Register (DMA0_POL)

0	R0P	DMAReq0 Polarity 0 Active high 1 Active low
1	A0P	DMAAck0 Polarity 0 Active high 1 Active low
2	E0P	EOT0[TC0] Polarity 0 Active high 1 Active low
3	R1P	DMAReq1 Polarity 0 Active high 1 Active low
4	A1P	DMAAck1 Polarity 0 Active high 1 Active low
5	E1P	EOT1[TC1] Polarity 0 Active high 1 Active low
6	R2P	DMAReq2 Polarity 0 Active high 1 Active low
7	A2P	DMAAck2 Polarity 0 Active high 1 Active low
8	E2P	EOT2[TC2] Polarity 0 Active high 1 Active low
9	R3P	DMAReq3 Polarity 0 Active high 1 Active low
10	A3P	DMAAck3 Polarity 0 Active high 1 Active low
11	E3P	EOT3[TC3] Polarity 0 Active high 1 Active low
12:31		Reserved

DCR 0x102, 0x10A, 0x112, 0x11A R/W

See *DMA Source Address Registers* on page 704



Figure 32-113. DMA Source Address High Registers (DMA0-SAH0-DMA0-SAH3)

0:31	USA	Upper 32-bits of source address for memory-to-memory and memory-to-peripheral transfers
------	-----	---

DCR 0x103, 0x10B, 0x113, 0x11B R/W

See DMA Source Address Registers) on page 704

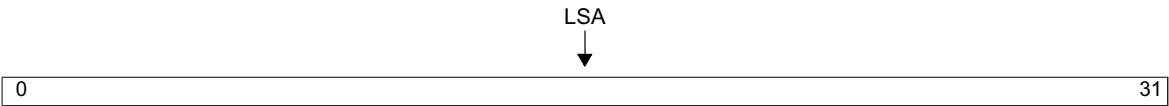


Figure 32-114. DMA Source Address Low Registers (DMA0-SAL0-DMA0-SAL3)

0:31	LSA	Lower 32-bits of source address for memory-to-memory and memory-to-peripheral transfers.
------	-----	--

DCR 0x123 R/W

See *DMA Scatter/Gather Command Register (DMA0_SGC)* on page 709

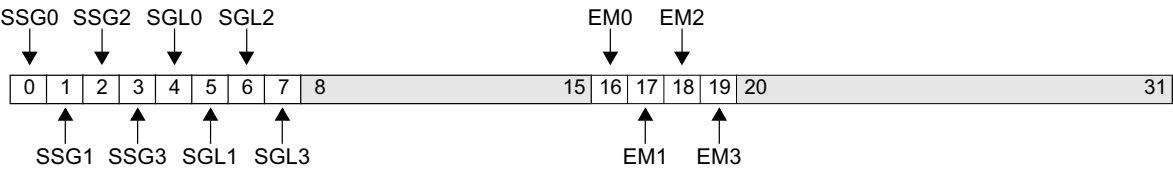


Figure 32-115. DMA Scatter/Gather Command Register (DMA0_SGC)

0:3	SSG[0:3]	Start Scatter/Gather for channels 0-3 0 Scatter/gather support is disabled 1 Scatter/gather support is enabled	To start a scatter/gather operation for channel n, EM[n] must also be set.
4:7	SGL[0:3]	Scatter/Gather Descriptor Table Location for channels 0-3 0 PLB memory space 1 OPB memory space	A channel's descriptor table may not cross between PLB and OPB address space. See <i>Scatter/Gather Transfers</i> on page 715.
8:15		Reserved	
16:19	EM[0:3]	Enable Mask for channels 0-3 0 Writes to SSG[n] and SGL[n] are ignored 1 Allow writing to SSG[n] and SGL[n]	To write SSG[n] or SGL[n], EM[n] must be set. Otherwise, writing SSG[n] or SGL[n] has no effect.
20:31		Reserved	

DCR 0x106, 0x10E, 0x116, 0x11E R/W

See DMA Scatter/Gather Descriptor Address Registers on page 707

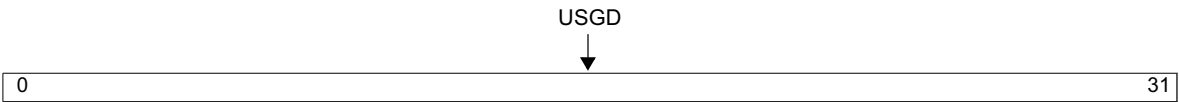


Figure 32-116. DMA Scatter/Gather Descriptor Address High Registers (DMA0_SGH0-DMA0_SGH3)

0:31	USGD	Upper 32-bits of address of next scatter/gather descriptor table.
------	------	---

DCR 0x107, 0x10F, 0x117, 0x11F R/W

See DMA Scatter/Gather Descriptor Address Registers on page 707

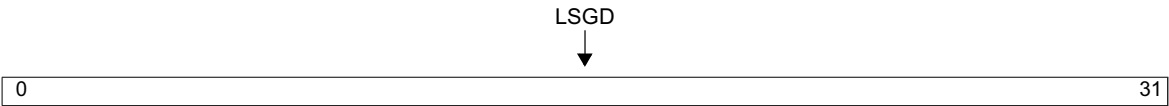


Figure 32-117. DMA Scatter/Gather Descriptor Address Low Registers (DMA0_SGL0-DMA0_SGL3)

0:31	LSGD	Lower 32-bits of address of next scatter/gather descriptor table.
------	------	---

DCR 0x125 R/W

See DMA Sleep Mode Register (DMA0_SLP) on page 710

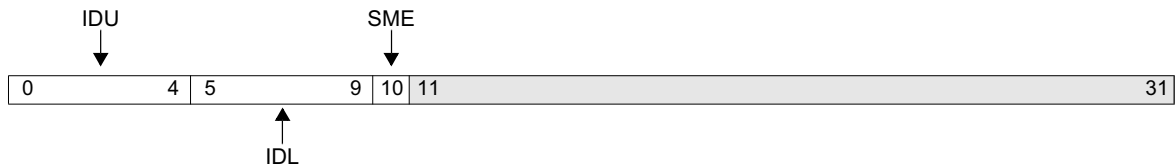
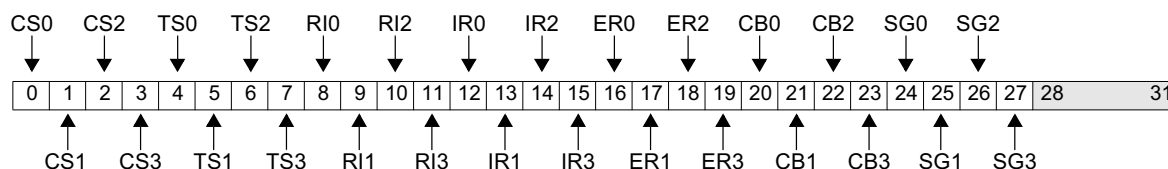


Figure 32-118. DMA Sleep Mode Register (DMA0_SLP)

0:4	IDU	Idle Timer Upper 0-31	Upper 5 bits of the idle timer.
5:9	IDL	Idle Timer Lower Hardcoded to 0b11111	Lower 5-bit portion of the idle timer. Writing this field has no effect.
10	SME	Sleep Mode Enable 0 Sleep disabled 1 Sleep enabled	If SME=1, also set CPM0_ER[DMA] to enable the Clock and Power Management macro to put the DMA controller to sleep.
11:31		Reserved	

DCR 0x120 Read/ClearSee *DMA Status Register (DMA0_SR)* on page 708*Figure 32-119. DMA Status Register (DMA0_SR)*

0:3	CS[0:3]	Channel 0-3 Terminal Count Status 0 Terminal count has not occurred 1 Terminal count has been reached	Set when the transfer count reaches 0 and the DMA0_CRn(TCE) bit is set.
4:7	TS[0:3]	Channel 0-3 End of Transfer Status 0 End of transfer has not been requested 1 End of transfer has been requested	Only valid for channels with DMA0_CRn[ETD]=0 (no memory-to-memory transfers).
8:11	RI[0:3]	Channel 0-3 Error Status 0 No error 1 Error occurred	See Errors section for more information.
12:15	IR[0:3]	Internal DMA Request 0 No internal DMA request pending 1 Internal DMA request is pending	
16:19	ER[0:3]	External DMA Request 0 No external DMA request pending 1 External DMA request is pending	
20:23	CB[0:3]	Channel Busy 0 Channel is idle 1 Channel currently active	During scatter/gather fetches, these bits are active.
24:27	SG[0:3]	Scatter/Gather Status 0 No scatter/gather operation in progress 1 Scatter/gather operation in progress	For multiple scatter/gather links, the scatter/gather status bit is set when the first link is loaded and is kept active until the last link is completed.
28:31		Reserved	

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x10–0x17 R/W

See *Peripheral Bank 0–7 Access Parameters (EBC0_B0AP-EBC0_B7AP)* on page 1009.

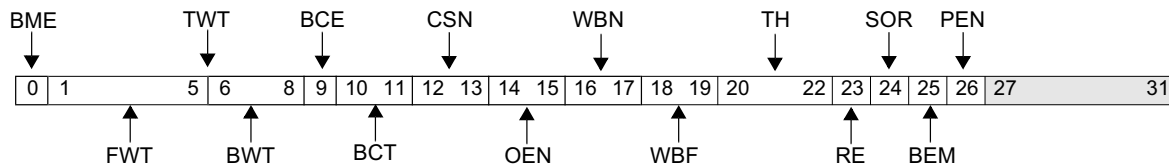


Figure 32-120. Peripheral Bank Access Parameters (EBC0_B0AP-EBC0_B7AP)

0	BME	Burst Mode Enable 0 Burst mode disabled 1 Burst mode enabled	
1:8	TWT	Transfer Wait 0 to 255 PerClk cycles	If bursting is disabled, BME=0, wait states on all non-burst transfers.
1:5	FWT	First Wait 0 to 31 PerClk cycles	If bursting is enabled, BME=1, number of wait states on the first transfer of a burst.
6:8	BWT	Burst Wait 0 to 7 PerClk cycles	If bursting is enabled, BME=1, number of wait states on non-first transfers of a burst.
9	BCE	Fixed Length Burst Reads Disabled Enabled	If BCE=1, all burst read operations consist of exactly BCT transfers. When BCE=0, burst read operations may read more than the requested amount of data.
10:11	BCT	Fixed Length Burst Count 2 transfers 4 transfers 8 transfers 16 transfers	The amount of data transferred depends upon the programmed bank width, EBC0_BnCR[BW].
12:13	CSN	Chip Select On Timing 0 to 3 PerClk cycles.	Number of cycles from peripheral address driven to $\overline{\text{PerCSn}}$ low.
14:15	OEN	Output Enable On Timing 0 to 3 PerClk cycles	Number of cycles from $\overline{\text{PerCSn}}$ low to $\overline{\text{PerOE}}$ low.
16:17	WBN	Write Byte Enable On Timing 0 to 3 PerClk cycles	If BEM=0, number of cycles from $\overline{\text{PerCSn}}$ low to $\overline{\text{PerWBE0:3}}$ active.
18:19	WBF	Write Byte Enable Off Timing (If bursting and ready pin input are disabled) 0 to 3 PerClk cycles	If BEM=0 and RE=0, number of cycles $\overline{\text{PerWBE0:3}}$ becomes inactive prior to $\overline{\text{PerCSn}}$ inactive.
20:22	TH	Transfer Hold 0 to 7 PerClk cycles	Contains the number of hold cycles inserted at the end of a transfer. Hold cycles insert idle bus cycles between transfers to enable slow peripherals to remove data from the data bus before the next transfer begins.
23	RE	Ready Enable PerReady input is disabled PerReady input is enabled	Set for Device Paced Transfers
24	SOR	Sampling of Ready for Device Paced Ready is asserted by the device one clock before data is ready on the external bus Ready is asserted by the device on the same clock that data is ready on the external bus	

Preliminary User's Manual

25	BEM	Byte Enable Mode PerWBE0:3 pins are only active on write cycles PerWBE0:3 pins are active for both read and write cycles
26	PEN	Parity Enable Disable Parity checking Enable Parity checking
27:31		Reserved

The EBCO implements odd parity.

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x00–0x07 R/W

See *Peripheral Bank Configuration Registers (EBC0_B0CR-EBC0_B7CR)* on page 1007.

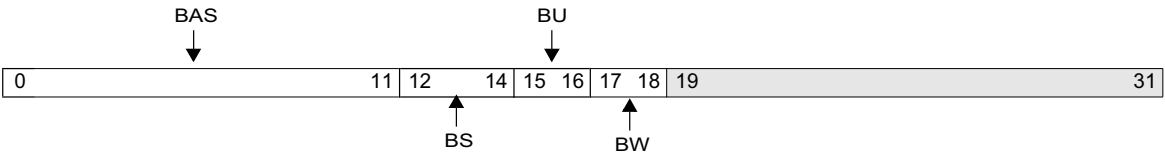


Figure 32-121. Peripheral Bank Configuration Registers (EBC0_B0CR-EBC0_B7CR)

0:11	BAS	Base Address Select	Specifies the bank starting address, which must be a multiple of the bank size.
12:14	BS	Bank Size 000 1 MB bank 001 2 MB bank 010 4 MB bank 011 8 MB bank 100 16 MB bank 101 32 MB bank 110 64 MB bank 111 128 MB bank	
15:16	BU	Bank Usage 00 Disabled 01 Read-only 10 Write-only 11 Read/Write	Specifies the type of accesses allowed for the bank. A protect error occurs if a write is attempted to a read-only bank or a read is attempted from a write-only bank.
17:18	BW	Bus Width 00 8-bit bus 01 16-bit bus 10 Reserved 11 32-bit bus	
19:31		Reserved	

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x20 R

See *Peripheral Bus Error Address Register (EBC0_BEAR)* on page 1013.

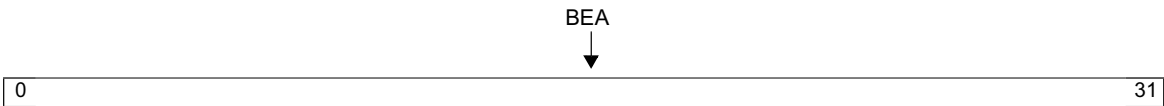


Figure 32-122. Peripheral Bus Error Address Register (EBC0_BEAR)

0:31	BEA	Bus Error Address
------	-----	-------------------

EBC0_BESR

Peripheral Bus Error Status Register

Preliminary User's Manual

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x21 R/W

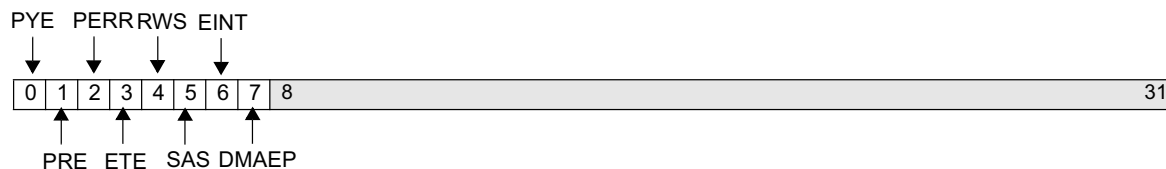
See *Peripheral Bus Error Status Register (EBC0_BESR)* on page 1014.

Figure 32-123. Bus Error Status Register (EBC0_BESR)

0	PYE	Parity Error Detected on Read 0 No Parity Error Detected 1 Parity Error Detected	O_ebco_opb_errAck is asserted.
1	PRE	Bank Protection Error 0 No Protection Error 1 Protection Error	O_ebco_opb_errAck is asserted for read or write.
2	PERR	PerErr 0 No PerErr Detected during EBCO transaction. 1 PerErr Detected during EBCO transaction.	If PerErr is detected at any other time during a read, errAck is not asserted. ErrAck is not asserted for write operations.
3	ETE	External Bus Timeout Error 0 No External Bus Timeout Error Detected 1 External Bus Timeout Error Detected	See PDT bit in <i>Figure 30.5.6</i> on page 1015 for timeout enable information.
4	RWS	Read/write Error Status 0 Errant operation was a write operation 1 Errant operation was a read operation	
5	SAS	Sequential Address status 0 Sequential Address Not Active during error detection 1 Sequential Address Active during error detection	
6	EINT	Error Interrupt This bit is a logical "OR" of the four different errors that may be reported and indicates an interrupt has been asserted to the Interrupt Controller	
7	DMAEP	DMA External Peripheral Error 0 No error 1 Error	DMA external peripheral error detected during EBCO transaction.
8:31		Reserved	

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x23 R/W

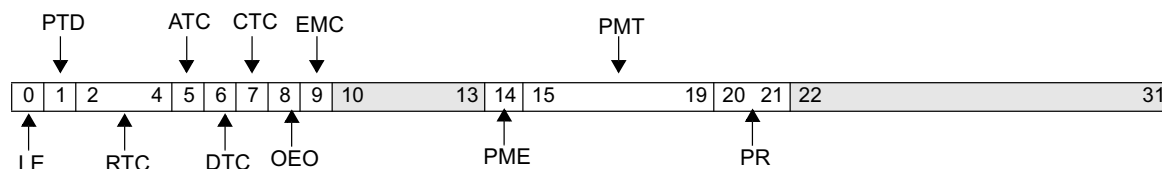
See *EBC Configuration Register (EBC0_CFG)* on page 1015.

Figure 32-124. EBC Configuration Register (EBC0_CFG)

0	LE	Lock Error 0 Do not lock error 1 Lock error	This bit is used for error locking in the EBC. If this bit is set, then the EBC will latch the first error detected and save it with associated parameters and address. If this bit is not set, then the EBC will continue to latch new errors and associated parameters and addresses that will overwrite previous errors.
1	PTD	Device-Paced Time-out Disable 0 Enabled time-outs 1 Disable time-outs	This bit is valid only when EBC0_BnAP[RE]=1. If PTD=1, the EBC waits indefinitely for assertion of PerReady during device-paced accesses.
2:4	RTC	Ready Timeout Count 000 16 PerClk cycles 001 32 PerClk cycles 010 64 PerClk cycles 011 128 PerClk cycles 100 256 PerClk cycles 101 512 PerClk cycles 110 1024 PerClk cycles 111 2048 PerClk cycles	When PTD=0, the number of clock cycles from the assertion of PerAddr0:31 until a time-out error occurs.
5	ATC	Address Bus High Impedance Control 0 External address bus is high impedance when EBC is idle 1 External address bus drive previous value when EBC is idle and has ownership of the peripheral interface	Default after reset is ATC=1. For details on how the ATC affects driver enables, See <i>Driver Enables</i> on page 985.
6	DTC	Data Bus High Impedance Control 0 External data bus is high impedance when EBC is idle 1 External data bus drive previous write data value when EBC is idle and has ownership of the peripheral interface	Default after reset is DTC=1. For details on how the DTC affects driver enables, See <i>Driver Enables</i> on page 985.
7	CTC	Control Signal High Impedance Control 0 External bus Control Signals are high impedance when EBC is idle 1 External bus Control Signals are driven inactive and held when EBC is idle and has ownership of the peripheral interface	Default after reset is CTC=1. For details on how the CTC affects driver enables, See <i>Driver Enables</i> on page 985.
8	OEO	External Bus Override High Impedance Control 0 External Bus Output Enable Override Disabled 1 External Bus Output Enable Override Enabled	Default after reset is OEO=1. For details on how the OEO affects driver enables, See <i>Driver Enables</i> on page 985.
9	EMC	External Master High Impedance Control 0 High impedance all EBC outputs when HOLD_ACK = 1. 1 High impedance all EBC outputs when HOLD_ACK = 1 except PerCS0:7 are always driven.	Default after reset is EMC=1. For details, See <i>Driver Enables</i> on page 985.
10:13		Reserved	

14	PME	Power Management Enable 0 Disabled 1 Enabled	
15:19	PMT	Power Management Timer 0-31	The EBC makes a sleep request to the Clock and Power Management unit when PME=1 and the EBC has been idle for 32*PMT PerClk cycles.
20:21	PR	Pending Request Timer 00 -16 01 - 32 10 - 64 11 - 128	Number of PerClk cycles to wait for a retried OPB operation ¹ to return before relinquishing external bus ownership back to the external master.
22:31		Reserved	

DCR 0x012 R/W

See *EBC Address Register (EBC0_CFGADDR)* on page 1007.

This register is used to determine offsets for the indirectly-accessed EBC DCRs.



Figure 32-125. EBC Address Register (EBC0_CFGADDR)

0:26		Reserved
27:31	DCRA	DCR Address Offset

DCR 0x013 R/W

See *EBC Data Register (EBC0_DATA)* on page 1007.

This register is used to access the indirectly-accessed EBC DCRs.

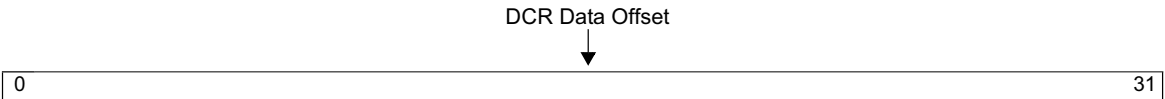


Figure 32-126. EBC Data Register (EBC0_CFGDATA)

0:31	DCRD	DCR Data Port
------	------	---------------

Preliminary User’s Manual

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x24 R/W

See EBC Core ID Register (EBC0_CID) on page 1016.

This register is used to access the indirectly-accessed EBC DCRs.

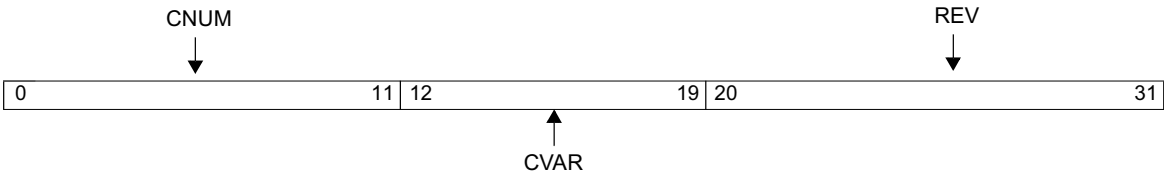


Figure 32-127. EBCO Core ID Register 0 (EBC0_CIDn)

0:11	CNUM	Core Number	'324
12:19	CVAR	Core Variation	'01'
20:31	REV	Revision Number	This value is the IBM SCCS revision number tracked and modified by the core designer.

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x02 R/W
See *EBMI Bus Error Address Register (EBM0_BEAR)* on page 975.

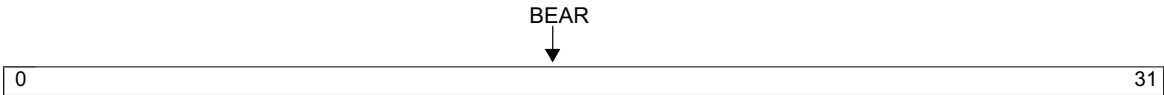


Figure 32-128. Peripheral Bus Error Address Register (EBM0_BEAR)

0:31	BEAR	Bus Error Address	The contents of this register are valid when any bit is set in the EBM0_BESR.
------	------	-------------------	---

Preliminary User’s Manual

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x04 R/W
See *EBMI Bus Error Mask Register (EBM0_BEMR)* on page 977.

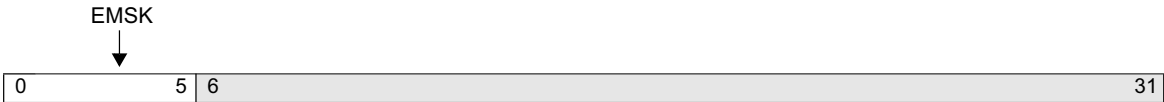


Figure 32-129. EBMI Bus Error Mask Register (EBM0_BEMR)

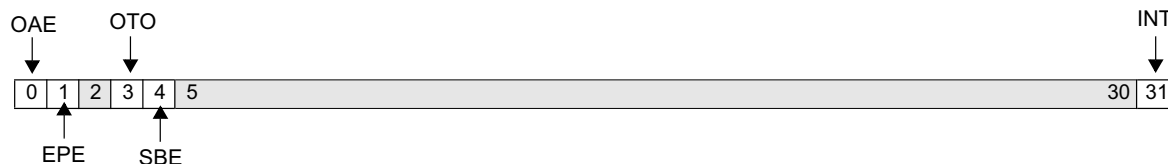
0:5	EMSK	Bus Error Mask 0 Reporting of this error is not masked 1 Reporting of this error is masked.
6:31		Reserved

EBM0_BESR

Bus Error Status Register

Preliminary User's Manual

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x03 R/W

See *EBMI Bus Error Status Register (EBM0_BESR)* on page 976.Figure 32-130. *EBMI Bus Error Status Register (EBM0_BESR)*

0	OAE	OPB Bus Ack Error 0 No ack error detected 1 Ack error detected	This bit is set when the EBMI detects an OPB error indication asserted during a data transfer on the OPB.
1	EPE	EXPB Bus Parity Error 0 No parity error detected 1 Parity error detected	This bit is set when the EBMI detects a parity error on incoming write data from the external master. The transfer with the parity error is not sent onto the OPB. Burst write operations must fill the buffer/terminate before they are sent onto the OPB. Consequently, none of the write data in the write buffer is forwarded to the OPB if a parity error is detected. All burst write data after the parity error is also discarded.
2		Reserved	
3	OTO	OPB Bus Timeout Error 0 No timeout detected 1 Timeout indication received when EBMI is reading or writing data on the OPB.	The current transfer on the OPB is terminated.
4	SBE	EXPB Special Cycle Error 0 No Special Cycle error detected. 1 External master has issued a non-Special Cycle operation when a Special Cycle operation was expected.	This error can only occur when using 16 or 8-bit external masters since the Special Cycle operation takes multiple transfers.
5:30			
31	INT	Interrupt Asserted 0 Interrupt line is not asserted. 1 Interrupt line is asserted to the chip internal interrupt controller.	This bit is a logical OR of all the other bits in the EBM0_BESR that are not masked in the EBM0_BEMR. This bit is not writable by software.

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x014 R/W

See *EBMI Configuration Address Register (EBM0_CFGADDR)* on page 972.

This register is used to determine offsets for the indirectly-accessed EBMI DCRs.



Figure 32-131. EBMI Configuration Address Register (EBM0_CFGADDR)

0:26		
27:31	DCRA	DCR Address Offset

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x015 R/W

See *EBMI Configuration Data Register (EBM0_CFGDATA)* on page 973.

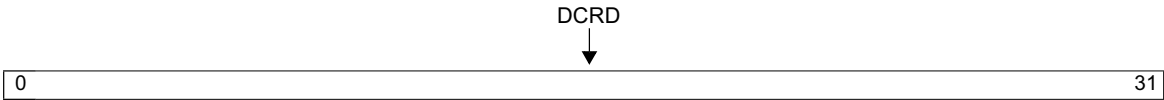


Figure 32-132. *EBMI Configuration Data Register (EBM0_CFGDATA)*

0:31	DCRD	DCR Data Port
------	------	---------------

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x011 Read-only

See *EBMI Core ID Register (EBM0_CID)* on page 980.

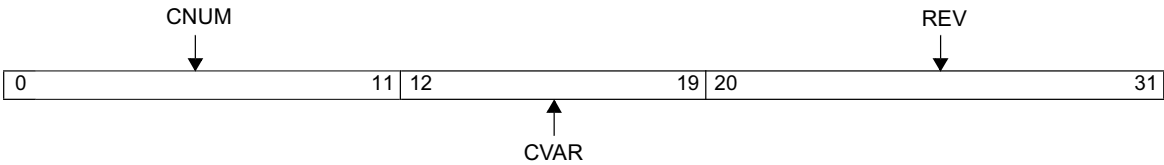


Figure 32-133. *EBMI Core ID Register 0 (EBM0_CID)*

0:11	CNUM	Core Number	'325'
12:19	CVAR	Core Variation	'01'
20:31	REV	Revision Number	This value is the IBM SCCS revision number tracked and modified by the core designer.

EBM0_CTL

Control Register

Preliminary User's Manual

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x00 R/W

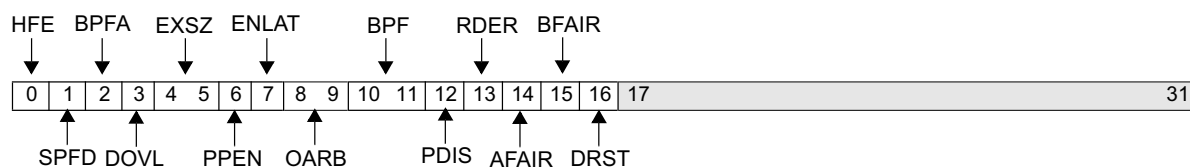
See *EBMI Control Register (EBM0_CTL)* on page 973.

Figure 32-134. EBMI Control Register (EBM0_CTL)

0	HFE	Hold First Error 0 EBM0_BEAR and EBM0_BESR contain information from first error detected. 1 EBM0_BEAR and EBM0_BESR contain information from last error detected.	
1	SPFD	Single Beat Word Prefetch Disable 0 4-byte read on OPB for any single-beat EXPB read request. 1 1 to 4- byte read on OPB for any single-beat EXPB read request, according to EXPB PerWBE0:3 signals.	This bit must be set if the EBMI is to read only the bytes that are requested for a single beat read.
2	BPFA	Burst Prefetch Ahead 0 Data is read from OPB into the data buffer when the data is required. 1 Data is read from OPB into the data buffer before the data is required, while the 1st buffer is being emptied.	This bit should be set by software if the external master routinely does long or many sequential burst read operations longer than the burst prefetch size of the buffer. The EBMI will prefetch 1 buffer ahead to maintain EXPB bus bandwidth. This bit is ignored if the Burst Prefetch mode below is set to '10' to disable the prefetch.
3	DOVL	Disable HoldA Arbitration Overlap 0 HoldAck asserted in response to an active HoldReq asserted by external master immediately. There is a possible 2-cycle delay if a Special Cycle operation is in progress from the previous tenure. 1 HoldAck is not asserted in response to an active HoldReq from the external master until all pending OPB operations and Special Cycle operations from a previous bus tenure are complete.	This bit should be left as 0 and the EBM0_FAIR register used to tune arbitration.
4:5	EXSZ	External Master Data Bus Size 00 8-bit data bus, PerData0:7 01 16-bit data bus, PerData0:15 10 32-bit data bus, PerData0:31 11 No external master	Software must initialize this register to indicate the size of the external master data bus.
6	PPEN	Enable second 32-byte buffer 0 Second buffer disabled. Use only 1 buffer. 1 Second buffer enabled. Ping/Pong enabled.	
7	ENLAT	Enable OPB Latency Counter 0 OPB Latency counter is disabled 1 OPB Latency counter is enabled	This bit enables the EBM0_LCNT register function.
8:9	OARB	OPB Arbitration Control Must be 0b01	
10:11	BPF	Burst Prefetch 00 8-beat (32-byte) read 01 4-beat (16-byte) read 10 1-beat (4/2/1-byte) read (no prefetch, read byte size of master) 11 Reserved, unused	For most applications, this should remain at '00.' If BPF=2'b10, the request will be forwarded exactly as received if SPFD=1. These bits control the number of bytes prefetched on a burst read from the external master.

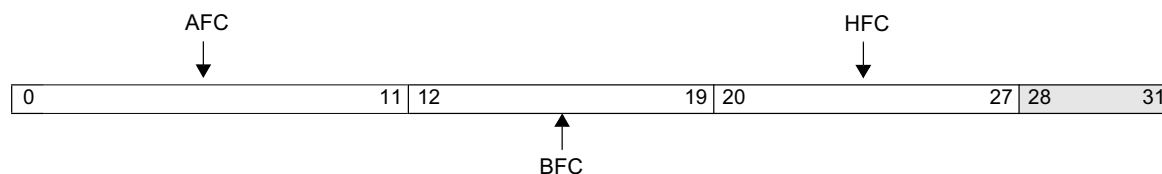
12	PDIS	EXPB Parity Checking Disabled 0 Write data parity is checked on EXPB 1 Write data parity is not checked on EXPB	Field enabled parity checking is only for external master transfers. This bit must be set by software if the external master does not support parity on PerData.
13	RDER	Enable Read Early Indication 0 Burst read data returned to external master when burst read from OPB is complete. 1 Burst read data returned to external master beginning when first beat from OPB is complete.	This mode bit is used to reduce latency for read burst operations.
14	AFAIR	Arbitration Fairness Counter Disable	This bit disables the AFCNT in the EBM0_FAIR register.
15	BFAIR	BReq Fairness Counter Disable	This bit disables the BFCNT in the EBM0_FAIR register.
16	DRST	Disable Reset Valid for New Tenure	This bit disables the clearing of the read buffer valid bits at the start of a new external master bus tenure. This bit should be set if the external master does long burst reads but often has to give up the EXPB before receiving all the data because BusReq is asserted.
17:31		Reserved	

EBM0_FAIR

EBMI Fairness Control Register

Preliminary User's Manual

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x08 R/W

See *EBMI Fairness Control Register (EBM0_FAIR)* on page 981.Figure 32-135. *EBMI Fairness Control Register 0 (EBM0_FAIR)*

0:11	AFC	Arbitration Fairness Count	<p>These bits indicate the number of OPB clocks EBMI will wait before asserting a signal to stop PLB-to-OPB traffic when an external master request is pending. Under most conditions, these bits should be set to a high value to lower the priority of the external master. These bits are used to tune the priority of requests from the PLB to the OPB vs. the priority of requests from the external master.</p> <p>If PLB-to-OPB traffic needs a relatively high priority, then a high value (for example x'F00') is written into this register by software.</p>
12:19	BFC	BusReq Fairness Count	<p>These bits indicate the number of PerClk cycles EBMI will wait before asserting BusReq to the external master. The BusReq fairness counter will begin decrementing when the EBCO has relinquished EXPB bus ownership. If BFC=0, BusReq may be asserted the same cycle as HoldAck.</p> <p>Under some conditions, the EBC may request ownership of the external bus and assert BusReq before the external master has asserted ExtReq. Setting the BFC bits to a non-zero value may allow the external master to make forward progress (if it is sensitive to this condition) by delaying BusReq until after the external master has asserted ExtReq.</p>
20:27	HFC	HoldReq Fairness Count	<p>These bits indicate the number of PerClk cycles EBMI will wait before forwarding HoldReq to the EBCO for the external master to gain bus ownership.</p> <p>This register is set to a high value to prioritize DMA transfers to and from the EBCO since DMA transfers are not affected by AFC.</p>
28:31		Reserved	

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x01 R/W

See *EBMI OPB Latency Count Register (EBM0_LCNT)* on page 975.



Figure 32-136. *EBMI OPB Latency Count Register (EBM0_LCNT)*

0:5	LCNT	Latency Counter This is the upper 6 bits of the initial load of a 10-bit OPB bus latency counter.
6:31		Reserved

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x07 R/W

See *EBMI Sleep Mode Register (EBM0_SLPMD)* on page 979.

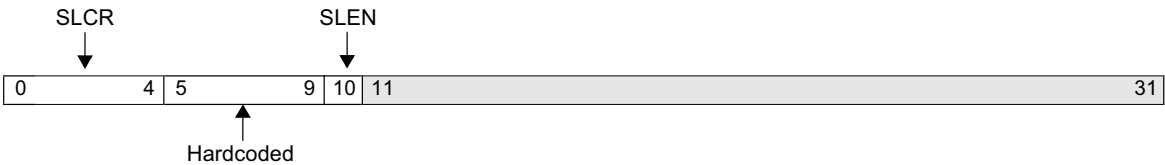


Figure 32-137. *EBMI Sleep Mode Register (EBM0_SLPMD)*

0:4	SLCR	Programmable timer values
5:9		Hardcoded to 1s
10	SLEN	Sleep mode enable bit 0 Sleep request not asserted 1 Sleep request asserted when sleep requirements are met Software must set this bit for the EBMI to assert its sleep request and be put to sleep.
11:31		Reserved

Preliminary User’s Manual

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x06 R/W
See EBM0 OPB Upper Address Mask Register (EBM0_UAM) on page 978.

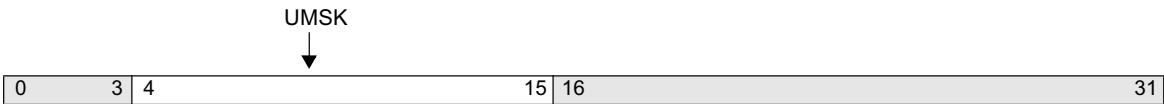


Figure 32-138. EBM0 Upper Address Mask (EBM0_UAM)

0:3		Reserved	
4:15	UMSK	OPB Upper Address Mask	This register must be initialized by software if the external master provides less than a complete 32-bit address.
16:31		Reserved	

DCR Accessed using EBM0_CFGADDR; EBM0_CFGDATA; Offset 0x05 R/W
See EBM0 OPB Upper Address Register (EBM0_UAR) on page 977.

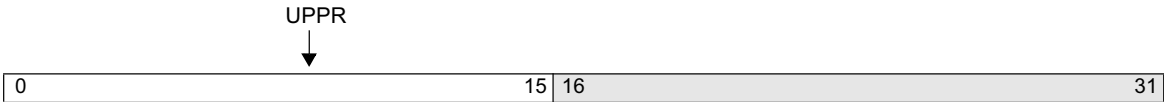


Figure 32-139. EBM0 Upper Address Register (EBM0_UAR)

0:15	UPPR	OPB Upper Address	This register must be initialized by software.
16:31		Reserved	

Preliminary User’s Manual

MMIO 0x1 40000840–00x1 4000084C (EMAC0), 0x1 40000940–00x1 4000094C (EMAC1), 0x1 40000C40–00x1 40000C4C (EMAC2), 0x1 40000E40–00x1 40000E4C (EMAC3)

See Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4) on page 854.

0	15	16	31
---	----	----	----

Figure 32-140. Group Address Hash Tables 1–4 (EMACx_GAHT1–EMACx_GAHT4)

0:15		Reserved
16:31		Group Address Hash Number

MMIO 0x1 4000081C (EMAC0), 0x1 4000091C (EMAC1), 0x1 40000C1C (EMAC2), 0x1 40000E1C (EMAC3)
See *Individual Address High (EMACx_IAHR)* on page 851.

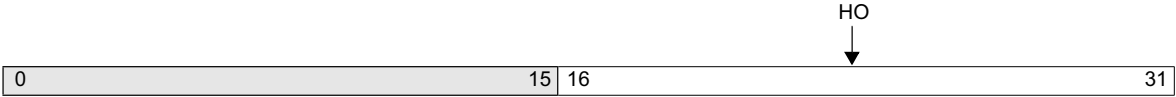


Figure 32-141. Individual Address High Register (EMACx_IAHR)

0:15		Reserved	
16:31	HO	Receive and Transmit High Order High-order halfword of the station unique individual address	This field contains bits 0:15 of the destination address (bit 0 is the most significant bit).

Preliminary User’s Manual

MMIO 0x1 40000830–0x1 4000083C (EMAC0), 0x1 40000930–0x1 4000093C (EMAC1), 0x1 40000C30–0x1 40000C3C (EMAC2), 0x1 40000E30–0x1 40000E3C (EMAC3)

See Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4) on page 854.

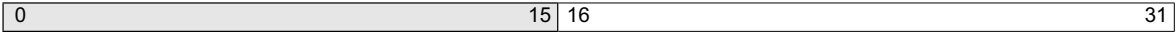


Figure 32-142. Individual Address Hash Tables 1–4 (EMACx_IAHT1–EMACx_IAHT4)

0:15		Reserved
16:31		Individual Address Hash Number

MMIO 0x1 40000820 (EMAC0), 0x1 40000920 (EMAC1), 0x1 40000C20 (EMAC2), 0x1 40000E20 (EMAC3)

See *Individual Address Low (EMACx_IALR)* on page 852.

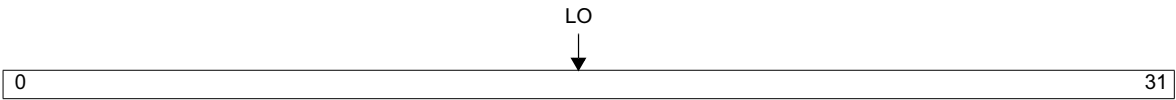


Figure 32-143. Individual Address Low Register (EMACx_IALR)

0:31	LO	Receive and Transmit Low Order Low-order bits of Receive Individual Address or Transmit Source Address
------	----	--

MMIO 0x1 40000870 (EMAC0), 0x1 40000970 (EMAC1), 0x1 40000C70 (EMAC2), 0x1 40000E70 (EMAC3)

See *Internal PCS Configuration Register (EMACx_IPCR)* on page 859.

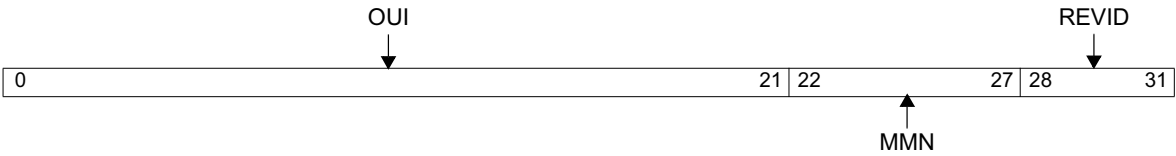


Figure 32-144. Internal PCS Configuration Register (EMACx_IPCR)

0:21	OUI	OUI Value
22:27	MMN	Manufacturer Model Number
28:31	REVID	Revision Number

MMIO 0x1 40000858 (EMAC0), 0x1 40000958 (EMAC1), 0x1 40000C58 (EMAC2), 0x1 40000E58 (EMAC3)

See *Inter-Packet Gap Value Register (EMACx_IPGVR)* on page 855.



Figure 32-145. *inter-Packet Gap Value Register (EMACx_IPGVR)*

0:25		Reserved
26:31		Inter-Packet Gap

MMIO 0x1 40000818 (EMAC0), 0x1 40000918 (EMAC1), 0x1 40000C18 (EMAC2), 0x1 40000E18 (EMAC3)

See *Interrupt Status Enable Register (EMACx_ISER)* on page 849.

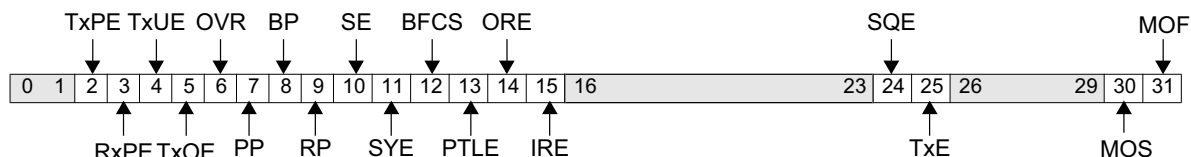


Figure 32-146. Interrupt Status Enable Register (EMACx_ISER)

0:1		Reserved
2	TxPE	TX Parity Error 0 No TX parity error 1 Tx parity error occurs when a transmit data bit is damaged at the transmit FIFO
3	RxPE	RX Parity Error 0 No RX parity error 1 Rx parity error occurs when a receive data bit is damaged at the receive FIFO or internal LPRA
4	TxUE	TX Underrun Event 0 No underrun event 1 Underrun event while packet is being transferred TX FIFO almost empty
5	RxOE	RX Overrun Event 0 No overrun event 1 Overrun event while packet is being received RX FIFO almost full
6	OVR	Overrun 0 Overrun error will not generate an interrupt. 1 Overrun error will generate an interrupt.
7	PP	Pause Packet 0 Received control pause packet will not generate an interrupt. 1 Received control pause packet will generate an interrupt.
8	BP	Bad Packet 0 Early termination on received packet will not generate an interrupt. 1 Early termination on received packet will generate an interrupt.
9	RP	Runt Packet 0 Received runt packet will not generate an interrupt. 1 Received runt packet will generate an interrupt.
10	SE	Short Event 0 Short event during receive will not generate an interrupt. 1 Short event during receive will generate an interrupt.
11	ALE	Alignment Error 0 Alignment error in received packet will not generate an interrupt. 1 Alignment error in received packet will generate an interrupt.

12	BFCS	Bad FCS 0 FCS error in received packet will not generate an interrupt. 1 FCS error in received packet will generate an interrupt.
13	PTLE	Packet Too Long Error 0 Oversized packets received will not generate an interrupt. 1 Oversized packet received will generate an interrupt.
14	ORE	Out Of Range Error 0 Out of range error on received packet will not generate an interrupt. 1 Out of range error on received packet will generate an interrupt.
15	IRE	In Range Error 0 In range error on received packet will not generate an interrupt. 1 In range error on received packet will generate an interrupt.
16:23		Reserved
24	SQE	SQE Error 0 SQE error on TX Channel will not generate an interrupt. 1 SQE error on TX Channel will generate an interrupt.
25	TxE	Transmit Error 0 TX error on TX Channel will not generate an interrupt. 1 TX error on TX Channel will generate an interrupt.
26:29		Reserved
30	MOS	MMA Operation Succeeded 0 Successful MMA Operation with a PHY will not generate an interrupt. 1 Successful MMA Operation with a PHY will generate an interrupt.
31	MOF	MMA Operation Failed 0 Unsuccessful MMA Operation with a PHY will not generate an interrupt. 1 Unsuccessful MMA Operation with a PHY will generate an interrupt.

MMIO 0x1 40000814 (EMAC0), 0x1 40000914 (EMAC1), 0x1 40000C14 (EMAC2), 0x1 40000E14 (EMAC3)

See *Interrupt Status Register (EMACx_ISR)* on page 847.

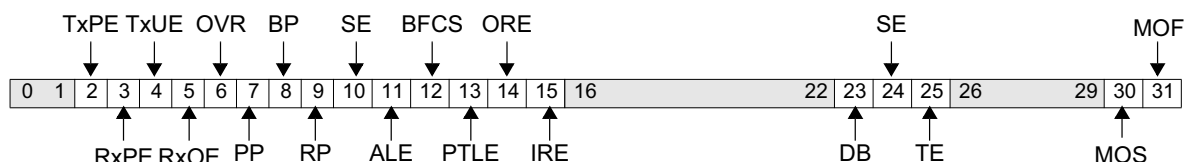


Figure 32-147. Interrupt Status Register (EMACx_ISR)

0:1		Reserved
2	TxPE	TX Parity Error 0 No TX parity error 1 Tx parity error occurs when a transmit data bit is damaged at the transmit FIFO
3	RxPE	RX Parity Error 0 No RX parity error 1 Rx parity error occurs when a receive data bit is damaged at the receive FIFO or internal LPRA
4	TxUE	TX Underrun Event 0 No underrun event 1 Underrun event while packet is being transferred
5	RxOE	RX Overrun Event 0 No overrun event 1 Overrun event while packet is being received
6	OVR	Overrun 0 No overrun error 1 Overrun error during reception of recent packet
7	PP	Pause Packet 0 Received packet is not a control pause packet 1 Received packet is a control pause packet
8	BP	Bad Packet 0 Receive operation OK 1 Early termination was initiated because of a packet error
9	RP	Runt Packet 0 No Runt packets received 1 Runt packet received
10	SE	Short Event 0 No short events 1 Duration of PHY_RX_DV signal less than ShortEventMaxTime constant
11	ALE	Alignment Error 0 No alignment error in received packet 1 Alignment error in received packet
12	BFCS	Bad FCS 0 No FCS error in received packet 1 Packet with an FCS error received

13	PTLE	Packet Too Long Error 0 No oversized packets received 1 Oversized packet received	Set if EMACx_RMR[ROP] = 1 and the received packet length exceeded the maximum allowed value: <ul style="list-style-type: none"> 1518 octets for standard packet (checked only if the length/type field of the transmitted packet contained length value) 1522 octets for VLAN tagged packet (checked only if the length/type field of the transmitted packet contained length value and jumbo support is disabled) 9018 octets for jumbo packet (with no VLAN support) 9022 octets for VLAN tagged Jumbo packet
14	ORE	Out Of Range Error 0 Received packet length field value OK 1 Received packet length field value greater than the maximum allowed LLC data size	Indicates that received packet has a length field value greater than the maximum allowed logical link control (LLC) data size (greater than 1500 and less than 1536).
15	IRE	In Range Error 0 Received packet does not contain an In Range Error 1 Received packet contains an In Range Error	
16:22		Reserved	
23	DB	Dead Bit 0 No transmit error or SQE for TX Channel 1 Transmit error or SQE has occurred for TX Channel	If EMACx_ISR[DB] = 1, EMAC does not request service for TX Channel from MAL, even if EMACx_TMR0[GNP] = 1. EMACx_ISR[DB] does not affect EMAC interrupt.
24	SE0	Signal Quality Error 0 No SQEs on TX Channel 1 SQE test failure during transmission of a packet from TX Channel	Applicable only in half-duplex mode during 10 Mbps operations; 0 in all other modes.
25	TE0	Transmit Error 0 TX Channel transmission OK 1 TX Channel transmission aborted	EMAC aborts the transmitted packet if one of the following events takes place: <ul style="list-style-type: none"> Late collision detection Excessive collision detection Excessive deferral TX FIFO underrun Loss of carrier sense
26:29		Reserved	Always 0
30	MOS	MMA Operation Succeeded 0 MMA_CONTROL addressed on the OPB 1 PHY transfer valid	The device driver should poll assertion of EMACx_ISR[MOS] or EMACx_ISR[MOF] before issuing a new command or before using data read from the PHY.
31	MOF	MMA Operation Failed 0 MMA_CONTROL addressed on the OPB 1 PHY transfer not valid	The device driver should poll assertion of EMACx_ISR[MOF] or EMACx_ISR[MOS] before issuing a new command or before using data read from the PHY.

MMIO 0x1 4000 0850 (EMAC0), 0x1 4000 0950 (EMAC1), 0x1 4000 0C50 (EMAC2), 0x1 4000 0E50 (EMAC3)

See *Last Source Address High (EMACx_LSAH)* on page 855.

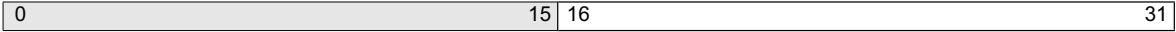


Figure 32-148. Last Source Address High Register (EMACx_LSAH)

0:15		Reserved
16:31		Last Source Address High-Order Halfword

MMIO 0x1 40000854 (EMAC0), 0x1 40000954 (EMAC1), 0x1 40000C54 (EMAC2), 0x1 40000E54 (EMAC3)

See *Last Source Address Low (EMACx_LSAL)* on page 855.



Figure 32-149. Last Source Address Low Register (EMACx_LSAL)

0:31	Last Source Address Low-Order Word
------	------------------------------------

MMIO 0x1 40000800 (EMAC0), 0x1 40000900 (EMAC1), 0x1 40000C00 (EMAC2), 0x1 40000E00 (EMAC3)

See *Mode Register 0 (EMACx_MR0)* on page 841.

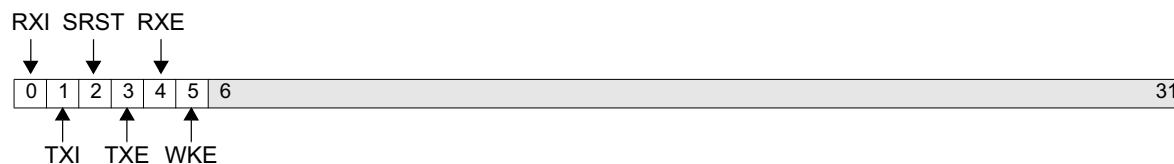


Figure 32-150. Mode Register 0 (EMACx_MR0)

0	RXI	Receive MAC Idle 0 RX MAC processing packet 1 RX MAC idle; RX packet processing complete	Read-only
1	TXI	Transmit MAC Idle 0 TX MAC processing packet 1 TX MAC idle; TX packet processing complete	Read-only
2	SRST	EMAC Software Reset 0 EMAC reset is complete 1 Reset the EMAC	Generates a general reset to EMAC through a software command. After setting this bit, EMAC hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive EMAC to the reset state. The bit is cleared by the hardware when the reset is completed. If EMACx_MR0[SRST] = 1, writing to any EMAC register, and reading any other bit in this register, is not supported.
3	TXE	Transmit MAC Enable 0 TX MAC is disabled 1 TX MAC is enabled	
4	RXE	Receive MAC Enable 0 RX MAC is disabled 1 RX MAC is enabled	
5	WKE	Wake-Up Enable 0 Incoming packets are not examined for wake-up packet 1 Examine incoming packets for wake-up packet	Software can change EMACx_MR0[WKE] only while EMACx_MR0[RXI] = 1 and EMACx_MR0[RXE] = 0.
6:31		Reserved	

EMACx_MR1

Mode Register 1

Preliminary User's Manual

MMIO 0x1 4000 0804 (EMAC0), 0x1 4000 0904 (EMAC1), 0x1 4000 0C04 (EMAC2), 0x1 4000 0E04 (EMAC3)

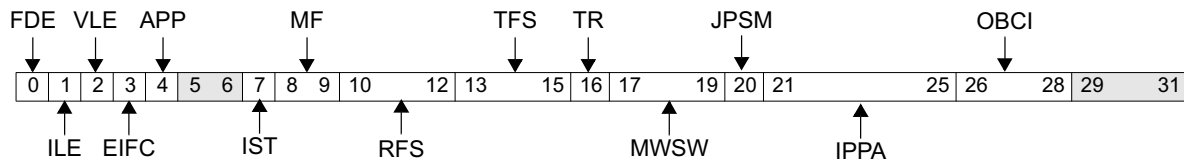
See *Mode Register 1 (EMACx_MR1)* on page 842.

Figure 32-151. Mode Register 1 (EMACx_MR1)

0	FDE	Full-Duplex Enable 0 Disable simultaneous transmit and receive 1 Enable simultaneous transmit and receive	
1	ILE	Internal Loop-back Enable 0 No wrap back 1 Transmitted packets wrapped back to receive FIFO	Full Duplex must also be set (EMACx_MR1[FDE]=1).
2	VLE	VLAN Enable 0 Disable processing of VLAN Tags 1 Enable processing of VLAN Tags	
3	EIFC	Enable Integrated Flow Control 0 Disable integrated flow control mechanism 1 Enable integrated flow control mechanism	Refer to <i>Flow Control</i> on page 829 for more details. Set EMACx_MR1[EIFC] = 0 in half-duplex mode.
4	APP	Allow Pause Packet 0 Disables processing of incoming control (pause) packets 1 Enables processing of incoming control (pause) packets	
5:6		Reserved	Always zero
7	IST	Ignore SQE test 0 Wait for end of SQE test period before activation of valid signal 1 Do not wait for end of SQE test period before activation of valid signal	EMACx_MR1[IST] = 0 only during half-duplex operation on 10 Mbps media.
8:9	MF	Medium Frequency 00 10 Mbps (Ethernet mode) 01 100 Mbps (Fast Ethernet mode) 10 1000 Mbps (Gigabit Ethernet mode) without using the internal GPCS device 11 1000 Mbps (Gigabit Ethernet mode) with the use of internal GPCS device	Defines the possible operational frequency on the MII/GMII interface. Gigabit ethernet mode settings are only valid for EMAC 2 and 3.
10:12	RFS	Receive (RX) FIFO Size 000 512 bytes 001 1 KB 010 2 KB 011 4 KB 100 8 KB 101 16 KB 110 Reserved	Note: The highest value for RX FIFO size is 4KB for EMAC0 and EMAC1, and 16KB for EMAC2 and EMAC3.

Preliminary User's Manual

13:15	TFS	Transmit (TX) FIFO Size 000 512 bytes 001 1 KB 010 2 KB 011 4 KB 100 8 KB 101 16 KB 110 Reserved	Note: The highest value for TX FIFO size is 2KB for EMAC0, EMAC1, EMAC2 and EMAC3.
16	TR	Transmit Request 0 Single packet 1 Multiple packets	Defines the different modes for using transmit channel of EMAC.
17:19	MWSW	Maximum Waiting Status Words 000 A packet is not sent until the status of the previous packet has been received 001 Allows up to one status to be pending when sending the next packet 010 Reserved	Defines the number of status words EMAC can wait for, and still continue transmission. A '0' value will force EMAC to wait for every status until requesting a transmission of new packet. For better performance set EMAC0_MR1[MWSW=001]
20	JPSM	Jumbo Packet Support Mode 0 Jumbo packet support mode disabled 1 Jumbo packet support mode enabled	When enabled, EMAC is capable of handling packets with a length of up to 9018 bytes. This bit can be set only when 1000 Mbps mode is chosen
21:25	IPPA	Internal PCS PHY Address	
26:28	OBCI	OPB Bus Clock Indication 000 50 MHz 001 66 MHz 010 83 MHz 011 100 MHz 100 Above 100 MHz	EMACx_MR1[TFS] and data clock of EMACx_MR1 are used for generation of EMC_MDC clock. Note: When operational frequency differs from this list, then the next greater frequency should be chosen.
29:31		Reserved	

MMIO 0x1 4000 086C (EMAC0), 0x1 4000 096C (EMAC1), 0x1 4000 0C6C (EMAC2), 0x1 4000 0C6C (EMAC3)
See *Received Octets Register (EMACx_OCRX)* on page 859.



Figure 32-152. Number of Octets Received (EMACx_OCRX)

0:31	OCRX	Number of octets (bytes) received.
------	------	------------------------------------

MMIO 0x1 40000868 (EMAC0), 0x1 40000968 (EMAC1), 0x1 40000C68 (EMAC2), 0x1 40000C68 (EMAC3)

See *Transmitted Octects (EMACx_OCTX)* on page 859.



Figure 32-153. Number of Octets Transmitted (EMACx_OCTX)

0:31	OCTX	Number of octets (bytes) transmitted.
------	------	---------------------------------------

MMIO 0x1 4000082C (EMAC0), 0x1 4000 092C (EMAC1), 0x1 40000C2C (EMAC2), 0x1 40000E2C (EMAC3)

See *Pause Timer Register (EMACx_PTR)* on page 853.

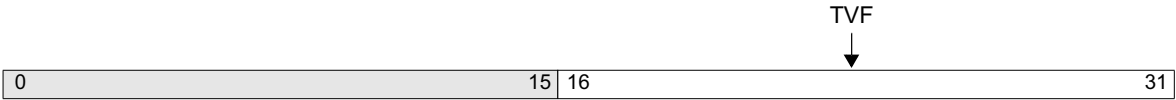


Figure 32-154. *Pause Timer Register (EMACx_PTR)*

0:15		Reserved
16:31	TVF	Timer Value Field

MMIO 0x1 4000 0810 (EMAC0), 0x1 4000 0910 (EMAC1), 0x1 4000 0C10 (EMAC2), 0x1 4000 0E10 (EMAC3)

See *Receive Mode Register (EMACx_RMR)* on page 846.

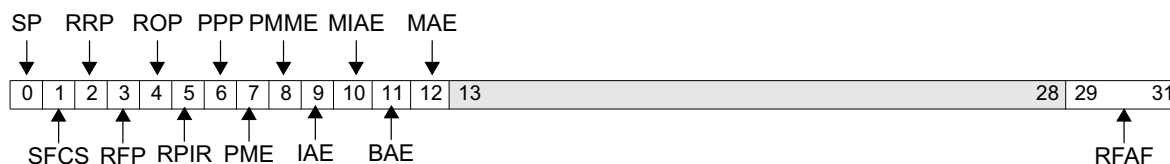


Figure 32-155. Receive Mode Register (EMACx_RMR)

0	SP	Strip Padding 0 Do not strip pad bytes from the received packet. 1 Strip pad/FCS bytes from the received packet.
1	SFCS	Strip FCS 0 Do not strip FCS bytes from the received packet. 1 Strip FCS bytes from the received packet.
2	RRP	Receive Runt Packets 0 Discard packets less than 64 bytes in length. 1 Receive packets less than 64 bytes in length.
3	RFP	Allow Receive Packets with a FCS Error 0 Discard packets containing a FCS error. 1 Receive packets containing a FCS error.
4	ROP	Receive Oversize Packet 0 Discard packets that activate Packet Is Too Long error. 1 Receive packets that activate Packet Is Too Long error.
5	RPIR	Receive Packets with In Range Error 0 Discard packets that activate In Range Error. 1 Receive packets that activate In Range Error.
6	PPP	Propagate Pause Packet 0 Do not propagate incoming pause packet to MAL; remove packet from FIFO. 1 Propagate incoming pause packet to MAL.
7	PME	Promiscuous Mode Enable 0 Do not enable promiscuous mode. 1 Accept all packets.
8	PMME	Promiscuous Multicast Mode Enable 0 Do not accept all multicast packets. 1 Accept all multicast packets.
9	IAE	Individual Address Enable 0 Do not compare address of received packets with content of individual address register. 1 Compare address of received packets with content of individual address register.
10	MIAE	Multiple Individual Address Enable 0 Do not compare address of received packets with hash table of individual addresses. 1 Compare address of received packets with hash table of individual addresses.
11	BAE	Broadcast Address Enable 0 Do not compare address of received packets with broadcast addresses. 1 Compare address of received packets with broadcast addresses.

12	MAE	<p>Multicast Address Enable</p> <p>0 Do not compare address of received packets with multicast addresses.</p> <p>1 Compare address of received packets with multicast addresses.</p>
13:28		Reserved
29:31	RFAF	<p>RX FIFO Almost Full</p> <p>000 Reserved</p> <p>001 Actual number of entries limit in FIFO is 2 and the limit in value of PHY clock cycles is 32</p> <p>010 Actual number of entries limit in FIFO is 4 and the limit in value of PHY clock cycles is 64</p> <p>011 Actual number of entries limit in FIFO is 8 and the limit in value of PHY clock cycles is 128</p> <p>100 Actual number of entries limit in FIFO is 16 and the limit in value of PHY clock cycles is 256</p> <p>101 Actual number of entries limit in FIFO is 32 and the limit in value of PHY clock cycles is 512</p> <p>110 Actual number of entries limit in FIFO is 64 and the limit in value of PHY clock cycles is 1024</p> <p>111 Actual number of entries limit in FIFO is 128 and the limit in value of PHY clock cycles is 2048</p> <p>When the number of occupied entries in the RX FIFO is greater than or equal to this limit, and while a packet is received, RX_FIFO_ALMOST_FULL interrupt is asserted.</p> <p>Note: The value of '111' is applicable only when 'Jumbo Packet' option is enabled. See <i>Transmit Mode Register 1 (EMACx_TMR1)</i> on page 845. Otherwise, writing a value of '111' will disable the RX FIFO almost full limit option, that is, no interrupt will occur. This is also set as default option.</p>

MMIO 0x1 40000864 (EMAC0), 0x1 40000964 (EMAC1), 0x1 40000C64 (EMAC2), 0x1 40000E64 (EMAC3)

See *Receive Low/High Water Mark Register (EMACx_RWMR)* on page 858.

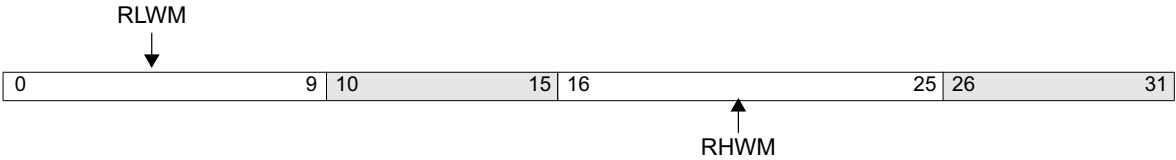


Figure 32-156. Receive Low/High Water Mark Register (EMACx_RWMR)

0:9	RLWM	Receive Low Water Mark
10:15		Reserved
16:25	RHWM	Receive High Water Mark
26:31		Reserved

MMIO 0x1 4000085C (EMAC0), 0x1 4000095C (EMAC1), 0x1 40000C5C (EMAC2), 0x1 40000E5C (EMAC3)

See *STA Control Register (EMACx_STACR)* on page 856.

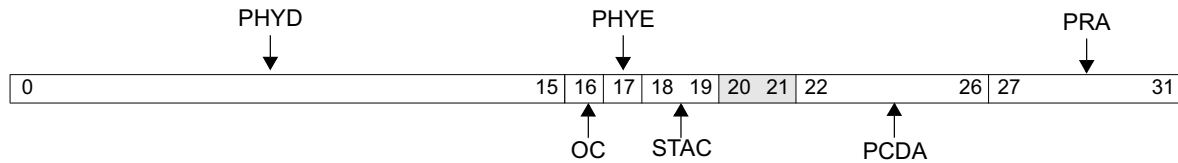


Figure 32-157. STA Control Register (EMACx_STACR)

0:15	PHYD	PHY data	Data to be sent to the PHY if the command is a write, or data is read from the PHY if the command is a read.
16	OC	Operation Complete 0 EMACx_STACR is addressed 1 PHY data transfer complete	
17	PHYE	PHY Error 0 Successful read transaction 1 Read transaction was not successful	EMACx_STACR[PHYE] = 0 when a read is successful.
18:19	STAC	STA Command 00 Reserved 01 Read 10 Write 11 Reserved	EMAC sets EMACx_STACR[STAC] = 0 when the command is completed.
20:21		Reserved	
22:26	PCDA	PHY Command Destination Address	This field contains the address of the PHY intended to receive the command
27:31	PRA	PHY Register Address	This field contains the PHY register address

MMIO 0x1 40000808 (EMAC0), 0x1 40000908 (EMAC1), 0x1 40000C08 (EMAC2), 0x1 40000E08 (EMAC3)

See *Transmit Mode Register 0 (EMACx_TMR0)* on page 844.

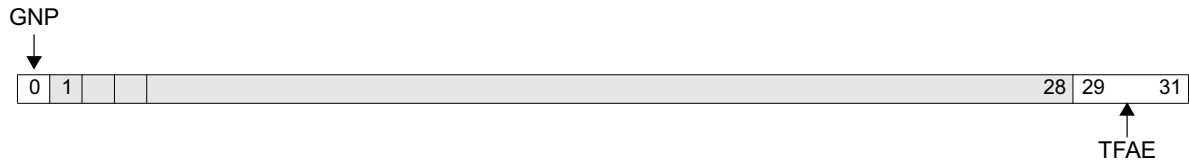


Figure 32-158. Transmit Mode Register 0 (EMACx_TMR0)

0	GNP	Get New Packet 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel	EMACx_TMR0[GNP0] = 0 if EMAC is programmed.
1:28		Reserved	
29:31	TFAE	TX FIFO Almost Empty 000 Reserved 001 Actual number of entries limit in FIFO is 2 and the limit in value of PHY clock cycles is 32 010 Actual number of entries limit in FIFO is 4 and the limit in value of PHY clock cycles is 64 011 Actual number of entries limit in FIFO is 8 and the limit in value of PHY clock cycles is 128 100 Actual number of entries limit in FIFO is 16 and the limit in value of PHY clock cycles is 256 101 Actual number of entries limit in FIFO is 32 and the limit in value of PHY clock cycles is 512 110 Actual number of entries limit in FIFO is 64 and the limit in value of PHY clock cycles is 1024 111 Actual number of entries limit in FIFO is 128 and the limit in value of PHY clock cycles is 2048	When the number of occupied entries in the TX FIFO is less than or equal to this limit, and while a packet is transmitted, TX_FIFO_ALMOST_EMPTY interrupt is asserted. Note: The value of '111' is applicable only when 'Jumbo Packet' option is enabled. See <i>Transmit Mode Register 1 (EMACx_TMR1)</i> on page 845. Otherwise, writing a value of '111' will disable the TX FIFO almost empty limit option, that is, no interrupt will occur. This is also set as default option.

MMIO 0x1 4000 080C (EMAC0), 0x1 4000 090C (EMAC1), 0x1 40000C0C (EMAC2), 0x1 40000E0C (EMAC3)
See *Transmit Mode Register 1 (EMACx_TMR1)* on page 845.

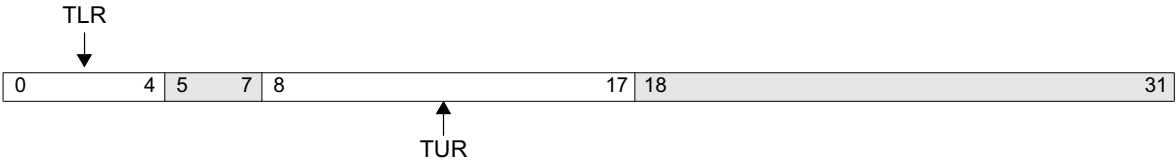


Figure 32-159. Transmit Mode Register 1 (EMACx_TMR1)

0:4	TLR	Transmit Low Request
5:7		Reserved
8:17	TUR	Transmit Urgent Request
18:31		Reserved

MMIO 0x1 40000860 (EMAC0), 0x1 40000960 (EMAC1), 0x1 40000C60 (EMAC2), 0x1 40000E60 (EMAC3)

See *Transmit Request Threshold Register (EMACx_TRTR)* on page 857.



Figure 32-160. Transmit Request Threshold Register (EMACx_TRTR)

0:7	TRT	Transmit Request Threshold The following number of bytes must be placed in the Transmit FIFO before initiating a transmit request. 0000_0000 64 bytes 0000_0001 128 bytes 0000_0010 192 bytes 0000_0011 256 bytes . . . 1111_1100 16.192 bytes 1111_1101 16.256 bytes 1111_1110 16.320 bytes 1111_1111 16.384 bytes
8:31		Reserved

MMIO 0x1 40000828 (EMAC0), 0x1 40000928 (EMAC1), 0x1 40000C28 (EMAC2), 0x1 40000E28 (EMAC3)

See *VLAN TCI Register (EMACx_VTCI)* on page 853.

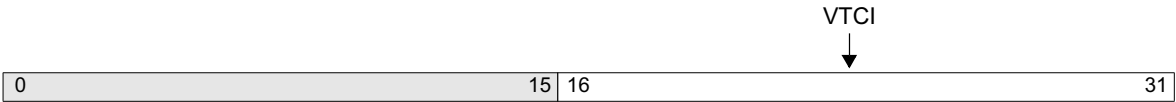


Figure 32-161. VLAN TCI Register (EMACx_VTCI)

0:15		Reserved
16:31	VTCI	VLAN TCI tag

Preliminary User’s Manual

MMIO 0x1 40000824 (EMAC0), 0x1 40000924 (EMAC1), 0x1 40000C24 (EMAC2), 0x1 40000E24 (EMAC3)

See *VLAN TPID Register (EMACx_VTPID)* on page 852.

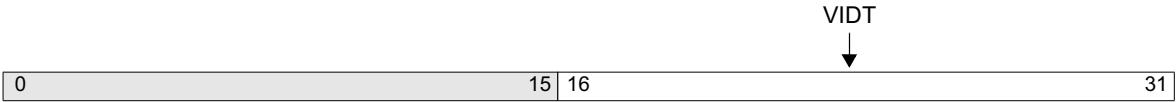


Figure 32-162. VLAN TPID Register (EMACx_VTPID)

0:15		Reserved
16:31	VIDT	VLAN ID tag

GPCSx_ANAR

GPCS Auto Negotiation Advertisement Register

Preliminary User's Manual

0x04 R/W accessed via EMACx_STACR[PHYD]

See GPCS Auto Negotiation Advertisement Register (GPCSx_ANAR) on page 864.

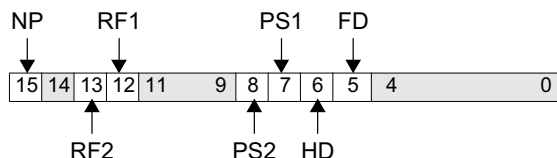


Figure 32-163. GPCS Auto Negotiation Advertisement Register (GPCSx_ANAR)

15	NP	NextPage	This bit shall be set to logic '1' in order to request next page transmission. Subsequent next pages may set the NextPage bit to logic '0' in order to communicate that there is no more next page information to be sent. The default value is '0'.
14		Reserved	Write as '0', ignore on read.
13	RF2	Remote Fault 2	This bit, in combination with bit 12, RF1, denotes the remote fault bits. The encoding is as follows: RF1 RF2 0 0 No error, link OK 0 1 Off-line 1 0 Link_Failure 1 1 Auto-Negotiation Error The default value is '00'.
12	RF1	Remote Fault 1	This bit, in combination with bit 13, RF2, denotes the remote fault bits. The encoding is as follows: RF1 RF2 0 0 No error, link OK 0 1 Off-line 1 0 Link_Failure 1 1 Auto-Negotiation Error The default value is '00'.
11:9		Reserved	Always 0
8	PS2	Pause 2	This bit, in combination with bit PS1, indicates the pause capability. The encoding is as follows: PS1 PS2 0 0 No Pause 0 1 Asymmetric Pause towards Link Partner 1 0 Symmetric Pause 1 1 Both Symmetric Pause and Asymmetric Pause towards Link Partner The default value is '00'
7	PS1	Pause 1	This bit, in combination with bit PS2, indicates the pause capability. The encoding is as follows: PS1 PS2 0 0 No Pause 0 1 Asymmetric Pause towards Link Partner 1 0 Symmetric Pause 1 1 Both Symmetric Pause and Asymmetric Pause towards Link Partner The default value is '00'

6	HD	Half Duplex	When this bit is set, the device is capable of operating in Half Duplex mode. The default value is '0'. Please note that in case Auto-Negotiation is enabled, this bit must be set according to the REG_FULL_DUPLEX input (that is the negated value of REG_FULL_DUPLEX).
5	FD	Full Duplex	When this bit is set, the device is capable of operating in Full Duplex mode. The default value is '0'. Please note that in case Auto-Negotiation is enabled, this bit must be set according to the REG_FULL_DUPLEX input (that is, the same value as REG_FULL_DUPLEX).
4:0			Always 0

0x06 R accessed via EMACx_STACR[PHYD]

See GPCS Auto Negotiation Expansion Register (GPCSx_ANER) on page 867.

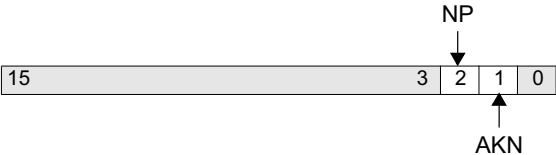


Figure 32-164. GPCS Auto Negotiation Expansion Register (GPCSx_ANER)

15:3		Reserved	Always 0
2	NP	Next Page Able	Indicates that the GPCS unit has the ability to engage in Next Page transactions. Always set to '1'.
1	AKN	Page Received	When this bit is set, a new page has been received and stored in the applicable Auto-Negotiation Link Partner Ability register (Base Page or Next Page). It is reset each time this register is read via the Management interface or by PCS reset. The default value is '0'.
0		Reserved	Always 0

0x08 R accessed via EMACx_STACR[PHYD]

See GPCS Auto Negotiation Next Page Transmit Register (GPCSx_ANLNPR) on page 869.

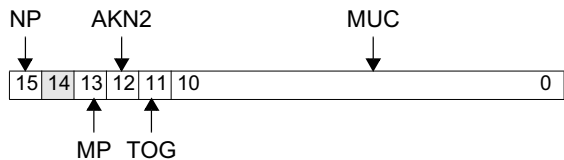


Figure 32-165. GPCS Auto Negotiation Next Page Transmit Register (GPCSx_ANLNPR)

15	NP	Next Page Enable 0 Last page 1 Additional Next Page(s) will follow	Used by the Next Page function to indicate whether or not this is the last Next Page to be transmitted. ‘
14		Reserved	Write as 0. Ignore on read
13	MP	Message Page 0 Unformatted Page 1 Message Page (default value)	Used by the Next Page function to differentiate a Message Page from an Unformatted Page.
12	AKN2	Acknowledge 2 0 Cannot comply with message (default value) 1 Will comply with message	Used by the Next Page function to indicate that a device has the ability to comply with the message.
11	TOG	Toggle	Used to ensure synchronization with the Link Partner during Next Page exchange. This bit shall always take the opposite value of the Toggle bit in the previously exchanged Link Code Word. The initial value of the Toggle bit in the first Next Page transmitted is the inverse of bit 11 in the base Link Code Word.
10:0	MUC	Message Unformatted Code	Used by the Next Page function to carry a single predefined Message Code. The default value is “00000000001”

GPCSx_ANLR

GPCS Auto Negotiation Link Partner Base Page Ability Register

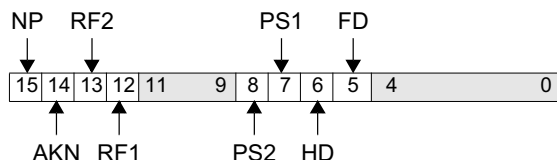
Preliminary User's Manual**0x05 R accessed via EMACx_STACR[PHYD]**See *GPCS Auto Negotiation Link Partner Base Page Ability Register (GPCSx_ANLR)* on page 866.

Figure 32-166. GPCS Auto Negotiation Link Partner Base Page Ability Register (GPCSx_ANLR)

15	NP	NextPage	When this bit is set to '1', the next page transmission is requested. The default value is '0'.
14	AKN	Acknowledge	When this bit is set, the Link Partner has successfully received at least three consecutive and matching rx_Config_Reg<D15:D0> values (ignoring the Acknowledge bit value). The default value is '0'.
13	RF2	Remote Fault 2	This bit, in combination with bit 12, RF1, denotes the remote fault bits. The encoding is as follows: RF1 RF2 0 0 No error, link OK 0 1 Off-line 1 0 Link_Failure 1 1 Auto-Negotiation Error The default value is '00'.
12	RF1	Remote Fault 1	This bit, in combination with bit 13, RF2, denotes the remote fault bits. The encoding is as follows: RF1 RF2 0 0 No error, link OK 0 1 Off-line 1 0 Link_Failure 1 1 Auto-Negotiation Error The default value is '00'.
11:9		Reserved	Always 0
8	PS2	Pause 2	This bit, in combination with bit PS1, indicates the pause capability. The encoding is as follows: PS1 PS2 0 0 No Pause 0 1 Asymmetric Pause towards Link Partner 1 0 Symmetric Pause 1 1 Both Symmetric Pause and Asymmetric Pause towards Link Partner The default value is '00'
7	PS1	Pause 1	This bit, in combination with bit PS2, indicates the pause capability. The encoding is as follows: PS1 PS2 0 0 No Pause 0 1 Asymmetric Pause towards Link Partner 1 0 Symmetric Pause 1 1 Both Symmetric Pause and Asymmetric Pause towards Link Partner The default value is '00'
6	HD	Half Duplex	When this bit is set, the device is capable of operating in Half Duplex mode. The default value is '0'.

5	FD	Full Duplex	When this bit is set, the device is capable of operating in Full Duplex mode. The default value is '0'.
4:0			Always 0

0x07 R accessed via EMACx_STACR[PHYD]

See GPCS Auto Negotiation Next Page Transmit Register (GPCSx_ANPTR) on page 868.

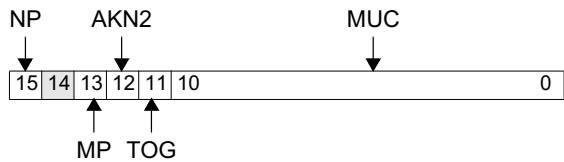


Figure 32-167. GPCS Auto Negotiation Next Page Transmit Register (GPCSx_ANPTR)

15	NP	Next Page Enable 0 Last page 1 Additional Next Page(s) will follow	Used by the Next Page function to indicate whether or not this is the last Next Page to be transmitted. ‘
14		Reserved	Write as 0. Ignore on read
13	MP	Message Page 0 Unformatted Page 1 Message Page (default value)	Used by the Next Page function to differentiate a Message Page from an Unformatted Page.
12	AKN2	Acknowledge 2 0 Cannot comply with message (default value) 1 Will comply with message	Used by the Next Page function to indicate that a device has the ability to comply with the message.
11	TOG	Toggle	Used to ensure synchronization with the Link Partner during Next Page exchange. This bit shall always take the opposite value of the Toggle bit in the previously exchanged Link Code Word. The initial value of the Toggle bit in the first Next Page transmitted is the inverse of bit 11 in the base Link Code Word.
10:0	MUC	Message Unformatted Code	Used by the Next Page function to carry a single predefined Message Code. The default value is “00000000001”

0x13 R/W accessed via EMACx_STACR[PHYD]

See GPCS Configuration Register (GPCSx_CFG) on page 873.

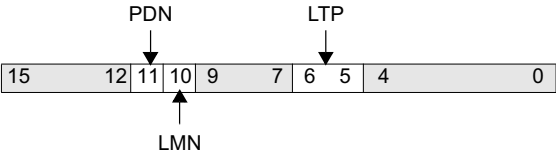


Figure 32-168. GPCS Configuration Register (GPCSx_CFG)

15:12		Reserved	
11	PDN	Power Down Negotiation 0 GPCS does not renegotiate before powering-down or isolating from GMII 1 GPCS renegotiates to announce to the Link Partner that it is going off-line before powering-down or isolating from GMII. Reset value is '0'.	
10	LMN	Loop Mode Negotiation 0 GPCS does not renegotiate before changing to the LoopBack mode. 1 GPCS renegotiates to announce to the Link Partner that it is going off-line before changing to the LoopBack mode Reset value is '0'.	
9:7		Reserved	Read as 0
6:5	LTP	Link Timer Period The value of these bits determines the period of the GPCS Link Timer. bit 6 bit 5 Link Timer Period 0 0 10 ms (+ 10 ms, -0 ms) 0 1 5 ms (+ 5 ms, -0 ms) 1 0 1 ms (+ 1 ms, -0 ms) 1 1 1 us (+ 1 us, -0 us) The reset value is '00'	
4:0		Reserved	Read as 0

GPCSx_CR

GPCS Control Register

Preliminary User's Manual

0x00 R/W accessed via EMACx_STACR[PHYD]

See GPCS Control Register (GPCSx_CR) on page 861.

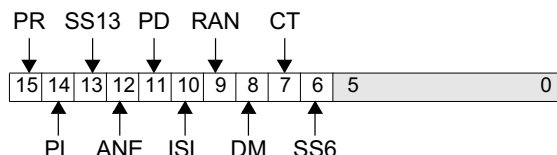


Figure 32-169. GPCS Control Register (GPCSx_CR)

15	PR	PhyReset	This is a soft reset bit. When this bit is set, the GPCS unit logic is reset, and all GPCS registers go to their default value. While the value of this bit is a logic '1', any attempt to write to the GPCS registers will have no effect on the registers. This bit is self-clearing.
14	PL	PhyLoop	When this bit is set, the GPCS unit asserts the PHY_LOOP_EN pin. Typically, the PHY_LOOP_EN pin is connected to the SERDES (such as EWRAP) so that the SERDES can be configured for LoopBack mode. The default value is '0'.
13	SS13	SpeedSelection13	This bit is read as '0'. In combination with bit 6, the SpeedSelection bit in this register, read as '1', it indicates that the GPCS unit works at a PHY speed of 1000 Mbps.
12	ANE	AutoNegEnable	When this bit is set, Auto-Negotiation is enabled. The default value is '0'.
11	PD	PowerDown	When this bit is set, the GPCS unit is placed in the Power Saving mode (the SYS_REG_CLK and PHY_RX_CLK1 can be stopped). When the GPCS unit is in the Power Saving mode, it responds only to the management transactions. The default value is '0'.
10	ISL	Isolate	When this bit is set, the GPCS unit is placed in the Isolation mode. When the GPCS unit is in the Isolation mode, it responds only to the management transactions. This bit has the same influence on GPCS unit power consumption as bit 11, the PowerDown bit in this register. The value is '1' after hard reset and '0' after soft reset.
9	RAN	RestartAutoNegotiation	When this bit is set, Auto-Negotiation is restarted. This bit is self-clearing. If Auto-Negotiation is disabled, then the value of this bit is always a logic '0' and any attempt to write a logic '1' to this bit is ignored. The default value is '0'.
8	DM	DuplexMode	If Auto-Negotiation is disabled (AutoNegEnable bit is cleared), this bit is used in order to configure the GPCS unit for Half or Full Duplex mode. If this bit is set, Full Duplex mode is selected. If this bit is cleared, Half Duplex mode is enabled. Please note that the value of this bit must be set to the same value as the REG_FULL_DUPLEX input. When Auto-Negotiation is enabled, this bit has no effect. The default value is '0'.
7	CT	CollisionTest	When this bit is set, the GPCS unit asserts the COL signal within 512 BT in response to TX_EN assertion, and deasserts the COL within 16 BT in response to TX_EN deassertion. The default value is '0'.
6	SS6	SpeedSelection6	This bit is read as '1'. In combination with bit 13, the SpeedSelection bit in this register, read as '0', it indicates that the GPCS unit works at a PHY speed of 1000 Mbps.
5:0		Reserved	

0x0F R accessed via EMACx_STACR[PHYD]

See *GPCS Extended Status Register (GPCSx_ESR)* on page 870.

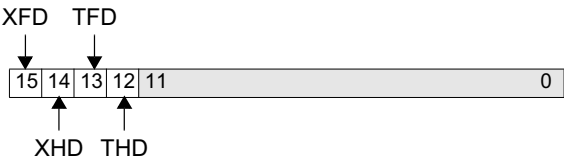


Figure 32-170. GPCS Extended Status Register (GPCSx_ESR)

15	XFD	1000BASE-X Full Duplex	When read as '1', indicates that the PHY has the ability to perform full duplex link transmission and reception. The value of this bit will be adhered to by the PHY_FD sideband signal.
14	XHD	1000BASE-X Half Duplex	When read as '1', indicates that the PHY has the ability to perform half duplex link transmission and reception. The value of this bit will be adhered to by the PHY_HD sideband signal.
13	TFD	1000BASE-T Full Duplex	Always 0
12	THD	1000BASE-T Half Duplex	Always 0
11:0		Reserved	Always 0

0x02 R accessed via EMACx_STACR[PHYD]

See GPCS ID0 Register (GPCSx_ID0) on page 863.

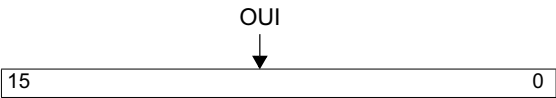


Figure 32-171. GPCS ID0 Register (GPCSx_ID0)

15:0	OUI	OUI(3:18)	The value of these bits adhere to the PHY_OUI(3:18) sideband signals. Bit 15, the OUI(3) bit of this register, corresponds to the PHY_OUI(3) sideband signal, and bit 0, the OUI(18) bit of this register, to the PHY_OUI(18) sideband signal.
------	-----	-----------	--

Preliminary User's Manual

0x03 R accessed via EMACx_STACR[PHYD]
See GPCS ID1 Register (GPCSx_ID1) on page 864.

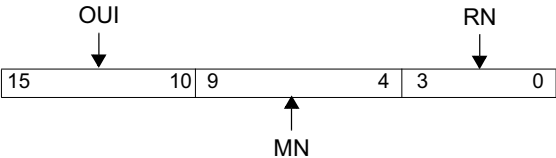


Figure 32-172. GPCS ID1 Register (GPCSx_ID1)

15:10	OUI	OUI(19:24)	The value of these bits adhere to the PHY_OUI(19:24) sideband signals. Bit 15, the OUI(19) bit of this register, corresponds to the PHY_OUI(19) sideband signal, and bit 10, the OUI(24) bit of this register, to the PHY_OUI(24) sideband signal.
9:4	MN	ModeNumber (5:0)	Manufacturer model number, bit (5:0). Bit 9 of this register corresponds to the ModeNumber(5) bit and bit 0 to the ModeNumber(0) bit. The value of these bits will adhere to the PHY_MODEL_NUMB(5:0) sideband signals.
3:0	RN	RevNumber(3:0)	Revision number, bit (3:0). Bit 3 of this register corresponds to the RevNumber(3) bit and bit 0 to the RevNumber(0) bit. The value of these bits will adhere to the sideband PHY_REV_NUMB(3:0) signals.

0x12 R/W accessed via EMACx_STACR[PHYD]

See GPCS Interrupt Status Enable Register (GPCSx_ISER) on page 872.

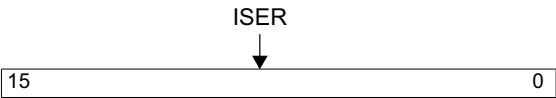


Figure 32-173. GPCS Interrupt Status Enable Register (GPCSx_ISER)

15:0	ISER	ISER	Each bit in the Interrupt Status Enable register corresponds to an associated bit in the Interrupt Status register. Status Enable bits in the Interrupt Status Enable register for reserved Status bits in the Interrupt Status Register are not implemented. These bits have no effect on write and return a logic '0' on read. Reset value '0'.
------	------	------	---

0x11 R/W accessed via EMACx_STACR[PHYD]

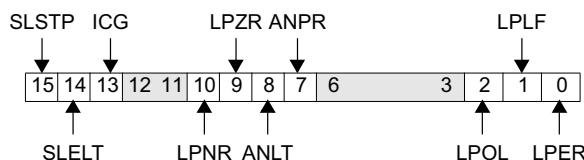
See *GPCS Interrupt Status Register (GPCSx_ISR)* on page 871.

Figure 32-174. GPCS Interrupt Status Register (GPCSx_ISR)

15	SLSTP	Synchronization Loss for Short Time Period. 0 Synchronization loss has not occurred 1 Synchronization loss has occurred	Reset value '0'.
14	SLELT	Synchronization Loss Exceeds Link Timer Period 0 Synchronization loss does not exceed link timer period 1 Synchronization loss exceeds link timer period	Reset value '0'.
13	ICG	Invalid code-group received 0 GPCS unit has not received an invalid code-group 1 GPCS unit has received an invalid code-group	When Auto-Negotiation is enabled, receiving an invalid code-group causes the GPCS unit to renegotiate. Reset value '0'.
12:11		Reserved	Always 0
10	LPNR	Link Partner No Response 0 Link partner responds to GPCS negotiation attempts 1 Link partner does not respond to GPCS negotiation attempts	Reset value '0'.
9	LPZR	Link Partner Zeroes Response 0 Link partner zeroes response duration is shorter than the GPCS unit link timer period. 1 Link partner zeroes response duration is greater than the GPCS unit link timer period.	Reset value '0'.
8	ANLT	Auto Negotiation Exceeds Link Timer Period 0 Auto negotiation does not exceed link timer period 1 Auto negotiation exceeds link timer period	When GPCS0_ISR[ANLT=1] auto-negotiation state machine is non-responsive for a time duration greater than the link timer period. Reset value '0'.
7	ANPR	Auto Negotiation Resolve Priority Error 0 Auto negotiation resolve priority error not detected 1 Auto negotiation resolve priority error detected	Auto-Negotiation Resolve Priority Error recognized by the GPCS unit. GPCS unit cannot resolve abilities with its Link Partner. Reset value '0'.
6:3		Reserved	Always 0
2	LPOL	Link Partner is off-line. 0 Link partner is not off-line 1 Link partner is off-line	Reset value '0'.
1	LPLF	Link Partner Link Failure 0 Link partner link successful 1 Link partner detected link failure	Reset value '0'.
0	LPER	Link Partner Auto Negotiation Error 0 Link partner auto negotiation error has not occurred 1 Link partner auto negotiation error has occurred	When GPCS0_ISR[LPER=1] Link Partner advertises that it cannot resolve abilities with the GPCS unit. Reset value '0'.

0x10 R accessed via EMACx_STACR[PHYD]

See GPCS Resolved Ability Register (GPCSx_RAR) on page 870.

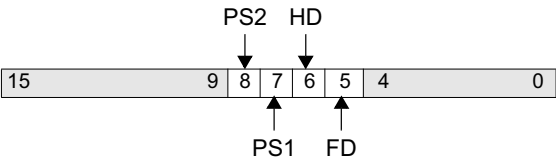


Figure 32-175. GPCS Resolved Ability Register (GPCSx_RAR)

15:9		Reserved	Always 0
8	PS2	Pause 2	This bit reflects the corresponding bit in the Link Partner ability base page register. The default value is '0'.
7	PS1	Pause 1	Pause, bit PS1. This bit reflects the corresponding bit in the Auto-Negotiation Link Partner Base Page Ability register. The default value is '0'.
6	HD	Half Duplex	If this bit is set, the local device will operate in Half Duplex mode.
5	FD	Full Duplex	If this bit is set, the local device will operate in Full Duplex mode.
4:0		Reserved	Always 0

0x01 R/W accessed via EMACx_STACR[PHYD]

See GPCS Status Register (GPCSx_SR) on page 862.

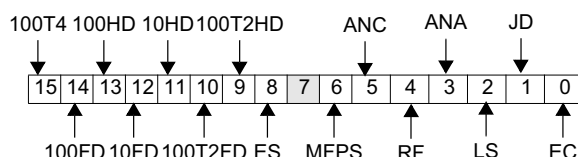


Figure 32-176. GPCS Status Register (GPCSx_SR)

15	100T4	100BASE-T4	Return '0' = PHY cannot perform 100BASE-T4.
14	100FD	100BASE-X Full Duplex	Return '0' = PHY cannot perform full duplex 100BASE-X.
13	100HD	100BASE-X Half Duplex	Return '0' = PHY cannot perform half duplex 100BASE-X.
12	10FD	10 Mbps Full Duplex	When this bit is set, Auto-Negotiation is enabled. The default value is '0'.
11	10HD	10 Mbps Half Duplex	Return '0' = PHY cannot operate at 10 Mbps in Full Duplex mode.
10	100T2FD	100BASE-T2 Full Duplex	Return '0' = PHY cannot operate at 10 Mbps in Half Duplex mode.
9	100T2HD	100BASE-T2 Half Duplex	Return '0' = PHY cannot perform full duplex 100BASE-T2.
8	ES	Extended Status	Return '1' = Extended status information in Register 15.
7		Reserved	Always '0'.
6	MFPS	MF Preamble Suppression	Return '0' = PHY does not use MII management frames.
5	ANC	Auto-Negotiation Complete	When read as '1', indicates that the Auto-Negotiation between the GPCS4 unit and its Link Partner has been completed and the contents of the following registers are valid: <ul style="list-style-type: none"> Auto-Negotiation Advertisement Register Auto-Negotiation Link Partner Base Page Ability Register Auto-Negotiation Expansion Register GPCS4 unit returns '0' in this bit if bit 12, the AutoNegEnable bit in the GPCS Control Register, is '0'. The default value is '0'.
4	RF	Remote Fault	When read as '1', indicates that remote fault condition has been detected. This bit will be set by the Auto-Negotiation block function on receipt of a base page with a non-zero Remote Fault field encoding. It will be cleared each time this register is read via the management interface or by PCS reset. In LoopBack mode, this bit is undefined. The default value is '0'.
3	ANA	Auto-Negotiation Ability	Return '1' = PHY can perform Auto-Negotiation.
2	LS	Link Status	This bit is set to a logic '1' when the <i>xmit</i> flag indicates DATA. When read as '1', indicates that a valid link has been established. The occurrence of a link failure condition clears Link Status bit.
1	JD	Jabber Detect	Always '0'
0	EC	Extended Capability	Return '1' = PHY has extended register capabilities.

MMIO 0x1 4000071C Read-Only

See *GPIO Input Register (GPIO0_IR)* on page 958.

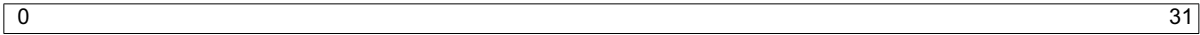


Figure 32-177. GPIO Input Register (GPIO0_IR)

0:31		GPIO register bits
------	--	--------------------

MMIO 0x1 40000718 R/W

See *GPIO Open Drain Register (GPIO0_ODR)* on page 957.



Figure 32-178. GPIO Open Drain Register (GPIO0_ODR)

0:31		GPIO0_ODR register bits
------	--	-------------------------

MMIO 0x1 40000700 R/W

See *GPIO Output Register (GPIO0_OR)* on page 956.

0	31
---	----

Figure 32-179. GPIO Output Register (GPIO0_OR)

0:31	GPIO0_OR register bits
------	------------------------

MMIO 0x1 40000704

See *GPIO Three-State Control Register (GPIO0_TCR)* on page 956.



Figure 32-180. GPIO Three-State Register (GPIO0_TCR)

0:31		GPIO0_TCR register bits
------	--	-------------------------

MMIO 0x140000A80–0x140000A98 R/

GPT Compare Timer Registers (GPT0_COMP0 - GPT0_COMP6) on page 476.

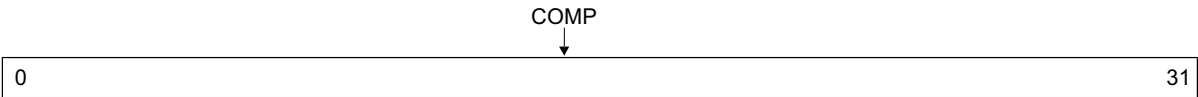


Figure 32-181. Compare Timer Register (GPT0_COMP0 - GPT0_COMP6)

0:31	COMP	Compare Timer
------	------	---------------

Preliminary User’s Manual

MMIO 0x140000A24 R/W

GPT Interrupt Enable Register (GPT0_IE) on page 475

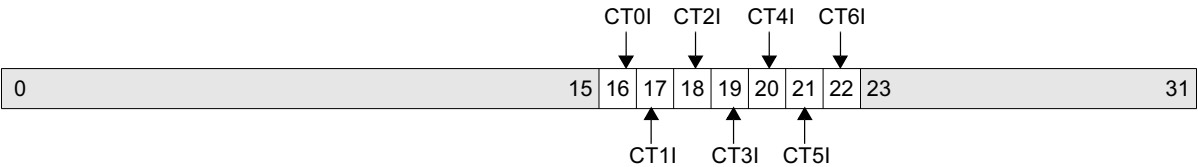
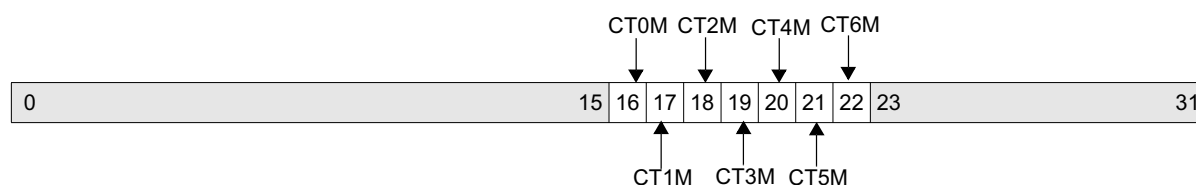


Figure 32-182. GPT Interrupt Enable Register (GPT0_IE)

0:15		Reserved
16	CT0I	Compare Timer 0 Interrupt Enable 0 Compare timer 0 interrupt enable disabled 1 Compare timer 0 interrupt enable enabled
17	CT1I	Compare Timer 1 Interrupt Enable 0 Compare timer 1 interrupt enable disabled 1 Compare timer 1 interrupt enable enabled
18	CT2I	Compare Timer 2 Interrupt Enable 0 Compare timer 2 interrupt enable disabled 1 Compare timer 2 interrupt enable enabled
19	CT3I	Compare Timer 3 Interrupt Enable 0 Compare timer 3 interrupt enable disabled 1 Compare timer 3 interrupt enable enabled
20	CT4I	Compare Timer 4 Interrupt Enable 0 Compare timer 4 interrupt enable disabled 1 Compare timer 4 interrupt enable enabled
21	CT5I	Compare Timer 5 Interrupt Enable 0 Compare timer 5 interrupt enable disabled 1 Compare timer 5 interrupt enable enabled
22	CT6I	Compare Timer 6 Interrupt Enable 0 Compare timer 6 interrupt enable disabled 1 Compare timer 6 interrupt enable enabled
23:31		Reserved

GPT0_IM

Interrupt Mask Register

Preliminary User's Manual**MMIO 0x140000A18 R/W***GPT Interrupt Mask Register (GPT0_IM) on page 472**Figure 32-183. GPT Interrupt Mask Register (GPT0_IM)*

0:15		Reserved
16	CT0M	Compare Timer 0 Interrupt Mask 0 Compare timer 0 interrupt mask disabled 1 Compare timer 0 interrupt mask enabled
17	CT1M	Compare Timer 1 Interrupt Mask 0 Compare timer 1 interrupt mask disabled 1 Compare timer 1 interrupt mask enabled
18	CT2M	Compare Timer 2 Interrupt Mask 0 Compare timer 2 interrupt mask disabled 1 Compare timer 2 interrupt mask enabled
19	CT3M	Compare Timer 3 Interrupt Mask 0 Compare timer 3 interrupt mask disabled 1 Compare timer 3 interrupt mask enabled
20	CT4M	Compare Timer 4 Interrupt Mask 0 Compare timer 4 interrupt mask disabled 1 Compare timer 4 interrupt mask enabled
21	CT5M	Compare Timer 5 Interrupt Mask 0 Compare timer 5 interrupt mask disabled 1 Compare timer 4 interrupt mask enabled
22	CT6M	Compare Timer 6 Interrupt Mask 0 Compare timer 6 interrupt mask disabled 1 Compare timer 6 interrupt mask enabled
23:31		Reserved

Preliminary User’s Manual

MMIO 0x140000A1C R/W

GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC) on page 474.

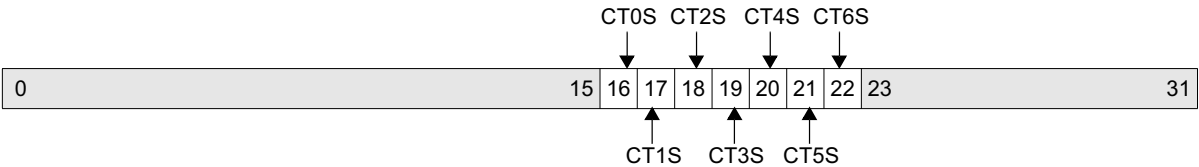


Figure 32-184. GPT Interrupt Status Register (GPT0_ISS and GPT0_ISC)

0:15		Reserved
16	CT0S	Compare Timer 0 Interrupt Status 0 Compare timer 0 interrupt status disabled 1 Compare timer 0 interrupt status enabled
17	CT1S	Compare Timer 1 Interrupt Status 0 Compare timer 1 interrupt status disabled 1 Compare timer 1 interrupt status enabled
18	CT2IS	Compare Timer 2 Interrupt Status 0 Compare timer 2 interrupt status disabled 1 Compare timer 2 interrupt status enabled
19	CT3S	Compare Timer 3 Interrupt Status 0 Compare timer 3 interrupt status disabled 1 Compare timer 3 interrupt status enabled
20	CT4S	Compare Timer 4 Interrupt Status 0 Compare timer 4 interrupt status disabled 1 Compare timer 4 interrupt status enabled
21	CT5S	Compare Timer 5 Interrupt Status 0 Compare timer 5 interrupt status disabled 1 Compare timer 5 interrupt status enabled
22	CT6S	Compare Timer 6 Interrupt Status 0 Compare timer 6 interrupt status disabled 1 Compare timer 6 interrupt status enabled
23:31		Reserved

MMIO 0x140000AC0–0x140000AD8 R/W

GPT Compare Mask Registers (GPT0_MASK0 - GPT0_MASK6) on page 476

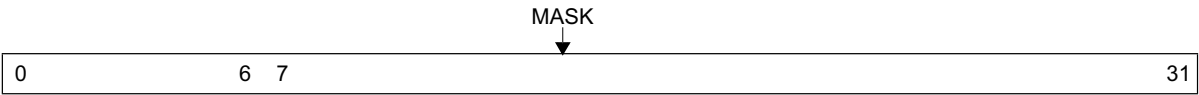


Figure 32-185. Compare Mask Register (GPT0_MASK0 - GPT0_MASK6)

0:31	MASK	Comparison Function 0 Comparison enabled 1 Comparison disabled	When set to 1, a valid comparison is assumed.
------	------	--	---

MMIO 0x140000A00 R/W

GPT Time Base Counter Register (GPT0_TBC) on page 472

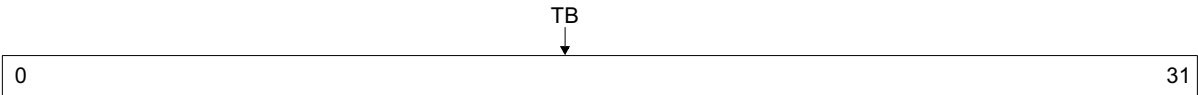


Figure 32-186. Time Base Counter Register (GPT0_TBC)

0:31	TB	Time Base
------	----	-----------

MMIO 0xFFFF0108 RW

IMU PLB Bus Error Address Register (IMU0_BEAR) on page 747



Figure 32-187. IMU PLB Bus Error Address Register (IMU0_BEAR)

0:11	QBA	Queue Base Address	The value stored in bits 0:11 can only be read, and it corresponds to the value of the Queue Base Address Low Register.
12:29	EA	Error Address	PLB address of the transfer involved in the: PLB Timeout, PLB Read Error or PLB Write Error conditions.
30:31		Reserved	Set to 00b

MMIO 0xFFFF0104 RW

IMU PLB Bus Error Masking Register (IMU0_BEMR) on page 747



Figure 32-188. IMU PLB Bus Error Masking Register (IMU0_BEMR)

0	EM	PLB Error Mask	IMU0_BEMR[EM] is used to mask the generation of the UIC2_IRQ[12] interrupt signal. If this bit is set, the interrupt is masked.
1:31		Reserved	

MMIO 0xFFFF0100 RW

IMU PLB Bus Error Status Register (IMU0_BESR) on page 746

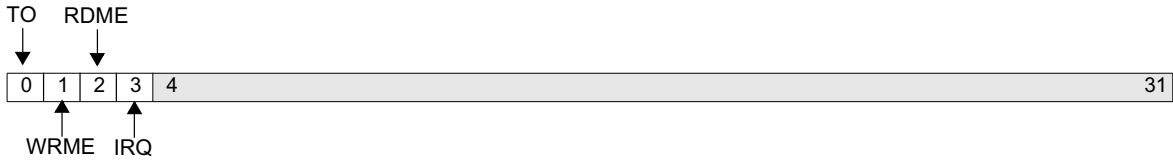


Figure 32-189. IMU PLB Bus Error Status Register (IMU0_BESR)

0	TO	PLB Timeout 0 PLB timeout cleared by using processor core read clear access 1 PLB timeout is set by using processor core read/write access	IMU0_BESR[TO] is set when a PLB timeout condition is detected in the PLB master port for a circular master transfer
1	WRME	PLB Write Master Error 0 PLB write master error cleared by using processor core read clear access 1 PLB write master error is set by using processor core read/write access	IMU0_BESR[WRME] is set when a PLB write master error condition is detected in the PLB master port for a circular master write transfer
2	RDME	PLB Read Master Error 0 PLB read master error cleared by using processor core read clear access 1 PLB read master error is set by using processor core read/write access	IMU0_BESR[RDME] is set when a PLB read master error condition is detected in the PLB master port for a circular master read transfer
3	IRQ	PLB IRQ Error 0 PLB IRQ error cleared by using processor core read clear access 1 PLB IRQ error is set by using processor core read/write access	IMU0_BESR[IRQ] is set when a PLB IRQ error condition is detected in the PLB master port for a circular master write transfer
4:31		Reserved	

MMIO 0xFFFF00D0 R/W

IMU Configuration Register (IMU0_CFG) on page 739

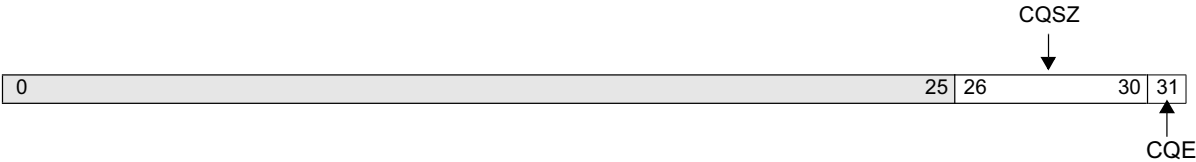


Figure 32-190. IMU Configuration Register (IMU0_CFG)

0:25		Reserved	
26:30	CQSZ	Circular Queue Size 00001b: 4K Entries 00010b: 8K Entries 00100b: 16 K Entries 01000b: 32 K Entries 10000b: 64 K Entries	
31	CQE	Circular Queue Enable 0 Circular queues disabled 1 Circular queues enabled	

MMIO 0xFFFF00A0 R/W

IMU Inbound Doorbell Register (IMU0_IDR) on page 735

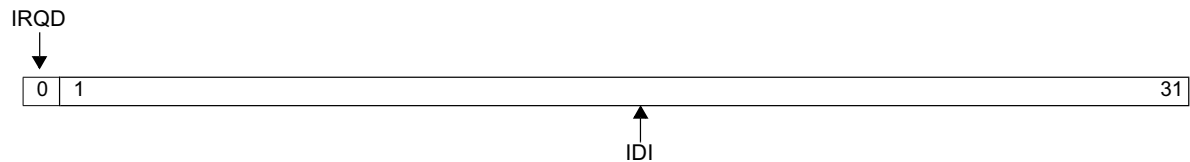


Figure 32-191. IMU Inbound Doorbell Register (IMU0_IDR)

0	IRQD	IRQ Doorbell Interrupt 0 IRQ doorbell interrupt is cleared 1 IRQ doorbell interrupt is set	When IMU0_IDR[IRQD=0] write 1 using processor core access to clear. When IMU0_IDR[IRQD=1] write 1 using device pci access to set.
1:31	IDI	Inbound Doorbell Interrupt 0 Inbound doorbell interrupt is cleared 1 Inbound doorbell interrupt is set	When IMU0_IDR[IDI=0] write 1 using processor core access to clear. When IMU0_IDR[IDI=1] write 1 using device pci access to set.

MMIO 0xFFFF00E0 R/W

IMU Inbound Free Head Pointer Register (IMU0_IFHPR) on page 742

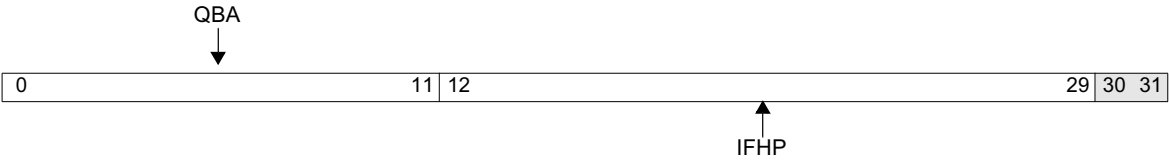


Figure 32-192. IMU Inbound Free Head Pointer Register (IMU0_IFHPR)

0:11	QBA	Queue Base Address	The value stored in bits 0:11 can only be read and correspond to the value of the Queue Base Address Low Register. Processor core access: Read Only Device on PCI access: Read Only
12:29	IFHP	Inbound Free Head Pointer	This is the offset of the pointer from the Queue Base Address Registers. Processor core access: Read/ Write Device on PCI access: Read/ Write
30:31		Reserved	Set to 00b.

MMIO 0xFFFF00E4 R/W

IMU Inbound Free Tail Pointer Register (IMU0_IFTPR) on page 742

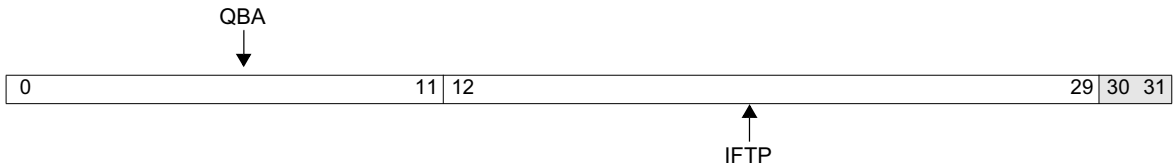


Figure 32-193. IMU Inbound Free Tail Pointer Register (IMU0_IFTPR)

0:11	QBA	Queue Base Address	The value stored in bits 0:11 can only be read, and it corresponds to the value of the Queue Base Address Low Register. Processor core access: Read Only Device on PCI access: Read Only
12:29	IFTP	Inbound Free Tail Pointer:	This is the offset of the Pointer from the Queue Base Address Registers. Processor core access: Read/ Write Device on PCI access: Read/ Write
30:31		Reserved	Set to 00b

MMIO 0xFFFF00A8 R/W

IMU Inbound Interrupt Mask Register (IMU0_IIMR) on page 736

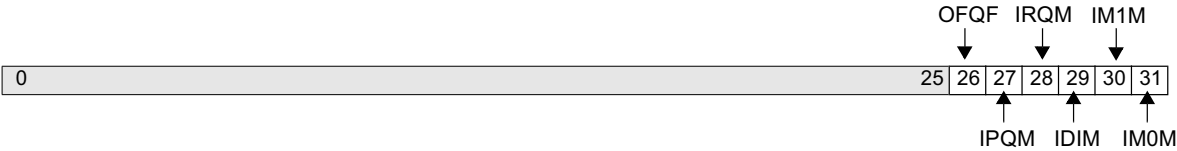
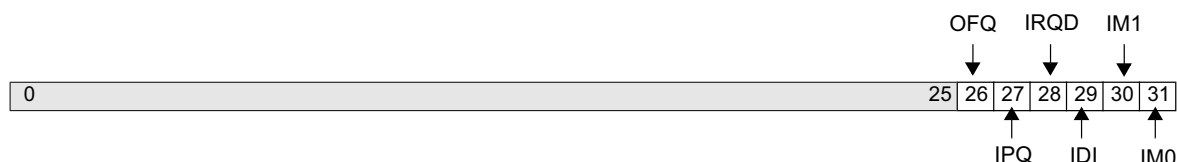


Figure 32-194. IMU Inbound Interrupt Mask Register (IMU0_IIMR)

0:25		Reserved	
26	OFQF	Outbound Free Queue Full Interrupt Mask	When set IMU0_IIMR[OFQF] masks the generation of UIC2_IRQ[6] signal.
27	IPQM	Inbound Post Queue Interrupt Mask	When set IMU0_IIMR[IPQM] masks the generation of UIC2_IRQ[7] signal.
28	IRQM	Inbound IRQ Doorbell Interrupt Mask	When set IMU0_IIMR[IRQM] masks the generation of UIC2_IRQ[8] signal.
29	IDIM	Inbound Doorbell Interrupt Mask	When set IMU0_IIMR[IDIM] masks the generation of UIC2_IRQ[9] signal.
30	IM1M	Inbound Message 1 Interrupt Mask	When set IMU0_IIMR[IM1M] masks the generation of UIC2_IRQ[11] signal.
31	IM0M	Inbound Message 0 Interrupt Mask	When set IMU0_IIMR[IM0M] masks the generation of UIC2_IRQ[10] signal.

IMU0_IISR

IMU Inbound Interrupt Status Register

Preliminary User's Manual**MMIO 0xFFFF00A4 R/W***IMU Inbound Interrupt Status Register (IMU0_IISR) on page 735**Figure 32-195. IMU Inbound Interrupt Status Register (IMU0_IISR)*

0:25		Reserved	
26	OFQ	<p>Outbound Free Queue Full Interrupt</p> <p>0 Outbound free queue full interrupt is cleared using the processor core or device on PCI access by writing 1</p> <p>1 Outbound free queue full interrupt is set when outbound free queue becomes full</p>	UIC2_IRQ[6] interrupt may be generated depending on the value of the mask bits.
27	IPQ	<p>Inbound Post Queue Interrupt</p> <p>0 Inbound post queue interrupt is cleared using processor core access</p> <p>1 Inbound post queue interrupt is set when inbound post queue is written</p>	<p>UIC2_IRQ[7] interrupt may be generated depending on the value of the mask bits.</p> <p>Note that this bit can only be cleared through processor core address access if the inbound post queue has reached the empty state. If a write cycle through the processor core address access intends to clear this bit while the inbound post queue is not empty, the write cycle will be ignored.</p> <p>Processor core access: Read/Write-to-Clear</p> <p>Device on PCI access: Read Only</p>
28	IRQD	<p>Inbound IRQ Doorbell Interrupt</p> <p>0 Inbound IRQ doorbell interrupt is cleared</p> <p>1 Inbound IRQ doorbell interrupt is set</p>	<p>UIC2_IRQ[8] interrupt may be generated depending on the value of the mask bits.</p> <p>Processor core access: Read Only</p> <p>Device on PCI access: Read Only</p>
29	IDI	<p>Inbound Doorbell Interrupt</p> <p>0 Inbound doorbell interrupt is cleared</p> <p>1 Inbound doorbell interrupt is set</p>	<p>UIC2_IRQ[9] interrupt may be generated depending on the value of the mask bits.</p> <p>Processor core access: Read Only</p> <p>Device on PCI access: Read Only</p>
30	IM1	<p>Inbound Message 1 Interrupt</p> <p>0 Inbound message 1 interrupt is cleared by writing 1 using processor core and device on pci access</p> <p>1 Inbound message 1 interrupt is set when IMU0_IMR1 is written</p>	UIC2_IRQ[11] interrupt may be generated depending on the value of the mask bits.
31	IM0	<p>Inbound Message 0 Interrupt</p> <p>0 Inbound message 0 interrupt is cleared by writing 1 using processor core and device on pci access</p> <p>1 Inbound message 0 interrupt is set when IMU0_IMR0 is written</p>	UIC2_IRQ[10] interrupt may be generated depending on the value of the mask bits.

MMIO 0xFFFF0090 IMU0_IMR0, 0xFFFF0094 IMU0_IMR1 R/W

IMU Inbound Message Registers 0:1 (IMU0_IMR0:IMU0_IMR1) on page 734

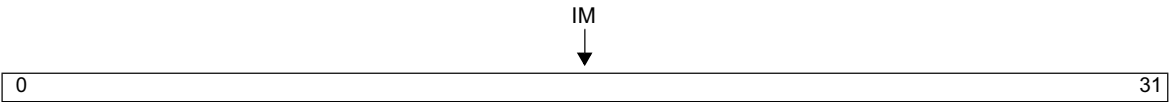


Figure 32-196. IMU Inbound Message Register (IMU0_IMR0 - IMU0_IMR1)

0:31	IM	Inbound Message	Contains a message for the PPC440GX processor core.
------	----	-----------------	---

MMIO 0xFFFF00E8 R/W

IMU Inbound Post Head Pointer Register (IMU0_IPHPR) on page 743

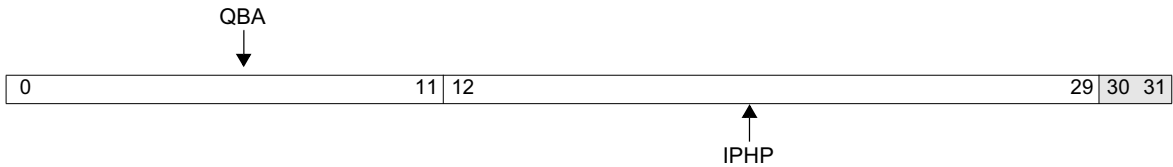


Figure 32-197. IMU Inbound Post Head Pointer Register (IMU0_IPHPR)

0:11	QBA	Queue Base Address	The value stored in bits 0:11 can only be read and corresponds to the value of the Queue Base Address Low Register. Processor core access: Read Only Device on PCI access: Read Only
12:29	IPHP	Inbound Post Head Pointer	This is the offset of the Pointer from the Queue Base Address Registers. Processor core access: Read/ Write Device on PCI access: Read/ Write
30:31			Set to 00b

MMIO 0xFFFF00EC R/W

IMU Inbound Post Tail Pointer Register (IMU0_IPTPR) on page 743

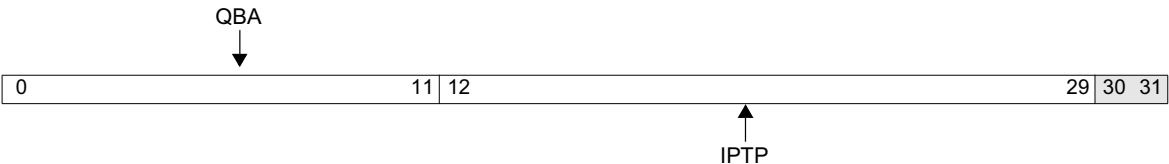


Figure 32-198. IMU Inbound Post Tail Pointer Register (IMU0_IPTPR)

0:11	QBA	Queue Base Address	The value stored in bits 0:11 can only be read, and it corresponds to the value of the Queue Base Address Low Register. Processor core access: Read Only Device on PCI access: Read Only
12:29	IPTP	Inbound Post Tail Pointer	This is the offset of the Pointer from the Queue Base Address Registers. Processor core access: Read/ Write Device on PCI access: Read/ Write
30:31		Reserved	Set to 00b

MMIO 0xFFFF0040 R/W

IMU Inbound Queue Port Register (IMU0_IQPR) on page 740



Figure 32-199. IMU Inbound Queue Port Register (IMU0_IQPR)

0:31	IQPR	
------	------	--

MMIO 0xFFFF010C R/W

IMU PLB IRQ Register (IMU0_MIRQ) on page 748



Figure 32-200. IMU PLB IRQ Register (IMU0_MIRQ)

0:7	IRQ	PLBS_IRQ	To clear any of these bits, write 1'b1 to these bits using address 10ch Access: Read/ Write-to-Clear
8:31		Reserved	

MMIO 0xFFFF00AC R/W

IMU Outbound Doorbell Register (IMU0_ODR) on page 737

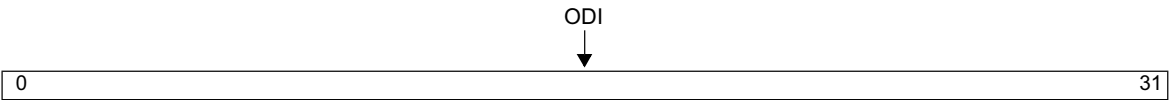


Figure 32-201. IMU Outbound Doorbell Register (IMU0_ODR)

0:31	ODI	<p>Outbound Doorbell Interrupt</p> <p>IMU0_ODR[ODI] bits are set by using processor core access</p> <p>IMU0_ODR[ODI] is cleared by using device on pci access</p> <p>When IMU0_ODR[ODI] bits are set, an interrupt signal may be asserted, depending on the mask bits.</p>
------	-----	--

MMIO 0xFFFF00F0 R/W

IMU Outbound Free Head Pointer Register (IMU0_OFHPR) on page 744



Figure 32-202. IMU Outbound Free Head Pointer Register (IMU0_OFHPR)

0:11	QBA	Queue Base Address	The value stored in bits 0:11 can only be read, and it corresponds to the value of the Queue Base Address Low Register. Processor core access: Read Only Device on PCI access: Read Only
12:29	OFHP	Outbound Free Head Pointer	This is the offset of the Pointer from the Queue Base Address Registers. Processor core access: Read/ Write Device on PCI access: Read/ Write
30:31		Reserved	Set to 00b

MMIO 0xFFFF00F4 R/W

IMU Outbound Free Tail Pointer Register (IMU0_OFTPR) on page 744

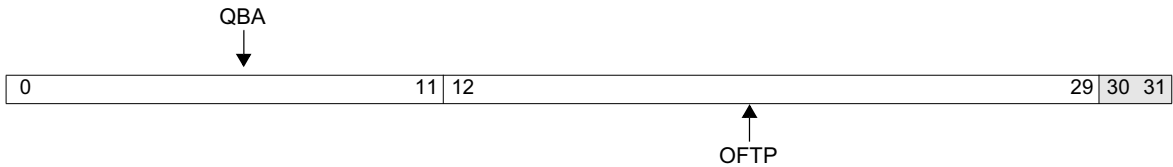


Figure 32-203. IMU Outbound Free Tail Pointer Register (IMU0_OFTPR)

0:11	QBA	Queue Base Address	The value stored in bits 0:11 can only be read, and it corresponds to the value of the Queue Base Address Low Register. Processor core access: Read Only Device on PCI access: Read Only
12:29	OFTP	Outbound Free Tail Pointer	This is the offset of the Pointer from the Queue Base Address Registers. Processor core access: Read/ Write Device on PCI access: Read/ Write
30:31		Reserved	Set to 00b

MMIO 0xFFFF00B4 R/W

IMU Outbound Interrupt Mask register (IMU0_OIMR) on page 738

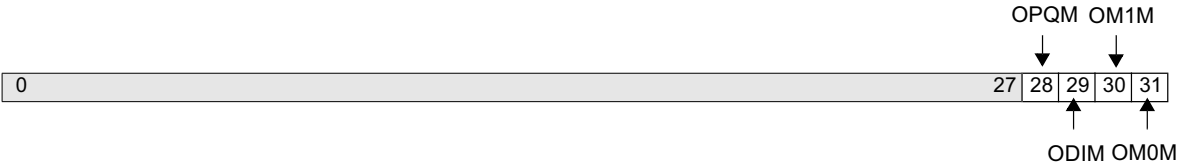


Figure 32-204. IMU Outbound Interrupt Mask Register (IMU0_OIMR)

0:27		Reserved	
28	OPQM	Outbound Post Queue Interrupt Mask	When IMU0_OIMR[OPQM] is set it masks the interrupt signal
29	ODIM	Outbound Doorbell Interrupt Mask	When IMU0_OIMR[ODIM] is set it masks the interrupt signal
30	OM1M	Outbound Message 1 Interrupt Mask	When IMU0_OIMR[OM1M] is set it masks the interrupt signal
31	OM0M	Outbound Message 0 Interrupt Mask	When IMU0_OIMR[OM0M] is set it masks the interrupt signal

MMIO 0xFFFF00B0 R/W

IMU Outbound Interrupt Status Register (IMU0_OISR) on page 737

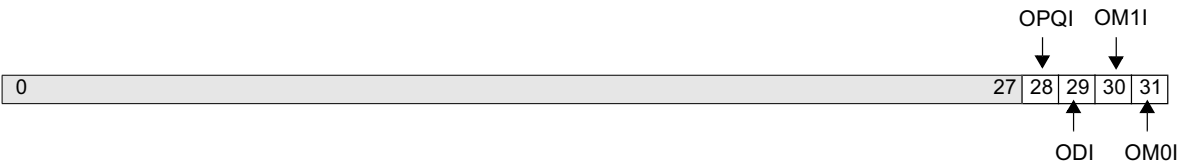


Figure 32-205. IMU Outbound Interrupt Status Register (IMU0_OISR)

0:27		Reserved	
28	OPQI	Outbound Post Queue Interrupt IMU0_OISR[OPQI] is set when valid data is loaded in the prefetch buffer. IMU0_OISR[OPQI] is cleared when data has been read from the prefetch buffer using the outbound queue port.	Processor core access: Read Only Device on PCI access: Read Only
29	ODI	Outbound Doorbell Interrupt IMU0_OISR[ODI] is set when IMU0_ODR[ODI]=1 IMU0_OISR[ODI] is cleared when IMU0_ODR[ODI]=0	Processor core access: Read Only Device on PCI access: Read Only
30	OM1I	Outbound Message 1 Interrupt 0 IMU0_OISR[OM1I] is cleared by using processor core or device on PCI access. 1 IMU0_OISR[OM1I] is set when IMU0_OMR1 is written.	
31	OM0I	Outbound Message 0 Interrupt 0 IMU0_OISR[OM0I] is cleared by using processor core or device on PCI access. 1 IMU0_OISR[OM0I] is set when IMU0_OMR0 is written.	

MMIO 0xFFFF0098 IMU0_OMR0, 0xFFFF009C IMU0_OMR1 R/W

IMU Outbound Message Registers 0:1 (IMU0_OMR0 - IMU0_OMR1) on page 734

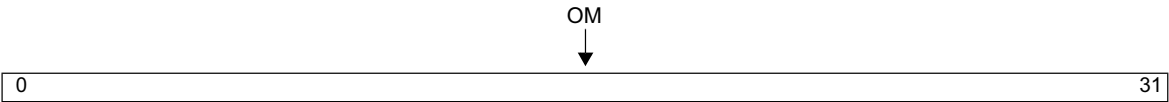


Figure 32-206. IMU Outbound Message Register (IMU0_OMR0 - IMU0_OMR1)

0:31	OM	Outbound Message	Contains a message for the IO devices.
------	----	------------------	--

MMIO 0xFFFF00F8 R/W

IMU Outbound Post Head Pointer Register (IMU0_OPHPR) on page 745

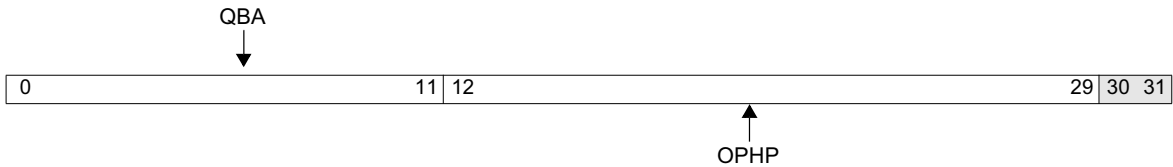


Figure 32-207. IMU Outbound Post Head Pointer Register (IMU0_OPHPR)

0:11	QBA	Queue Base Address	The value stored in bits 0:11 can only be read, and it corresponds to the value of the Queue Base Address Low Register. Processor core access: Read Only Device on PCI access: Read Only
12:29	OPHP	Outbound Post Head Pointer	The value stored in bits 0:11 can only be read, and it corresponds to the value of the Queue Base Address Registers. Processor core access: Read Only Device on PCI access: Read Only
30:31		Reserved	Set to 00b

MMIO 0xFFFF00FC R/W

IMU Outbound Post Tail Pointer Register (IMU0_OPTPR) on page 745

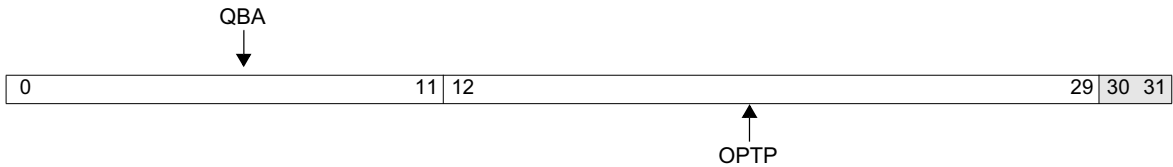


Figure 32-208. IMU Outbound Post Tail Pointer Register (IMU0_OPTPR)

0:11	QBA	Queue Base Address	The value stored in bits 0:11 can only be read, and it corresponds to the value of the Queue Base Address Low Register. Processor core access: Read Only Device on PCI access: Read Only
12:29	OPTP	Outbound Post Tail Pointer	This is the offset of the Pointer from the Queue Base Address Registers. Processor core access: Read/ Write Device on PCI access: Read/ Write
30:31		Reserved	Set to 00b

MMIO 0xFFFF0044 R/W

IMU Outbound Queue Port Register (IMU0_OQPR) on page 741



Figure 32-209. IMU Outbound Queue Port Register (IMU0_OQPR)

0:31	OQPR	
------	------	--

MMIO 0xFFFF0110 R/W

IMU PLB Programmable Register (IMU0_PPR) on page 748

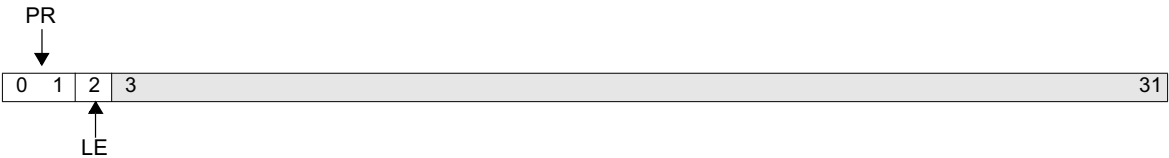


Figure 32-210. IMU PLB Programmable Register (IMU0_PPR)

0:1	PR	PLB Priority 00 Lowest 01 Highest
2	LE	PLB Lock Error
3:31		Reserved

MMIO 0xFFFF00D4 R/W

IMU Queue Base Address High Register (IMU0_QBAHR) on page 739

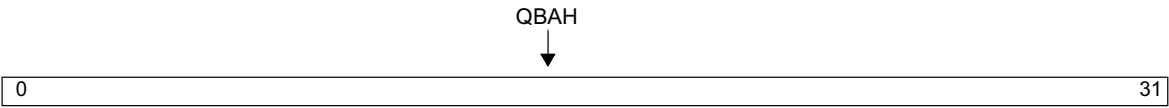


Figure 32-211. IMU Queue Base Address High Register (IMU0_QBAHR)

0:31	QBAH	Queue Base Address High
------	------	-------------------------

MMIO 0xFFFF00D8 R/W

IMU Queue Base Address Low Register (IMU0_QBALR) on page 740



Figure 32-212. IMU Queue Base Address Low Register (IMU0_QBALR)

0:11	QBA	Queue Base Address
12:31		Reserved

MMIO 0xFFFF0134 R/W

IMU Revision ID Register (IMU0_REVID) on page 749

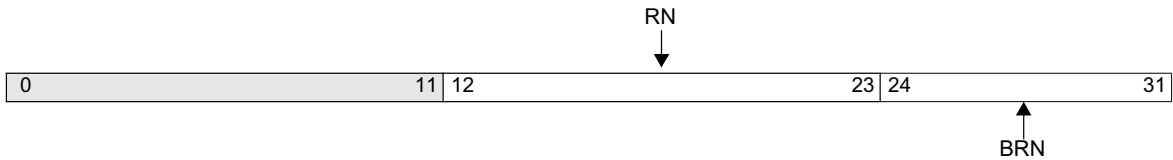


Figure 32-213. IMU Revision ID Register (IMU0_REVID)

0:11		Reserved	
12:23	RN	Revision Number	Access: read only
24:31	BRN	Branch Revision Number	Access: read only

Preliminary User’s Manual

MMIO 0xFFFF0130 R/W

IMU Sleep Register (IMU0_SLP) on page 749

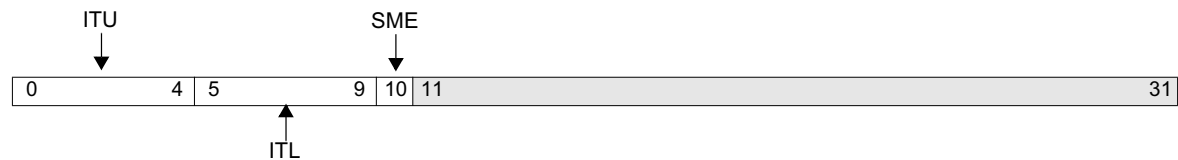


Figure 32-214. IMU PLB Sleep Register (IMU0_SLP)

0:4	ITU	IDLE Timer Upper	Upper 5 bits of the IMU IDLE timer. Access: Read/ Write
5:9	ITL	IDLE Timer Lower	Lower 5 bits of the IMU idle timer (hardcoded to 11111b). Access: Read Only
10	SME	Sleep Mode Enable 0 Sleep mode Disabled 1 Sleep Mode Enabled	Access: Read/ Write
11:31		Reserved	

MMIO 0xFFFF0138 R/W

IMU Core Status Register (IMU0_SR) on page 750



Figure 32-215. IMU Core Status Register (IMU0_SR)

0	CQLB	IMU Circular Queue Logic Busy	Access: read only
1:31		Reserved	

Preliminary User’s Manual

MMIO 0x1 4000040C IIC0, 0x1 4000050c IIC1 R/W

See IICx Clock Divide Register (IICx_CLKDIV) on page 941.

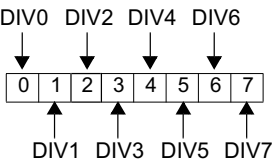


Figure 32-216. IICx Clock Divide Register (IICx_CLKDIV)

0	DIV0	Divisor bit 0
1	DIV1	Divisor bit 1
2	DIV2	Divisor bit 2
3	DIV3	Divisor bit 3
4	DIV4	Divisor bit 4
5	DIV5	Divisor bit 5
6	DIV6	Divisor bit 6
7	DIV7	Divisor bit 7

MMIO 0x1 40000406 IIC0, 0x1 40000506 IIC1 R/W

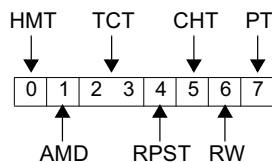
See *IICx Control Register (IICx_CNTL)* on page 932.

Figure 32-217. IICx Control Register (IICx_CNTL)

0	HMT	Halt Master Transfer 0 Normal transfer operation. 1 Issue Stop signal on the IIC bus as soon as possible to halt master transfer.	If no transfer is in progress, no action is taken. IICx_CNTL[PT] needs not be set. If IICx_MDCNTL[EINT] = 1, an interrupt is generated.
1	AMD	Addressing Mode 0 Use 7-bit addressing. 1 Use 10-bit addressing.	Does not affect slave transfers.
2:3	TCT	Transfer Count 00 Transfer one byte. 01 Transfer two bytes. 10 Transfer three bytes. 11 Transfer four bytes.	
4	RPST	Repeated Start 0 Normal start operation 1 Use repeated Start function to start transfer.	
5	CHT	Chain Transfer 0 Transfer is only or last transfer. 1 Transfer is one of a sequence of transfers (but not last in sequence).	Completion of a requested transfer causes a Stop condition to be generated on the IIC bus.
6	RW	Read/Write 0 Transfer is a write. 1 Transfer is a read.	
7	PT	Pending Transfer 0 Most recent requested transfer is complete. 1 Start transfer if bus is free.	

Preliminary User’s Manual

MMIO 0x1 40000410 IIC0, 0x1 40000510 IIC1 R/W

See IICx Direct Control Register (IICx_DIRECTCNTL) on page 947.

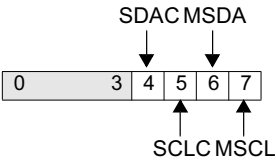


Figure 32-218. IICx Direct Control Register (IICx_DIRECTCNTL)

0:3		Reserved
4	SDAC	IICSDA Output Control Directly controls the IICSDA output. 0 IICSDA is a logic 0 1 IICSDA is a logic 1
5	SCLC	IICSCSCL Output Control Directly controls the IICSCSCL output 0 IICSCSCL is a logic 0 1 IICSCSCL is a logic 1
6	MSDA	Monitor IICSDA Used to monitor the IICSDA input 0 IICSDA is a logic 0 1 IICSDA is a logic 1 Read-only
7	MSCL	Monitor IICSCSCL. Used to monitor the IICSCSCL input. 0 IICSCSCL is a logic 0 1 IICSCSCL is a logic 1 Read-only

IICx_EXTSTS

IICxExtended Status

Preliminary User's Manual

MMIO 0x1 40000409 IIC0, 0x1 40000509 IIC1 R/W

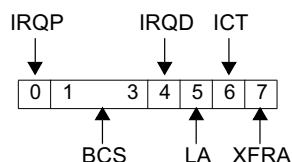
See *IICx Extended Status Register (IICx_EXTSTS)* on page 937.

Figure 32-219. IICx Extended Status Register (IICx_EXTSTS)

0	IRQP	<p>IRQ Pending</p> <p>0 No IRQ is pending.</p> <p>1 An IRQ is active, another IRQ is on-deck, and another interrupt-generating condition has occurred.</p>	<ul style="list-style-type: none"> IICx_EXTSTS[IRQP] might be set momentarily while an IRQ moves from the Pending to the On-deck state. An interrupt remains pending, IICx_EXTSTS[IRQP]=1, until the current on-deck interrupt becomes active, IICx_STS[IRQD]=0 and IICx_STS[IRQA]=1. Writing 1 to IICx_EXTSTS[IRQP] clears the field. When the IIC interrupt is disabled, IICx_MDCNTL[IRQP] = 0, IICx_EXTSTS[IRQP] should be ignored.
1:3	BCS	<p>Bus Control State</p> <p>000 Unused; if this value is read an error occurred.</p> <p>001 Slave-selected state; the IIC interface has detected and decoded a slave transfer request on the IIC bus.</p> <p>010 Slave Transfer state; the IIC interface has detected but has not decoded a slave transfer request on the IIC bus.</p> <p>011 Master Transfer state; entered after a master transfer request has started on the IIC bus.</p> <p>100 Free Bus state; the bus is free and no transfer request is pending.</p> <p>101 Busy Bus state; the bus is busy.</p> <p>110 Unknown state; value after IIC reset.</p> <p>111 Unused; if this value is read an error occurred.</p>	<p>Read-only.</p>
4	IRQD	<p>IRQ On-Deck</p> <p>0 No IRQ is on-deck.</p> <p>1 An interrupt is active, and another interrupt-generating condition has occurred.</p>	<ul style="list-style-type: none"> IICx_EXTSTS[IRQD] might be set momentarily while an IRQ moves from the On-deck to the Active state. An interrupt remains on-deck, IICx_EXTSTS[IRQD] = 1, until the current active interrupt is no longer active, IICx_STS[IRQA] = 0. If IICx_EXTSTS[IRQP] = 1, IICx_EXTSTS[IRQD] is set on the next OPB clock. Writing 1 to IICx_EXTSTS[IRQD] clears the field. When the IIC interrupt is disabled, IICx_MDCNTL[IRQP]=0, IICx_EXTSTS[IRQD] should be ignored.
5	LA	<p>Lost Arbitration</p> <p>0 Normal operation.</p> <p>1 Loss of arbitration has ended the requested master transfer.</p>	<ul style="list-style-type: none"> If arbitration is lost, any requested master transaction may have terminated prematurely. Read data may be incomplete and not all write data may have been written. If arbitration is lost during a repeat start, the master may not own the IIC bus.

6	ICT	<p>Incomplete Transfer</p> <p>0 Normal operation.</p> <p>1 Some of the bytes of the requested master transfer were not transferred.</p>	<p>For an incomplete transfer, read the transfer count, IICx_XFRCNT, to determine how bytes were transferred.</p>
7	XFRA	<p>Transfer Aborted</p> <p>0 No transfer is pending, or transfer is in progress.</p> <p>1 A requested master transfer was aborted by a NACK during the transfer of the address byte, or was aborted because arbitration was lost. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.</p>	<p>Transfer aborted. When set to a 1, a requested master transfer was aborted by a NOT acknowledge during the transfer of the address byte. It is also set to a 1 when a requested master transfer loses data. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.</p>

MMIO 0x1 40000405 IIC0, 0x1 40000505 IIC1 R/W

See IICx High Master Address Register (IICx_HMADR) on page 931.

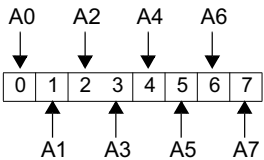


Figure 32-220. IICx High Master Address Register (IICx_HMADR)

0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

Preliminary User’s Manual

MMIO 0x1 4000040B IIC0, 0x1 4000050B IIC1 R/W

See IICx High Slave Address Register (IICx_HSADR) on page 940.

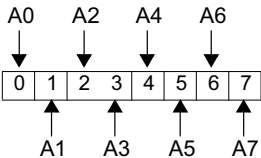


Figure 32-221. IICx High Slave Address Register (IICx_HSADR)

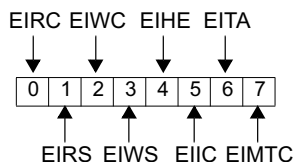
0	A0	Address bit 0	1 for 10-bit addresses
1	A1	Address bit 1	1 for 10-bit addresses
2	A2	Address bit 2	1 for 10-bit addresses
3	A3	Address bit 3	1 for 10-bit addresses
4	A4	Address bit 4	0 for 10-bit addresses
5	A5	Address bit 5	MSb for 10-bit addresses
6	A6	Address bit 6	Next to MSb for 10-bit addresses
7	A7	Address bit 7	Don't care for 10-bit addresses

IICx_INTRMSK

IICx Interrupt Mask

Preliminary User's Manual

MMIO 0x1 4000040D IIC0, 0x1 4000050D IIC1 R/W

See *IICx Interrupt Mask Register (IICx_INTRMSK)* on page 943.Figure 32-222. *IICx Interrupt Mask Register (IICx_INTRMSK)*

0	EIRC	Enable IRQ on Slave Read Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave read on the IIC bus. Note: IICx_XTCNTLSS[SR] = 1 indicates a Slave Read Complete.
1	EIRS	Enable IRQ on Slave Read Needs Service 0 Disable 1 Enable	The interrupt is activated upon receipt of a slave read on the IIC bus and the slave buffer was empty or went empty and more data was requested on the IIC bus. Note: IICx_XTCNTLSS[SRS] = 1 indicates a Slave Read Needs Service.
2	EIWC	Enable IRQ on Slave Write Complete 0 Disable 1 Enable	The interrupt is activated upon receipt of a Stop during a slave write on the IIC bus. Note: IICx_XTCNTLSS[SWC] = 1 indicates a Slave Write Complete.
3	EIWS	Enable IRQ on Slave Write Needs Service 0 Disable 1 Enable	The interrupt is activated when the slave buffer becomes full during a slave write on the IIC bus. Note: IICx_XTCNTLSS[SWS] = 1 indicates a Slave Write Needs Service.
4	EIHE	Enable IRQ on Halt Executed 0 Disable 1 Enable	
5	EIIC	Enable IRQ on Incomplete Transfer 0 Disable 1 Enable	
6	EITA	Enable IRQ on Transfer Aborted 0 Disable 1 Enable	
7	EIMTC	Enable IRQ on Requested Master Transfer Complete 0 Disable 1 Enable	

Preliminary User’s Manual

MMIO 0x1 40000404 IIC0, 0x1 40000504 IIC1 R/W
See IICx Low Master Address Register (IICx_LMADR) on page 930.

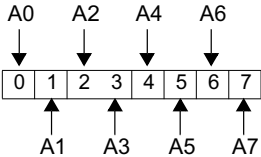


Figure 32-223. IICx Low Master Address Register (IICx_LMADR)

0	A0	Address bit 0
1	A1	Address bit 1
2	A2	Address bit 2
3	A3	Address bit 3
4	A4	Address bit 4
5	A5	Address bit 5
6	A6	Address bit 6LSb for 7-bit addresses
7	A7	Address bit 7LSb for 10-bit addresses; don't care for 7-bit addresses

MMIO 0x1 4000040A IIC0, 0x1 4000050A IIC1 R/W

See *IICx Low Slave Address Register (IICx_LSADR)* on page 939.

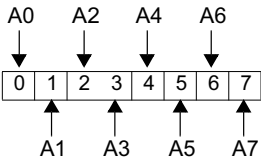


Figure 32-224. IICx Low Slave Address Register (IICx_LSADR)

0	A0	Address bit 0
1	A1	Address bit 1
2	A2	Address bit 2
3	A3	Address bit 3
4	A4	Address bit 4
5	A5	Address bit 5
6	A6	Address bit 6 LSb for 7-bit addresses
7	A7	Address bit 7 LSb for 10-bit addresses; don't care for 7-bit addresses

Preliminary User’s Manual

MMIO 0x1 40000400 IIC0, 0x1 40000500

See IICx Master Data Buffer (IICx_MDBUF) on page 928.



Figure 32-225. IICx Master Data Buffer (IICx_MDBUF)

0		Data bit
1		Data bit
2		Data bit
3		Data bit
4		Data bit
5		Data bit
6		Data bit
7		Data bit

MMIO 0x1 40000407 IIC0, 0x1 40000507 IIC1 R/W

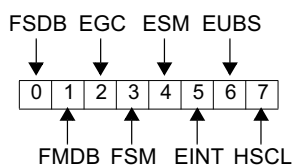
See *IICx Mode Control Register (IICx_MDCNTL)* on page 933.

Figure 32-226. IICx Mode Control Register (IICx_MDCNTL)

0	FSDB	Flush Slave Data Buffer 0 Normal operation 1 Set slave data buffer to empty.	Cleared after buffer is emptied.
1	FMDB	Flush Master Data Buffer 0 Normal operation 1 Set master data buffer to empty.	Cleared after buffer is emptied.
2	EGC	Enable General Call 0 Ignore general call on IIC bus. 1 Respond to general call on IIC bus.	IICx_MDCNTL[ESM] overrides this field; if IICx_MDCNTL[ESM] = 1, a general call is ignored.
3	FSM	Fast/Standard Mode 0 IIC transfers run at 100 kHz (standard mode). 1 IIC transfers run at 400 kHz (fast mode).	
4	ESM	Enable Slave Mode 0 Slave transfers are ignored. 1 Slave transfers are enabled.	Program IICx_LSADR and IICx_HSADR before setting this field.
5	EINT	Enable Interrupt 0 Interrupts are disabled. 1 Enables interrupts for interrupts enabled in IICx_NTRMSK.	
6	EUBS	Exit Unknown IIC Bus State 0 Normal operation. 1 IIC bus control state machine exits unknown bus state, if in an unknown state.	If the IIC bus control state machine is in a known state, setting IICx_MDCNTL[EUBS] = 1 has no effect.
7	HSCL	Hold IIC Serial Clock Low 0 If slave is not ready, issue a NACK in response to slave transfer request. 1 If slave is not ready, hold the IIC_SCL signal low until slave is ready.	This field is used only when in slave mode.

Preliminary User’s Manual

MMIO 0x1 40000402 IIC0, 0x1 40000502 IIC1 R/W

See *IICx Slave Data Buffer (IICx_SDBUF)* on page 929.



Figure 32-227. IICx Slave Data Buffer (IICx_SDBUF)

0		Data bit
1		Data bit
2		Data bit
3		Data bit
4		Data bit
5		Data bit
6		Data bit
7		Data bit

MMIO 0x1 40000408 IIC0, 0x1 40000508 IIC1 R/W

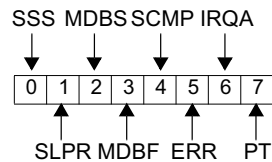
See *IICx Status Register (IICx_STS)* on page 935.

Figure 32-228. IICx Status Register (IICx_STS)

0	SSS	Slave Status Set 0 No slave operations are in progress. 1 Slave operation is in progress.	Read-only; this field is set when any of the following fields are set: IICx_XTCNTLSS[SRX, SRS, SWC, SWS].
1	SLPR	Sleep Request 0 Normal operation. 1 Sleep mode (SDR0_ER[IIC] = 1).	Read-only. The IIC interface is awakened when a start signal is detected on the IIC bus or when the SDR0_ER[IICx] is cleared.
2	MDBS	Master Data Buffer Status 0 Master data buffer is empty. 1 Master data buffer contains data.	Read-only.
3	MDBF	Master Data Buffer Full 0 Master data buffer is not full. 1 Master data buffer is full.	Read-only.
4	SCMP	Stop Complete 0 No request to halt transfer, or master data transfer, is complete. 1 Request to halt transfer, or master data transfer, is complete.	To clear IICx_STS[SCMP], set IICx_STS[SCMP] = 1.
5	ERR	Error 0 No error has occurred. 1 One of the following fields is set: IICx_EXTSTS[LA, ICT, XFRA] = 1.	Read-only.
6	IRQA	IRQ Active 0 No IIC interrupt has been sent to the universal interrupt controller (UIC). 1 An IIC interrupt has been sent to the UIC.	To clear IICx_STS[IRQA], set IICx_STS[IRQA] = 1. If IICx_MDCNTL[EINT] = 0, then IICx_STS[IRQA] is not set.
7	PT	Pending Transfer 0 No transfer is pending, or transfer is in progress. 1 Transfer is pending.	Read-only.

MMIO 0x1 4000040E IIC0 0x1 4000050E IIC1 R/W

See *IICx Transfer Count Register (IICx_XFRCNT)* on page 944.

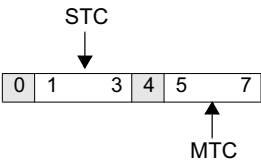


Figure 32-229. IICx Transfer Count Register (IICx_XFRCNT)

0		Reserved
1:3	STC	Slave Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved
4		Reserved
5:7	MTC	Master Transfer Count 000 0 bytes transferred 001 1 byte transferred 010 2 bytes transferred 011 3 bytes transferred 100 4 bytes transferred 101 Reserved 110 Reserved 111 Reserved

IICx_XTCNTLSS

IICx Extended Control and Slave Status

Preliminary User's Manual

MMIO 0x1 4000040F IIC0, 0x1 4000050F IIC1 R/W

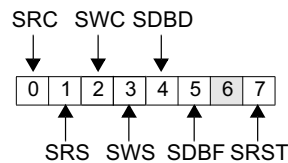
See *IICx Extended Control and Slave Status Register (IICx_XTCNTLSS)* on page 945.

Figure 32-230. IICx Extended Control and Slave Status Register (IICx_XTCNTLSS)

0	SRC	<p>Slave Read Complete</p> <p>0 Normal operation, or IICx_MDCNTL[HSCL] = 0, IICx_SDBUF is empty, and a read operation is in progress.</p> <p>1 A NACK or Stop condition was received over the IIC bus, or a repeated Start condition ended a read operation.</p>	Check whether the read operation emptied IICx_SDBUF.
1	SRS	<p>Slave Read Needs Service</p> <p>0 Normal operation or slave read does not need service.</p> <p>1 IICx_SDBUF is empty, and a read operation was requested on the IIC bus.</p> <p>The set condition may also indicate that IICx_SDBUF is empty due to a slave read and additional data is requested by the master.</p>	<p>1. If IICx_MDCNTL[HSCL]=0 and IICx_SDBUF contains no data, the slave will issue a NACK and IICx_XTCNTLSS[SRS] is set.</p> <p>2. If IICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master request additional data.</p> <p>3. If IICx_MDCNTL[HSCL]=0, and IICx_SDBUF contains no data, the slave will hold IIC_SCL low to indicate the slave is busy. IICx_XTCNTLSS[SRS] is set until the IICx_SDBUF is filled. Once filled, IIC_SCL is released, IICx_XTCNTLSS[SRS] is cleared, and the slave sends the data.</p> <p>4. If IICx_MDCNTL[HSCL]=1, and IICx_SDBUF contains data, the slave will send the data. IICx_XTCNTLSS[SRS] is not set unless the master requests additional data.</p>
2	SWC	<p>Slave Write Complete</p> <p>0 Normal operation or slave write in progress.</p> <p>1 A Stop signal was received during a write operation, or a repeated Start condition ended a write operation.</p>	
3	SWS	<p>Slave Write Needs Service</p> <p>0 Normal operation or slave write does not need service.</p> <p>1 IICx_SDBUF is full during a slave write.</p>	<p>1. If IICx_MDCNTL[HSCL] = 1 and IICx_SDBUF is full, the slave will hold IIC_SCL low to indicate the slave is busy. IICx_XTCNTLSS[SWS] is set until IICx_SDBUF is empty. Once empty, IIC_SCL is released, IICx_XTCNTLSS[SWS] is cleared, and the slave receives the data.</p> <p>2. If IICx_MDCNTL[HSCL] = 0 and IICx_SDBUF is full, the slave will issue a NACK and IICx_XTCNTLSS[SWS] is set.</p>
4	SDBD	<p>Slave Data Buffer Has Data</p> <p>0 IICx_SDBUF is empty</p> <p>1 IICx_SDBUF contains data</p>	Read-only
5	SDBF	<p>Slave Data Buffer Full</p> <p>0 IICx_SDBUF is not full</p> <p>1 IICx_SDBUF is full</p>	Read-only
6		Reserved	
7	SRST	<p>Soft Reset</p> <p>0 Normal operation</p> <p>1 Soft reset</p>	

DCR 0x032 R/W

L2 Cache Address Register (L2C0_ADDR) on page 333



Figure 32-231. L2 Cache Address Register (L2C0_ADDR)

0:31	ADDR	Address
------	------	---------

DCR 0x030 R/W

L2 Cache Configuration Register (L2C0_CFG) on page 329

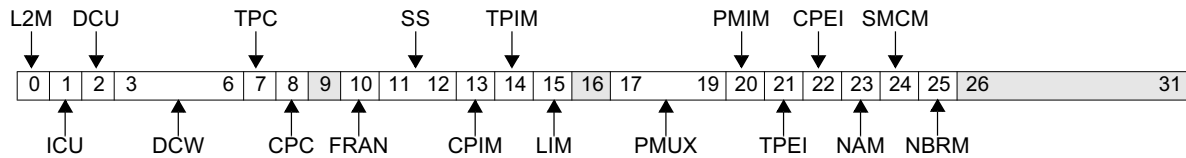


Figure 32-232. L2 Cache Configuration Register (L2C0_CFG)

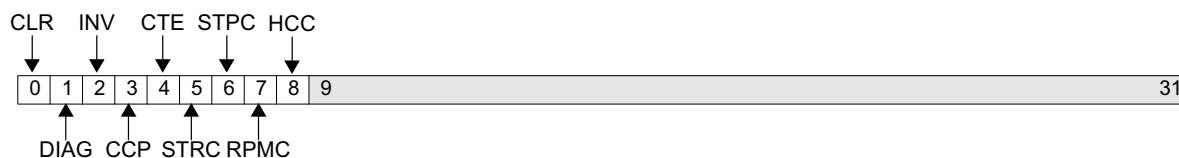
0	L2M	L2 Mode 0 Set SRAMs for SRAM core use 1 Set SRAM for L2 use
1	ICU	Instruction cache unit enabled 0 Instruction cache unit does not use L2 1 Instruction cache unit enabled to use L2
2	DCU	Data cache unit enabled 0 Data cache unit does not use L2 1 Data cache unit enabled to use L2
3:6	DCW	Disable cache way 0000 Enable ways 0, 1, 2 and 3 0001 Enable ways 0, 1, and 2 0011 Enable ways 0 and 1 0111 Enable way 0 All other combinations reserved
7	TPC	Tag Parity Check Enable 0 Tag parity check disabled 1 Tag parity check enabled
8	CPC	Cache Parity Check Enable 0 Cache parity check disabled 1 Cache parity check enabled
9		Reserved
10	FRAN	Fast Read Acknowledge Enable 0 Fast read acknowledge disabled 1 Fast read acknowledge enabled For best L2 performance set L2C0_CFG[FRAN] = 1.
11:12	SS	SRAM Size 00 256KB 01 Reserved 10 Reserved 11 Reserved PPC440GX Embedded Processor supports 256KB SRAM
13	CPIM	Cache Parity Error Interrupt Mask 0 Cache parity error interrupt disabled 1 Cache parity error interrupt enabled
14	TPIM	Tag Parity Error Interrupt Mask 0 Tag parity error interrupt disabled 1 Tag parity error interrupt enabled
15	LIM	LRU Code Point Error Interrupt Mask 0 LRU code point error interrupt disabled 1 LRU code point error interrupt enabled
16		Reserved

Preliminary User's Manual

17:19	PMUX	Performance Monitoring Mask Control 000 Monitor snooping 001 Monitor I-fetching 010 Monitor D-fetching 011 Monitor D-storing 100 Reserved 101 Reserved 110 Reserved 111 Reserved	
20	PMIM	Performance Monitor Interrupt Mask 0 Performance monitor interrupt disabled 1 Performance monitor interrupt enabled	
21	TPEI	Tag Parity Error Inject 0 Tag parity error inject disabled 1 Tag parity error inject enabled	
22	CPEI	Cache Parity Error Inject 0 Cache parity error inject disabled 1 Cache parity error inject enabled	
23	NAM	No Abort Mode 0 No abort mode disabled 1 No abort mode enabled	
24	SMCM	Self Modifying Code Mode 0 Self modifying code mode disabled 1 Self modifying code mode enabled	
25	NBRM	No Block Request Mode 0 No block request mode disabled 1 No block request mode enabled	See above for additional information
26:31		Reserved	

L2C0_CMD

L2 Cache Command Register

Preliminary User's Manual**DCR 0x031 R/W***L2 Cache Command Register (L2C0_CMD) on page 331**Figure 32-233. L2 Cache Command Register (L2C0_CMD)*

0	CLR	Clear Command 0 Clear command disabled 1 Clear command enabled	L2C0_ADDR bits 16:26 are used to specify the tag indexed to be cleared. Setting L2C0_CMD[CLR] to 1 causes all ways at the specified index to be set invalid with good parity. The LRU bits are also cleared.
1	DIAG	Diagnostic Class of Commands 0 Diagnostic class of commands disabled 1 Diagnostic class of commands enabled	For diagnostic command description see <i>L2 Cache Address Register (L2C0_ADDR)</i> on page 333.
2	INV	Invalidate Command 0 Invalidate command disabled 1 Invalidate command enabled	This command will invalidate the address specified in the L2C0_ADDR register.
3	CCP	Clear Cache Parity Error 0 Clear cache parity error disabled 1 Clear cache parity error enabled	This command will reset the cache trap address and way registers within the L2 such that they can trap a new error.
4	CTE	Clear Tag Error 0 Clear tag error disabled 1 Clear tag error enabled	This command will reset the tag trap address and way registers within the L2 such that they can trap a new error.
5	STRC	Start Performance Monitor Counters 0 Start performance monitor counters disabled 1 Start performance monitor counters enabled	
6	STPC	Stop Performance Monitor Counters 0 Stop performance monitor counters disabled 1 Stop performance monitor counters enabled	
7	RPMC	Reset Performance Monitor Counters 0 Reset performance monitor counters disabled 1 Reset performance monitor counters enabled	
8	HCC	Hardware Clear Command 0 Hardware clear command disabled 1 Hardware clear command enabled	Setting L2C0_CFG[HCC = 1] causes all indexes and ways of the cache to be cleared by hardware. Before issuing this command, the L2C0_ADDR field must be set to 0.
9:31		Reserved	

Preliminary User's Manual

DCR 0x033 R/only

L2 Cache Data Register (L2C0_DATA) on page 334

0	31
---	----

Figure 32-234. L2 Cache Data Register (L2C0_DATA)

0:31	DATA	Data
------	------	------

DCR 0x035 R/only

L2 Cache Revision ID Register (L2C0_REVID) on page 335



Figure 32-235. L2 Cache Revision ID Register (L2C0_REVID)

0:31	REVID	Core Revision ID
------	-------	------------------

Preliminary User’s Manual

DCR 0x036:0x037 R/W

L2 Cache Snoop Register 0:1 (L2C0_SNP0:LC20_SNP1) on page 336

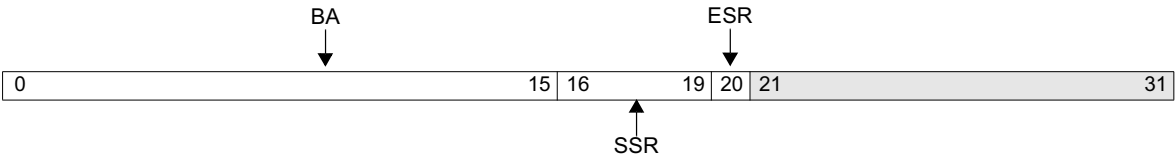


Figure 32-236. L2 Cache Snoop Register 0:1 (L2C0_SNP0:L2C0_SNP1)

0:15	BA	Base Address	L2C0_SNP0:L2C0_SNP1[0:15] specifies the starting address of the snooping region where L2C0_SNP0:L2C0_SNP1[0:3] refers to PLB upper address bits 28:31 and L2C0_SNP0:L2C0_SNP1[4:15] refers to PLB address bits 0:11.
16:19	SSR	Size of Snooping Region 0000 1MB 0001 2MB 0010 4MB 0011 8MB 0100 16MB 0101 32MB 0110 64MB 0111 128MB 1000 256MB 1001 512MB 1010 1GB 1011 2GB 1100 4GB 1101 8GB 1110 16GB 1111 32GB	
20	ESR	Enable Snoop Region 0 Snoop region disabled 1 Snoop region enabled	
21:31		Reserved	

DCR 0x034 R/only

L2 Cache Status Register (L2C0_SR) on page 335

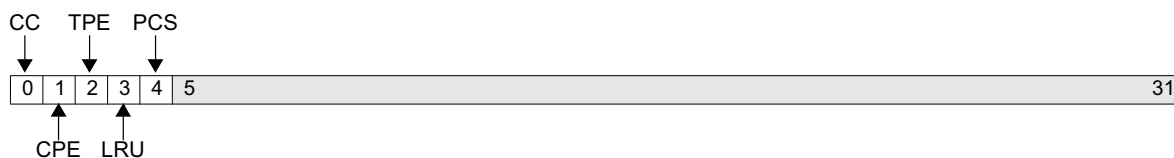


Figure 32-237. L2 Cache Status Register (L2C0_SR)

0	CC	Command Complete 0 Command issued in L2_CMD not complete 1 Command issued in L2_CMD complete	When L2C0_CMD[CLR], L2C0_CMD[HCC], L2C0_CMD[DIAG], or L2C0_CMD[INV] is issued, L2C0_SR[CC] is cleared and then set when the command is complete.
1	CPE	Cache Parity Error 0 No cache parity error present 1 Cache parity error present	If L2C0_SR[CPE] = 1 and error is unmasked, an interrupt will be presented
2	TPE	Tag Parity Error 0 No tag parity error present 1 Tag parity error present	If L2C0_SR[TPE] = 1 and error is unmasked, an interrupt will be presented
3	LRU	LRU Code Point Error 0 No LRU error present 1 LRU error present	If L2C0_SR[LRU] = 1 and error is unmasked, an interrupt will be presented
4	PCS	Performance Counter Stopped 0 Performance counter is running 1 Performance counter is stopped	If L2C0_SR[PCS] = 1 and condition is unmasked, an interrupt will be presented
5:31		Reserved	

DCR 0x180 Read/Write

See *MAL Configuration Register (MAL0_CFG)* on page 776.

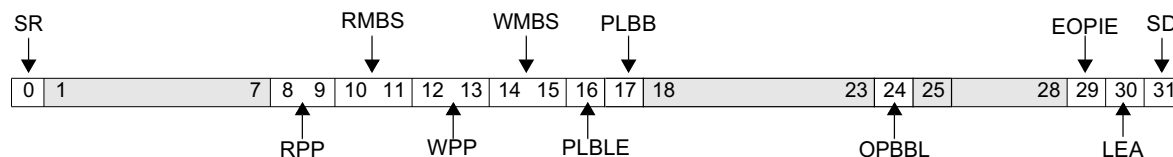


Figure 32-238. MAL Configuration Register (MAL0_CFG)

0	SR	MAL Software Reset 0 MAL reset is complete 1 Reset the MAL	Generates a general reset to MAL through a software command. After setting this bit, MAL hardware (registers, interface and internal state machines) returns to the power-on reset value. The software writes 1 to this bit in order to drive MAL to the reset state. The bit is cleared by the hardware when the reset is completed (one system clock).
1:7		Reserved	
8:9	RPP	Read PLB Priority 00 Lowest 01 10 11 Highest	Determines the priority of MAL requests on the PLB for read transactions.
10:11	RMBS	Read Max Burst Size 00 Max burst size of 4 01 Max burst size of 8 10 Max burst size of 16 11 Max burst size of 32 (recommended)	Maximum PLB burst size for read transactions. Determines the maximum data transfers (RdDacks) per read burst transaction.
12:13	WPP	Write PLB Priority 00 Lowest 01 10 11 Highest	Determines the priority of MAL requests on the PLB for write transactions.
14:15	WMBS	Write Max Burst Size 00 Max burst size of 4 01 Max burst size of 8 10 Max burst size of 16 11 Max burst size of 32 (recommended)	Maximum PLB burst size for write transactions. Determines the maximum data transfers (WrDacks) per write burst transaction.
16	PLBLE	PLB Lock Error 0 LOCKERROR signal not applied to the PLB slave 1 LOCKERROR signal applied to the PLB slave	When this bit is set, MAL applies the LOCKERROR signal to the PLB slave when it is the initiator during PLB transactions.
17	PLBB	PLB Burst 0 Burst transactions not allowed 1 Burst transactions allowed	When this bit is reset, MAL is not allowed to perform burst transactions.
18:23		Reserved	
24	OPBBL	OPB Bus Lock 0 OPB not locked 1 OPB locked	When this bit is set, MAL locks the OPB during data transfers to and from the COMMAs.
25:28		Reserved	

29	EOPIE	<p>End of Packet Interrupt Enable</p> <p>0 Generate interrupt on every end-of-packet only if the buffers I bit is set</p> <p>1 Generate interrupt is on every end-of-packet</p>	<p>When this bit is set, an interrupt is generated on every end of packet (both transmit and receive).</p> <p>When clear, end of packet/buffer interrupt is generated only if the buffers I bit is set (1).</p> <p>Note: An interrupt is generated for every descriptor on which the I bit is set, regardless of the state of the EOPIE bit.</p>
30	LEA	<p>Locked Error Active</p> <p>0 Handle errors in a non-locked mode</p> <p>1 Handle errors in locked mode</p>	<p>Determines MAL's error handling mode. When this bit is set, MAL will handle errors in the locked mode, otherwise it will handle errors in a non-locked mode.</p>
31	SD	<p>MAL Scroll Descriptor</p> <p>0 Do not scroll to the first descriptor of the next packet</p> <p>1 Scroll to the first descriptor of the next packet</p>	<p>Determines whether or not MAL should scroll to the first descriptor of the next packet, following an early packet termination initiated by the related COMMAC. When set, Scrolling mode is active.</p>

DCR 0x181 Read/Clear

See *MAL Error Status Register (MAL0_ESR)* on page 780.

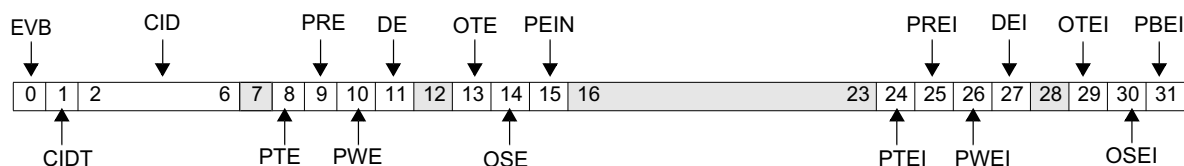


Figure 32-239. MAL Error Status Register (MAL0_ESR)

0	EVB	<p>Error Valid Bit</p> <p>0 Bit 1:15 are available for latching new error information.</p> <p>1 Bits 1:15 contain last error. A new error cannot be latched.</p>	<p>When this bit is set, bits 1-6 include the ID of the erroneous channel (in case of OPB errors). Bits 11-15 indicate the type of error.</p> <p>In non-locked mode, the error indication describes the last error that had occurred. In locked mode, the error is the first one that had occurred after this bit was cleared.</p> <p>This bit is set when an error occurs and remains set until reset by the software. In locked mode, new errors cannot be latched in the error lock indication fields if this bit is set</p>
1	CIDT	<p>Channel ID Type</p> <p>0 Channel ID represents TX channel</p> <p>1 Channel ID represents RX channel</p>	
2:6	CID	Channel ID Number	<p>Indicates the number of the channel that caused the error.</p> <p>Note: An error on the PLB cannot be related to a channel. The error condition may be resolved by using the error information optionally locked in the PLB slave.</p>
7		Reserved	
8	PTE	<p>PLB Timeout Error</p> <p>0 No error</p> <p>1 PLB timeout error has occurred</p>	Indicates the error is a PLB timeout
9	PRE	<p>PLB Read Error</p> <p>0 No error</p> <p>1 PLB read error has occurred</p>	
10	PWE	<p>PLB Write Error</p> <p>0 No error</p> <p>1 PLB write error has occurred</p>	
11	DE	<p>Descriptor Error</p> <p>0 No error</p> <p>1 Non-valid descriptor</p>	Indicates that the error is a non-valid descriptor, which is <i>not</i> the first descriptor in a TX packet.
12		Reserved	
13	OTE	<p>OPB Timeout Error</p> <p>0 No error</p> <p>1 OPB timeout error has occurred</p>	Indicates the error is an OPB timeout.
14	OSE	<p>OPB Slave Error</p> <p>0 No error</p> <p>1 OPB slave error occurred</p>	Indicates the error is an error indication asserted by an OPB slave.
15	PEIN	<p>PLB Bus Error Indication</p> <p>0 No error</p> <p>1 PLB bus error occurred</p>	When this bit is set, the detected error is a PLB error. There is no meaning to the Channel ID field in this case.
16:23		Reserved	
24	PTEI	<p>PLB Timeout Error Interrupt</p> <p>0 No error</p> <p>1 PLB timeout error occurred</p>	This bit is set following a PLB timeout error indication. Set condition for this bit generates a maskable interrupt.

25	PREI	PLB Read Error Interrupt 0 No error 1 PLB read error occurred	This bit is set following a PLB read error indication. Set condition for this bit generates a maskable interrupt.
26	PWEI	PLB Write Error Interrupt 0 No error 1 PLB write error occurred	This bit is set following a PLB write error indication. Set condition for this bit generates a maskable interrupt.
27	DEI	Descriptor Error Interrupt 0 No error 1 Descriptor data error recognized	A descriptor data error is recognized during access to the descriptor table. This error indication is asserted when a non-valid descriptor is accessed, which is <i>not</i> the first descriptor in a TX packet. Set condition for this bit generates a maskable interrupt.
28		Reserved	
29	OTEI	OPB Timeout Error Interrupt 0 No error 1 OPB time-out	This bit is set following an OPB time out error indication. Set condition for this bit generates a maskable interrupt.
30	OSEI	OPB Slave Error Interrupt 0 No error 1 OPB error from a slave	This bit is set following an OPB error indicated by the slave. Set condition for this bit generates a maskable interrupt.
31	PBEI	PLB Bus Error Interrupt 0 No error 1 PLB error indication	This bit is set following a PLB error indication (from the PLB slave). Set condition for this bit generates a maskable interrupt.

DCR 0x182 Read/Write

See *MAL Interrupt Enable Register (MAL0_IER)* on page 783.

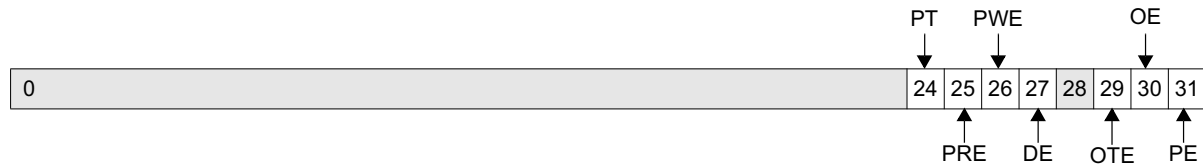


Figure 32-240. MAL Interrupt Enable Register (MAL0_IER)

0:23		Reserved
24	PT	PLB Timeout Interrupt 0 PLB timeout interrupt disabled 1 PLB timeout interrupt enabled
25	PRE	PLB Read Error Interrupt 0 PLB read error interrupt disabled 1 PLB read error interrupt enabled
26	PWE	PLB Write Error Interrupt 0 PLB write error interrupt disabled 1 PLB write error interrupt enabled
27	DE	Descriptor Error Interrupt 0 Descriptor error interrupt disabled 1 Descriptor error interrupt enabled
28		Reserved
29	OTE	OPB Timeout Error Interrupt 0 OPB timeout error interrupt disabled 1 OPB timeout error interrupt enabled
30	OE	OPB Error Interrupt 0 OPB slave error interrupt disabled 0 OPB slave error interrupt enabled
31	PE	PLB Error Interrupt 0 PLB error interrupt disabled 1 PLB error interrupt enabled

DCR 0x1E0 - 0x1E1 Read/Write

See *RX Channel Buffer Size Register (MAL0_RCBSx)* on page 788.

0	23	24	31
---	----	----	----

Figure 32-241. RX Channel Buffer Size Register (MAL0_RCBSx)

0:23		Reserved
24:31		Receive Channel Buffer Size

DCR 0x195 Read/Write

See *Descriptor Base Address Registers (MAL0_TXBADDR, MAL0_RXBADDR)* on page 785.



Figure 32-242. RX Descriptor Base Address Register (MAL0_RXBADDR)

0:27		Reserved
28:31	DBA	Descriptor Base Address

DCR 0x191 Read/Write

See *Channel-Active Set and Channel-Active Reset registers* on page 777.



Figure 32-243. RX Channel_Active Reset Register (MAL0_RXCARR)

0:3	RCAR	Receive Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, and so on). When 0 is written to the bit, channel operation is disabled. MAL0 has four receive channels.
4:31		Reserved	

DCR 0x190 Read/Write

See *Channel-Active Set and Channel-Active Reset registers* on page 777.



Figure 32-244. RX Channel_Active Set Register (MAL0_RXCASR)

0:3	RCAS	Receive Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has four receive channels.
4:31		Reserved	

DCR 0x1C0 MAL0_RXCTP0R, 0x1C1 MAL0RXCTP1R Read/Write

See *Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTPxR)* on page 787.



Figure 32-245. RX Channel Table Pointer Register (MAL0_RXCTPxR)

0:31		Channel Table Pointer	Pointer to the base address of the buffer descriptor table used by the channel. The value should point to a location in memory accommodating an aligned doubleword (the three least significant bits of the pointer must be 000). MAL0 has four receive channels.
------	--	-----------------------	--

DCR 0x193 Read/Clear

See *Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)* on page 783.



Figure 32-246. RX Descriptor Error Interrupt Register (MAL0_RXDEIR)

0:3	RXDE	Receive Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). When one or more bits are set, MAL RXDE interrupt is set. Writing 1 to a bit clears it. MAL0 has four receive channels.
4:31		Reserved	

DCR 0x192 Read/Clear

See *End of Buffer Interrupt Status Registers* on page 779.



Figure 32-247. RX End of Buffer Interrupt Status Register (MAL0_RXEOBISR)

0:3	RCEI	Receive Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). Writing 1 to a bit clears it. MAL0 has four receive channels.
4:31		Reserved	

Preliminary User’s Manual

DCR 0x194 Read/Write

See *PLB Storage Attribute Registers (MAL0_TXTATTRR, MAL0_RXTATTRR)* on page 785.

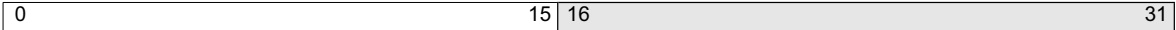


Figure 32-248. RX PLB Attribute Register (MAL0_RXTATTRR)

0:15		RX PLB Storage Attributes
16:31		Reserved

DCR 0x189 Read/Write

See *Descriptor Base Address Registers (MAL0_TXBADDR, MAL0_RXBADDR)* on page 785.



Figure 32-249. TX Descriptor Base Address Register (MAL0_TXBADDR)

0:27		Reserved
28:31	DBA	Descriptor Base Address

DCR 0x185 Read/Write

See *Channel-Active Set and Channel-Active Reset registers* on page 777.



Figure 32-250. TX Channel Active Reset Register (MAL0_TXCARR)

0:3	TCAR	Transmit Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is disabled. MAL0 has four transmit channels.
4:31		Reserved	

DCR 0x184 Read/Write

See *Channel-Active Set and Channel-Active Reset registers* on page 777.



Figure 32-251. TX Channel_Active Set Register (MAL0_TXCASR)

0:3	TCAS	Transmit Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, and so on). When 1 is written to the bit, channel operation is enabled. MAL0 has four transmit channels.
4:31		Reserved	

DCR 0x1A0–0x1A3 Read/Write

See *Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTPxR)* on page 787.

0	31
---	----

Figure 32-252. TX Channel Table Pointer Register (MAL0_TXCTPxR)

0:31		Channel Table Pointer	Pointer to the base address of the buffer descriptor table used by the channel. The value entered should point to a location in memory accommodating an aligned doubleword (the three least significant bits of the pointer must be 000). MAL0 has four transmit channels.
------	--	-----------------------	---

DCR 0x187 Read/Clear

See *Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)* on page 783.



Figure 32-253. TX Descriptor Error Interrupt Register (MAL0_TXDEIR)

0:3	TXDE	Transmit Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). When one or more bits are set to 1, MAL TXDE interrupt is set. Writing 1 to a bit clears it. MAL0 has four transmit channels.
4:31		Reserved	

Preliminary User’s Manual

DCR 0x186 Read/Clear

See *End of Buffer Interrupt Status Registers* on page 779.



Figure 32-254. TX End of Buffer Interrupt Status Register (MAL0_TXEOBISR)

0:3	TCEI	Transmit Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, and so on). Writing 1 to a bit clears it. MAL0 has four transmit channels.
4:31		Reserved	

DCR 0x188 Read/Write

See *PLB Storage Attribute Registers (MAL0_TXTATTRR, MAL0_RXTATTRR)* on page 785.

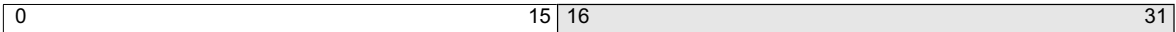


Figure 32-255. TX PLB Attribute Register (MAL0_TXTATTRR)

0:15		TX PLB Storage Attributes
16:31		Reserved

Preliminary User’s Manual

MMIO 0x140000601 Read/Write

See OPB Arbiter Control Register (OPBA0_CR) on page 97.

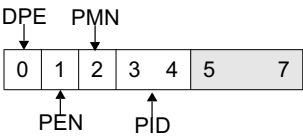


Figure 32-256. OPB Arbiter Control Register (OPBA0_CR)

0	DPE	Dynamic Priority Enable 0 Dynamic priority disabled 1 Dynamic priority enabled
1	PEN	Park Enable 0 Park disabled 1 Park enabled
2	PMN	Park on Master Not Last 0 Park on master last 1 Park on master not last
3:4	PID	Parked Master ID 00 Master ID 0 of OPB master device connected to M0_request and OPB_M0Grant 01 Master ID 1 of OPB master device connected to M1_request and OPB_M1GrantReserved 10 Master ID 2 of OPB master device connected to M2_request and OPB_M2Grant 11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved Master 0 is the PLB to OPB bridge controller; master 1 is the external bus master interface, and master 2 is the DMA controller. Master 3 is unused.
5:7		Reserved

MMIO 0x14000600 Read/Write

See *OPB Arbiter Priority Register (OPBA0_PR)* on page 96.

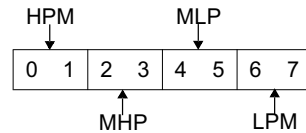


Figure 32-257. OPB Arbiter Priority Register (OPBA0_PR)

0:1	HPM	<p>High Priority Master ID</p> <p>00 Master ID 0 of OPB master device connected to M0_request and OPB_M0Grant</p> <p>01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved</p> <p>10 Master ID of OPB master device connected to M2_request and OPB_M2GrantReserved</p> <p>11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved</p>	At reset, this priority is assigned to PLB to OPB bridge.
2:3	MHP	<p>Medium High Priority master ID</p> <p>00 Master ID of OPB master device connected to M0_request and OPB_M0Grant</p> <p>01 Master ID of OPB master device connected to M1_request and OPB_M1Grant</p> <p>10 Master ID of OPB master device connected to M2_request and OPB_M2Grant</p> <p>11 Master ID of OPB master device connected to M3_request and OPB_M3Grant</p>	At reset, this priority is assigned to the external bus master interface.
4:5	MLP	<p>Medium Low Priority master ID</p> <p>00 Master ID of OPB master device connected to M0_request and OPB_M0GrantReserved</p> <p>01 Master ID of OPB master device connected to M1_request and OPB_M1GrantReserved</p> <p>10 Master ID 2 of OPB master device connected to M2_request and OPB_M2Grant</p> <p>11 Master ID of OPB master device connected to M3_request and OPB_M3GrantReserved</p>	At reset, this priority is assigned to DMA
6:7	LPM	<p>Low Priority Master ID</p> <p>00 Master ID of OPB master device connected to M0_request and OPB_M0Grant</p> <p>01 Master ID of OPB master device connected to M1_request and OPB_M1Grant</p> <p>10 Master ID of OPB master device connected to M2_request and OPB_M2Grant</p> <p>11 Master ID of OPB master device connected to M3_request and OPB_M3Grant</p>	

DCR 0x0A8 Read/Write

See *OPB to PLB Bridge Control Register (OPB0_BCTRL)* on page 98.



Figure 32-258. OPB to PLB Bridge Control Register (OPB0_BCTRL)

0:1	PRI	PLB Priority Bits. 00 Lowest 01 10 11 Highest	Used to determine the priority of OPB to PLB bridge requests on the PLB
2:31		Reserved	

DCR 0x0AB Read-Only

See *OPB to PLB Bridge Error Address Register High (OPB0_BEARH)* on page 100.



Figure 32-259. OPB to PLB Bridge Error Address Register High (OPB0_BEARH)

0:27		Reserved
28:31	10'h0B3	Upper address of bus error

DCR 0x0AA Read-Only

See *OPB to PLB Bridge Error Address Register Low (OPB0_BEARL)* on page 99.



Figure 32-260. *OPB to PLB Bridge Error Address Register Low (OPB0_BEARL)*

0:31	0x0B2	Lower address of bus error
------	-------	----------------------------

DCR 0x0A9 Read-Only

See *OPB to PLB Bridge Status Register (OPB0_BSTAT)* on page 99.

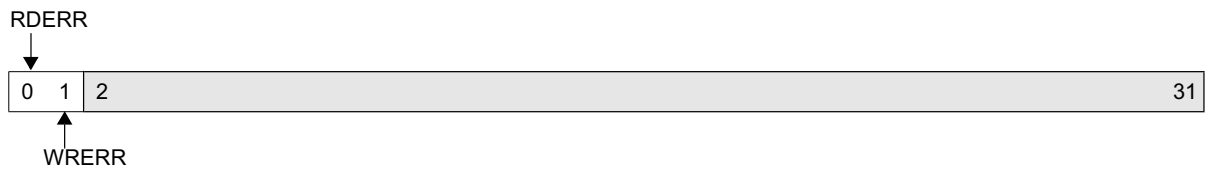


Figure 32-261. OPB to PLB Bridge Status Register (OPB0_BSTAT)

0	RDERR	Read Error Bit Reset to 0	This error bit is set whenever a PLB slave device reports a read error to the bridge. The error is additionally reported to the OPB master through BGI_errAck. The OPB to PLB bridge status register read error bit should be polled to detect errors, and the SEAR and SEGR of the PLB slave consulted for details of the error condition.
1	WRERR	Write Error Bit Reset to 0	This error bit is set whenever a PLB slave device reports a write error to the bridge. For posted single writes and errors occurring after the deassertion of OPB_seqAddr on burst writes, no error is reported to the OPB master through BGI_errAck. An error is reported to the OPB master through BGI_errAck for a previous write error if OPB_seqAddr is still asserted (but note that the error does not correspond to the transfer during which it is reported).
2:31		Reserved	

DCR 0x0AC Read-Only

See *OPB to PLB Bridge Error Address Register Low (OPB0_BEARL)* on page 99.

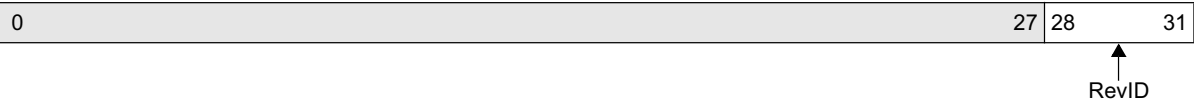
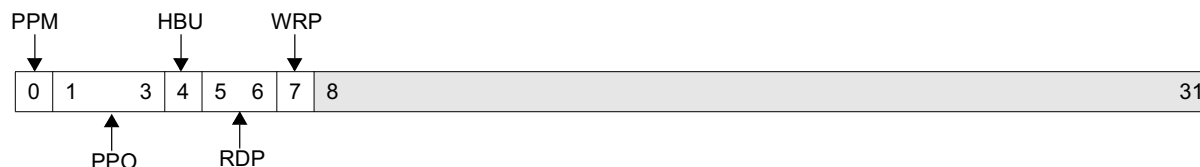


Figure 32-262. OPB to PLB Bridge Revision ID Register (OPB0_REVID)

0:27		Reserved
28:31	0x0B2	This value indicates what revision level this core is. The revision ID for this core is 1.

PLB0_ACR

PLB Arbiter Control Register

Preliminary User's Manual**DCR 0x083 ReadWrite**See *PLB Arbiter Control Register (PLB0_ACR)* on page 84.Figure 32-263. *PLB Arbiter Control Register (PLB0_ACR)*

0	PPM	PLB Priority Mode 0 Fixed 1 Fair	Note: It is recommended to configure PLB0_ACR[PPM] = 1.
1:3	PPO	PLB Priority Order 000 Masters 0, 1, 2, 3, 4, 5, 6, 7 001 Masters 1, 2, 3, 4, 5, 6, 7, 0 010 Masters 2, 3, 4, 5, 6, 7, 0, 1 011 Masters 3, 4, 5, 6, 7, 0, 1, 2 100 Masters 4, 5, 6, 7, 0, 1, 2, 3 101 Masters 5, 6, 7, 0, 1, 2, 3, 4 110 Masters 6, 7, 0, 1, 2, 3, 4, 5 111 Masters 7, 0, 1, 2, 3, 4, 5, 6	The PLB priority order (PPO) will remain a constant value when PLB priority mode (PPM) bit 0 is set to 0 (fixed). However, the PLB priority order (PPO) bits 1:3 are constantly changing when PLB priority mode (PPM) bit 0 is set to 1 (fair)
4	HBU	High Bus Utilization 0 Disabled 1 Enabled	If read and write pipelining are disabled this feature has no effect on arbiter operation.
5:6	RDP	Read Pipeline Control 00 Read pipelining disabled 01 2 Deep read pipe 10 3 Deep read pipe 11 4 Deep read pipe	
7	WRP	Write Pipeline Control 0 Write Pipeline Disabled 1 2 Deep Write Pipe	
8:31		Reserved	

Preliminary User’s Manual

DCR 0x087 Read/Write

See *PLB Error Address Register High (PLB0_BEARH)* on page 88

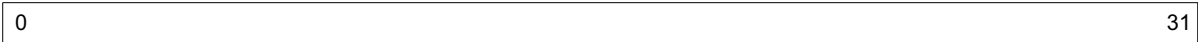


Figure 32-264. PLB Error Address Register (PLB0_BEARH)

0:31		Upper address of bus timeout error
------	--	------------------------------------

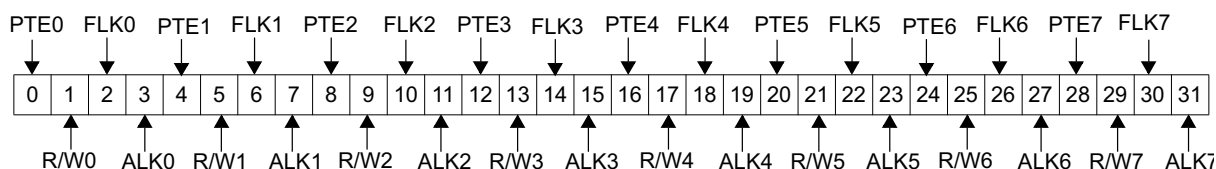
DCR 0x086 Read/Write

See *PLB Error Address Register (PLB0_BEARL)* on page 87.

0	31
---	----

Figure 32-265. PLB Error Address Register (PLB0_BEARL)

0:31		Lower address of bus timeout error
------	--	------------------------------------

DCR 0x084 Read/ClearSee *PLB Error Status Register (PLB0_BESR)* on page 85Figure 32-266. *PLB Error Status Register (PLB0_BESR)*

0	PTE0	Master 0 PLB Timeout Error Status 0 No master 0 timeout error 1 Master 0 timeout error	Master 0 is the read-only instruction cache unit (ICU)
1	R/W0	Master 0 Read/Write Status 0 Master 0 error operation was a write 1 Master 0 ICU error operation was a read	
2	FLK0	Master 0 PLB0_BESR Field Lock 0 Master 0 PLB0_BESR field is unlocked 1 Master 0 field is locked	
3	ALK0	Master 0 PLB0_BEAR Address Lock 0 Master 0 PLB0_BEAR is unlocked 1 Master 0 PLB0_BEAR is locked	
4	PTE1	Master 1 PLB Timeout Error Status 0 No master 1 timeout error 1 Master 1 timeout error	Master 1 is the read-only data cache unit (DCU)
5	R/W1	Master 1 Read/Write Status 0 Master 1 error operation was a write 1 Master 1 error operation was a read	
6	FLK1	Master 1 PLB0_BESR Field Lock 0 Master 1 PLB0_BESR field is unlocked 1 Master 1 PLB0_BESR field is locked	
7	ALK1	Master 1 PLB0_BEAR Address Lock 0 Master 1 PLB0_BEAR is unlocked 1 Master 1 PLB0_BEAR is locked	
8	PTE2	Master 2 PLB Timeout Error Status 0 No master 2 timeout error 1 Master 2 timeout error	Master 2 is the write-only data cache unit (DCU)
9	R/W2	Master 2 Read/Write Status 0 Master 2 error operation was a write 1 Master 2 error operation was a read	
10	FLK2	Master 2 PLB0_BESR Field Lock 0 Master 2 PLB0_BESR field is unlocked 1 Master 2 PLB0_BESR field is locked	
11	ALK2	Master 2 PLB0_BEAR Address Lock 0 Master 2 PLB0_BEAR is unlocked 1 Master 2 PLB0_BEAR is locked	
12	PTE3	Master 3 PLB Timeout Error Status 0 No Master 3 timeout error 1 Master 3 timeout error	Master 3 is the PCIX bridge controller

PLB0_BESR (cont.)

PLB Error Status Register

Preliminary User's Manual

13	R/W3	Master 3 Read/Write Status 0 Master 3 error operation was a write 1 Master 3 error operation was a read	
14	FLK3	Master 3 PLB0_BESR Field Lock 0 Master 3 PLB0_BESR field is unlocked 1 Master 3 PLB0_BESR field is locked	
15	ALK3	Master 3 PLB0_BEAR Address Lock 0 Master 3 PLB0_BEAR is unlocked 1 Master 3 PLB0_BEAR is locked	
16	PTE4	Master 4 PLB Timeout Error Status 0 No master 4 timeout error 1 Master 4 timeout error	Master 4 is the I2O messaging unit
17	R/W4	Master 4 Read/Write Status 0 Master 4 error operation was a write 1 Master 4 error operation was a read	
18	FLK4	Master 4 PLB0_BESR Field Lock 0 Master 4 PLB0_BESR field is unlocked 1 Master 4 field is locked	
19	ALK4	Master 4 PLB0_BEAR Address Lock 0 Master 4 PLB0_BEAR is unlocked 1 Master 4 PLB0_BEAR is locked	
20	PTE5	Master 5 PLB Timeout Error Status 0 No master 5 timeout error 1 Master 5 timeout error	Master 5 is the MAL controller
21	R/W5	Master 5 Read/Write Status 0 Master 5 error operation was a write 1 Master 5 error operation was a read	
22	FLK5	Master 5 PLB0_BESR Field Lock 0 Master 5 PLB0_BESR field is unlocked 1 Master 5 PLB0_BESR field is locked	
23	ALK5	Master 5 PLB0_BEAR Address Lock 0 Master 5 PLB0_BEAR is unlocked 1 Master 5 PLB0_BEAR is locked	
24	PTE6	Master 6 PLB Timeout Error Status 0 No master 6 timeout error 1 Master 6 timeout error	Master 6 is the DMA controller
25	R/W6	Master 6 Read/Write Status 0 Master 6 error operation was a write 1 Master 6 error operation was a read	
26	FLK6	Master 6 PLB0_BESR Field Lock 0 Master 6 PLB0_BESR field is unlocked 1 Master 6 PLB0_BESR field is locked	
27	ALK6	Master 6 PLB0_BEAR Address Lock 0 Master 6 PLB0_BEAR is unlocked 1 Master 6 PLB0_BEAR is locked	
28	PTE7	Master 7 PLB Timeout Error Status 0 No Master 7 timeout error 1 Master 7 timeout error	Master 7 is the OPB to PLB bridge controller
29	R/W7	Master 7 Read/Write Status 0 Master 7 error operation was a write 1 Master 7 error operation was a read	

30	FLK7	Master 7 PLB0_BESR Field Lock 0 Master 7 PLB0_BESR field is unlocked 1 Master 7 PLB0_BESR field is locked
31	ALK7	Master 7 PLB0_BEAR Address Lock 0 Master 7 PLB0_BEAR is unlocked 1 Master 7 PLB0_BEAR is locked

DCR 0x082 Read-Only

See *PLB Revision ID Register (PLB0_REVID)* on page 84

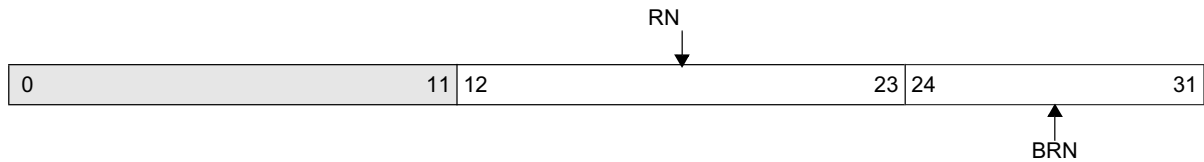


Figure 32-267. PLB Arbiter Revision ID Register (PLB0_REVID)

0:11		Reserved	
12:23	RN	Revision number	Corresponds to the RCS revision of the source RTL
24:31	BRN	Branch revision number	Corresponds to the RCS branch revision of the source RTL

MMIO 0x2 0EC80014 Read/Write (PLB), 0x17-0x14 Read/Write (PCI-X)

See *PCI-X BAR0 High Register (PCIX0_BAR0H)* on page 646

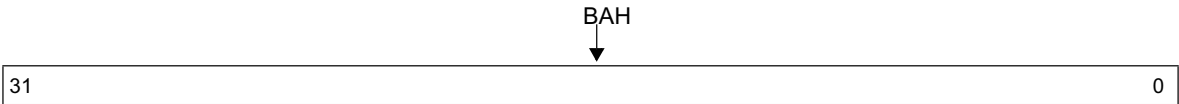


Figure 32-268. PCI-X BAR0 High Register (PCIX0_BAR0H)

31:0	BAH	Base Address High	Defines the upper 32 bits of PCI memory space that is mapped to PLB. Only corresponding bits that are 1 in the size field of the PIM0 Size/Attribute High Register are writable. Bits that are 0 in the size field of the PIM0 Size/Attribute High Register cause the corresponding Base Address bits to be always 0.
------	-----	-------------------	---

MMIO 0x2 0EC80010 Read/Write (PLB), 0x13-0x10 Read/Write (PCI-X)

See *PCI-X BAR0 Low Register (PCIX0_BAR0L)* on page 645

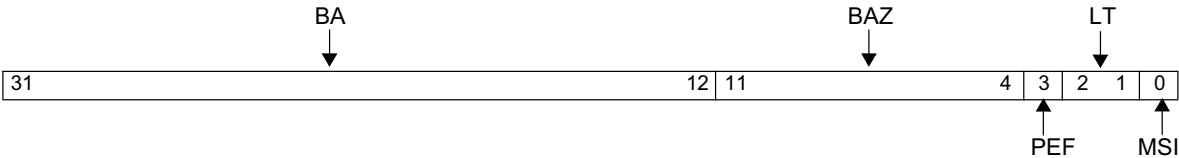


Figure 32-269. PCI-X BAR0 Low Register (PCIX0_BAR0L)

31:12	BA	Base Address	These bits determine the lower 32 bits of the PCI Memory address space where this region is located. Only corresponding bits that are 1 in the size field of the PIM0 Size/Attribute Low Register are writable. Bits that are 0 in the size field of the PIM0 Size/Attribute Low Register cause the corresponding Base Address bits to be always 0.
11:4	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.
3	PEF	Prefetchable	This bit determines if the region is prefetchable. Its value is determined by the PIM0 Size/Attribute Low, Prefetch Enable bit.
2:1	LT	Location Type	These bits are always 10 to indicate that the memory space can be located anywhere in the 64-bit address space.
0	MSI	Memory Space Indicator	This bit is always 0 to indicate memory space (rather than I/O).

MMIO 0x2 0EC80018 Read/Write (PLB), 0x1B-0x18 Read/Write (PCI-X)

See *PCI-X BAR1 Register (PCIX0_BAR1)* on page 647



Figure 32-270. PCI-X BAR1 Register (PCIX0_BAR1)

31:8	BA	Base Address	These bits determine the PCI I/O address space where this region is located.
7:2	BAZ	Base Address always zero	These bits are always 0 since the minimum size of this range is 256 bytes.
1		Reserved	Returns zero when read.
0	MSI	Memory Space Indicator	This bit is always 1 to indicate I/O space (rather than memory).

MMIO 0x2 0EC80020 Read/Write (PLB), 0x23-0x20 Read/Write (PCI-X)

See *PCI-X BAR2 High Register (PCIX0_BAR2H)* on page 649

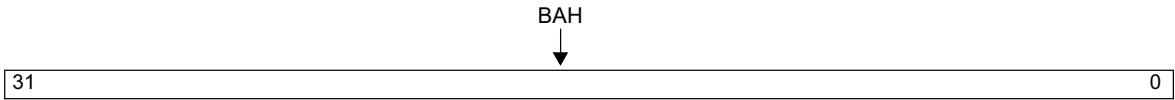


Figure 32-271. PCI-X BAR2 High Register (PCIX0_BAR2H)

31:0	BAH	Base Address High	Defines the upper 32 bits of PCI memory space that is mapped to PLB. Only corresponding bits that are 1 in the size field of the PIM2 Size/Attribute High Register are writable. Bits that are 0 in the size field of the PIM2 Size/Attribute High Register cause the corresponding Base Address bits to be always 0.
------	-----	-------------------	---

MMIO 0x2 0EC8001CRead/Write (PLB), 0x1F-0x1C Read/Write (PCI-X)

See *PCI-X BAR2 Low Register (PCIX0_BAR2L)* on page 648

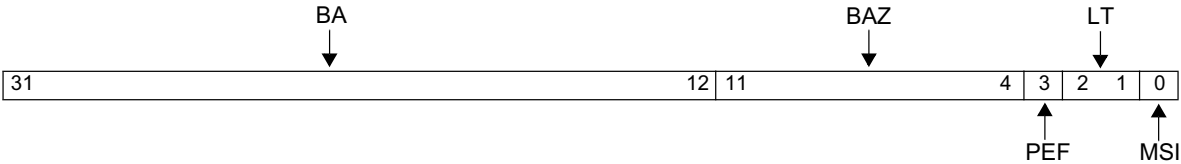


Figure 32-272. PCI-X BAR2 Low Register (PCIX0_BAR2L)

31:12	BA	Base Address	These bits determine the lower 32-bits of the PCI Memory address space where this region is located. Only corresponding bits that are 1 in the size field of the PIM2 Size/Attribute Low Register are writable. Bits that are 0 in the size field of the PIM2 Size/Attribute Low Register cause the corresponding Base Address bits to be always 0.
11:4	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.
3	PEF	Prefetchable	This bit determines if the region is prefetchable. Its value is determine by the PIM2 Size/Attribute Low, Prefetch Enable bit.
2:1	LT	Location Type	These bits are always 10 to indicate that the memory space can be located anywhere in the 64-bit address space.
0	MSI	Memory Space Indicator	This bit is always 0 to indicate memory space (rather than I/O).

MMIO 0x2 0EC8000F Read-Only (PLB), 0x0F Read-Only (PCI-X)

See *PCI-X Built-in Self Test (BIST) Control Register (PCIX0_BIST)* on page 644

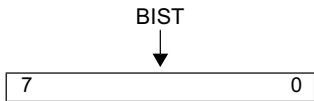


Figure 32-273. PCI-X BIST Control Register (PCIX0_BIST)

7:0	BIST	PCI Built-in-Self Test (BIST) Control
-----	------	---------------------------------------

MMIO 0x2 0EC80040 Read/Write (PLB), 0x43-0x40 Read/Write (PCI-X)

See *PCI-X Bridge Options 1 (PCIX0_BRDGOPT1)* on page 653

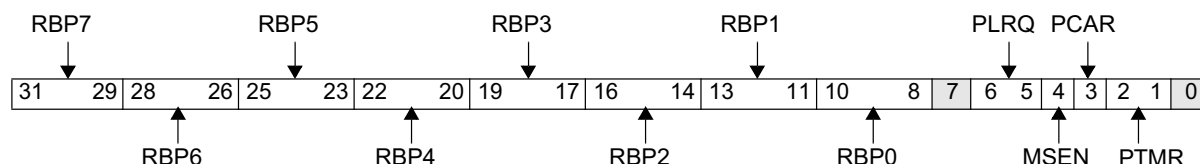


Figure 32-274. PCI-X Bridge Options 1 (PCIX0_BRDGOPT1)

31:29	RBP7	Outbund Read Buffer Mapping for PLB master 7:OPB to PLB Bridge	Must be set 000.
28:26	RBP6	Outbund Read Buffer Mapping for PLB master 6:Direct Memory Access Controller	Must be set to 001.
25:23	RBP5	Outbund Read Buffer Mapping for PLB master 5:Media Access Layer	Must be set to 000.
22:20	RBP4	Outbund Read Buffer Mapping for PLB master 4:I2O Messaging Unit	Must be set to 000.
19:17	RBP3	Outbund Read Buffer Mapping for PLB master 3:PCIX Bridge Controller	Must be set to 000.
16:14	RBP2	Outbund Read Buffer Mapping for PLB master 2:Processor Core Data Cache Write Unit	Must be set to 000.
13:11	RBP1	Outbund Read Buffer Mapping for PLB master 1:Processor Core Data Cache Read Unit	Must be set to 000.
10:8	RBP0	Outbund Read Buffer Mapping for PLB master 0:Processor Core Instruction Cache Read Unit	Must be set to 000.
7		Reserved	This bit must be set to zero.
6:5	PLRQ	PLB Request Priority	This bit controls how the PLB-PCIX Bridge PLB master controls the PLB arbitration priority for all PLB accesses. 11b: highest 10b: next highest 01b: next highest 00b: lowest
4	MSEN	Inbound MSI Enable	This bit enables Inbound MSI. This is typically only used in Host-Bridge Mode. When high, the Inbound MSI logic drives the inbound interrupt to UIC.
3	PCAR	PCI Arbiter Park Mode	This bit defines how the internal PCI arbiter will handle bus parking. A value of 0 means that the arbiter will park on requester 0 (the bridge PCI master). A value of 1 means that the arbiter will park on the last master granted the bus. This bit must not be changed while PCI masters are active.
2:1	PTMR	PCI Target Memory Read Command interpretation	This field allows PLB-PCIX Bridge to be forced to treat a PCI memory read as a memory read multiple or memory read line with respect to the burst size implied by these read commands. This is for masters that use memory read for multiple beat bursts. 00 Memory read 01 Memory read line 10 Memory read multiple 11 Reserved
0		Reserved	Must be set to 0.

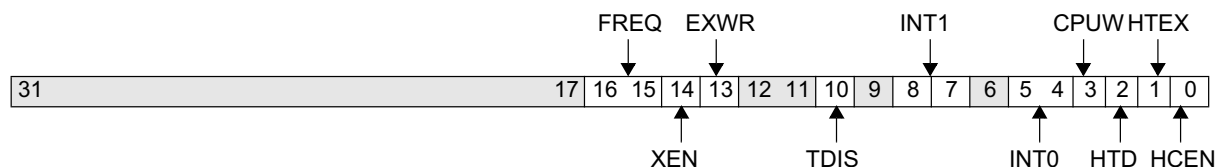
MMIO 0x2 0EC80044 Read/Write (PLB), 0x47-0x44 Read/Write (PCI-X)See *PCI-X Bridge Options 2 (PCIX0_BRDGOPT2)* on page 654

Figure 32-275. PCI-X Bridge Options 2 (PCIX0_BRDGOPT2)

31:17		Reserved	Always read as zero.
16:15	FREQ	Frequency Range for PCIX 00 100-133 MHz 01 66-100 MHz 10 50-66 MHz 11 reserved. Frequency Range for PCI conv 00 reserved 01 reserved 10 33-66 MHz 11 0-33 MHz.	Read only value that reports the PCI frequency range
14	XEN	PCI-X Mode Enabled	Read only value that reports if X mode was selected.
13	EXWR	External Write to PCI Command Interrupt	When the PCI Command Register is written from the PCI side (inbound config write), this bit is set.
12:11		Reserved	Must be set to 0.
10	TDIS	PCI Discard Timer Disable	When 1, PLB-PCIX Bridge never discards delayed read data.
9		Reserved	Must be set to 0.
8:7	INT1	PCI Interrupt Source 1	Reset to 01. Do not change.
6		Reserved	Must be set to 0.
5:4	INT0	PCI Interrupt Source 0	This field has a value of 00 after reset. Do not change. Resets to 00. Setting this field to 01 will mask the generation of outbound interrupts, both via INTA and MSI.
3	CPUW	Local CPU wait	This bit controls the value of the PCI local CPU wait (PCWE) strapping bit, which prevents the local CPU from running when this bit is 1. By causing this bit to be 1 at reset, the host PCI master is given time to initialize the internal register set of the PLB-PCIX Bridge before the local CPU boots. This is typically used to set up boot code, when the local CPU is not booting from local ROM.
2	HTD	Host Configuration Enable Timer Disable	If this bit is set, the Host Configuration Enable timer never expires. See Host Configuration Enable bit below.
1	HTEX	Host Configuration Enable Timer Expired	This bit is set if the Host Configuration Enable timer expired. See Host Configuration Enable bit below.

Preliminary User's Manual

0	HCEN	Host Configuration Enable	<p>This bit controls host PCI access to the PCI configuration registers. If this bit is a 0, all host attempts to access PCI configuration registers (internal registers) of the PLB-PCIX Bridge are retried. By causing this bit to be 0 at reset the local CPU is given time to initialize the internal register set before the host sees it.</p> <p>The reset value of this bit is determined by the PCI host configuration enable (PHCE) strapping bit. If PCI host configuration enable (PHCE) strapping bit is tied to a 1 (high), this bit has a value of 1 after reset. If PCI host configuration enable (PHCE) strapping bit is tied to a 0 (low), this bit has a value of 0 after reset.</p> <p>If this bit is cleared following reset, and the Host Configuration Enable Timer Disable bit is cleared, and if this bit does not get set by the local CPU within 2^{24} PLB clocks (126 ms at 133 MHz), then it is set automatically, and the Host Configuration Enable Timer Expired bit is set in this register.</p> <p>As per the PCI Specification, Version 2.3, as little as 500 ms is allowed for configuration to be set up. Thus, if PLB is run below 34 MHz, this timer will not expire before this time period.</p>
---	------	---------------------------	--

MMIO 0x2 0EC8000C Read/Write (PLB), 0x0C Read/Write (PCI-X)

See *PCI-X Cache Line Size Register (PCIX0_CACHELS)* on page 642

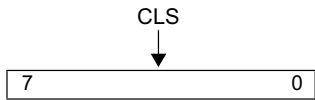


Figure 32-276. *PCI-X Cache Line Size Register (PCIX0_CACHELS)*

7:0	CLS	PCI Cache Line Size
-----	-----	---------------------

Preliminary User’s Manual

MMIO 0x2 0EC80034 Read-Only (PLB), 0x34 Read-Only (PCI-X)

See *PCI-X Capabilities Pointer (PCIX0_CAP)* on page 651

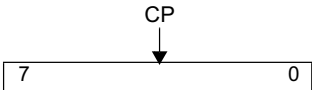


Figure 32-277. *PCI-X Capabilities Pointer (PCIX0_CAP)*

7:0	CP	PCI Capabilities Pointer
-----	----	--------------------------

MMIO 0x2 0EC80009 Read/Write (PLB), 0x0B-0x09 Read-Only (PCI)-X)

See *PCI-X Class Register (PCIX0_CLS)* on page 642

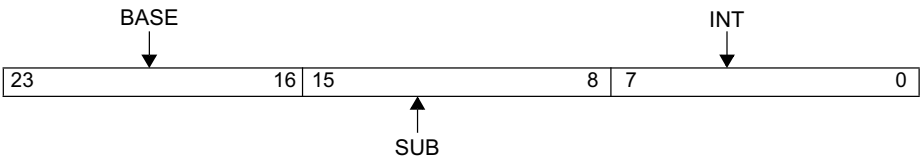


Figure 32-278. PCI-X Class Register (PCIX0_CLS)

23:16	BASE	Base Class	Reset to 0x06, which indicates bridge device.
15:8	SUB	Subclass	Reset to 00, which indicates host bridge.
7:0	INT	Interface Class	Reset to 00.

MMIO 0x2 0EC80004 R/W (PLB) 0x05-0x04 R/W (PCI-X)

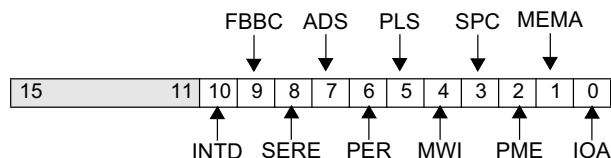
See *PCI-X Command Register (PCIX0_CMD)* on page 639

Figure 32-279. PCI-X Command Register (PCIX0_CMD)

15:11		Reserved	These bits are reserved, and return zeros when read.
10	INTD	Interrupt Disable	This bit disables the PLB-PCIX Bridge from asserting INTA#. A value of 0 enables the assertion of its INTA# signal. A value of 1 disabled the assertion of its INTA# signal. This bit is 0 at reset.
9	FBBC	Fast Back-to-Back Capable Read-only; returns 0 when read.	Fast back-to-back write enable. This bit enables PCI masters to perform fast back-to-back transactions to different devices. The PLB-PCIX Bridge does not perform fast back-to-back transactions, therefore, this bit is read-only and returns zero when read.
8	SERE	SERR# Enable	Enable PCI_SERR#. Enables driving PCI_SERR# to report the detection of an error out to the PCI bus. If disabled, PCI_SERR# is never asserted regardless of the detection of any error.
7	ADS	Address Stepping	Address stepping wait states. PLB-PCIX bridge does not address step (except when generating a Configuration Type 0 cycle), therefore, this bit is read-only and returns zero when read.
6	PER	Parity Error Response	Parity error response. This bit enables the following types of PCI bus parity errors: PCI data bus parity errors while PCI is master PCI data bus parity errors while PCI is target PCI address bus parity errors When parity error response is disabled (set to 0), detection of these errors is masked and PERR# is not asserted, although parity is still generated.
5	PLS	Palette Snooping	Enable special palette snooping. The PLB-PCIX Bridge macro is not a VGA device, therefore, this bit is read-only and returns zero when read.
4	MWI	Memory Write and Invalidate	Enable memory write and invalidate command support. When high, PLB-PCIX Bridge is allowed to generate MWI transactions. Only used in PCI-Conv mode. This bit is 0 at reset.
3	SPC	Special Cycle	Enable special cycle operations. PLB-PCIX Bridge never monitors special cycles, therefore, this bit is read-only and returns zero when read.
2	PME	PCI Master Enable	Enables PLB-PCIX Bridge to master cycles on the PCI bus. When this bit is 0, PLB-PCI Bridge only responds as a PLB slave to accesses to the internal register set. However, the PLB-PCIX Bridge can still master transfers for split completions and to complete outbound writes that were posted prior to setting this bit to 0.
1	MEMA	Memory Access	Controls response of PLB-PCIX Bridge as a PCI memory target. A value of 1 enables PLB-PCIX Bridge to respond as a target. This bit is 0 (disabled) at reset.
0	IOA	I/O Access	Controls response of PLB-PCIX Bridge as a PCI I/O target. A value of 1 enables PLB-PCIX Bridge to respond as a target. This bit is 0 (disabled) at reset.

MMIO 0x2 0EC80002 Read/Write (PLB), 0x03-0x02 Read-Only (PCI-X)

See *PCI-X Device ID Register (PCIX0_DEVID)* on page 638

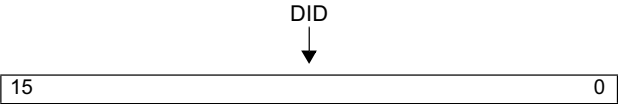


Figure 32-280. *PCI-X Device ID Register (PCIX0_DEVID)*

15:0	DID	Device ID
------	-----	-----------

MMIO 0x2 0EC80030 Read/Write (PLB), 0x33-0x30 Read/Write (PCI-X)

See *PCI-X Expansion ROM Base Address Register (PCIX0_EROMBA)* on page 650

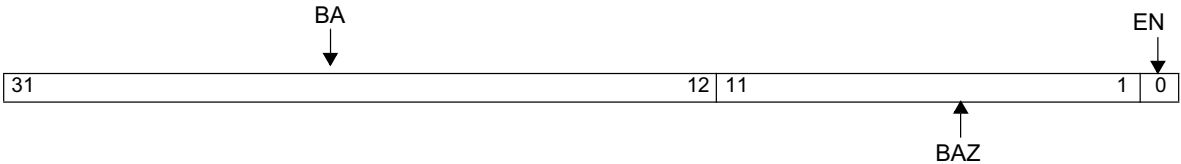


Figure 32-281. PCI-X Expansion ROM Base Address Register (PCIX0_EROMBA)

31:12	BA	Base Address	These bits determine the PCI I/O address space where this region is located.
11:1	BAZ	Base Address - always zero	These bits are always 0 since the minimum size of this range is 4 KB.
0	EN	Enable	When 1, the expansion ROM address space of the PLB-PCIX Bridge is enabled and the BAR2 address space is disabled. When 0, the expansion ROM address space is disabled and the BAR2 address space is enabled, translated through PIM2.

PCIX0_ERREN

PCI-X Error Enable

Preliminary User's Manual

MMIO 0x2 0EC80050 Read/Write (PLB), 0x53-0x50 Read/Write (PCI-X)

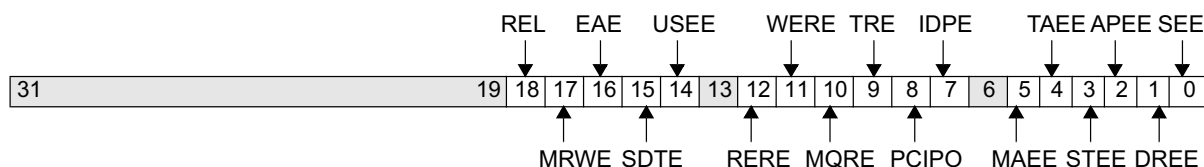
See *PCI-X Error Enable (PCIX0_ERREN)* on page 656

Figure 32-282. PCI-X Error Enable (PCIX0_ERREN)

31:19		Reserved	Always read as zero.
18	REL	Route Error Locally	When set, errors are routed locally using ERROR_INT. Assertion of PLB MERR is not affected by this bit. When cleared, errors are routed to the PCI using PCI_SERR#.
17	MRWE	MRdErr/MWrErr Assertion Enable	This bit enables the assertion of PLB Read MERR or PLB Write MERR when the PLB-PCIX Bridge detects an error.
16	EAE	ERROR_INT Assertion Enable	This bit enables the assertion of ERROR_INT when the PLB-PCIX Bridge detects an error.
15	SDTE	Outbound Split Discard Timer Enable	This bit enables the outbound split timers. If this bit is 0, then the PLB-PCIX Bridge waits an infinite amount of time for outbound split completions.
14	USEE	Unexpected Split Completion Error Enable	When this bit is 1, unexpected split completions will generate an error. When this bit is 0, unexpected split completions will be accepted, but no error is generated.
13		Reserved	Always read as zero.
12	RERE	MRdErr Receive Enable	This bit enables the detection of PLB Read MERR when the PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
11	WERE	MWrErr Receive Enable	This bit enables the detection of PLB Write MERR when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
10	MQRE	Mirq Receive Enable	This bit enables the detection of MIRQ when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
9	TRE	Timeout Receive Enable	This bit enables the detection of Timeout when PLB-PCIX Bridge is the PLB master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
8	PCIPO	PCI Parity Option	This bit should normally be left 1. It may be set to 0 to mask all types of PCI parity error checking. This bit is 1 after reset.
7	IDPE	PCI Inbound Write Data Parity Error Enable	This bit enables the detection of PCI data parity errors on inbound writes. Note: The Detected Parity Error bit of the PCI Status Register and the masking of PCI PERR# is not affected by the above, but the assertion of ERROR_INT and PCI_SERR# is.
6		Reserved	Always read as zero.
5	MAEE	Master Abort Error Enable	This bit enables the detection of master aborts as an error condition, when PLB-PCIX Bridge is the PCI master. If this bit is set, PLB-PCIX Bridge may drive PLB Read MERR or PLB Write MERR on the PLB or PCI_SERR# on the PCI bus.
4	TAE	Target Abort Error Enable	This bit enables the detection of a target abort received while PLB-PCIX Bridge is the PCI master to be detected as an error condition. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# on the PCI bus.

3	STEE	Split Transaction Error Enable	This bit enables the detection of split transaction errors (PCI-X only), while the PLB-PCIX Bridge is the PCI master. If this bit is set, the PLB-PCIX Bridge may drive ERROR_INT to the local CPU or PCI_SERR# to the PCI bus.
2	APEE	Addr/Attrib Parity Error Enable	This bit enables the detection of Address or Attribute (PCI-X only) parity errors while the PLB-PCIX Bridge is the PCI target (including split response). If this bit is set, the PLB-PCIX Bridge may drive PCI_SERR# on the PCI bus.
1	DRTE	Delayed Read Discard Timer Expired Error Enable	This bit enables the detection of Delayed Read Discard Timer expiration as an error. If this bit is set, the PLB-PCIX Bridge may drive the ERROR_INT signal to the local CPU or PCI_SERR# on the PCI bus.
0	SEE	PCI_SERR# Received as Error Enable	This bit enables the detection of PCI_SERR# as an error. If this bit is set, the PLB-PCIX Bridge may drive the ERROR_INT signal to the local CPU. Note: The PLB-PCIX Bridge may be receiving this error, while at the same time, it is driving PCI_SERR# in response to some other error.

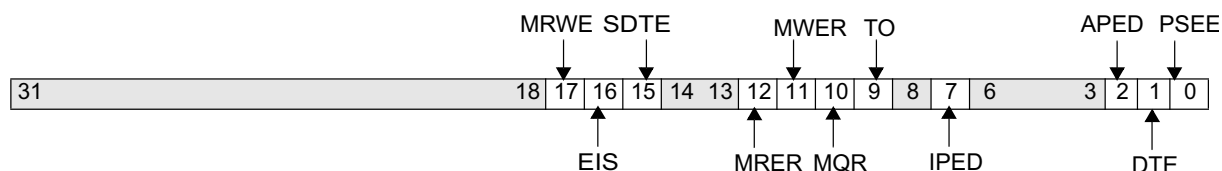
MMIO 0x2 0EC80054 Read/Write (PLB), 057-054 Read/Write (PCI-X)See *PCI-X Error Status (PCIX0_ERRSTS)* on page 657

Figure 32-283. PCI-X Error Status (PCIX0_ERRSTS)

31:18		Reserved	Always read as zero.
17	MRWE	MRdErr or MWrErr Signaled	This bit is set whenever an error occurs that causes PLB-PCIX Bridge to assert PLB Read MERR or PLB Write MERR.
16	EIS	ERROR_INT Signaled	This bit is set whenever an error occurs that causes PLB-PCIX Bridge to assert ERROR_INT.
15	SDTE	Split Discard Timer Expired	This bit is set when an outbound split discard timer expires prior to receiving the split completion.
14:13		Reserved	Always read as zero.
12	MRER	MRdErr Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives PLB Read MERR.
11	MWER	MWErr Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives PLB Write MERR.
10	MQR	Mirq Received	This bit is set when PLB-PCIX Bridge is a PLB master and receives MIRQ.
9	TO	Timeout	This bit is set when PLB-PCIX Bridge is a PLB master and receives Timeout.
8		Reserved	Always read as zero.
7	IPED	PCI Inbound Write Data Parity Error Detected	This bit is set when a PCI inbound write data parity error is detected and the PCI Inbound Write Data Parity Error Enable bit of the Error Enable Register is set.
6		Reserved	Always read as zero.
5		Reserved	Always read as zero. Note: The status bit for the detection of master aborts is in the PCI Status Register.
4		Reserved	Always read as zero. Note: The status bit for the detection of target aborts is in the PCI Status Register.
3		Reserved	Always read as zero. Note: The split completion error is indicated in the PCI-X Status Register.
2	APED	Addr/Attrib Parity Error Detected	This bit is set when the PLB-PCIX Bridge is the target on the PCI bus and detects an address or attribute parity error and the Address/Attribute Parity Error Enable bit of the Error Enable Register is set and the Parity Error Response bit of the PCI Command Register is set.
1	DTE	Delayed Read Discard Timer Expired	This bit is set when the PLB-PCIX Bridge detects that the Discard Timer Expires bit (PCI-Conv Only) and the Delayed Read Discard Timer Expired Error Enable bit of the Error Enable Register is set.
0	PSEE	PCI_SERR# Received	This bit is set when PCI_SERR# is asserted and the PCI_SERR# Receive as Error Enable bit of the Error Enable Register is set.

Preliminary User’s Manual

PCIX0_HDTYPE
MMIO 0x2 0EC8000E Read-Only (PLB), 0x0E Read-Only (PCI-X)
See *PCI-X Header Type Register (PCIX0_HDTYPE)* on page 643

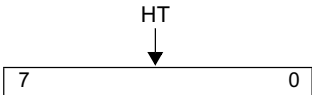


Figure 32-284. PCI-X Header Type Register (PCIX0_HDTYPE)

7:0	HT	PCI Header Type
-----	----	-----------------

MMIO 0x2 0EC801F8 Write-Only (PLB), (PCIX0_BAR0H || PCIX0_BAR0L) + (0xFB-0xF8) Write-Only (PCI-X)

See *PCI-X Inbound Message MSI (PCIX0_IM)* on page 690



Figure 32-285. PCI-X Inbound Message MSI (PCIX0_IM)

31:4		Reserved	Returns zero when read.
3:0	IMSI	MSI In	Inbound MSIs write this register. The data value written causes the corresponding pci_msi_int0:12 to go active to the interrupt controller.

Preliminary User’s Manual

MMIO 0x2 0EC8003C Read/Write (PLB), 0x3C Read/Write (PCI-X)
See *PCI-X Interrupt Line Register (PCIX0_INTLN)* on page 651

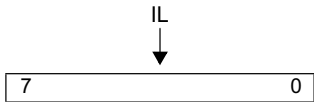


Figure 32-286. PCI-X Interrupt Line Register (PCIX0_INTLN)

7:0	IL	PCI Interrupt Line
-----	----	--------------------

MMIO 0x2 0EC8003D Read-Only (PLB), 0x3D read-Only (PCI-X)

See *PCI-X Interrupt Pin Register (PCIX0_INTPN)* on page 651

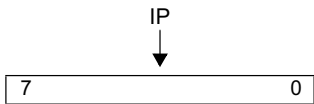


Figure 32-287. PCI-X Interrupt Pin Register (PCIX0_INTPN)

7:0	IP	PCI Interrupt Pin
-----	----	-------------------

Preliminary User’s Manual

MMIO 0x2 0EC8000D Read/Write (PLB) 0x0D Read/Write (PCI-X)

See *PCI-X Latency Timer Register (PCIX0_LATTIM)* on page 643

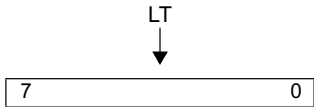


Figure 32-288. PCI-X Latency Timer Register (PCIX0_LATTIM)

7:0	LT	PCI Latency Timer
-----	----	-------------------

MMIO 0x2 0EC8003F Read-Only (PLB), 0x3F Read-Only (PCI-X)

See *PCI-X Maximum Latency Register (PCIX0_MAXLTNCY)* on page 652

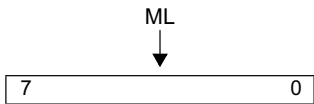


Figure 32-289. *PCI-X Maximum Latency Register (PCIX0_MAXLTNCY)*

7:0	ML	PCI Maximum Latency
-----	----	---------------------

Preliminary User’s Manual

MMIO 0x2 0EC8003E Read-Only (PLB), 0x3E Read-Only (PCI-X)
See *PCI-X Minimum Grant Register (PCIX0_MINGNT)* on page 652

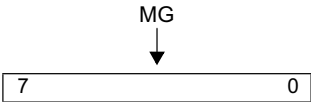


Figure 32-290. PCI-X Minimum Grant Register (PCIX0_MINGNT)

7:0	MG	PCI Minimum Grant
-----	----	-------------------

MMIO 0x2 0EC80104 RW (PLB), (PCIX0_BAR0H || PCIX0_BAR0L) + (0x07-0x04) RW (PCI-X)

See *PCI-X Message In High (PCIX0_MSGIH)* on page 689

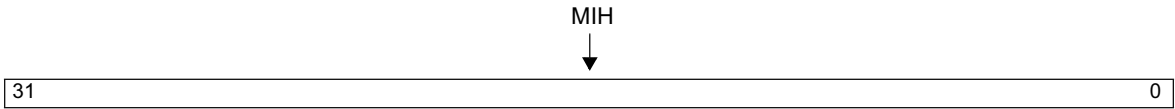


Figure 32-291. *PCI-X Message In High (PCIX0_MSGIH)*

31:0	MIH	Message In High	This holds the upper 32 bits of a 64-bit inbound message.
------	-----	-----------------	---

MMIO 0x2 0EC80100 R/W (PLB), (PCIX0_BAR0H || PCIX0_BAR0L) + (0x03-0x00) R/W (PCI-X)

See *PCI-X Message In Low (PCIX0_MSGIL)* on page 689

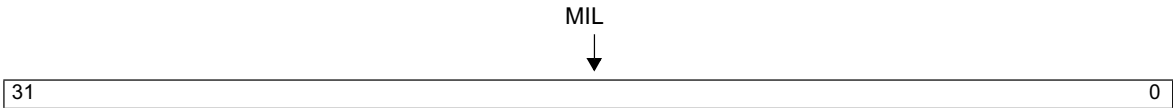


Figure 32-292. *PCI-X Message In Low (PCIX0_MSGIL)*

31:0	MIL	Message In Low	This holds an inbound message. If bit 0 is high, then an inbound interrupt is generated.
------	-----	----------------	--

MMIO 0x2 0EC8010C RW (PLB), (PCIX0_BAR0H || PCIX0_BAR0L) + (0x0F-0x0C RW (PCI-X)

See *PCI-X Message Out High (PCIX0_MSGOH)* on page 690

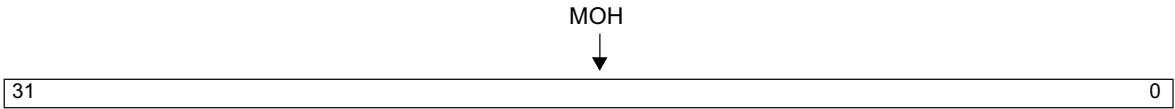


Figure 32-293. *PCI-X Message Out High (PCIX0_MSGOH)*

31:0	MOH	Message Out High	This holds the upper 32 bits of a 64-bit outbound message.
------	-----	------------------	--

MMIO 0x2 0EC80108 RW (PLB), (PCIX0_BAR0H || PCIX0_BAR0L) + (0x0B-0x08) RW (PCI-X)

See *PCI-X Message Out Low (PCIX0_MSGOL)* on page 690

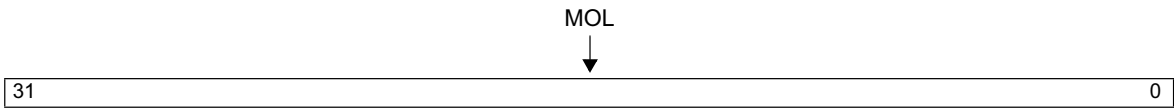


Figure 32-294. *PCI-X Message Out Low (PCIX0_MSGOL)*

31:0	MOL	Message Out Low	This holds an outbound message. If bit 0 is high, then an out-bound interrupt is generated.
------	-----	-----------------	---

MMIO 0x2 0EC800C0 Read-Only (PLB), 0xC0 read-Only (PCI-X)
See *PCI-X MSI Capability Identifier Register (PCIX0_OMCAPID)* on page 676

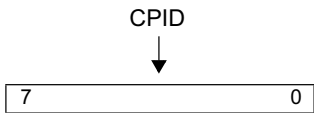


Figure 32-295. PCI-X MSI Capabilities Identifier (PCIX0_OMCAPID)

7:0	CPID	Message Signaled Interrupts (MSI) Capabilities Identifier
-----	------	---

MMIO 0x2 0EC800C4 Read/Write (PLB), 0xC7-0xC4 Read/Write (PCI-X)

See *PCI-X Message Address Register (PCIX0_OMMA)* on page 677

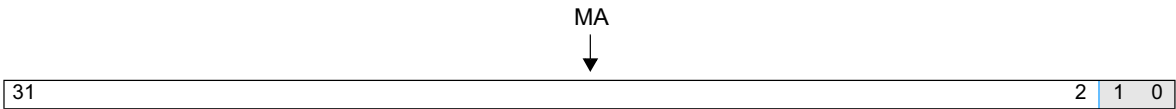


Figure 32-296. PCI-X Message Address Register (PCIX0_OMMA)

31:2	MA	Message Address	Indicates system specific message address. If the Message Enable bit (bit 0 of the Message Control Register) is set, the contents of this register specify the doubleword aligned address pci_ad[31:2] for the MSI memory write transaction. pci_ad[1:0] are driven to zero during the address phase. This field is read/write.
1:0		Reserved	Returns zero when read.

MMIO 0x2 0EC800C2 Read/Write (PLB), 0xC3-0xC2 Read/Write (PCI-X)
See *PCI-X Message Control Register (PCIX0_OMMC)* on page 677

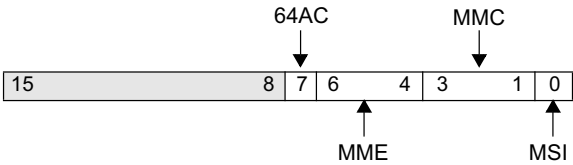


Figure 32-297. PCI-X Message Control Register (PCIX0_OMMC)

15:8		Reserved	Returns zero when read.
7	64AC	64-bit Address Capable	This bit is hardwired to a 1 and indicates that PLB-PCIX Bridge is capable of generating a 64-bit message address.
6:4	MME	Multiple Message Enable	System software writes to this field to indicate the number of allocated messages for PLB-PCIX Bridge. The number of allocated messages is aligned to a power of 2. Bit 6:5 of this field are hardwired to 0 and only bit 4 is writable. This field's state after reset is 000b.
3:1	MMC	Multiple Message Capable	System software reads this field to determine the number of messages requested by PLB-PCIX Bridge. The PLB-PCIX Bridge requests two messages and is indicated by hardwiring this field to a value of 001b.
0	MSI	MSI Enable 0 MSI disabled 1 MSI enabled	This read/write bit is used by the system to enable or disable MSI capability. This bit state after reset is 0 (MSI is disabled after reset).

MMIO 0x2 0EC800CC Read/Write (PLB), 0xCD-0xCC Read/Write (PCI-X)

See *PCI-X Message Data Register (PCIX0_OMMDATA)* on page 678

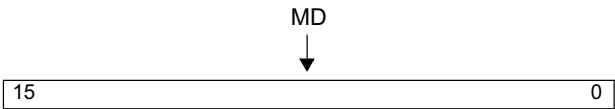


Figure 32-298. PCI-X Message Data Register (PCIX0_OMMDATA)

15:0	MD	Message Data	System-specified message. This field is written by the system. When PLB-PCIX Bridge issues an MSI message, it writes this value to the previously defined message address. If PLB-PCIX Bridge is allocated two messages by the system, the least significant bit of this field determines which message is being reported. If PLB-PCIX Bridge is allocated only one message by the system, then PLB-PCIX Bridge cannot modify this data on a MSI write.
------	----	--------------	---

MMIO 0x2 0EC800CE Read/Write (PLB), 0xCE Read/Write (PCI-X)

See *PCI-X Message End of Interrupt Register (PCIX0_OMMEOI)* on page 679



Figure 32-299. PCI-X Message End of Interrupt Register (PCIX0_OMMEOI)

7:1		Reserved	Returns zero when read.
0	MEOI	Message End of Interrupt	Following the generation of an outbound MSI, the outbound MSI unit must be rearmed before another outbound MSI can be generated. Writing a 1 to this bit rearms the MSI function. This bit automatically clears itself (always reads zero). EOI is required only when using MSI.

MMIO 0x2 0EC800C8 Read/Write (PLB), 0xCB-0XC8 Read/Write (PCI-X)

See *PCI-X Message Upper Address Register (PCIX0_OMMUA)* on page 678

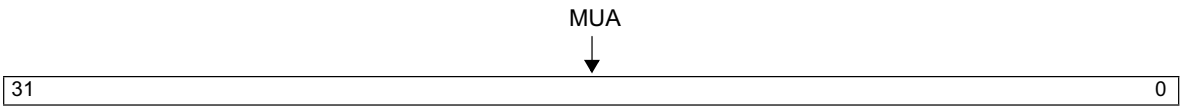


Figure 32-300. PCI-X Message Upper Address Register (PCIX0_OMMUA)

31:0	MUA	Message Upper Address	PLB-PCIX Bridge supports a 64-bit message address (bit 7 in Message Control Register is set to 1). The contents of this register specify the upper 32 bits of a 64-bit message address pci_ad[63:32]. If the contents of this register are zero, PLB-PCIX Bridge uses the 32-bit address specified by the Message Address Register. This field is read/write and is configured by the system.
------	-----	-----------------------	---

MMIO 0x2 0EC800C1 Read/Write (PLB), 0xC1 Read/Write (PCI-X)
See *PCI-X Next Item Pointer (PCIX0_OMNIPTR)* on page 676

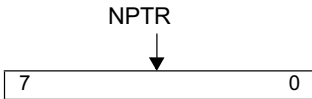


Figure 32-301. PCI-X MSI Next Item Pointer Register(PCIX0_OMNIPTR)

7:0	NPTR	Message Signaled Interrupts (MSI) Next Item Pointer
-----	------	---

MMIO 0x2 0EC800DC Read-Only (PLB), 0xDC Read-Only (PCI-X)

See *PCI-X Capability Identifier (PCIX0_PCIXCAPID)* on page 683

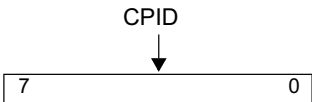


Figure 32-302. *PCI-X Capability Identifier (PCIX0_PCIXCAPID)*

7:0	CPID	Capability Identifier
-----	------	-----------------------

MMIO 0x2 0EC800E8 Read-Only (PLB), 0xEB-0xE8 Read-Only (PCI-X)
See *PCI-X Internal Core Device ID Register (PCIX0_PCIXCID)* on page 686

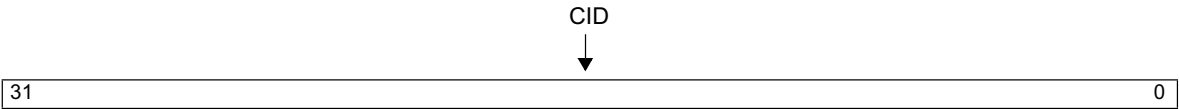


Figure 32-303. *PCI-X Internal Core Device ID Register (PCIX0_PCIXCID)*

31:0	CID	Internal Core Device ID	Read only.
------	-----	-------------------------	------------

Preliminary User’s Manual

MMIO 0x2 0EC800DE Read/Write (PLB), 0cDF-0xDE Read/Write (PCI-X)
See *PCI-X Command Register (PCIX0_PCIXCMD)* on page 684

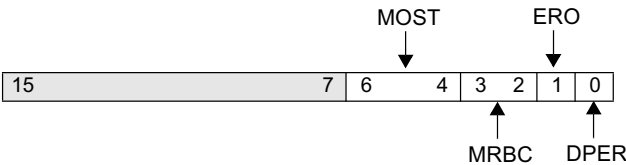


Figure 32-304. PCI-X Command Register (PCIX0_PCIXCMD)

15:7		Reserved	Returns zero when read.
6:4	MOST	Maximum Outstanding Split Transactions	This field indicates the number of outstanding split transactions allowed. If it is written to value less than the DMOS field of the PCIX0_PCIXSTS register, the PLB-PCIX Bridge limits the number of outstanding split transactions accordingly.
3:2	MRBC	Maximum Memory-Read Byte Count	This field defaults to 00 to indicate that the PLB-PCIX Bridge can request a maximum byte count of 512 bytes in a single memory read transaction. The system will not set this field higher than 00 because 512 bytes is the designed maximum memory read byte count.
1	ERO	Enable Relaxed Ordering	This bit is hardwired to a 0, indicating that the PLB-PCIX Bridge never sets the Relaxed Ordering attribute bit.
0	DPER	Data Parity Error Recovery Enable	This bit is writable but has no effect.

MMIO 0x2 0EC800E4 Read/Write (PLB), 0xE7-0xE4 Read/Write (PCI-X)
See *PCI-X Internal Debug Register (PCIX0_PCIXIDR)* on page 686



Figure 32-305. PCI-X Internal Debug Register (PCIX0_PCIXIDR)

31:0		Reserved
------	--	----------

MMIO 0x2 0EC800DD Read-Only (PLB), 0xDD Read-Only (PCI-X)

See *PCI-X Next Item Pointer Register (PCIX0_PCIXNIPTR)* on page 683

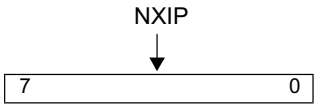


Figure 32-306. *PCI-X Next Item Pointer Register (PCIX0_PCIXNIPTR)*

7:0	NXIP	Next Item Pointer
-----	------	-------------------

MMIO 0x2 0EC800EC Read-Only (PLB), 0xEF-0xEC Read-Only (PCI-X)
See *PCI-X Internal Core Revision ID Register (PCIX0_PCIXRID)* on page 686

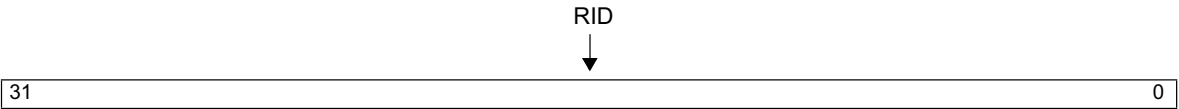


Figure 32-307. PCI-X Internal Core Revision ID Register (PCIX0_PCIXRID)

31:0	RID	Internal Core Revision ID	Read only.
------	-----	---------------------------	------------

MMIO 0x2 0EC800E0 Read/Write (PLB), 0xE3-0xE0 Read/Write (PCI-X)

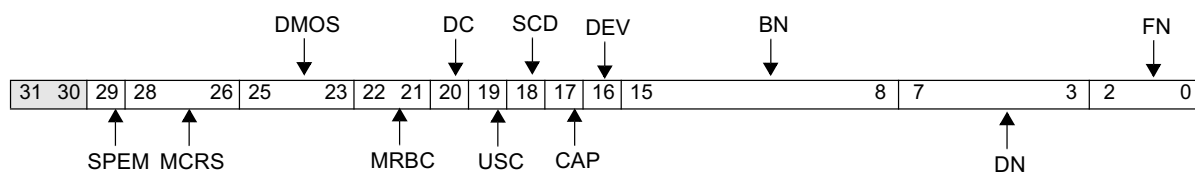
See *PCI-X Status Register (PCIX0_PCIXSTS)* on page 684

Figure 32-308. PCI-X Status Register (PCIX0_PCIXSTS)

31:30		Reserved	Returns zero when read.
29	SPEM	Received Split Completion Error Message	This bit is set if a split completion error message is received. Writing a value of 1 to this bit clears this bit.
28:26	MCRS	Designed Max Cumulative Read Size	This field is hardwired to 001b to indicate that the maximum cumulative outbound read size is less than or equal to 2 KB (it is 1.6 KB).
25:23	DMOS	Designed Max Outstanding Splits	This field indicates the maximum number of split transactions that the PLB-PCIX Bridge can have outstanding. Its reset value depends on the number of special purpose outbound read buffers present. The reset value is 010b, indicating that the PLB-PCIX Bridge can have three outstanding split completions.
22:21	MRBC	Designed Max Memory Read Byte Count	This field is hardwired to 00b to indicate that the maximum read byte count that the PLB-PCIX Bridge generates on outbound reads is 512 bytes.
20	DC	Device Complexity	This bit is hardwired to a 0 indicating that the PLB-PCIX Bridge is a simple device. (It does not handle LOCK# and never retries or disconnects split completions).
19	USC	Unexpected Split Completion	This bit is set if an unexpected split completion with Initiator Bus Number and Initiator Number of the PLB-PCIX Bridge is received. Writing a value of 1 clears this bit. This bit is hardwired to 0, because this error is not checked.
18	SCD	Split Completion Discarded	This bit is set if the PLB-PCIX Bridge discards a split completion because the requestor would not accept it. This bit is hardwired to 0, because PLB-PCIX Bridge never discards a split completion.
17	CAP	133 MHz Capable	This bit indicates 133 MHz capability. This bit is hardwired to 1.
16	DEV	64-bit Device	This bit indicates the size of the data bus of the chip containing the PLB-PCIX Bridge. 1 means the bus is 64 bits. Its value is equal to the PCI initialize Req64 enable (PR64E) strapping bit.
15:8	BN	Bus Number	This field indicates the number of the bus segment for the PLB-PCIX Bridge containing this function. These bits are read/writable from the PLB side, but read only from the PCI side.
7:3	DN	Device Number	This field indicates the number of the device for the PLB-PCIX Bridge containing this function. These bits are read/writable from the PLB side, but read only from the PCI side.
2:0	FN	Function Number	This field indicates the number of the function for the PLB-PCIX Bridge containing this function. These bits are hardwired to 000b.

MMIO 0x2 0EC800A0 Read/Write (PLB), 0xA3-0xA0 Read/Write (PCI-X)
See *PCI-X PIM 0 Local High Address (PCIX0_PIM0LAH)* on page 669

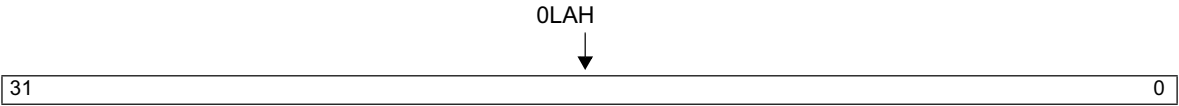


Figure 32-309. PCI-X PIM 0 Local High Address (PCIX0_PIM0LAH)

31:0	0LAH	Local Address High	Defines the upper 32 bits of the starting address of range 0 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the PIM 0 Size/Attribute High Register are actually passed to the PLB address.
------	------	--------------------	--

MMIO 0x2 0EC8007C Read/Write (PLB), 0x9f-0x9C Read/Write (PCI-X)

See *PCI-X PIM 0 Local Low Address (PCIX0_PIM0LAL)* on page 669



Figure 32-310. PCI-X PIM 0 Local Low Address (PCIX0_PIM0LAL)

31:12	0LAL	Local Address Low	Defines the starting address of range 0 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the PIM 0 Size/Attribute Low Register are actually passed to the PLB address.
11:0		Reserved	Returns zero when read.

PCIX0_PIM0SAL

PCI-X PIM 0 Size/Attribute Low Register

Preliminary User's Manual

MMIO 0x2 0EC80098 Read/Write (PLB), 0x9B-0x98 Read/Write (PCI-X)

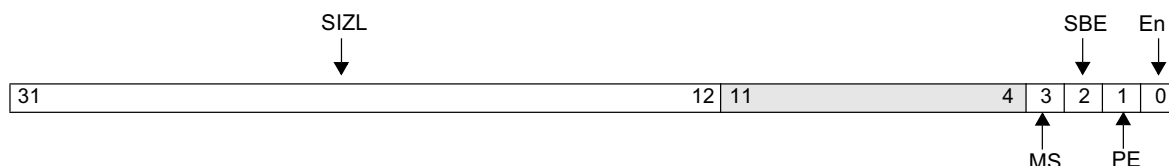
See *PCI-X PIM0 Size/Attribute Register (PCIX0_PIM0SAL)* on page 667

Figure 32-311. PCI-X PIM 0 Size/Attribute Low Register (PCIX0_PIM0SAL)

31:12	SIZL	Size Low of PIM0	The size of the PIM0 can range from 4K bytes to 2^{64} bytes and must be a power of 2. To determine the value to program in this field, use the following process: 1. Represent the desired size with a 64-bit number. Use only one bit set in that number since it is a power of two. Use all zeros to represent 2^{64} bytes. 2. Set all the bits to the left of the set bit. 3. Store bits [31:12] of the resulting number in this field. For example, if the desired PIM0 size is 16M, start with $16 \times 1024 \times 1024 = 0000_0000_0100_0000h$. Set all the bits to the left of the set bit ($FFFF_FFFF_FF00_0000h$). Bits [31:12] of that number are $FF00_0h$; therefore, $FF00_0h$ should be stored in this field.
11:4		Reserved	Always read as zero.
3	MS	Map split using PIM1	This bit determines where the first 1KB of BAR0 is mapped when Split BAR0 Enable is active. If high, the first 1 KB of BAR0 is mapped to the PLB, with the PLB address being determined by the PIM1 Local Address. This is typically used for I2O support, where an I2O core is located on the PLB. If low, the first 1 KB of BAR0 is mapped internally, enabling access to the Simple Message Passing and Inbound MSI. If Split BAR0 Enable is low, then this bit is a don't care.
2	SBE	Split BAR0 Enable	This bit enables splitting the BAR0-specified region into two ranges. When this bit is low, the entire BAR0 region is mapped to one region of PLB space, as determined by PIM0 Local Address registers. When this bit is high, the BAR0 range is divided into two regions: the first 1 KB of the BAR0 range is mapped to either a PLB region as determined by the PIM1 Local Address registers, or to the internal registers for using Simple Message Passing and/or Inbound MSI. The second region is the remainder of the BAR0 range and is mapped to the PLB address space as determined by the PIM0 Local Address Register. If this bit is low, the entire BAR0 address space is mapped to PLB address space as determined by the PIM0 Local Address Register. This bit is 0 after reset.
1	PE	Prefetch Enable	This bit determines whether this region is prefetchable. The value of this bit is also readable in BAR0 low, bit 3.
0	En	Enable	If set, enables the PCI BAR0 registers and decoders.

MMIO 0x2 0EC80098 Read/Write (PLB), 0x9B-0x98 Read/Write (PCI-X)

See *PCI-X PIM0 Size/Attribute High Register (PCIX0_PIM0SAH)* on page 668

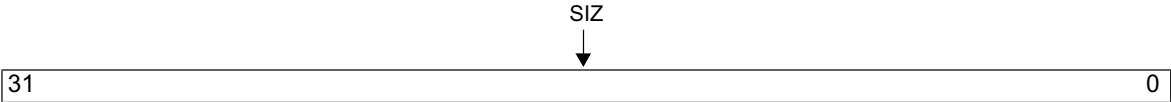


Figure 32-312. PCI-X PIM 0 Size/Attribute High Register (PCIX0_PIM0SAH)

31:0	SIZ	Size High of PIM0	<p>The size of the PIM0 can range from 4K bytes to 2⁶⁴ bytes and must be a power of 2. To determine the value to program in this field, use the following process:</p> <ol style="list-style-type: none">1. Represent the desired size with a 64-bit number. Use only one bit set in that number since it is a power of two. Use all zeros to represent 2⁶⁴ bytes.2. Set all the bits to the left of the set bit.3. Store bits [63:32] of the resulting number in this field. <p>For example, if the desired PIM0 size is 16M, start with 16x1024x1024 = 0000_0000_0100_0000h. Set all the bits to the left of the set bit (FFFF_FFFF_FF00_0000h). Bits [63:32] of that number are FFFF_FFFFh; therefore, FFFF_FFFFh should be stored in this field.</p>
------	-----	-------------------	--

MMIO 0x2 0EC800AC Read/Write (PLB), 0xAF-0xAC Read/Write (PCI-X)

See *PCI-X PIM1 Local High Address (PCIX0_PIM1LAH)* on page 671

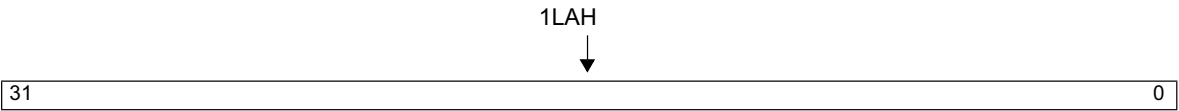


Figure 32-313. *PCI-X PIM 1 Local High Address (PCIX0_PIM1LAH)*

31:0	1LAH	Local Address High	Defines the upper 32 bits of the starting address of range 1 in PLB space that is mapped to PCI I/O space. Only the bits that are 1 in the PIM 0 Size/Attribute High Register are actually passed to the PLB address.
------	------	--------------------	---

MMIO 0x2 0EC800A8 Read/Write (PLB), 0xAB-0xA8 Read/Write (PCI-X)

See *PCI-X PIM 1 Local Low Address (PCIX0_PIM1LAL)* on page 671



Figure 32-314. PCI-X PIM 1 Local Low Address (PCIX0_PIM1LAL)

31:12	1LAL	Local Address Low	Defines the starting address of range 1 in PLB space that is mapped to PCI I/O space. Only the bits that are 1 in the PIM 1 Size/Attribute Register are actually passed to the PLB address. Only bits 31:12 are writable.
11:0		Reserved	Returns zero when read.

MMIO 0x2 0EC800A4 Read/Write (PLB), 0xA7-0xA4 Read/Write (PCI-X)

See *PCI-X PIM1 Size/Attribute Register (PCIX0_PIM1SA)* on page 670



Figure 32-315. PCI-X PIM 1 Size/Attribute Register (PCIX0_PIM1SA)

31:8	SIZE	Size	SIZE defines the size of the region of PCI I/O space that is mapped to local (PLB) space through PIM 1. The size for the PIM1 is hardcoded to 256 bytes; therefore, read to these bits returns all ones. This size has no effect on the PIM0 1 KB region.
7:1		Reserved	Returns zero when read.
0	En	Enable	If set, enables the PCI BAR1 registers and decoder.

MMIO 0x2 0EC800B8 Read/Write (PLB), 0xBB-0xB8 Read/Write (PCI-X)

See *PCI-X PIM 2 Local Address High (PCIX0_PIM2LAH)* on page 675

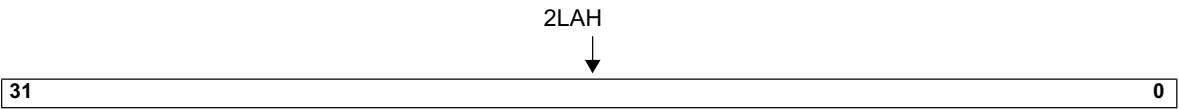


Figure 32-316. PCI-X PIM 2 Local Address High (PCIX0_PIM2LAH)

31:0	2LAH	Local Address High	Defines the upper 32 bits of the starting address of range 2 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the PIM 2 Size/Attribute High Register are actually passed to the PLB address.
------	------	--------------------	--

MMIO 0x2 0EC800B4 Read/Write (PLB), 0xB7-0xB4 Read/Write (PCI-X)

See *PCI-X PIM2 Local Low Address (PCIX0_PIM2LAL)* on page 674



Figure 32-317. PCI-X PIM 2 Local Low Address (PCIX0_PIM2LAL)

31:12	2LAL	Local Address Low	Defines the starting address of range 2 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the PIM 2 Size/Attribute Low Register are actually passed to the PLB address. Only bits 31:12 are writable.
11:0		Reserved	Returns zero when read.

MMIO 0x2 0EC800B0 Read/Write (PLB), 0xB3-0xB0 Read/Write (PCI-X)

See *PCI-X PIM 2 Size/Attribute Low Register (PCIX0_PIM2SAL)* on page 672

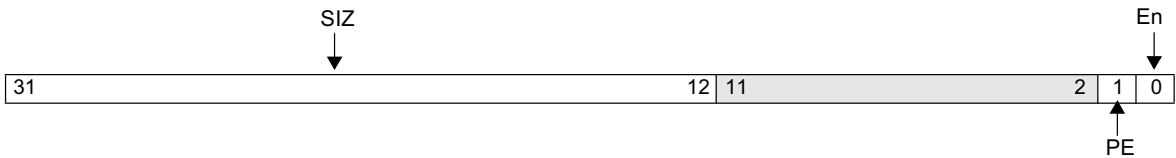
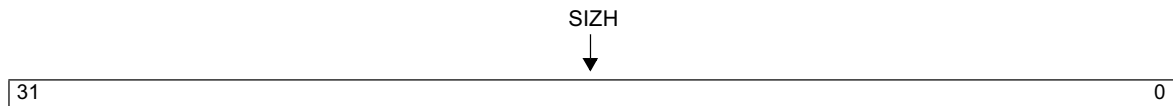


Figure 32-318. PCI-X PIM 2 Size/Attribute Low Register (PCIX0_PIM2SAL)

31:12	SIZ	Size Low of PIM2	<p>The size of the PIM2 can range from 4K bytes to 2^{64} bytes and must be a power of 2. To determine the value to program in this field, use the following process:</p> <ol style="list-style-type: none">1. Represent the desired size with a 64-bit number. Use only one bit set in that number since it is a power of two. Use all zeros to represent 2^{64} bytes2. Set all the bits to the left of the set bit.3. Store bits [31:12] of the resulting number in this field. <p>For example, if the desired PIM2 size is 16M, start with $16 \times 1024 \times 1024 = 0000_0000_0100_0000h$. Set all the bits to the left of the set bit (FFFF_FFFF_FF00_0000h). Bits [31:12] of that number are FF00_0h; therefore, FF00_0h should be stored in this field.</p>
11:2		Reserved	Returns zero when read.
1	PE	Prefetch Enable	This bit determines if this region is prefetchable. The value of this bit is also readable in BAR2 low, bit 3.
0	En	Enable	If set, enables the PCI PIM2 BAR Register and the Expansion ROM BAR Register and the decoder.

See *PCI PIM2 Size/Attribute High Register (PCIX0_PIM2SAH)* on page 673



31:0	SIZH	<p>The size of the PIM2 can range from 4K bytes to 2^{64} bytes and must be a power of 2. To determine the value to program in this field, use the following process:</p> <ol style="list-style-type: none"> 1. Represent the desired size with a 64-bit number. Use only one bit set in that number since it is a power of two. Use all zeros to represent 2^{64} bytes. 2. Set all the bits to the left of the set bit. 3. Store bits [63:32] of the resulting number in this field. <p>For example, if the desired PIM2 size is 16M, start with $16 \times 1024 \times 1024 = 0000_0000_0100_0000h$. Set all the bits to the left of the set bit (FFFF_FFFF_FF00_0000h). Bits [63:32] of that number are FFFF_FFFFh; therefore, FFFF_FFFFh should be stored in this field.</p>
------	------	--

MMIO 0x2 0EC80060 Read-Only (PLB), 0x63-0x60 Read-Only (PCI-X)

See *PCI-X PLB Slave Error Address High (PCIX0_PLBBEARH)* on page 660

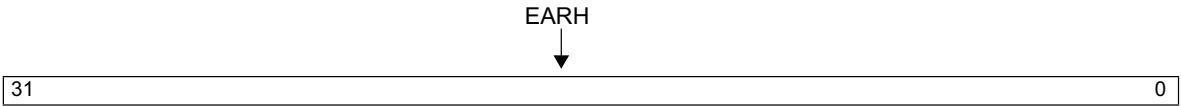


Figure 32-320. PCI-X PLB Slave Error Address High (PCIX0_PLBBEARH)

31:0	EARH	PLB Slave Error Address High
------	------	------------------------------

MMIO 0x2 0EC8005C Read-Only PLB, 0x5f-0x5C Read-Only (PCI-X)
See *PCI-X PLB Slave Error Address Low (PCIX0_PLBBEARL)* on page 659

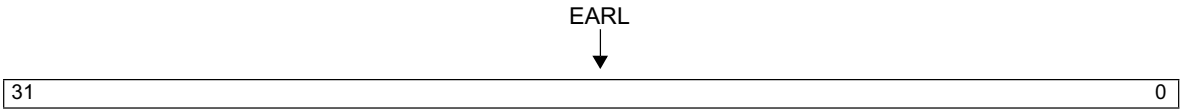


Figure 32-321. PCI-X PLB Slave Error Address Low (PCIX0_PLBBEARL)

31:0	EARL	PLB Slave Error Address Low
------	------	-----------------------------

MMIO 0x2 0EC80058 Read-Only (PLB), 0x5b-0x58 (PCI-X)

See *PCI-X PLB Slave Error Attribute Register (PCIX0_PLBBESR)* on page 659

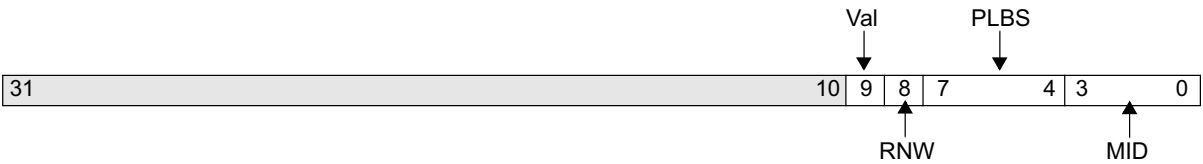


Figure 32-322. PCI-X PLB Slave Error Attribute Register (PCIX0_PLBBESR)

31:10		Reserved	Always read as zero.
9	Val	Valid	Indicates if bits 8:0 are valid (This bit is only meaningful when an outbound error is indicated in the PCIX0_ERRSTS register). Note: It is possible for a parity error to be detected too late for the PLB information to be saved.
8	RNW	RNW	RNW from the PLB master
7:4	PLBS	size	Size from the PLB master
3:0	MID	masterID	MasterID of the PLB master

MMIO 0x2 0EC800D2 Read-Only (PLB), 0xD3-0xD2 Read-Only (PCI-X)

See *PCI-X Power Management Capabilities Register (PCIX0_PMC)* on page 680

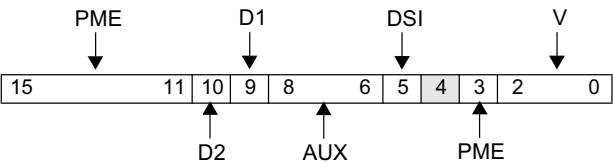


Figure 32-323. PCI-X Power Management Capabilities Register (PCIX0_PMC)

15:11	PME	PME_Support	The PLB-PCIX Bridge does not support PME#; therefore, all the bits are hardwired to 0.
10	D2	D2_Support	This bit determines if the D2 Power Management State is supported. PLB-PCIX Bridge does not support the D2 Power Management State; therefore, this bit is hardwired to a 0.
9	D1	D1_Support	This bit determines if the D1 Power Management State is supported. PLB-PCIX Bridge supports the D1 Power Management State; therefore, this bit is hardwired to 1.
8:6	AUX	Aux_Current	PLB-PCIX Bridge does not support PME#; therefore, this field is unsupported. All the bits are hardwired to 0.
5	DSI	Device Specific Initialization (DSI)	The DSI bit indicates whether special initialization of this function is required (beyond the standard PCI configuration header) before the generic class device driver is able to use it. This bit is hardwired to 0.
4		Reserved	Returns zero when read.
3	PME	PME Clock	This bit is hardwired to 0 indicating that the function does not support PME# generation in any state.
2:0	V	Version	Returns a value of 010b on reads, indicating that this function complies with the <i>PCI Bus Power Management Interface Specification</i> , Version 1.1.

MMIO 0x2 0EC800D0 Read/Write (PLB), 0xD0 Read-Only (PCI-X)

See *PCI-X Power Management Capability Identifier Register (PCIX0_PMCAPID)* on page 679

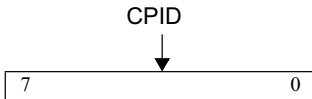


Figure 32-324. *PCI-X Power Management Capabilities Identifier (PCIX0_PMCAPID)*

7:0	CPID	Capabilities Identifier	When read by system software as 01h indicates that PLB-PCIX Bridge supports power management and the data structure currently being pointed to is the PCI power management capability structure.
-----	------	-------------------------	--

MMIO 0x2 0EC800D6 Read-Only (PLB), 0xD6 Read-Only (PCI-X)

See *PCI-X PMCSR PCI-to-PLB-PCIX Bridge Support Extensions (PCIX0_PMCSRBSE)* on page 681

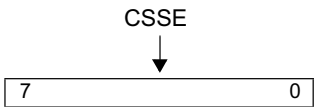


Figure 32-325. *PCI-X PMCSR PCI-to-PLB-PCIX Bridge Support Extensions (PCIX0_PMCSRBSE)*

7:0	CSSE	Power Management Control/Status PCI-to-PLB-PCIX Bridge Support Extensions.	Required for all PCI-to-PCI bridges. Always read as zero.
-----	------	--	---

MMIO 0x2 0EC800D4 Read/Write (PLB), 0xD5-0xD4 Read/Write (PCI-X)

See *PCI-X Power Management Control/Status Register (PCIX0_PMCSR)* on page 681

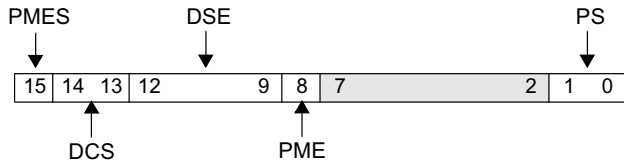


Figure 32-326. *PCI-X Power Management Control/Status Register (PCIX0_PMCSR)*

15	PMES	PME_Status	The PLB-PCIX Bridge does not support PME#; therefore, this bit is hardwired to a 0.
14:13	DSC	Data_Scale	The PLB-PCIX Bridge does not support the Data Register; therefore, these bits are hardwired to 00.
12:9	DSE	Data_Select	The PLB-PCIX Bridge does not support the Data Register; therefore, these bits are hardwired to 0000.
8	PME	PME_En	PLB-PCIX Bridge does not support PME# generation; therefore, this bit is hardwired to a 0.
7:2		Reserved	Returns zero when read.
1:0	PS	PowerState 00 D0 01 D1 10 D2 11 D3-Hot	This 2-bit R/W field is used both to determine the current power state of a function and to set the function into a new power state. If software attempts to write an unsupported, optional state to this field, the value of this field does not change. Writing this register may cause the PMSC_RR to change.

MMIO 0x2 0EC800D7 Read-Only (PLB), 0xD7 Read-Only (PCI-X)

See *PCI-X Power Management Data (PCIX0_PMDATA)* on page 682

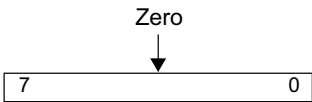


Figure 32-327. PCI-X Power Management Data (PCIX0_PMDATA)

7:0	Zero	Optional register	Returns zero when read.
-----	------	-------------------	-------------------------

MMIO 0x2 0EC800D1 Read-Write (PLB), 0xD1 Read-Only (PCI-X)

See *PCI-X Power Management Next Item Pointer Register (PCIX0_PMNIPTR)* on page 680

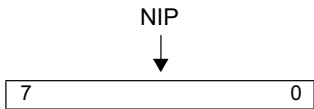


Figure 32-328. PCI-X Power Management Next Item Pointer (PCIX0_PMNIPTR)

7:0	NIP	Next Item Pointer	Describes the location of the next item in the function's capability list. The Next Item Pointer defaults to pointing to the capability structure for PCI-X located at DCh in the configuration space. This can be overwritten with 00h if support for PCI-X is not desired.
-----	-----	-------------------	--

MMIO 0x2 0EC800D8 Read/Write (PLB), 0xD8 Read/Write (PCI-X)

See *PCI-X Power Management State Change Request Register (PCIX0_PMSCRR)* on page 682

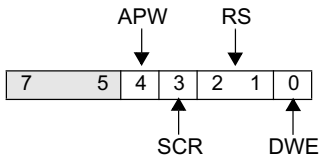


Figure 32-329. PCI-X Power Management State Change Request Register (PCIX0_PMSCRR)

7:5		Reserved	Returns zero when read.
4	APW	Accept PMCSR Write	The local processor sets this bit when it is ready to change the power management state. The bit is cleared when the host configuration write to the PMCSR is accepted. This bit is always a 1 if the Delayed Write Enable bit (bit 0) is a 0. (It is possible for the local processor to write a 0 to this bit).
3	SCR	State Change Request	The PLB-PCIX Bridge sets this bit when there is a host write to the PMCS for a power management state change request. This drives an interrupt to the local processor informing it of a state change request. The local processor must simultaneously clear this bit and set the Accept PMCSR Write bit when it is ready to change the state. After clearing this bit, new request will not be detected until the outstanding delayed write is accepted. (It is possible for the local processor to write a 1 to this bit.)
2:1	RS	Requested State	This field indicates the new power management state requested using the delayed host write to the PMCSR.
0	DWE	Delayed Write Enable 1 - Delayed write 0 - Immediate write	When 1, any configuration write to the PMCSR is completed as a delayed write. All writes to PMCSR are retried until the local processor sets the Accept PMCSR Write bit (bit 4). When 0, any configuration write to the PMCSR is completed immediately. This bit is a don't care if a host write to PMCSR requests a state change from D3hot to D0. The default state is 0 to prevent bus hang if the PM_Int service routine is not ready to go.

MMIO 0x2 0EC8006C Read/Write (PLB), 0x6F-0x6C Read/Write (PCI-X)

See *PCI-X POM 0 Local High Address (PCIX0_POM0LAH)* on page 661

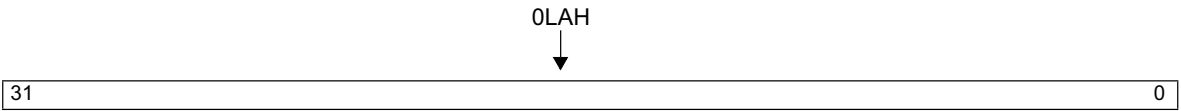


Figure 32-330. *PCI-X POM 0 Local High Address (PCIX0_POM0LAH)*

31:0	0LAH	Local Address High	Defines the upper 32 bits of the starting address of range 0 in PLB space that is mapped to PCI memory. All bits are writable.
------	------	--------------------	--

MMIO 0x2 0EC80068 Read/Write (PLB), 0x6B-0x68 Read/Write (PCI-X)

See *PCI-X POM 0 Local Low Address (PCIX0_POM0LAL)* on page 660



Figure 32-331. PCI-X POM 0 Local Low Address (PCIX0_POM0LAL)

31:20	0LAL	Local Address Low	Defines the starting address of range 0 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the POM 0 size are actually used to determine the starting address, all other bits are don't cares.
19:0		Reserved	Returns zero when read.

Preliminary User’s Manual

MMIO 0x2 0EC80078 Read/Write (PLB), 0x7B-0x78 Read/Write (PCI-X)
See PCI-X POM 0 PCI Address High (PCIX0_POM0PCIAH) on page 662

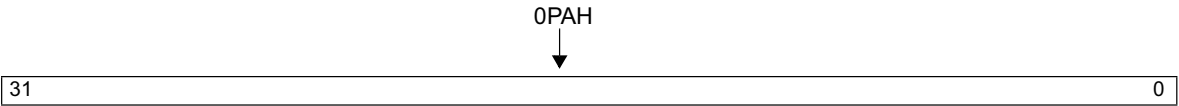


Figure 32-332. PCI-X POM 0 PCI Address High (PCIX0_POM0PCIAH)

31:0	0PAH	PCI High Address
------	------	------------------

MMIO 0x2 0EC80074 Read/Write (PLB), 0x77-0x74 Read/Write (PCI-X)
See *PCI-X POM 0 PCI Address Low (PCIX0_POM0PCIAL)* on page 662



Figure 32-333. PCI-X POM 0 PCI Address Low (PCIX0_POM0PCIAL)

31:20	0PAL	PCI Low Address
19:0		Reserved Always read as zero.

Preliminary User’s Manual

MMIO 0x2 0EC80070 Read/Write (PLB), 0x73-0x70 Read/Write (PCI-X)
PCI-X POM 0 Size/Attribute Register (PCIX0_POM0SA) on page 661



Figure 32-334. PCI-X POM 0 Size/Attribute Register (PCIX0_POM0SA)

31:20	SIZE	Size of POM0.	<p>The size of the POM0 can range from 1M to 4G and must be a power of 2. To determine the value to program in this field, use the following process:</p> <ol style="list-style-type: none">1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. Use all zeros to represent 4 GB.2 Set all the bits to the left of the set bit.3. Store bits [31:20] of the resulting number in this field. <p>For example, if the desired POM0 size is 16M, start with 16x1024x1024 = 0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:20] of that number are FF0h; therefore, FF0h should be stored in this field.</p>
19:1		Reserved	Always read as zero.
0	En	Enable Mapping 0 Disable 1 Enable	<p>This bit determines if range 0 is enabled to map PLB space to PCI memory space.</p> <p>Note: The POM 0 Local Address, POM 0 PCI Low Address, and POM 0 PCI High Address must be initialized before enabling.</p> <p>This field has a value of 0 after reset.</p>

MMIO 0x2 0EC80080 Read/Write (PLB), 0x83-0x80 Read/Write (PCI-X)
See *PCI-X POM 1 Local Address High (PCIX0_POM1LAH)* on page 663

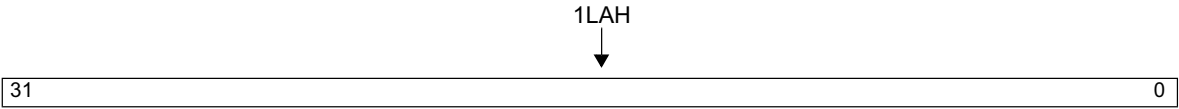


Figure 32-335. PCI-X POM 1 Local Address High (PCIX0_POM1LAH)

31:0	1LAH	Local Address High	Defines the starting address of range 1 in PLB space that is mapped to PCI memory. Only the bits that are 1 in the POM 1 Size are actually used to determine the starting address. All other bits are don't cares.
------	------	--------------------	--

MMIO 0x2 0EC8007C Read/Write (PLB), 0x7F-0x7C Read/Write (PCI-X)

See *PCI-X POM 1 Local Address Low (PCIX0_POM1LAL)* on page 663



Figure 32-336. PCI-X POM 1 Local Address Low (PCIX0_POM1LAL)

31:20	1LAL	Local Address Low	Defines the starting address of range 1 in PLB space that is mapped to PCI memory.
19:0		Reserved	Returns zero when read.

MMIO 0x2 0EC80084 Read/Write (PLB), 0x87-0x84 Read/Write (PCI-X)
See *PCI-X POM 1 Size/Attribute Register (PCIX0_POM1SA)* on page 664

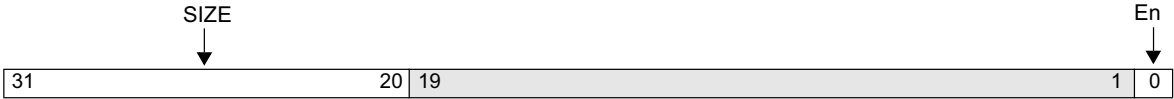


Figure 32-337. PCI-X POM 1 Size/Attribute Register (PCIX0_POM1SA)

31:20	SIZE	Size of POM 1	<p>The size of the POM 1 can range from 1M to 4G and must be a power of 2. To determine the value to program in this field, use the following process:</p> <ol style="list-style-type: none">1. Represent the desired size with a 32-bit number. Use only one bit set in that number since it is a power of two. Use all zeros to represent 4 GB.2 Set all the bits to the left of the set bit.3. Store bits [31:20] of the resulting number in this field. <p>For example, if the desired POM 1 size is 16M, start with 16x1024x1024 = 0100_0000h. Set all the bits to the left of the set bit (FF00_0000h). Bits [31:20] of that number are FF0h; therefore, FF0h should be stored in this field.</p>
19:1		Reserved	Always read as zero.
0	En	Enable mapping 1 Enable 0 Disable	<p>This bit determines if range 1 is enabled to map PLB space to PCI memory space.</p> <p>Note: The POM 1 Local Address, POM 1 PCI Low Address, and POM 1 PCI High Address must be initialized before enabling.</p> <p>This field has a value of 0 after reset.</p>

Preliminary User’s Manual

MMIO 0x2 0EC8008C Read/Write (PLB), 0x8F-0x8C Read/Write (PCI-X)
See *PCI-X POM 1 PCI Address High (PCIX0_POM1PCIAH)* on page 665

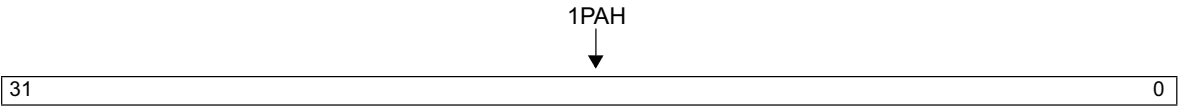


Figure 32-338. PCI-X POM 1 PCI Address High (PCIX0_POM1PCIAH)

31:0	1PAH	PCI Address High
------	------	------------------

MMIO 0x2 0EC80088 Read/Write (PLB), 0x8b-0x88 Read/Write (PCI-X)

See *PCI-X POM 1 PCI Address Low (PCIX0_POM1PCIAL)* on page 665



Figure 32-339. PCI-X POM 1 PCI Address Low (PCIX0_POM1PCIAL)

31:20	1PAL	PCI Address Low
19:0		Reserved Returns zero when read.

MMIO 0x2 0EC80090 Read/Write (PLB), 0x93-0x90 Read/Write (PCI-X)

See *PCI-X POM 2 Size/Attribute Register (PCIX0_POM2SA)* on page 666



Figure 32-340. PCI-X POM 2 Size/Attribute Register (PCIX0_POM2SA)

31:1		Reserved	Always read as zero.
0	En	Enables mapping 1 Enable 0 Disable	This bit determines if range 2 is enabled to map PLB space to PCI memory space. A value of 1 enables the mapping.

MMIO 0x2 0EC80008 Read/Write (PLB), 0x08 Read-Only (PCI-X)
See *PCI-X Revision ID Register (PCIX0_REVID)* on page 641

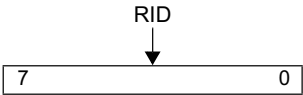


Figure 32-341. PCI-X Revision ID Register (PCIX0_REVID)

7:0	RID	Revision ID	Revision level of device.
-----	-----	-------------	---------------------------

MMIO 0x2 0EC8002E Read/Write (PLB), 0x2F-0x2E Read-Only (PCI-X)

See *PCI-X Subsystem ID Register (PCIX0_SBSYSID)* on page 650

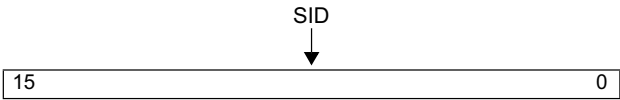


Figure 32-342. *PCI-X Subsystem ID Register (PCIX0_SBSYSID)*

15:0	SID	PCI Subsystem ID
------	-----	------------------

MMIO 0x2 0EC8002C Read/Write (PLB), 0x2D-0x2C Read-Only (PCI-X)
See *PCI-X Subsystem Vendor ID Register (PCIX0_SBSYSVID)* on page 649

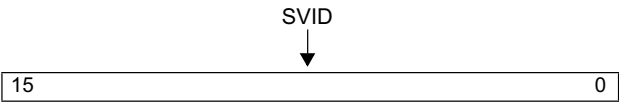


Figure 32-343. PCI-X Subsystem Vendor ID Register (PCIX0_SBSYSVID)

15:0	SVID	PCI Subsystem Vendor ID
------	------	-------------------------

MMIO 0x2 0EC80006 Read/Write (PLB), 0x07-0x06 Read/Write (PCI-X)

See *PCI-X Status Register (PCIX0 STATUS)* on page 640

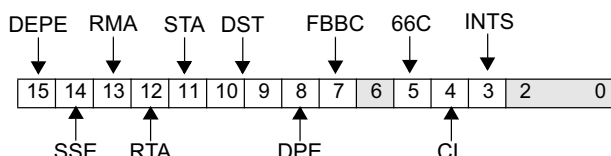


Figure 32-344. PCI-X Status Register (PCIX0 STATUS)

15	DEPE	Detected Parity Error Write 1 to clear.	PLB-PCIX Bridge sets this bit whenever it detects a PCI bus parity error. This bit is maskable only by bit 8 (PO) of the PCI Error Enable register. This bit is set when the following occurs: PCI address bus parity error detected when PLB-PCIX Bridge is a target. PCI data bus parity error detected when a PCI master writes to PLB memory (PLB-PCIX Bridge is the target). PCI data bus parity error detected when PLB-PCIX Bridge masters a PCI read cycle. Writing a 1 to this bit resets it to 0.
14	SSE	Signaled System Error Write 1 to clear.	Signaled system error. PLB-PCIX Bridge sets this bit if it asserts PCI_SERR#. Writing a 1 to this bit resets it to 0.
13	RMA	Received Master Abort Write 1 to clear.	This bit is set whenever PLB-PCIX Bridge terminates a PCI cycle for which it is the master with master abort (except configuration and special cycles) and the Master Abort Error Enable bit of the Error Enable Register is set. Writing a 1 to this bit resets it to 0.
12	RTA	Received Target Abort Write 1 to clear.	PLB-PCIX Bridge sets this bit whenever a PCI cycle for which it is the master is terminated with target abort and the Target Abort Error Enable bit of the Error Enable Register is set. Writing a 1 to this bit resets it to 0.
11	STA	Signaled Target Abort Write 1 to clear.	PLB-PCIX Bridge never signals a target abort. This bit is always 0.
10:9	DST	$\overline{\text{PCIDevSel}}$ Response Timing Read-only.	PCI_DEVSEL# response timing. PLB-PCIX Bridge asserts PCI_DEVSEL# on the second clock (also known as medium response time) after PCI_FRAME# is asserted by a PCI master attempting to access memory on the PLB side of the bridge. These bits are read-only and always return 01b when read.
8	DPE	Data Parity Error Detected Write 1 to clear.	This bit is set when the following two conditions are met: PLB-PCIX Bridge detects a data parity error (PCI_PERR# asserted) when it is the master on a PCI read cycle, or it is the master when it samples PCI_PERR# asserted on a PCI write cycle. The Parity Error Response bit (bit 6 of the PCI Command Register) is set. Writing a 1 to this bit resets it to 0.
7	FBBC	Fast Back-to-Back Capable Read-only; returns zero when read.	Indicates that the PCI target is capable of accepting fast back-to-back transactions when the transactions are not to the same agent. PLB-PCIX Bridge target does accept this type of fast back-to-back transaction, therefore, this bit is read-only and is 1 when in PCI-Conv mode. When in PCI-X mode, it is always 0.
6		Reserved	This bit is hardcoded to zero. This bit was the UDF support bit in PCI 2.1.
5	66C	66 MHz Capable	Indicates that the device is able to run at 66 MHz. Hardcoded to 1.
4	CL	Capabilities List	Indicates whether or not this device implements the pointer for a new capabilities linked list at offset 34h. This bit is read-only and returns 1 when read, indicating that the value read at offset 34h is a pointer in configuration space to a linked list of new capabilities.
3	INTS	Interrupt Status	This read-only bit reflects the state of the outbound interrupt in the PLB-PCIX Bridge. Only when the Interrupt Disable bit in the PCICx_CMD register is 0, and this Interrupt Status bit is 1, will the PLB-PCIX Bridge's INTA# signal be asserted. Setting the Interrupt Disable bit to 1 has no effect on the state of this bit.
2:0		Reserved	These bits are reserved, and return zeros when read.

MMIO 0x2 0EC80000 Read/Write (PLB), 0x01-0x00 Read-Only (PCI-X)
See *PCI-X Vendor ID Register (PCIX0_VENDID)* on page 638

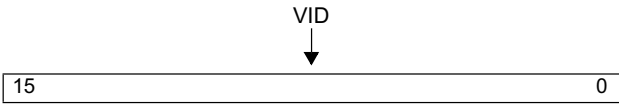


Figure 32-345. PCI-X Vendor ID Register (PCIX0_VENDID)

15:0	VID	Vendor ID
------	-----	-----------

MMIO 0x2 0EC800F2 Read/Write (PLB), 0xF2 Read/Write (PCI-X)

See *PCI-X VPD Address Register (PCIX0_VPDADR)* on page 688

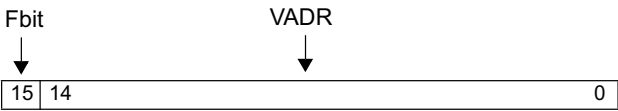


Figure 32-346. PCI-X VPD Address Register (PCIX0_VPDADR)

15	Fbit	Flag Bit	The flag bit is written from the PCI side when the VPD address is written. It is written by the PLB side to indicate the completion of a read or write operation. Please see the PCI Specification, Version 2.3, Appendix I for the detailed operation of the flag bit.
14:0	VADR	VPD Address	The VPD address is writable from the PCI side. It is used to specify the address of the VPD datum to be accessed. This field is not writable from the PLB side.

MMIO 0x2 0EC800F0 Read (PLB), 0xF0 Read (PCIX)

See *PCI-X VPD Capability Identifier - Cap_ID (Offset = 0) (PCI0_VPDCAPID)* on page 687

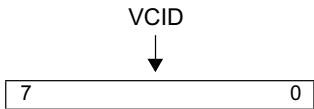


Figure 32-347. PCI-X VPD Capability Identifier - Cap_ID (Offset=0) (PCIX0_VPDCAPID)

7:0	VCID	Capability Identifier
-----	------	-----------------------

MMIO 0x2 0EC800F4 Read/Write (PLB), 0xF4 Read/Write (PCIX)

See *PCI-X VPD Data Register (PCIX0_VPDDATA)* on page 688

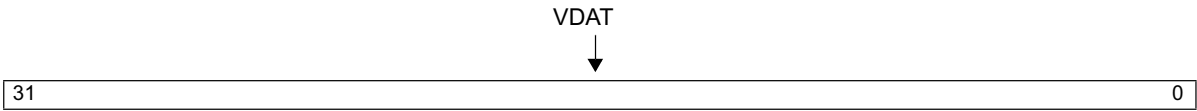


Figure 32-348. PCI-X VPD Data Register (PCIX0_VPDDATA)

31:0	VDAT	VPD Data	The VPD data register is writable from both the PCI and PLB sides. The PCI side puts data to be written to VPD in this register. The PLB side puts data that was read from VPD in this register.
------	------	----------	--

MMIO 0x2 0EC800F1 Read/Write (PLB), 0xF1 Read (PCIX)

See *PCI-X VPD Next Item Pointer Register (PCIX0_VPDNIPTR)* on page 687

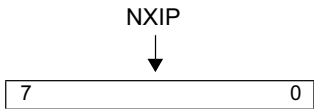


Figure 32-349. PCI-X VPD Next Item Pointer Register (PCIX0_VPDNIPTR)

7:0	NXIP	Next Item Pointer
-----	------	-------------------

DCR 0x093 Read-Only

See *PLB to OPB Bridge Error Address Register High (POB0_BEARH)* on page 91.



Figure 32-350. PLB to OPB Bridge Error Address Register High (POB0_BEARH)

0:27		Reserved
28:31	10'h0B3	Upper address of bus error

DCR 0x092 Read-Only

See *PLB to OPB Bridge Error Address Register Low (POB0_BEARL)* on page 91.



Figure 32-351. *PLB to OPB Bridge Error Address Register Low (POB0_BEARL)*

0:31	0x0B2	Lower address of bus error
------	-------	----------------------------

DCR 0x090 Read/Clear

See *PLB to OPB Bridge Error Status Register 0 (POB0_BESR0)* on page 89.

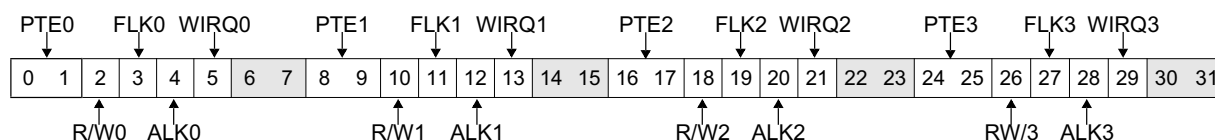


Figure 32-352. PLB to OPB Bridge Error Status Register 0 (POB0_BESR0)

0:1	PTE0	PLB Timeout Error Status Master 0 00 No master 0 error occurred 01 Master 0 timeout error occurred 10 Master 0 slave error occurred 11 Reserved Master 0 is the read-only instruction cache unit (ICU)
2	R/W0	Read Write Status Master 0 0 Master 0 error operation is a write 1 Master 0 error operation is a read
3	FLK0	POB0_BESR0 Field Lock Master 0 0 Master 0 POB0_BESR0 field is unlocked 1 Master 0 POB0_BESR0 field is locked
4	ALK0	POB0_BEAR Address Lock Master 0 0 Master 0 POB0_BEAR address is unlocked 1 Master 0 POB0_BEAR address is locked
5	WIRQ0	Write Error Interrupt Master 0 0 No write error detected - master 0 interrupt request is inactive 1 Write error detected - master 0 interrupt request is active
6:7		Reserved
8:9	PTE1	PLB Timeout Error Status Master 1 00 No master 1 error occurred 01 Master 1 timeout error occurred 10 Master 1 slave error occurred 11 Reserved Master 1 is the read-only data cache unit (DCU)
10	R/W1	Read/Write Status Master 1 0 Master 1 error operation is a write 1 Master 1 error operation is a read
11	FLK1	POB0_BESR0 Field Lock Master 1 0 Master 1 POB0_BESR0 field is unlocked 1 Master 1 POB0_BESR0 field is locked
12	ALK1	POB0_BEAR Address Lock Master 1 0 Master 1 POB0_BEAR address is unlocked 1 Master 1 POB0_BEAR address is locked
13	WIRQ1	Write Error Interrupt Master 1 0 No write error detected - master 1 interrupt request is inactive 1 Write error detected - master 1 interrupt request is active
14:15		Reserved

POB0_BESR0 (cont.)

PLB to OPB Bridge Error Status Register 0

Preliminary User's Manual

16:17	PTE2	PLB Timeout Error Status Master 2 00 No master 2 error occurred 01 Master 2 timeout error occurred 10 Master 2 slave error occurred 11 Reserved	Master 2 is the write-only data cache unit (DCU)
18	R/W2	Read/Write Status Master 2 0 Master 2 error operation is a write 1 Master 2 error operation is a read	
19	FLK2	POB0_BESR0 Field Lock Master 2 0 Master 2 POB0_BESR0 field is unlocked 1 Master 2 POB0_BESR0 field is locked	
20	ALK2	POB0_BEAR Address Lock Master 2 0 Master 2 POB0_BEAR address is unlocked 1 Master 2 POB0_BEAR address is locked	
21	WIRQ2	Write Error Interrupt Master 2 0 No write error detected - master 2 interrupt request is inactive 1 Write error detected - master 2 interrupt request is active	
22:23		Reserved	
24:25	PTE3	PLB Timeout Error Status Master 3 00 No master 3 error occurred 01 Master 3 timeout error occurred 10 Master 3 slave error occurred 11 Reserved	Master 3 is the PCIX bridge controller
26	R/W3	Read/Write Status Master 3 0 Master 3 error operation is a write 1 Master 3 error operation is a read	
27	FLK3	POB0_BESR0 Field Lock Master 3 0 Master 3 POB0_BESR0 field is unlocked 1 Master 3 POB0_BESR0 field is locked	
28	ALK3	POB0_BEAR Address Lock Master 3 0 Master 3 POB0_BEAR address is unlocked 1 Master 3 POB0_BEAR address is locked	
29	WIRQ3	Write Error Interrupt Master 3 0 No write error detected - master 3 interrupt request is inactive 1 Write error detected - master 3 interrupt request is active	
30:31		Reserved	

DCR 0x094 Read/Clear

See *PLB to OPB Bridge Error Status Register 1 (POB0_BESR1)* on page 92.

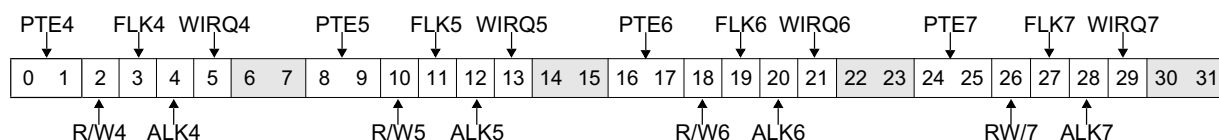


Figure 32-353. *PLB to OPB Bridge Error Status Register 1 (POB0_BESR1)*

0:1	PTE4	PLB Timeout Error Status Master 4 00 No master 4 error occurred 01 Master 4 timeout error occurred 10 Master 4 slave error occurred 11 Reserved Master 4 is the I2O messaging unit
2	R/W4	Read Write Status Master 4 0 Master 4 error operation is a read 1 Master 4 error operation is a write
3	FLK4	POB0_BESR0 Field Lock Master 4 0 Master 4 POB0_BESR0 field is unlocked 1 Master 4 POB0_BESR0 field is locked
4	ALK4	POB0_BEAR Address Lock Master 4 0 Master 4 POB0_BEAR address is unlocked 1 Master 4 POB0_BEAR address is locked
5	WIRQ4	Write Error Interrupt Master 4 0 No write error detected - master 4 interrupt request is inactive 1 Write error detected - master 4 interrupt request is active
6:7		Reserved
8:9	PTE5	PLB Timeout Error Status Master 5 00 No master 5 error occurred 01 Master 5 timeout error occurred 10 Master 5 slave error occurred 11 Reserved Master 5 is the MAL controller
10	R/W5	Read/Write Status Master 5 0 Master 5 error operation is a write 1 Master 5 error operation is a read
11	FLK5	POB0_BESR1 Field Lock Master 5 0 Master 5 POB0_BESR1 field is unlocked 1 Master 5 POB0_BESR1 field is locked
12	ALK5	POB0_BEAR Address Lock Master 5 0 Master 5 POB0_BEAR address is unlocked 1 Master 5 POB0_BEAR address is locked
13	WIRQ5	Write Error Interrupt Master 5 0 No write error detected - master 5 interrupt request is inactive 1 Write error detected - master 5 interrupt request is active
14:15		Reserved
16:17	PTE6	PLB Timeout Error Status Master 6 00 No master 6 error occurred 01 Master 6 timeout error occurred 10 Master 6 slave error occurred 11 Reserved Master 6 is the DMA controller

POB0_BESR1 (cont.)

PLB to OPB Bridge Error Status Register 1

Preliminary User's Manual

18	R/W6	Read/Write Status Master 6 0 Master 6 error operation is a write 1 Master 6 error operation is a read
19	FLK6	POB0_BESR1 Field Lock Master 6 0 Master 6 POB0_BESR1 field is unlocked 1 Master 6 POB0_BESR1 field is locked
20	ALK6	POB0_BEAR Address Lock Master 6 0 Master 6 POB0_BEAR address is unlocked 1 Master 6 POB0_BEAR address is locked
21	WIRQ6	Write Error Interrupt Master 6 0 No write error detected - master 6 interrupt request is inactive 1 Write error detected - master 6 interrupt request is active
22:23		Reserved
24:25	PTE7	PLB Timeout Error Status Master 7 00 No master 7 error occurred 01 Master 7 timeout error occurred 10 Master 7 slave error occurred 11 Reserved Master 7 is the OPB to PLB bridge controller
26	R/W7	Read/Write Status Master 7 0 Master 7 error operation is a write 1 Master 7 error operation is a read
27	FLK7	POB0_BESR1 Field Lock Master 7 0 Master 7 POB0_BESR1 field is unlocked 1 Master 7 POB0_BESR1 field is locked
28	ALK7	POB0_BEAR Address Lock Master 7 0 Master 7 POB0_BEAR address is unlocked 1 Master 7 POB0_BEAR address is locked
29	WIRQ7	Write Error Interrupt Master 7 0 No write error detected - master 7 interrupt request is inactive 1 Write error detected - master 7 interrupt request is active
30:31		Reserved

DCR 0x096 Read/Clear

See *PLB to OPB Bridge Configuration Register (POB0_CONFIG)* on page 94.

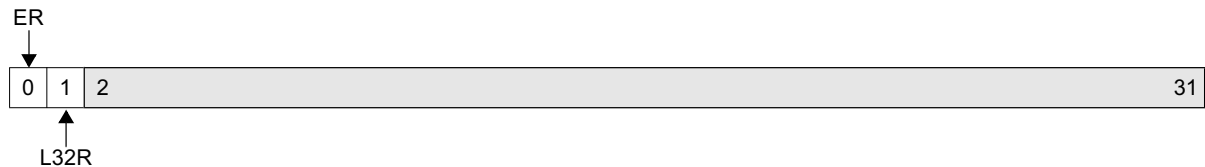


Figure 32-354. PLB to OPB Bridge Configuration Register (POB0_CONFIG)

0	ER	Enable re arbitration 0 PLB to OPB bridge re arbitration is enabled 1 PLB to OPB bridge re arbitration is disabled
1	L32R	Line 32 bit read 0 PLB to OPB bridge reads line operations at requested size of master 1 PLB to OPB bridge reads line operations at 32 bit slave size regardless of requesting master size
2:31		Reserved

DCR 0x098 Read/Clear

See *PLB to OPB Bridge Burst Latency Timer (POB0_LATENCY)* on page 94.

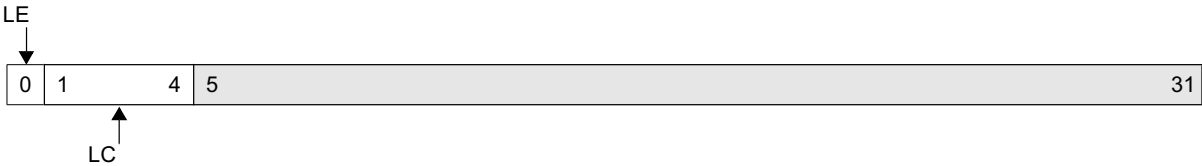


Figure 32-355. *PLB to OPB Bridge Burst Latency Timer Register (POB0_LATENCY)*

0	LE	Latency Enable 0 Latency timer disabled 1 Latency timer enabled
1:4	LC	Latency Count 0000 - Minimum count value. PLB to OPB bridge will count 16 OPB_xferAcks during a read or write burst sequence and if OPB_pendReq is sampled high at the end of the count - then the burst sequence is terminated. 1111 - Maximum count value. PLB to OPB bridge will count 16 blocks of 16 OPB_xferAcks, (or 256 xferAcks) during a read or write burst sequence, and if OPB_pendReq is sampled high at the end of count - then the burst sequence is terminated.
5:31		Reserved

DCR 0x09A Read/Clear

See *PLB to OPB Bridge Revision ID (POB0_REVID)* on page 95.

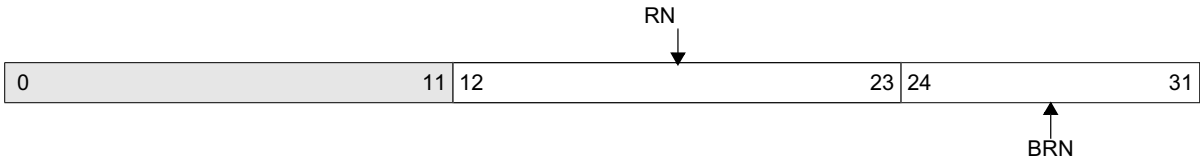


Figure 32-356. PLB to OPB Bridge Revision ID Register (POB0_REVID)

0:11		Reserved	
12:23	RN	Revision number	Corresponds to the RCS revision of the source RTL
24:31	BRN	Branch revision number	Corresponds to the RCS branch revision of the source RTL

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x23 R/W

See *Cycle Counter Register (PPM0_CCR)* on page 116.



Figure 32-357. Cycle Counter Register (PPM0_CCR)

0:31	CCV	32-bit Clock Count Value	This value can range from 0x0h to 0xFFFFFFFFh. Its contents are decremented by 1 (for every PLB clock) if the enable bit in the CR register is active. This counter will stop decrementing when a value of 0x0h reached.
------	-----	--------------------------	--

DCR Offset 0x016 R/W

See . PPM Configuration Address Register (PPM0_CFGADDR) on page 111.



Figure 32-358. PPM Configuration Address Register (PPM0_CFGADDR)

0:23			
24:31	DCRA	8-bit PPM Register Offset Value	This value can range from 0x000000 to 0x7F. Its contents are used as the indirect DCR address for accessing a PPM register.

DCR 0x017 R/W

See *PPM Configuration Data Register (PPM0_CFGDATA)* on page 112.

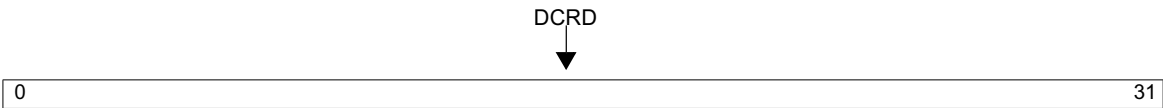


Figure 32-359. PPM Configuration Data Register (PPM0_CFGDATA)

0:31	DCRD	32-bit Data Value	This value can range from 0x00000000 to 0xFFFFFFFF. Its contents contain the value of the PPM register as indicated by the PPM_CFGADDR register contents.
------	------	-------------------	---

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x23 R/W

See *Control Register (PPM0_CR)* on page 114.

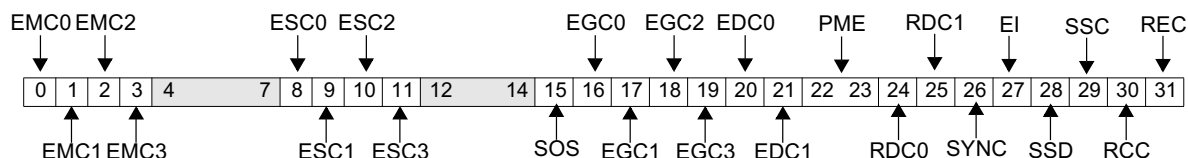


Figure 32-360. Control Register (PPM0_CR)

0	EMC0	Enable Master Counter 0 0 Disable master counter 0. 1 Enable master counter 0.
1	EMC1	Enable Master Counter 1 0 Disable master counter 1. 1 Enable master counter 1.
2	EMC2	Enable Master Counter 2 0 Disable master counter 2 1 Enable master counter 2
3	EMC3	Enable Master Counter 3 0 Disable master counter 3 1 Enable master counter 3
4:7		Reserved
8	ESC0	Enable Slave Counter 0 0 Disable slave counter 0 1 Enable slave counter 0
9	ESC1	Enable Slave Counter 1 0 Disable slave counter 1 1 Enable slave counter 1
10	ESC2	Enable Slave Counter 2 0 Disable slave counter 2 1 Enable slave counter 2
11	ESC3	Enable Slave Counter 3 0 Disable slave counter 3 1 Enable slave counter 3
12:14		Reserved
15	SOS	Synchout Selector 0 Drives PPM_synchOutSSC signal with value of SSC bit and PPM_synchOutSSD signal with value of SSD bit. 1 Asserts PPM_synchOutSSC signal when master counter 0, slave counter 0, generic counter 0, or cycle counter has overflow/underrun its value. Asserts PPM_synchOutSSD signal when duration counter 0 occurrence total counter, duration counter 1 occurrence total counter, duration counter 0 total value counter, and duration counter 1 total value counter has overrun their contents. Note: The associated interrupts must be enabled for this feature to work.
16	EGC0	Enable Generic Counter 0 0 Disable generic counter 0 1 Enable generic counter 0
17	EGC1	Enable Generic Counter 1 0 Disable generic counter 1 1 Enable generic counter 1

18	EGC2	Enable Generic Counter 2 0 Disable generic counter 2. 1 Enable generic counter 2.
19	EGC3	Enable Generic Counter 3 0 Disable generic counter 3. 1 Enable generic counter 3.
20	EDC0	Enable Duration Counter Set 0 0 Disable duration counter 0. 1 Enable duration counter 0.
21	EDC1	Enable Duration Counter Set 1 0 Disable duration counter 1. 1 Enable duration counter 1.
22:23	PME	Power Mode Enable 00 Normal mode 01 Low power mode 10 Ultra low power mode 11 Reserved
24	RDC0	Reset Event Counters 0 No action. 1 Reset all DC0 counters.
25	RDC1	Reset Event Counters 0 No action. 1 Reset all DC1 counters.
26	SYNC	SyncIn Enable 0 Disable syncIn functionality. 1 Enable syncIn functionality.
27	EI	Enable Interrupts 0 Disable interrupts. 1 Enable interrupts
28	SSD	Start/Stop Duration Events 0 Stop all duration events 1 Start all duration events
29	SSC	Start/Stop Cycle Counter 0 Stop decrementing cycle counter. 1 Start decrementing cycle counter.
30	RCC	Reset Cycle Counters 0 No action. 1 Reset cycle counter.
31	REC	Reset Event Counters 0 No action. 1 Reset all PPM0_MCRn, PPM0_SCRn and PPM_GCRn.

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x35, 0x36 R/W

See *Duration Counter Minimum Register 0:1 (PPM0_DCMNR0-PPM0_DCMNR1)* on page 127.



Figure 32-361. Duration Counter Minimum Registers 0:1 (PPM0_DCMNR0-PPM0_DCMNR1)

0:7		Reserved
8:31	MNV	Least Min value

This value can range from 0x000000 to 0xFFFFFF. Its contents are the least minimum duration value calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x33, 0x34 R/W

See *Duration Counter Maximum Register 0:1 (PPM0_DCMXR0-PPM0_DCMXR1)* on page 126.



Figure 32-362. Duration Counter Maximum Registers 0:1 (PPM0_DCMXR0-PPM0_DCMXR1)

0:7		Reserved	
8:31	MXV	Highest Max value	This value can range from 0x0 to 0xFFFFF. Its contents are the highest maximum duration value calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x39, 0x3A R/W

See *Duration Counter Occurrence Total Register 0:1 (PPM0_DCOTR0-PPM0_DCOTR1)* on page 128.



Figure 32-363. Duration Counter Occurrence Total Registers 0:1 (PPM0_DCOTR0-1)

0:7		Reserved	
8:31	DOT	Total number of event occurrences	This value can range from 0x0 to 0xFFFFF. Its contents are incremented once for each occurrence of an event as selected in the corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x31 0x32 R/W

See *Duration Counter Selection Register 0:1 (PPM0_DCSR0-PPM0_DCSR1)* on page 125.

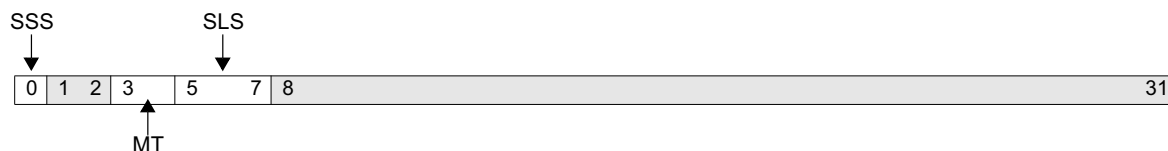


Figure 32-364. Duration Counter Selection Registers 0:1 (PPM0_DCSR0-PPM0_DCSR1)

0	SSS	Single Shot Selection 0 Disable single shot measurement 1 Enable single shot measurement	Selects which Slave's signal transitions to count with this counter set.
1:2		Reserved	
3:4	MT	Measurement Type 00 Write tenure 01 Read tenure 10 Wait tenure 11 Reserved	Selects the type of slave duration measurement to be performed (see Table 3-2 PLB Event Duration Measurement on page 109).
5:7	SLS	Slave Selection 000 Selects SI0 001 Selects SI1 010 Selects SI2 011 Selects SI3 100 Selects SI4 101 Selects SI5 110 Selects SI6 111 Selects SI7	Selects the desired PLB Slave device in which the measurement is to be performed Slave 0 is DDR_SDRAM Slave 1 is PCIX Slave 2 is SRAM Slave 3 is IMU Slave 4 is reserved Slave 5 is PLB to OPB bridge Slave 6 is reserved Slave 7 is reserved
8:31		Reserved	

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x37 0x38 R/W

See *Duration Counter Total Value Register 0:1 (PPM0_DCTVR0-PPM0_DCTVR1)* on page 127.

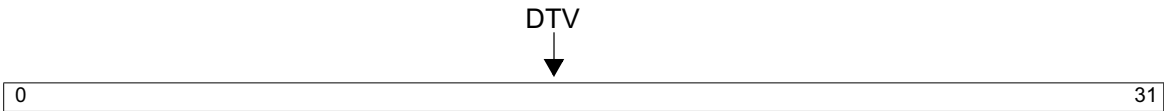


Figure 32-365. Duration Total Value Register 0:1 (PPM0_DCTVR0-PPM0_DCTVR1)

0:31	DTV	Duration Total Value	This value can range from 0x0 to 0xFFFFFFFF. The value is incremented in accordance with the duration values calculated via the selection in its corresponding Duration Selection register. This register can only be updated if its corresponding enable bit in the CR register is active.
------	-----	----------------------	---

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x32D, 0x2E, 0x2F, 0x30 R/W

See Generic Pipeline Event Counter Register 0:3 (PPM0_GCR0-PPM0_GCR3) on page 125.



Figure 32-366. Generic Pipeline Event Counter Registers 0:3 (PPM0_GCR0-PPM0_GCR3)

0:7		Reserved	
8:31	GECV	Generic Event Counter Value	This value can range from 0x000000 to 0xFFFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding GCounter Selection register. This register can only be updated if its corresponding GCn enable bit in the CR register is active

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x19, 0x1A, 0x1B, 0x1C R/W

See Generic Event Counter Selection Register 0:3 (PPM0_GCSR0-PPM0_GCSR3) on page 123.



Figure 32-367. Generic Event Counter Selection Registers 0:3 (PPM0_GCSR0-PPM0_GCSR3)

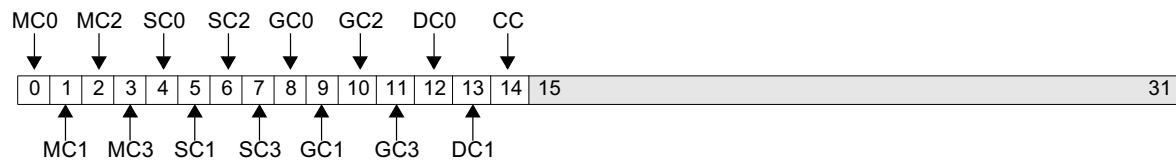
0-3	GES	Generic Event Selection		Selects the external signal n, or the pipeline-depth n signal as input to this counter. Note: Using the pipeline depth selection(s) one can determine the depth of PLB pipelining that has been reached under specific applications.
		0000	Selects G0_input	
		0001	Selects G1_input	
		0010	Selects G2_input	
		0011	Selects G3_input	
		0100	Selects write-pipeline-depth 1	
		0101	Selects write-pipeline-depth 2	
		0110	Selects read-pipeline-depth 1	
		0111	Selects read-pipeline-depth 2	
		1000	Selects read-pipeline-depth 3	
		1001	Selects read-pipeline-depth 4	
4:31		Reserved		

PPM0_ISR

Interrupt Status Register

Preliminary User's Manual

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x19, 0x1A, 0x1B, 0x1C R/W

See *Interrupt Status Register (PPM0_ISR)* on page 112.Figure 32-368. *Interrupt Status Register (PPM0_ISR)*

0	MC0	Master Counter 0 Interrupt Status 0 Master counter 0 interrupt did not occur 1 Master counter 0 interrupt occurred
1	MC1	Master Counter 1 Interrupt Status 0 Master counter 1 interrupt did not occur 1 Master counter 1 interrupt occurred
2	MC2	Master Counter 2 Interrupt Status 0 Master counter 2 interrupt did not occur 1 Master counter 2 interrupt occurred
3	MC3	Master Counter 3 Interrupt Status 0 Master counter 3 interrupt did not occur 1 Master counter 3 interrupt occurred
4	SC0	Slave Counter 0 Interrupt Status 0 Slave counter 0 interrupt did not occur 1 Slave counter 0 interrupt occurred
5	SC1	Slave Counter 1 Interrupt Status 0 Slave counter 1 interrupt did not occur 1 Slave counter 1 interrupt occurred
6	SC2	Slave Counter 2 Interrupt Status 0 Slave counter 2 interrupt did not occur 1 Slave counter 2 interrupt occurred
7	SC3	Slave Counter 3 Interrupt Status 0 Slave counter 3 interrupt did not occur 1 Slave counter 3 interrupt occurred
8	GC0	Generic Counter 0 Interrupt Status 0 Generic counter 0 interrupt did not occur 1 Generic counter 0 interrupt occurred
9	GC1	Generic Counter 1 Interrupt Status 0 Generic counter 1 interrupt did not occur 1 Generic counter 1 interrupt occurred
10	GC2	Generic Counter 2 Interrupt Status 0 Generic counter 2 interrupt did not occur 1 Generic counter 2 interrupt occurred
11	GC3	Generic Counter 3 Interrupt Status 0 Generic counter 3 interrupt did not occur 1 Generic counter 3 interrupt occurred
12	DC0	Duration Counter 0 Interrupt Status 0 Duration counter 0 interrupt did not occur 1 Duration counter 0 interrupt occurred

13	DC1	Duration Counter 1 Interrupt Status 0 Duration counter 1 interrupt did not occur 1 Duration counter 1 interrupt occurred
14	CC	Cycle Counter Interrupt Status 0 Cycle counter interrupt did not occur 1 Cycle counter interrupt occurred
15:31		Reserved

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x7 R/W
See Lower Address Mask Register (PPM0_LAMR) on page 118.

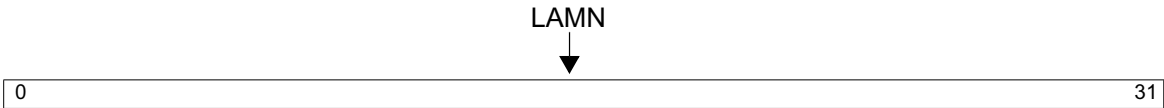


Figure 32-369. Lower Address Mask Register (PPM0_LAMR)

0:31	LAMn	Lower AddressBit-n Mask 0 Disable bit-n matching. 1 Enable bit-n matching.	If enabled, the corresponding bit in the LAR register is match against the same address bit-n of the PLB lower address bus.
------	------	--	--

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x5 R/W

See *Lower Address Register (PPM0_LAR)* on page 117.

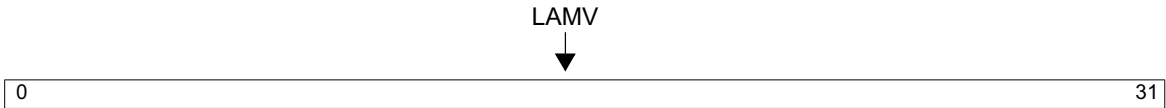


Figure 32-370. Lower Address Register (PPM0_LAR)

0:31	LAMV	Lower Address Match Value	If enabled, this value is match against the upper address bits of the PLB transaction event. Its contents can range from 0x00000000 to 0xFFFFFFFF.
------	------	---------------------------	--

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x1D, 0x1E, 0x1F, 0x20 R/W

See Master Event Counter Register 0:3 (PPM0_MCR0-PPM0_MCR3) on page 123.



Figure 32-371. Master Event Counter Registers 0:3 (PPM0_MCR0-PPM0_MCR3)

0:7		Reserved	
8:31	MECV	Master Event Counter Value	This value can range from 0x000000 to 0xFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding Mcounter Selection register. This register will only be updated if its corresponding MCn enable bit in the CR register is active.

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x9, 0xA, 0xB, 0xC R/W

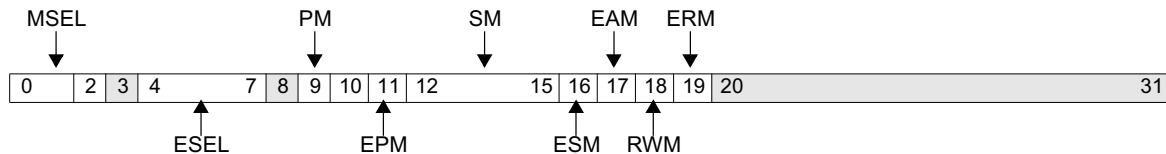
See *Master Event Counter Selection Register 0:3 (PPM0_MCSR0-PPM0_MCSR3)* on page 119.

Figure 32-372. Master Event Counter Selection Register 0:7 (PPM0_MCSR0-PPM0_MCSR7)

0:2	MSEL	Master Selection 000 Selects Master 0 001 Selects Master 1 010 Selects Master 2 011 Selects Master 3 100 Selects Master 4 101 Selects Master 5 110 Selects Master 6 111 Selects Master 7	Selects which master's signal transitions to count with this counter set. Master 0 is ICU Read Master 1 is DCU Read Master 2 is DCU Write Master 3 is PCIX Master 4 is IMU Master 5 is MAL Master 6 is DMA Master 7 is OPB to PLB bridge
3		Reserved	
4:7	ESEL	Master Event Selection 0000 Selects PLB_MnAddrAck 0001 Selects PLB_MnRdDack 0010 Selects PLB_MnWrDack 0011 Selects PLB_MnRdErr 0100 Selects PLB_MnWrErr 0101 Selects PLB_MnRequest 0110 Selects PLB_MnAbort 1000 Selects Mn_WrBurst 1001 Selects Mn_RdBurst 1010-1111 Reserved	Selects the master's PLB event to be tracked (refer to <i>Table 3-1 PLB Event Occurrence</i> on page 107).
8		Reserved	
9:10	PM	Priority Matching 00 Lowest priority 01 Low priority 10 High priority 11 Highest priority	Sets the priority decode value, which will be used if EPM bit is set.
11	EPM	Enable Priority Matching 0 Disable priority matching 1 Enable priority matching	Selects the priority bits to be used when tracking an event Note: This feature is functional when MES bits=0101 only. This bit is ignored for all other MES values.

12:15	SM	PLB transaction Size Matching 0000 One to 4 bytes 0001 4-word line 0010 8-word line 0011 16-word line 0100 Reserved 0101 Reserved 0110 Reserved 0111 Reserved 1000 Burst transfer - bytes 1001 Burst transfer - half words 1010 Burst transfer - words 1011 Burst transfer - double words 1100 Burst transfer - quad words 1101 Burst transfer - octal words 1110 Reserved 1111 Reserved	Sets the PLB transaction size decode value, which will be used during event tracking if the ESM bit is set.
16	ESM	Enable Size Matching 0 Disable transaction Size matching 1 Enable transaction Size matching	Selects the SM bits to be used when tracking an event Note: This feature is functional when MES bits=0000 only. This bit is ignored for all other values.
17	EAM	Enable Address Matching 0 Disable address matching feature 1 Enable address matching feature	Selects the contents of the UAR/LAR registers to be used when tracking an event Note: This feature is functional when MES bits=0000 only. This bit is ignored for all other values.
18	RWM	RnW Matching 0 Write transaction matching 1 Read transaction matching	Selects the transaction matching criteria for Mn_Request transactions only.
19	ERM	Enable RnW Matching 0 Disable matching 1 Enable matching	Selects the priority bits to be used when tracking an event Note: This feature is functional when MES bits=0000 or 0110 only. This bit is ignored for all other values.
20:31		Reserved	

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x8 Read Only

See *Revision ID Register (PPM0_RIDR)* on page 118.

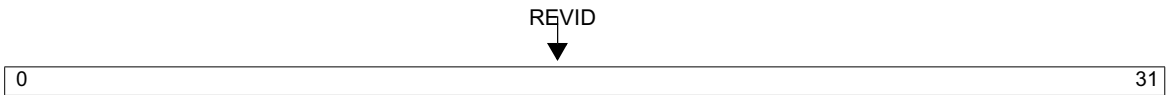


Figure 32-373. Revision ID Register (PPM0_RIDR)

0:31	REVID	RevID value: 0xC27E341x	Read Only field, where x is the Revision #. The default value for x in PPC440GX Embedded Processor is 1.
------	-------	----------------------------	--

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x25, 0x26,0x27,0x28 R/W
See SLave Event Counter Register 0:3 (PPM0_SCR0-PPM0_SCR3) on page 124.



Figure 32-374. Slave Event Counter Registers 0:3 (PPM0_SCR0-PPM0_SCR3)

0:7		Reserved	
8:31	SECV	Slave Event Counter Value	This value can range from 0x000000 to 0xFFFFFFFF. Its contents are incremented once for each event occurrence as selected in its corresponding SCounter Selection register. This register can only be updated if its corresponding SCn enable bit in the CR register is active

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x11, 0x12,0x13,0x14 R/W

See *Slave Event Counter Selection Register 0:7 (PPM0_SCSR0-PPM0_SCSR7)* on page 121.

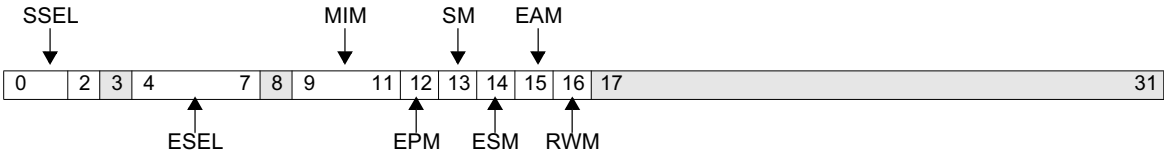


Figure 32-375. Slave Event Counter Selection Register 0:3 (PPM0_SCSR0-PPM0_SCSR3)

0:2	SSEL	Slave Selection	Selects which Slave's signal transitions to count with this counter set.
		000 Selects Slave 0	Slave 0 is DDR_SDRAM
		001 Selects Slave 1	Slave 1 is PCIX
		010 Selects Slave 2	Slave 2 is SRAM
		011 Selects Slave 3	Slave 3 is IMU
		100 Selects Slave 4	Slave 4 is reserved
		101 Selects Slave 5	Slave 5 is PLB to OPB bridge
		110 Selects Slave 6	Slave 6 is reserved
3		Reserved	
4:7	ESEL	Slave Event Selection	Selects the PLB slave event to be tracked.
		0000 Selects PLB_SlnAddrAck	
		0001 Selects PLB_SlnRdDack	
		0010 Selects PLB_SlnWrDack	
		0011 Selects PLB_SlnRearbitrate	
		0100 Selects Sln_Wait	
		0101 Selects Sln_WrComp	
		0110 Selects Sln_RdComp	
		0111-1111 Reserved	

8		Reserved	
9:11	MIM	Master ID Matching 000 Selects Master 0 001 Selects Master 1 010 Selects Master 2 011 Selects Master 3 100 Selects Master 4 101 Selects Master 5 110 Selects Master 6 111 Selects Master 7	Sets the MasterID decode value, which will be used if EMIM bit is set. Master 0 is ICU Read Master 1 is DCU Read Master 2 is DCU Write Master 3 is PCIX Master 4 is IMU Master 5 is MAL Master 6 is DMA Master 7 is OPB to PLB Bridge
12	EMIM	Enable Master ID Matching 0 Disable master ID matching 1 Enable master ID matching	Selects the MIM bits to be used when tracking an event Note: This feature is functional when SES bits=0000 only. This bit is ignored for all other SES values.
13	RWM	RnW Matching 0 Write transaction matching 1 Read transaction matching	Selects the Read or Write value that will be used when ERM bit is set.
14	ERM	Enable RnW Matching 0 Disable matching 1 Enable matching	Selects the RWM bit to be used when tracking an event Note: This feature is functional when SES bits=0000 or 0011 only. This bit is ignored for all other SES values.
15	AVM	Address Valid Matching 0 PAVvalid matching 1 SAVvalid matching	Selects either the PAVvalid or SAVvalid signals to used when the EVM bit is set.
16	EVM	Enable Address Valid Matching 0 Disable matching 1 Enable matching	Enables the Primary of Secondary Address Valid matching feature to be used when tracking an event Note: This feature is functional when SES bits= 0011 only. This bit is ignored for all other SES values.
17:31		Reserved	

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x6 R/W

See Upper Address Mask Register (PPM0_UAMR) on page 117.

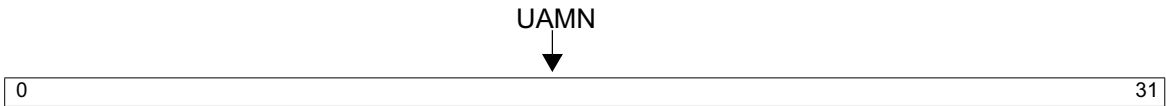


Figure 32-376. Upper Address Mask Register (PPM0_UAMR)

0:31	UAMn	Upper Address Bit-n Mask 0 Disable bit-n matching. 1 Enable bit-n matching.	If enabled, the corresponding bit in the UAR register is match against the same address bit-n of the PLB upper address bus.
------	------	---	---

DCR Accessed using PPM0_CFGADDR; PPM0_CFGDATA; Offset 0x4 R/W
See Upper Address Register (PPM0_UAR) on page 116.

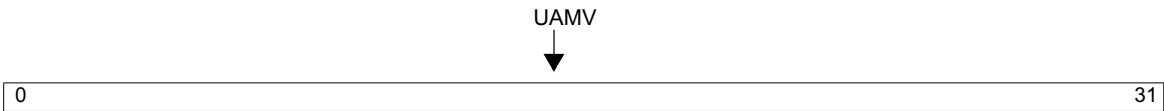


Figure 32-377. Upper Address Register (PPM0_UAR)

0:31	UAMV	Upper Address Match Value	If enabled, this value is matched against the upper PLB address bits UAbus(0-31) of the PLB transaction event.
------	------	---------------------------	--

Preliminary User’s Manual

MMIO 0x1 40000790 Read/Write

See RGMII Function Enable Register (RGMII0_FER) on page 812.

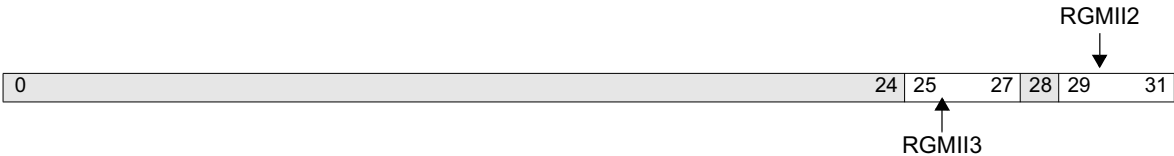


Figure 32-378. RGMII Function Enable Register (RGMII0_FER)

0:24		Reserved
25:27	RGMII3	<div>EMAC3 RGMII Enable</div> <div>0xx EMAC3 RGMII PHY interface is disabled.</div> <div>100 EMAC3 RTBI PHY interface is enabled.</div> <div>101 EMAC3 RGMII PHY interface is enabled.</div> <div>110 EMAC3 TBI PHY interface is enabled.</div> <div>111 EMAC3 GMII PHY interface is enabled.</div> <div>don't care when disabled</div> <div>reduced mode when RTBI PHY enabled</div> <div>reduced mode when RGMII PHY enabled</div> <div>passthru mode when TBI PHY enabled</div> <div>passthru mode when GMII PHY enabled</div> <div>See table note below.</div>
28		Reserved
29:31	RGMII2	<div>EMAC2 RGMII Enable</div> <div>0xx EMAC2 RGMII PHY interface is disabled.</div> <div>100 EMAC2 RTBI PHY interface is enabled.</div> <div>101 EMAC2 RGMII PHY interface is enabled.</div> <div>110 EMAC2 TBI PHY interface is enabled.</div> <div>111 EMAC2 GMII PHY interface is enabled.</div> <div>don't care when disabled</div> <div>reduced mode when RTBI PHY enabled</div> <div>reduced mode when RGMII PHY enabled</div> <div>passthru mode when TBI PHY enabled</div> <div>passthru mode when GMII PHY enabled</div> <div>See table note below.</div>
Note: In reduced mode, the two channels operate independently. In pass-thru mode, only one channel operates. If both channels are enabled and set to pass-thru, or one channel is reduced and the other is pass-thru, these are invalid settings resulting in both channels being disabled.		

MMIO 0x1 40000794 Read/Write

See *SMI Status Register (ZMII0_SMIISR)* on page 803

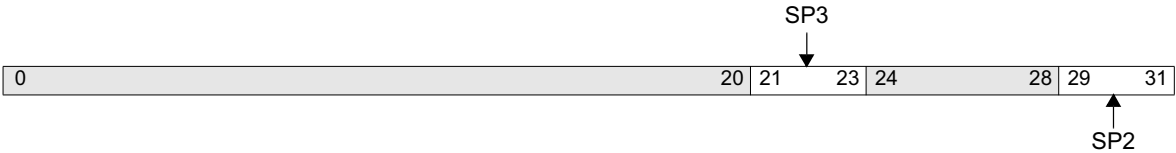


Figure 32-379. Speed Selection Register (RGMII0_SSR)

0:20		Reserved
21:23	SP3	EMAC3 Speed Selection 000 10 Mbps selected. 010 100 Mbps selected. 100 1000 Mbps selected
24:28		Reserved
29:31	SP2	EMAC2 Speed Selection 000 10 Mbps selected. 010 100 Mbps selected. 100 1000 Mbps selected

Preliminary User’s Manual

DCR Accessed using SDRAM0_CFGADDR; SDRA4M0_CFGDATA; Offset 0x40–0x4C R/W

See Memory 0 - 3 Configuration (SDRAM0_B0CR-SDRAM0_B3CR) on page 562.

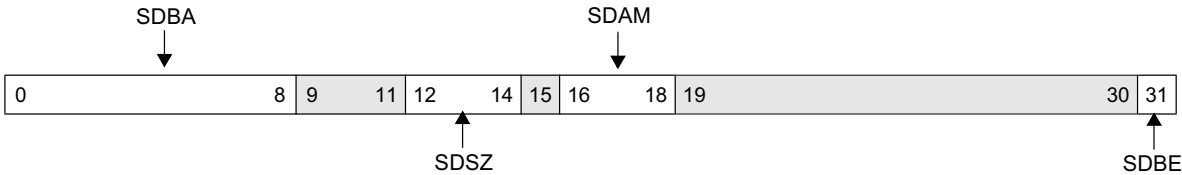


Figure 32-380. Memory 0-3 Configuration (SDRAM0_B0CR-SDRAM0_B3CR)

0:8	SDBA	Base Address
9:11		Reserved
12:14	SDSZ	Size 000 Reserved 001 8 MB 010 16 MB 011 32 MB 100 64 MB 101 128 MB 110 256 MB 111 512 MB
15		Reserved
16:18	SDAM	Addressing Mode 000 Mode 1 001 Mode 2 010 Mode 3 011 Mode 4 1xx Reserved See Table 18-6 DDR SDRAM Addressing Modes for details.
19:30		Reserved
31	SDBE	Memory Bank Enable

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x10 Read-Only
See Master Bus Error Address Register (SDRAM0_BEAR) on page 554.

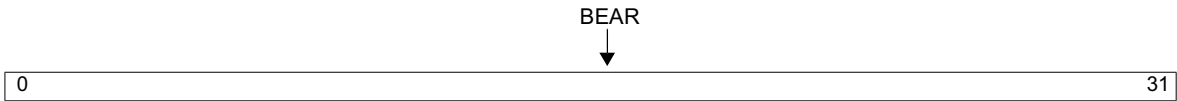


Figure 32-381. Bus Error Address Register (SDRAM0_BEAR)

0:31	BEAR	Address of Bus Error	
------	------	----------------------	--

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x00 R/W

See *Bus Error Syndrome Register 0 (SDRAM0_BESR0)* on page 551.

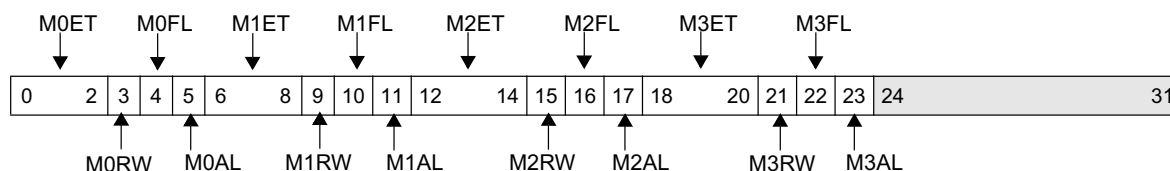


Figure 32-382. Bus Error Syndrome Register 0 (SDRAM0_BESR0)

0:2	M0ET	Master 0 Error Type 000 No Error 001 Reserved 010 ECC Uncorrectable Error 100 Reserved	Master 0 is the read-only instruction cache unit (ICU)
3	M0RW	Master 0 Read/Write Status 0 Write Error 1 Read Error	
4	M0FL	Master 0 Field Lock 0 BESR0 Unlocked 1 BESR0 Locked	
5	M0AL	Master 0 Address Lock 0 BEAR Not Locked 1 BEAR Locked	
6:8	M1ET	Master 1 Error Type 000 No Error 001 Reserved 010 ECC Uncorrectable Error 100 Reserved	Master 1 is the read-only data cache unit (DCU)
9	M1RW	Master 1 Read/Write Status 0 Reserved 1 Read Error	
10	M1FL	Master 1 Field Lock 0 BESR1 Unlocked 1 BESR1 Locked	
11	M1AL	Master 1 Address Lock 0 BEAR Not Locked 1 BEAR Locked	
12:14	M2ET	Master 2 Error Type 000 No Error 001 Reserved 010 ECC Uncorrectable Error 100 Reserved	Master 2 is the write-only data cache unit (DCU)
15	M2RW	Master 2 Read/Write Status 0 Write Error 1 Reserved	
16	M2FL	Master 2 Field Lock 0 BESR2 Unlocked 1 BESR2 Locked	
17	M2AL	Master 2 Address Lock 0 BEAR Not Locked 1 BEAR Locked	

18:20	M3ET	Master 3 Error Type 000 001 Reserved 010 ECC Uncorrectable Error 100 Reserved	Master 3 is the PCIX bridge controller
21	M3RW	Master 3 Read/Write Status 0 Write Error 1 Read Error	
22	M3FL	Master 3 Field Lock 0 BESR3 Unlocked 1 BESR3 Locked	
23	M3AL	Master Address Lock 0 BEAR Not Locked 1 BEAR Locked	
24:31		Reserved	

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x08 R/W

See *Bus Error Syndrome Register 1 (SDRAM0_BESR1)* on page 553.

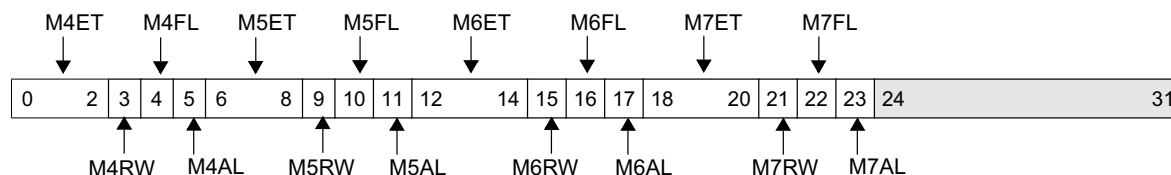


Figure 32-383. Bus Error Syndrome Register 1 (SDRAM0_BESR1)

0:2	M4ET	Master 4 Error Type 000 No Error 001 Reserved 010 ECC Error 100 Reserved	Master 4 is I2O messaging unit (IMU)
3	M4RW	Master 4 Read/Write 0 Write Error 1 Read Error	
4	M4FL	Master 4 Field Lock 0 BESR4 Unlocked 1 BESR4 Locked	
5	M4AL	Master 4 Address Lock 0 BEAR Not Locked 1 BEAR Locked by	
6:8	M5ET	Master 5 Error Type 000 No Error 001 Reserved 010 ECC Uncorrectable Error 100 Reserved	Master 5 is media access layer (MAL)
9	M5RW	Master 5 Read/Write 0 Write Error 1 Read Error	
10	M5FL	Master 5 Field Lock 0 BESR5 Unlocked 1 BESR5 Locked	
11	M5AL	Master 5 Address Lock 0 BEAR Not Locked 1 BEAR Locked	

12:14	M6ET	Master 6 Error Type 000 No Error 001 Reserved 010 ECC Uncorrectable Error 100 Reserved	Master 6 direct memory access controller (DMA)
15	M6RW	Master 6 Read/Write 0 Write Error 1 Read Error	
16	M6FL	Master 6 Field Lock 0 BESR6 Unlocked 1 BESR6 Locked	
17	M6AL	Master 6 Address Lock 0 BEAR Not Locked 1 BEAR Locked	
18:20	M7ET	Master 7 Error Type 000 No Error 001 Reserved 010 ECC Uncorrectable Error 100 Reserved	Master 7 is OPB to PLB bridge controller
21	M7RW	Master 7 Read/Write 0 Write Error 1 Read Error	
22	M7FL	Master 7 Field Lock 0 BESR7 Unlocked 1 BESR7 Locked	
23	M7AL	Master 7 Address Lock 0 BEAR Not Locked 1 BEAR Locked	
24:31		Reserved	

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x20 R/W

See *Memory Controller Options 0 (SDRAM0_CFG0)* on page 556.

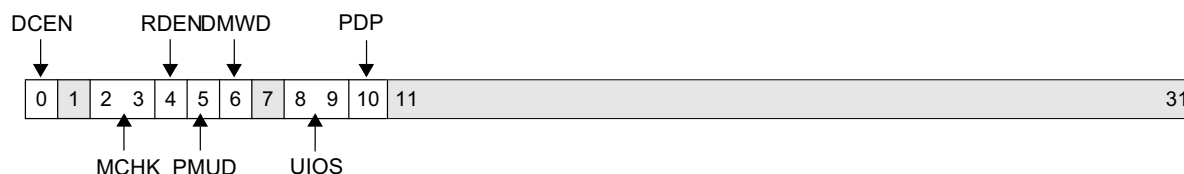


Figure 32-384. Memory Controller Options 0 (SDRAM0_CFG0)

0	DCEN	DDR SDRAM Controller Enable 0 DDR SDRAM disabled 1 DDR SDRAM enabled	When SDRAM0_CFG0[DCEN=1], DDR SDRAM is initialized using the power-on sequence and subsequently available for access. All DDR SDRAM controller configuration registers should be initialized and valid when SDRAM0_CFG0[DCEN=1].
1		Reserved	
2:3	MCHK	Memory Data Error Checking 00 None 01 Reserved 10 ECC generation only (no checking or correction) 11 ECC checking and correction	
4	RDEN	Registered DIMM Enable 0 Disable 1 Enable	
5	PMU	Page Management Unit 0 Page management unit enabled 1 Page management unit disabled	When SDRAM0_CFG[PMU=0], up to 8 open pages are maintained for subsequent accesses. When SDRAM0_CFG[PMU=1], the accessed page is opened for a given access, remains open for the duration of the access (allowing page hits within PLB sequential bursts) and is precharged (using read/write with autoprecharge command) upon completion of the access.
6	DMWD	DDR SDRAM Width 0 32-bit 1 64-bit	
7		Reserved	
8:9	UIOS	Unused I/O State 00 Active (High-Z on Read, Driven on Write) - switching DQS 01 Active (High-Z on Read, Driven on Write) -static values 10 Static (Always Driven) - static values 11 High-Z	This field can be used to control the state of the unused I/O for various configurations through the corresponding I/O driver data and enable signals. Unused I/O are defined as the I/O associated with a function that is not enabled. Example: ECC[0:7], DM[8], and DQS[8] would be unused I/O when ECC is not enabled. Likewise, DATA[32:63], DM[4:7], and DQS[4:7] would be unused I/O in 32-bit mode. All unused I/O receivers will be gated off.

10	PDP	Page Deallocation Policy 0 Pseudo Least Recently Allocated 1 Least Recently Used	<p>This field selects the page deallocation policy when page mode is enabled. Page deallocation occurs when the PMU has 8 open pages and an access does not target a page or bank address associated with one of those open pages. A PMU entry must be deallocated or replaced (and the corresponding page closed).</p> <p>With PMU_DP=0, the PMU entry that has been open longest will be deallocated. With PMU_DP=1, the page that was least recently used (least recently accessed) will be deallocated.</p> <p>Note: This distinction is only relevant for memory sub-systems employing more than two physical banks (chip selects) of memory.</p>
11:31		Reserved	

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x21 R/W

See *Memory Controller Options 1 (SDRAM0_CFG1)* on page 559.

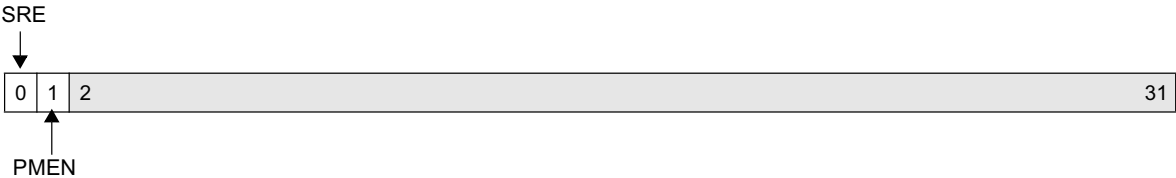


Figure 32-385. Memory Controller Options 1 (SDRAM0_CFG1)

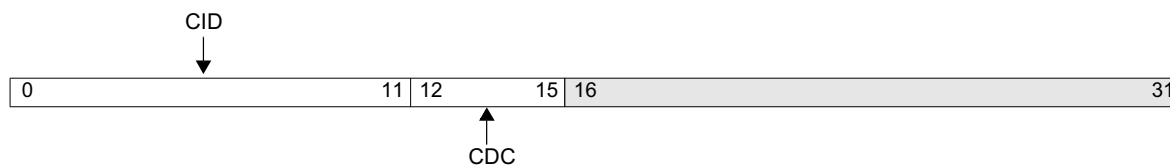
0	SRE	Self-Refresh Entry 0 Exit 1 Enter	This bit may be set by software (using DCR write). Once set, the SDRAM is maintained in self-refresh mode until this bit is reset by software. The DDR_SDRAM controller responds to this bit independent of the state of SDRAM0_CFG0[DCEN].
1	PMEN	Power Management Enable 0 Power management disabled 1 Power management enabled	See <i>Power Management</i> on page 597 for specific details.
2:31		Reserved	

SDRAM0_CID

Core ID Register

Preliminary User's Manual

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0xA4 Read/Only

See *Controller ID Register (SDRAM0_CID)* on page 579.*Figure 32-386. Controller ID Register (SDRAM0_CID)*

0:11	CID	Core ID identifies the IBM specific core number associated with the DDR SDRAM.
12:15	CDC	Core Derivative Code B Base Library Core D Derivative of Base Library Core
16:31		Reserved

Preliminary User’s Manual

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x82 R/W

See *DDR SDRAM Clock Timing Register (SDRAM0_CLKTR)* on page 571.

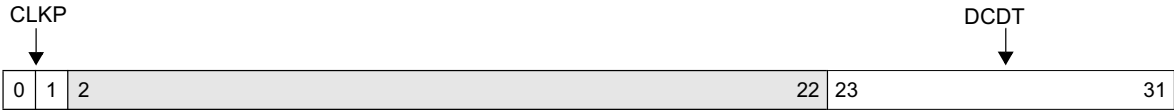


Figure 32-387. *DDR_SDRAM Clock Timing Register (SDRAM0_CLKTR)*

0:1	CLKP	Write Clock Phase 00 Advance 0 degrees 01 Advance 90 degrees 10 Advance 180 degrees 11 Reserved In most applications, CLKP should be set to 90 degrees phase advance.
2:22		Reserved
23:31	DCDT	DDR Clock Delay Tuning Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 23:29 select increments of 1 full delay element. Bits 30:31 select increments of 1/4 delay element. This field should never be programmed to exceed a delay equivalent to 1/4 the MemClkOut0 frequency.

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x22 R/W

See *DDR SDRAM Device Options (SDRAM0_DEVOPT)* on page 559.

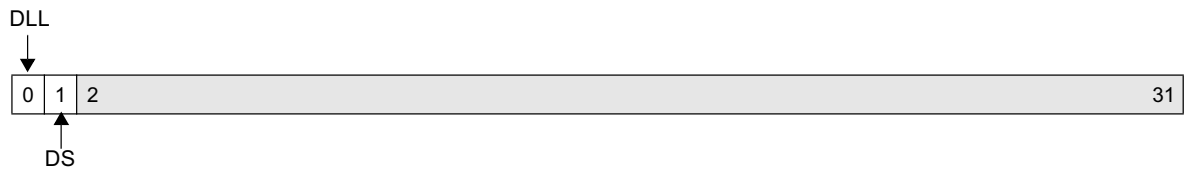


Figure 32-388. *DDR_SDRAM Device Options (SDRAM0_DEVOPT)*

0	DLL	DDR_SDRAM Device DLL Disable 0 Device DLL enabled 1 Device DLL disabled	Enable DLL for normal operation. The DLL setting controls the state of address bit during the initialization of the Extended Mode Register.
1	DS	DDR_SDRAM Device I/O Drive Strength 0 Drive strength is normal 1 Drive strength is weak	Clear DS for normal operation. The DS setting controls the state of address bit MemAddr1 during the initialization of the Extended Mode Register.
2:31		Reserved	

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x84 R/W

See *Delay Line Calibration Register (SDRAM0_DLYCAL)* on page 577.

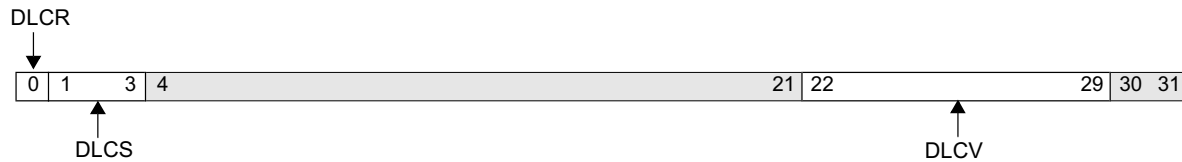


Figure 32-389. Delay Line Calibration Register (SDRAM0_DLYCAL)

0	DLCR	Delay Line Calibration Request 0 Delay Line Calibration not requested 1 Delay Line Calibration requested	Setting DLCR bit initiates the delay line calibration. During the re-initialization process, the DLCR bit is cleared. The delay line calibration occurs automatically following Sys-Reset or upon request by setting the DLCR bit. The results of the calibration are provided to assist in the setting of SDRAM0_CLKTR, SDRAM0_WDDCTR and SDRAM0_TR1.
1:3	DLCS	Delay Line Calibration Status 000 Delay Line Calibration not run 001 Delay Line Calibration in progress 010 Delay Line Calibration complete 100 Delay Line Calibration error	This two bit field indicates the real-time status of the calibration process at the time of the register read. Once complete (DLCS = 010), the DLY_VAL field is valid. A delay line calibration status of "error" indicates that the calibration process completed without successfully determining the value of DLCV. Sufficient delay stages have been included in the calibration delay line such that this completion status should never occur within the supported operating frequency range.
4:21		Reserved	
22:29	DLCV	Delay Line Calibration Value	This 8-bit binary encoded value indicates the number of delay elements in a single 1x half clock cycle. Bit 22 is Most Significant Bit; bit 29 is Least Significant Bit.
30:31		Reserved	

SDRAM0_ECCESR

ECC Error Status Register

Preliminary User's Manual

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x98 R/W

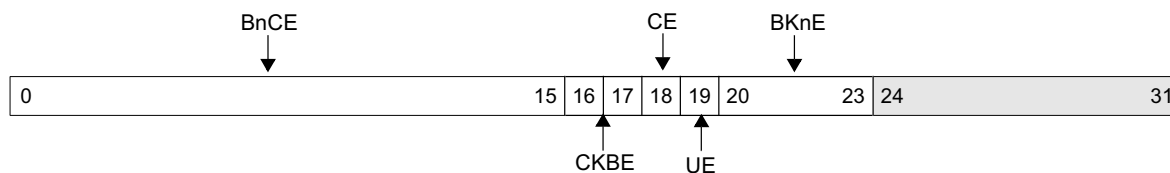
See *ECC Error Status Register (SDRAM0_ECCESR)* on page 578.

Figure 32-390. ECC Error Status Register (SDRAM0_ECCESR)

0:15	BnCE	Byte Lane n Corrected Error (where n=0-15) 0 No Error 1 Error Occurred in Byte Lane n
16:17	CKBE	Error Detected in Checkbits 64-bit Mode 00 No Error 01 Reserved 10 Error in Checkbits 11 Reserved 32-bit Mode 00 No Error 01 Error in Lower Checkbits 10 Error in Upper Checkbits 11 Error in Upper and Lower Checkbits
18	CE	Correctable Error
19	UE	Uncorrectable Error
20:23	BKnE	Bank n Error (where n=0-3) 0 No Error 1 Error Occurred in Bank n
24:31		Reserved

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x24 Read-only

See *Memory Controller Status (SDRAM0_MCSTS)* on page 560.

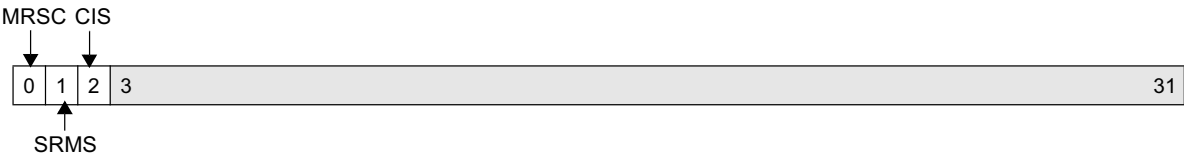


Figure 32-391. Memory Controller Status (SDRAM0_MCSTS)

0	MRSC	MRS Command Complete 0 MRS Not Complete 1 MRS Complete	This bit will be set to "1" once the DDR SDRAM controller has successfully completed the Mode Register Set Command, which results from setting DC_EN in SDRAM0_CFG0. Clearing DC_EN will clear this bit in the following cycle.
1	SRMS	Self-Refresh Mode Status 0 SDRAM is not in Self-Refresh Mode 1 SDRAM is in Self-Refresh Mode	This bit will be set upon the successful completion of Self-Refresh Mode entry, that is, the SDRAM device has been placed in Self-Refresh Mode. This bit will be cleared when Self-Refresh Mode has been exited. This bit applies to the software-initiated Self-Refresh Mode using SDRAM0_CFG0[SRE].
2	CIS	Core Idle Status 0 Busy 1 Idle	Indicates that there are no current or pending queued read/write accesses.
3:31		Reserved	

SDRAM0_MIRQ

Master Write Interrupt

Preliminary User's Manual

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x11 - 0x12 R/W

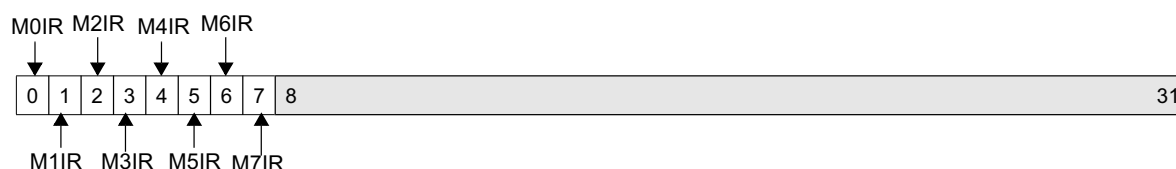
See *Master Write Interrupt (SDRAM0_MIRQ)* on page 555

Figure 32-392. Master Write Interrupt (SDRAM0_MIRQ)

0	M0IR	Master 0 Write Interrupt 0 No Error 1 Write Error	Master 0 is processor core instruction cache unit (ICU)
1	M1IR	Master 1 Write Interrupt 0 No Error 1 Write Error	Master 1 is processor core data cache read unit (DCU)
2	M2IR	Master 2 Write Interrupt 0 No Error 1 Write Error	Master 2 is processor core data cache write unit (DCU)
3	M3IR	Master 3 Write Interrupt 0 No Error 1 Write Error	Master 3 is PCIX bridge controller (PCIX)
4	M4IR	Master 4 Write Interrupt 0 No Error 1 Write Error	Master 4 is I20 messaging unit (IMU)
5	M5IR	Master 5 Write Interrupt 0 No Error 1 Write Error	Master 5 is media access layer (MAL)
6	M6IR	Master 6 Write Interrupt 0 No Error 1 Write Error	Master 6 is direct memory access controller (DMA)
7	M7IR	Master 7 Write Interrupt 0 No Error 1 Write Error	Master 7 is OPB to PLB bridge controller
8:31		Reserved	

Preliminary User’s Manual

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x34 R/W

See *Power Management Idle Timer (SDRAM0_PMIT)* on page 561.



Figure 32-393. Power Management Idle Timer (SDRAM0_PMIT)

0:4	PMC	Power Management Count	Programmable
5:31		Reserved	bits 5:9 hardcoded to 1, and bits 10:31 hardcoded to zero.

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0xA8 Read-only

See Revision ID Register (SDRAM0_RID) on page 579.

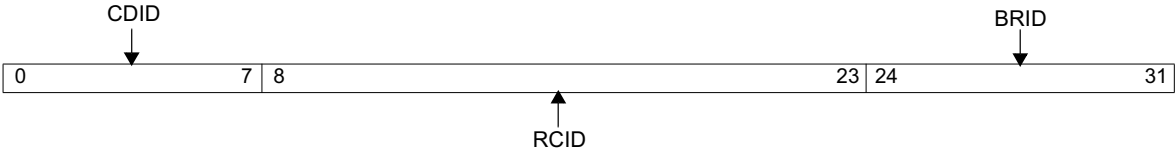


Figure 32-394. Revision ID Register (SDRAM0_RID)

0:7	CDID	Controller Derivative ID nn = controller derivative ID number nn = 00 indicates the Base Library Controller (CDC = B) nn = A non-zero values indicates the specific controller derivative ID number and is tracked by development (CDC = D).
8:23	RCID	Controller Revision Control ID cccc indicates the controller SCCS revision control ID associated with the specific version of the controller.
24:31	BRID	Controller Branch Revision Control ID bb indicates the core SCCS revision control branch ID associated with the specific version of the controller. This field is used to associate a specific Netlist with the corresponding RTL branch in the event of a modification at the Netlist level. bb = 00 indicates the an unmodified (post synthesis) netlist bb = non-zero value indicates the branch ID for the RTL and the corresponding netlist modification.

Preliminary User’s Manual

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x30 R/W

See Refresh Timer Register (SDRAM0_RTR) on page 560.



Figure 32-395. Refresh Timing Register (SDRAM0_RTR)

0:1		Reserved	Hardcoded to zero
2:12	RINT	Refresh Interval	Programmable
13:31		Reserved	Hardcoded to zero

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x18 R/W

See *PLB Slave Interface Options (SDRAM0_SLIO)* on page 556.

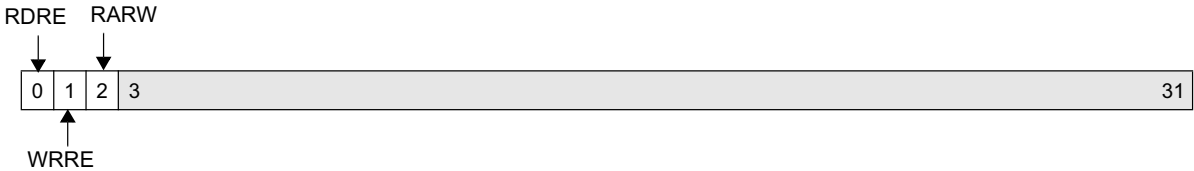


Figure 32-396. PLB Slave Interface Options (SDRAM0_SLIO)

0	RDRE	PLB Slave Read Rearbitrate Enable 0 PLB slave read rearbitrate disabled 1 PLB slave read rearbitrate enabled
1	WRRE	PLB Slave Write Rearbitrate Enable 0 PLB slave write rearbitrate disabled 1 PLB slave write rearbitrate enabled
2	RARW	PLB Read Around Write Enable 0 PLB read around write disabled 1 PLB read around write enabled
3:31		Reserved

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x80 R/W

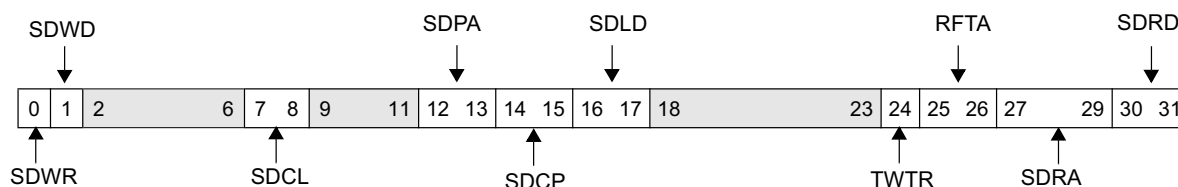
See *SDRAM Timing Register 0 (SDRAM0_TR0)* on page 563.

Figure 32-397. SDRAM Timing Register 0 (SDRAM0_TR0)

0	SDWR	DDR SDRAM Write Recovery 0 2 CLK 1 3 CLK	T_{wr}
1	SDWD	DDR SDRAM Write to Write delay when crossing a chip select boundary 0 0 CLK 1 1 CLK	This field configures the delay between back to back write cycles that cross chip select boundaries. A setting of 0 enables seamless writes when crossing chip select boundaries, while a setting of 1 will insert a single clock cycle delay between back-to-back writes when crossing a chip select boundary.
2:6		Reserved	
7:8	SDCL	DDR SDRAM CAS_Latency 00 Reserved 01 2 CLK 10 2.5 CLK 11 3 CLK	T_{aa} This field indicates the DDR SDRAM device CAS latency (independent of SDRAM0_CFG0[R_DIMM_EN]) and is used directly during the SDRAM device mode set command to configure the DDR SDRAM. See the section on "Registered DIMM Support" for the supported configurations and CAS_Latency settings. Note: Setting SD_CL to 2.5 CLK generally requires that SDRAM0_TR1[RDCD] be set to 1, SDRAM0_TR1[RDSS] be set to T2 Sample, and SDRAM0_TR1[RDSL] be set to Stage 3.
9:11		Reserved	
12:13	SDPA	DDR SDRAM CBR Precharge Command to next Activate Command minimum 00 Reserved 01 2 CLK 10 3 CLK 11 4 CLK	T_{rp}
14:15	SDCP	DDR SDRAM Read/Write Command to Precharge Command 00 2 CLK 01 3 CLK 10 4 CLK 11 Reserved	$T_{ras} - T_{rcd}$ The minimum value is calculated by first determining the values of T_{ras} and T_{rcd} in units of clock cycles (divide the respective device-specific AC timing specifications by the clock cycle time and round up to the nearest whole clock). Then apply the formula provided. SDCP settings that violate this rule (that is, settings less than the minimum calculated above for the device-specific timing parameters), independent of the programmed value of SDRD, are not supported.
16:17	SDLD	DDR SDRAM Command Leadoff 00 1 CLK 01 2 CLK 10 Reserved 11 Reserved	
18:23		Reserved	

SDRAM0_TR0 (contd.)

Timing Register 0

Preliminary User's Manual

24	TWTR	DDR SDRAM Write-To-Read Command 0 0 Selects Twtr of 1 clock 1 1 Selects Twtr of 2 clocks	Defaults to 0. Defines Write-To-Read.
25:26	RFTA	DDR SDRAM Refresh-To-Activate Command 00 Adds 0 clocks to SD_RFTA value programmed in SDTR1[27:29] 01 Adds 1 clock to SD_RFTA value programmed in SDTR1[27:29] 10 Adds 2 clocks to SD_RFTA value programmed in SDTR1[27:29] 11 Reserved	Defines the Refresh to Activate Field Adder and is used in conjunction with SDTR1[27:29]. Should only set to a non-zero value if SDTR1[27:29] = 3'b111. Example: Setting SDTR1[27:29] = 3'b111 selects RFTA of 13 clock cycles. Additionally, setting SDTR1[25:26] = 2'b01 will provide a total RFTA of 14 clocks.
27:29	SDRA	DDR SDRAM CBR Refresh Command to next Activate Command minimum 000 6 CLK 001 7 CLK 010 8 CLK 011 9 CLK 100 10 CLK 101 11 CLK 110 12 CLK 111 13 CLK	T_{rfc}
30:31	SDRD	DDR SDRAM RAS to CAS Delay 00 Reserved 01 2 CLK 10 3 CLK 11 4 CLK	T_{rcd}

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x81R/W

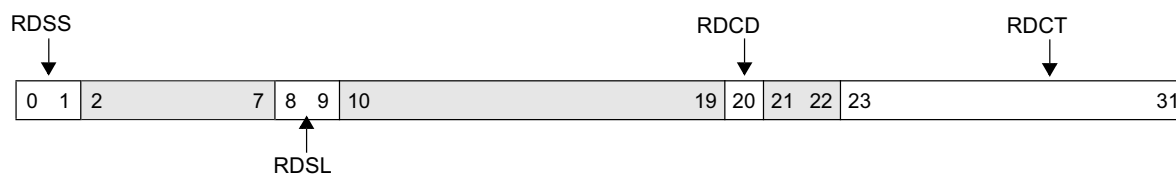
See *SDRAM Timing Register 1 (SDRAM0_TR1)* on page 565.

Figure 32-398. SDRAM Timing Register 1 (SDRAM0_TR1)

0:1	RDSS	Read Sample Cycle Select 00 T0 Sample 01 T1 Sample 10 T2 Sample 11 T3 Sample	Note: Set RDSS to T2 Sample for CAS latency 2.5 devices.
2:7		Reserved	
8:9	RDSL	Read Sample Stage Select 00 Stage 1 01 Stage 2 10 Stage 3 11 Reserved	Note: Set RDSL to Stage 3 for CAS latency 2.5 devices.
10:19		Reserved	
20	RDCD	Read Clock Delay - Stage 2 0 0 clock delay 1 1/2 clock delay	Note: Set RDCD to 1 for CAS latency 2.5 devices.
21:22		Reserved	
23:31	RDCT	Read Clock Delay Tuning Bits	Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 30:31, select increments of 1/4 delay element. Bits 23:29, select increments of 1 full delay element. This field should never be programmed to exceed a delay equivalent to 1/2 the Read Clock frequency.

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x38 R/W

See PLB UABus Base Address (SDRAM0_UABBA) on page 561.

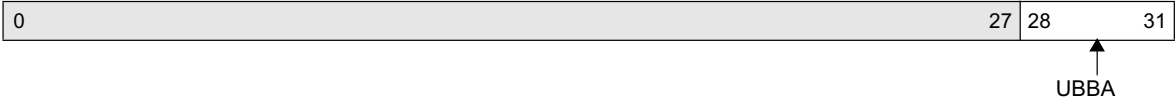


Figure 32-399. PLB UABus Base Address (SDRAM0_UABBA)

0:27		Reserved	
28:31	UBBA	PLB UABus Base Address	Defines the 4 GB aligned region in which the physical memory is located.

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x83 R/W

See *Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR)* on page 574.



Figure 32-400. Write Data/DM/DQS Clock Timing Register (SDRAM0_WDDCTR)

0:1	WRCP	Write Clock Phase 00 Advance 0 degrees 01 Advance 90 degrees 10 Advance 180 degrees 11 Reserved
2:22		Reserved
23:31	DCD	DDR Write Data/DM/DQS Clock Delay Tuning Bits Bit 23 is Most Significant Bit; bit 31 is Least Significant Bit. Bits 23:29 select increments of 1 full delay element. Bits 30:31 select increments of 1/4 delay element.

DCR 0x024 Read/Write

See *Bus Error Address Register (SRAM0_BEAR)* on page 537.

0	31
---	----

Figure 32-401. Bus Error Address Register (SRAM0_BEAR)

0:31	ABE	Address of Bus Error (asynchronous)
------	-----	-------------------------------------

DCR 0x025 Read/Write

See *Bus Error Status Register 0 (SRAM0_BESR0)* on page 538.

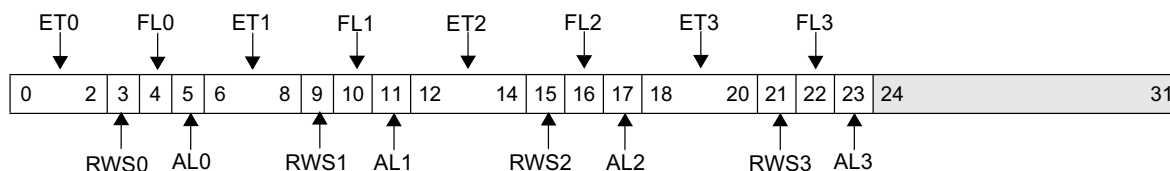


Figure 32-402. Bus Error Status Register 0 (SRAM0_BESR0)

0:2	ET0	Error type for master 0 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 0 is processor core instruction cache unit
3	RWS0	Read/write status for master 0 0 Error operation was a write operation 1 Error operation was a read operation	
4	FL0	Field lock for master 0 0 Fields are unlocked 1 Fields are locked	
5	AL0	BEAR address lock for master 0 0 BEAR address unlocked 1 BEAR address locked	
6:8	ET1	Error type for master 1 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 1 is processor core data cache read unit
9	RWS1	Read/write status for master 1 0 Error operation was a write operation 1 Error operation was a read operation	
10	FL1	Field lock for master 1 0 Fields are unlocked 1 Fields are locked	
11	AL1	BEAR address lock for master 1 0 BEAR address unlocked 1 BEAR address locked	

12:14	ET2	Error type for master 2 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 2 is processor core data cache write unit
15	RWS2	Read/write status for master 2 0 Error operation was a write operation 1 Error operation was a read operation	
16	FL2	Field lock for master 2 0 Fields are unlocked 1 Fields are locked	
17	AL2	BEAR address lock for master 2 0 BEAR address unlocked 1 BEAR address locked	
18:20	ET3	Error type for master 3 (PCI-X) 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	
21	RWS3	Read/write status for master 3 0 Error operation was a write operation 1 Error operation was a read operation	Master 3 is the PCIX controller
22	FL3	Field lock for master 3 0 Fields are unlocked 1 Fields are locked	
23	AL3	0 BEAR address lock for master 3 0 BEAR address unlocked 1 BEAR address locked	
24:31		Reserved	

DCR 0x026 Read/Write

See *Bus Error Status Register 1 (SRAM0_BESR1)* on page 540.

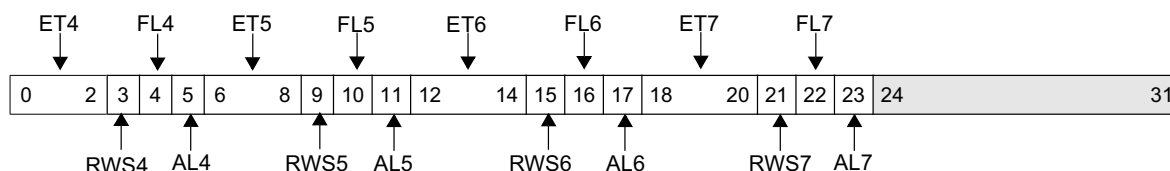


Figure 32-403. Bus Error Status Register 1 (SRAM0_BESR1)

0:2	ET4	Error type for master 4 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 4 is I2O messaging unit
3	RWS4	Read/write status for master 4 0 Error operation was a write operation 1 Error operation was a read operation	
4	FL4	Field lock for master 4 0 Fields are unlocked 1 Fields are locked	
5	AL4	BEAR address lock for master 4 0 BEAR address unlocked 1 BEAR address locked	
6:8	ET5	Error type for master 5 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 5 is media access layer
9	RWS5	Read/write status for master 5 0 Error operation was a write operation 1 Error operation was a read operation	
10	FL5	Field lock for master 5 0 Fields are unlocked 1 Fields are locked	
11	AL5	BEAR address lock for master 5 0 BEAR address unlocked 1 BEAR address locked	

SRAM0_BESR1 (cont.)

SRAM Bus Error Status Register 1

Preliminary User's Manual

12:14	ET6	Error type for master 6 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 6 is direct memory access (DMA) controller
15	RWS6	Read/write status for master 6 0 Error operation was a write operation 1 Error operation was a read operation	
16	FL6	Field lock for master 6 0 Fields are unlocked 1 Fields are locked	
17	AL6	BEAR address lock for master 6 0 BEAR address unlocked 1 BEAR address locked	
18:20	ET7	Error type for master 7 000 No error 001 Reserved 010 Parity error 011 Reserved 100 Protection error 101 Reserved 110 Reserved 111 Reserved	Master 7 is OPB to PLB bridge
21	RWS7	Read/write status for master 7 0 Error operation was a write operation 1 Error operation was a read operation	
22	FL7	Field lock for master 7 0 Fields are unlocked 1 Fields are locked	
23	AL7	BEAR address lock for master 7 0 BEAR address unlocked 1 BEAR address locked	
24:31		Reserved	

DCR 0x28 Read only

See *Core ID Register (SRAM0_CID)* on page 542.

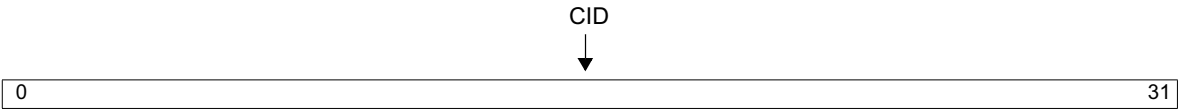


Figure 32-404. SRAM Internal Core Device ID Register (SRAM0_CID)

0:31	CID	Internal Core Device ID	H322B0000 (Read only).
------	-----	-------------------------	------------------------

DCR 0x2A Read/Write

See *Data Parity Checking Register (SRAM0_DPC)* on page 543.



Figure 32-405. Data Parity Checking Register (SRAM0_DPC)

0	DPC	Data Parity Check: 0 Disabled 1 Enabled	When set to 1, this bit enables data parity generation during write transactions and read transactions. When set to 0, this bit disables both functions.
1:31		Reserved	

DCR 0x029 Read only

See *Revision ID Register (SRAM0_REVID)* on page 543.

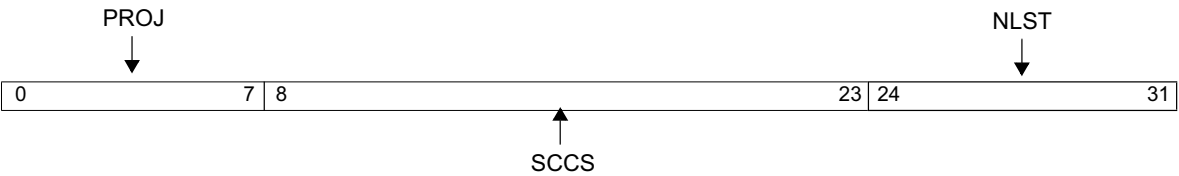


Figure 32-406. SRAM Internal Core Revision ID Register (SRAM0_REVID)

0:7	PROJ	Project Number	Read only.
8:23	SCCS	SCCS Version	Read only.
24:31	NLST	Netlist Number	Read only

DCR 0x027 Read/Write

See *Power Management Register (SRAM0_PMEG)* on page 542.

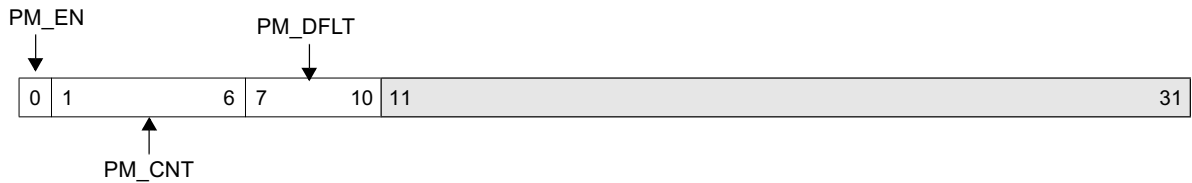


Figure 32-407. Power Management Register (SRAM0_PMEG)

0	PM_EN	Power Management enable: 0 Sleep mode disabled 1 Sleep mode enabled	
1:6	PM_CNT	Power Management Counter	The value (n) programmed into these register bits are the multiples of 16 clock cycles (n x 16). If PM_EN is set to 1, SRAM controller will go to sleep mode after being idle for (n x 16) clock cycles.
7:10	PM_DFLT	Power Management Default Wait Interval Hardwired to 1111.	The default value of 16 clock cycles. If PM_EN is set to 1, SRAM controller will go to sleep mode after being idle for 16 clock cycles.
11:31		Reserved	

Preliminary User’s Manual

DCR 0x020–0x023 Read/Write

See *SRAM Bank Configuration Registers (SRAM0_SB0CR - SRAM0_SB3CR)* on page 536.

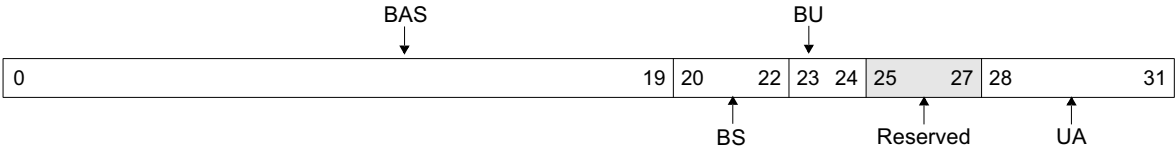


Figure 32-408. Memory Configuration (SRAM0_SB0CR - SRAM0_SB3CR)

0: 19	BAS	Base Address Select	Specifies the starting address of the SRAM bank.
20:22	BS	Bank Size 000 reserved 001 reserved 010 Reserved 011 Reserved 100 64 KB 101 Reserved 110 Reserved 111 Reserved	Specifies the size of the logical bank address range.
23:24	BU	Bank Usage 00 Disabled 01 Bank is valid for read only (RO) 10 Reserved 11 Bank is valid for read/write (R/W)	
25:27		Reserved	
28:31	UA	Four least significant bits of the upper 32-bit address.	

TAHx_MR

TAHx Mode Register

Preliminary User's Manual

MMIO 0x140000B60 TAH0_MR, 0x140000D60 TAH1_MR R/W

TAH Mode Register (TAHx_MR) on page 895.

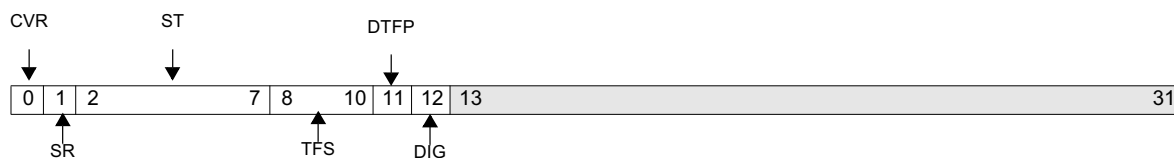


Figure 32-409. TAH Mode Register (TAHx_MR)

0	CVR	Checksum Verification on Receive 0 Checksum for IP, TCP and UDP packets disabled 1 Checksum for IP, TCP and UDP packets enabled
1	SR	TAH Software Reset 0 TAH reset is complete 1 Reset the TAH
2:7	ST	Send Threshold 000000 - 256 bytes 000001 - 256 bytes 000010 - 512 bytes
8:10	TFS	Transmit FIFO Size 001 Transmit FIFO size is 2Kbyte 010 Transmit FIFO size is 4Kbyte 011 Transmit FIFO size is 6Kbyte 100 Transmit FIFO size is 8Kbyte 101 Transmit FIFO size is 10Kbyte Note: The default is 001. Maximum of 10K FIFO is used.
11	DTFP	Disable Transmit FIFO Parity Protection 0 TAH checks for parity errors in the transmit FIFO 1 TAH does not check for parity errors in the transmit FIFO
12	DIG	Disable Interrupt Generation 0 TAH generates interrupts when errors are detected 1 TAH does not generate interrupts when errors are detected
13:31		Reserved

MMIO 0x140000B50 TAH0_REVID R/only, 0x140000D50 TAH1_REVID R/only

TAH Revision ID Register (TAHx_REVID) on page 895

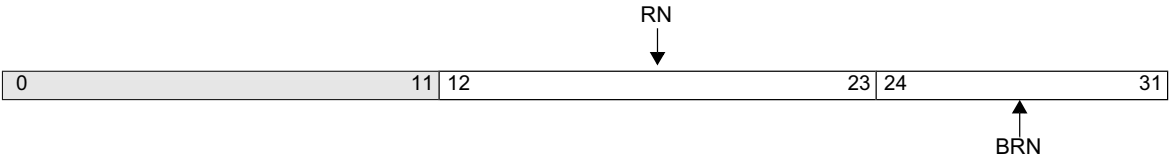


Figure 32-410. TAHx Revision ID Register (TAHx_REVID)

0:11		Reserved	
12:23	RN	Revision Number	Set to 1
24:31	BRN	Branch Revision Number	Set to 0

MMIO 0x140000B64-0x140000B78 TAH0_SSR0:SSR5 R/W, 0x140000D64-0x140000D78 TAH1_SSR0:SSR5 R/W
TAH Segment Size Registers 0:5 (TAHx_SSR0-TAHx_SSR5) on page 897

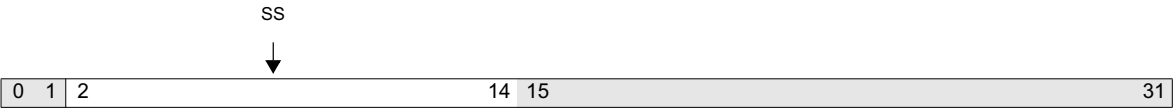


Figure 32-411. TAH Segment Size Register 0:5 (TAHx_SSR0-TAHx_SSR5)

0:1		Reserved
2:14	SS	Segment Size Defines the size of the segment in multiples of 2 bytes to be used when TCP segmentation is enabled.
15:31		Reserved

MMIO 0x140000B7C TAH0_TSR ReadOnly, 0x140000D7C TAH1_TSR ReadOnly

TAH Transmit Status Register (TAHx_TSR) on page 898

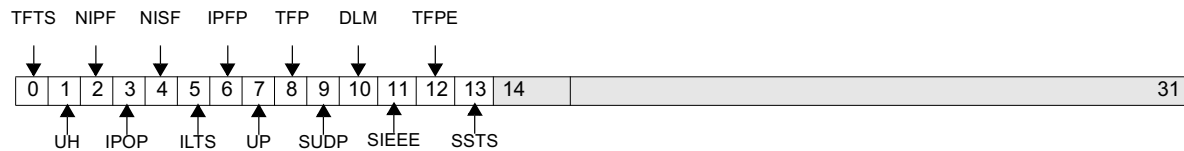


Figure 32-412. TAH Transmit Status Register (TAHx_TSR)

0	TFTS	Transmit FIFO Too Small The size of the transmit FIFO is not more than 80bytes bigger than the size of the packet when hardware checksum is enabled; or the size of the transmit FIFO is no more than 512 bytes bigger than the size of a segment when hardware segmentation is enabled.	
1	UH	Unrecognized Header The packet is in a format that is not supported by hardware	
2	NIPF	Not IP Version 4 Format The packet is not in IP version 4 format	
3	IPOP	IP Option Present IP option is present in the packet	
4	NISF	No IEEE SNAP Format The packet uses the IEEE 802 format but the SNAP header is incorrect	
5	ILTS	IP Length Too Short The total length in the IP header is smaller than 40 bytes.	
6	IPFP	IP Fragment Present The packet is part of an IP fragment.	
7	UP	Unsupported Protocol The protocol in use is neither TCP or UDP.	
8	TFP	TCP Flags Present Some TCP flags other than ACK are set.	
9	SUDP	Segmentation for UDP UDP packets cannot be segmented.	
10	DLM	Data Length Mismatch Amount of send data is smaller than the size of the TCP/IP header or the value in the IP length field.	If more send data is present, the extra data will be discarded with no error.
11	SIEEE	Segmentation for IEEE The size of the segment exceeds 1500 bytes for IEEE formatted packets when hardware segmentation is enabled.	
12	TFPE	Transmit FIFO Parity Error Parity error is detected in the Transmit FIFO.	
13	SSTS	Segment Size Too Small The size of the segment is less than 168 bytes for DIX formatted segments or 176 bytes for IEEE formatted segments when hardware segmentation is enabled	
14:31		Reserved	

MMIO 0x140000200 (UART0), 0x140000300 (UART1) Read/Write

See *UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)* on page 914.



Figure 32-413. UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)

8:15		Data bits
Note: UARTx_DLL is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

MMIO 0x140000201 (UART0), 0x140000301(UART 1) Read/Write

See *UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM)* on page 914.

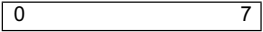


Figure 32-414. UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM)

0:7		Data bits
Note: UARTx_DLM is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

MMIO 0x140000202 (UART0), 0x140000302 Write-Only

See *FIFO Control Registers (UARTx_FCR)* on page 909.

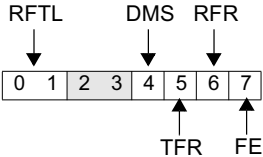


Figure 32-415. UART FIFO Control Registers (UARTx_FCR)

0:1	RFTL	Receiver FIFO Trigger Level 00 01 byte 01 16 bytes 10 32 bytes 11 56 bytes	
2:3		Reserved	
4	DMS	DMA Mode Select 0 Mode 0 = single transfer 1 Mode 1 = multiple transfers	Select single or multiple transfer mode if UARTx_FCR[7] = 1.
5	TFR	Transmitter FIFO Reset 0 Operation complete 1 Reset the transmitter FIFO	A 1 written to this bit clears all bytes in the transmitter FIFO and resets all of its counter logic to 0. The transmitter shift register is not cleared. This bit is self-clearing.
6	RFR	Receiver FIFO Reset 0 Operation complete 1 Reset the receiver FIFO	A 1 written to this bit clears all bytes in the receiver FIFO and resets all of its counter logic to 0. The receiver shift register is not cleared. This bit is self-clearing.
7	FE	FIFO Enable 0 Disable FIFOs 1 Enable FIFOs	When set to 1, both the receiver and transmitter FIFOs are enabled. When set to 0, both receiver and transmitter FIFOs are reset. Data is automatically cleared from both FIFOs when changing to and from FIFO and 16450 modes. Programming other bits will be ignored if this bit is not a 1.
Note: UARTx_FCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

MMIO 0x140000201 (UART0), 0x140000301 (UART1) Read/Write

See *Interrupt Enable Registers (UARTx_IER)* on page 906.

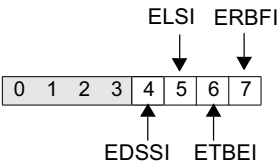


Figure 32-416. UART Interrupt Enable Registers (UARTx_IER)

0:3		Reserved	Always 0.
4	EDSSI	Modem Status Interrupt 0 Disable modem status interrupt 1 Enable modem status interrupt	
5	ELSI	Receiver Line Status Interrupt enable 0 Disable receiver line status interrupt 1 Enable receiver line status interrupt	
6	ETBEI	Transmitter Holding Register Empty Interrupt enable 0 Disable transmitter holding register empty interrupt 1 Enable transmitter holding register empty interrupt	
7	ERBFI	Received Data Available Interrupt enable 0 Disable received data available interrupt 1 Enable received data available interrupt	In FIFO mode, timeout interrupts follow the enable/disable state of ERBFI.
Note: UARTx_IER is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

MMIO 0x140000202 (UART0), 0x140000302 (UART1) Read-Only

See *Interrupt Identification Registers (UARTx_IIR)* on page 907.

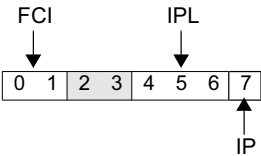


Figure 32-417. UART Interrupt Identification Registers (UARTx_IIR)

0:1	FCI	FIFO Control Indicator 00 FIFOs disabled (UARTx_FCR[FC] = 0) 01 Reserved 10 Reserved 11 FIFOs enabled (UARTx_FCR[FC] = 1)
2:3		Reserved
4:6	IPL	Interrupt Priority Level 000 Priority level 4 001 Priority level 3 010 Priority level 2 011 Priority level 1 100 Reserved 101 Reserved 110 Priority level 2 111 Reserved Note: Priority 1 is highest priority.
7	IP	Interrupt Pending 0 Interrupt is pending 1 No interrupt pending When set to 0, IIR contents can be used as a pointer to the appropriate interrupt service routine.
Note: UARTx_IIR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

MMIO 0x140000203 (UART0), 0x140000303 (UART1) Read/Write

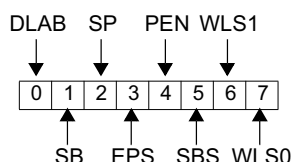
See *UART Line Control Registers (UARTx_LCR)* on page 910.

Figure 32-418. UART Line Control Registers (UARTx_LCR)

0	DLAB	Divisor Latch Access Bit 0 Address RBR, THR and IER with LTADR2-0 for read or write operation 1 Address Divisor Latches with LTADR2-0 for read or write operation	
1	SB	Set Break 0 Disable Break 1 Enable Break	Causes a break condition to be transmitted to the UART when the core is receiving. SOUT is forced to the spacing state (0). This bit acts only on SOUT and has no effect on the transmitter logic.
2	SP	Sticky Parity 0 Disable sticky parity 1 Enable sticky parity	If UARTx_LCR[EPS] = 1 and UARTx_LCR[PEN] = 1, the parity bit is transmitted and checked as 0. If UARTx_LCR[EPS] = 0 and UARTx_LCR[PEN] = 1, the parity bit is transmitted and checked as 1.
3	EPS	Even Parity Select 0 Generate odd parity 1 Generate even parity	This bit is significant only if UARTx_LCR[PEN] = 1.
4	PEN	Parity Enable 0 Disable parity checking 1 Enable parity checking	
5	SBS	Stop Bit Select 0 Characters have 1 stop bit 1 Characters have 1.5 or 2 stop bits	If UARTx_LCR[WPS1,WPS0] = 00, characters have 1.5 stop bits. For any other value of UARTx_LCR[WPS1,WPS0], characters have 2 stop bits. The receiver checks the first stop bit only, regardless of how many stop bits are selected.
6	WLS1	Word Length Select Bits 1 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters	
7	WLS0	Word Length Select Bits 0 00 Use 5-bit characters 01 Use 6-bit characters 10 Use 7-bit characters 11 Use 8-bit characters	

Note: UARTx_LCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.

UARTx_LSR

UART Line Status Registers

Preliminary User's Manual

MMIO 0x140000205 (UART0), 0x140000305 (UART1) R/W

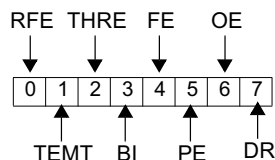
See *Line Status Registers (UARTx_LSR)* on page 912.

Figure 32-419. UART Line Status Registers (UARTx_LSR)

0	RFE	<p>Receiver FIFO Error Indicator</p> <p>0 In FIFO mode, reset to 0 when the processor reads the UARTx_LSR, provided there are no subsequent errors in the FIFO.</p> <p>1 There are one or more instances of parity error, framing error or break indication in the FIFO.</p>	Always 0 in 16450 mode.
1	TEMT	<p>Transmitter Empty Indicator</p> <p>0 Reset to 0 whenever the THR or the transmitter shift register contain a character. In FIFO mode, it is reset to 0 whenever the transmitter FIFO or the transmitter shift register contain a character.</p> <p>1 Set to 1 when the THR and the Transmitter shift register are both empty. In FIFO mode, it is set to 1 when the transmitter FIFO and the transmitter shift register are both empty.</p>	
2	THRE	<p>Transmitter Holding Register Empty Indicator</p> <p>0 Concurrent reset to 0 with the loading of the THR by the processor. In FIFO mode it is reset to 0 when at least one byte is written to the transmitter FIFO.</p> <p>1 Set to 1 when the UART is ready to accept a new character for transmission. In FIFO mode, this bit is set when the transmitter FIFO is empty.</p>	When UARTx_IER[THRE] = 1, the UART issues an interrupt to the PPC440GX interrupt controller. This bit is set to 1 when a character is transferred from the THR to the transmitter shift register.
3	BI	<p>Break Interrupt Indicator.</p> <p>0 Reset to 0 whenever processor reads Line Status Register (LSR).</p> <p>1 Set to 1 whenever the received data input is held at the spacing level (0) for longer than a full word transmission time.</p>	The full word transmission time is the time required for the start bit, data bits (can be 5–8 bits), parity and stop bits. In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Only one 0 character is loaded into the receiver FIFO when a break occurs. After the next valid start bit is received and has gone into the marking state, the next character transfer is enabled. Error causes a Receiver Line Status Interrupt.
4	FE	<p>Framing Error Indicator.</p> <p>0 Reset to 0 whenever processor reads LSR.</p> <p>1 Set to 1 whenever stop bit following the last data bit or parity bit is detected as 0 (spacing level). Indicates that a valid stop bit was not found in the received character.</p>	Error causes a Receiver Line Status Interrupt.
5	PE	<p>Parity Error Indicator.</p> <p>0 Reset to 0 whenever processor reads UARTx_LSR.</p> <p>1 Indicates that the received data character does not have the correct parity as determined by the even parity select bit (UARTx_LCR[EPS]). Set to 1 upon detection of a parity error.</p>	In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Error causes a Receiver Line Status Interrupt.

6	OE	<p>Overrun Error Indicator.</p> <p>0 Reset to 0 whenever processor reads UARTx_LSR.</p> <p>1 Data in the RBR was read by the processor before the next character was transferred into the UARTx_RBR, hence the original data was lost.</p>	<p>In FIFO mode, if the incoming data continues to fill the FIFO beyond the trigger level, an OE occurs only after the FIFO is completely full and the entire next character has been received in the receiver shift register. The processor is informed of the OE immediately upon occurrence. The character in the shift register will be overwritten and will not be transferred to the FIFO. Error causes a Receiver Line Status Interrupt.</p>
7	DR	<p>Receiver Data Ready Indicator.</p> <p>0 Reset to 0 when all data has been read from the receiver FIFO or the UARTx_RBR.</p> <p>1 An entire incoming character has been received into the UARTx_RBR or receiver FIFO.</p>	
Note: UARTx_LSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

UARTx_MCR

UART Modem Control Registers

Preliminary User's Manual

MMIO 0x140000204 (UART0), 0x140000304 (UART1) Read/Write

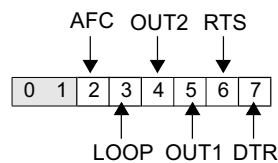
See *Modem Control Registers (UARTx_MCR)* on page 911.

Figure 32-420. UART Modem Control Registers (UARTx_MCR)

0:1		Reserved	Always 0.
2	AFC	Auto Flow Control 0 Hardware flow control disabled 1 Hardware flow control enabled	
3	LOOP	Loopback Mode 0 Disabled 1 Enabled	<p>Provides a local loopback feature for diagnostic testing of the UART. The following occurs:</p> <ol style="list-style-type: none"> 1. SOUT is set to the marking state (logic 1) SIN is disconnected. 2. The output of the transmitter shift register feeds the input of the receiver shift register. 3. The four modem control inputs $\overline{\text{DSR}}$, $\overline{\text{CTS}}$, $\overline{\text{RI}}$, and $\overline{\text{DCD}}$ are disconnected. 4. The four modem control outputs $\overline{\text{DTR}}$, $\overline{\text{RTS}}$, $\overline{\text{OUT1}}$, and $\overline{\text{OUT2}}$ are set to a logic 1 (their inactive state). 5. The four modem control outputs are connected internally to the four modem control inputs. <p>Transmitted data is immediately received to verify the UART transmit and receive data paths.</p> <p>Receiver and transmitter interrupts are operational. Their sources are external to the UART. Also operational are the modem control interrupts, but their source is the low-order 4 bits of UARTx_MCR instead of the modem control inputs to the UART. UARTx_IER still controls the interrupts.</p>
4	OUT2	User Output 2 0 $\overline{\text{OUT2}}$ inactive (1) 1 $\overline{\text{OUT2}}$ active (0)	The $\overline{\text{OUT2}}$ bit may be written and read but it provides no function
5	OUT1	User Output 1 0 $\overline{\text{OUT1}}$ inactive (1) 1 $\overline{\text{OUT1}}$ active (0)	The $\overline{\text{OUT1}}$ bit may be written and read but it provides no function
6	RTS	Request To Send 0 RTS inactive (1) 1 RTS active (0)	
7	DTR	Data Terminal Ready 0 $\overline{\text{DTR}}$ inactive (1) 1 $\overline{\text{DTR}}$ active (0)	
Note: UARTx_MCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

MMIO 0x140000206 (UART0), 0x140000306 (UART1) Read/Write

See *Modem Control Registers (UARTx_MCR)* on page 911.

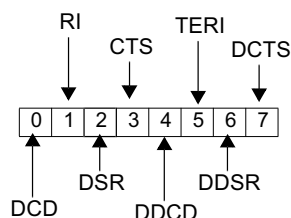


Figure 32-421. UART Modem Status Registers (UARTx_MSR)

0	DCD	Data Carrier Detect	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT2].
1	RI	Ring Indicator	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT1].
2	DSR	Data Set Ready	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[DTR].
3	CTS	Clear To Send	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[RTS].
4	DDCD	Delta Data Carrier Detect 0 Set when processor reads the Modem Status Register 1 $\overline{\text{DCD}}$ input changed state	Indicates that the $\overline{\text{DCD}}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
5	TERI	Trailing Edge of Ring Indicator 0 Set when processor reads the Modem Status Register 1 $\overline{\text{RI}}$ input changed from 0 to 1	Indicates that the $\overline{\text{RI}}$ input to the UART changed from 0 to 1 since the processor last read the Modem Status Register. A modem status interrupt is generated.
6	DDSR	Delta Data Set Ready 0 Set when processor reads the Modem Status Register 1 DSR input changed state	Indicates that the $\overline{\text{DSR}}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
7	DCTS	Delta Clear To Send 0 Set when processor reads the Modem Status Register 1 $\overline{\text{CTS}}$ input changed state	Indicates that the $\overline{\text{CTS}}$ input to the UART has changed state since the processor last read the Modem Status Register. A modem status interrupt is generated.
Note: UARTx_MSR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.			

MMIO 0x140000200, 0x140000300 Read-Only

See *Receiver Buffer Registers (UARTx_RBR)* on page 906.



Figure 32-422. *UART Receiver Buffer Registers (UARTx_RBR)*

0:7		Data bit
Note: UARTx_RBR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

MMIO 0x140000207 (UART0), 0x140000307 (UART1) Read/Write

See *Scratchpad Registers (UARTx_SCR)* on page 914.

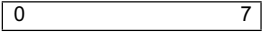


Figure 32-423. Scratchpad Registers (UARTx_SCR)

0:7		Data bits
Note: UARTx_SCR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

MMIO 0x140000200 (UART0), 0x140000300 (UART1) Write-Only

See *Transmitter Holding Registers (UARTx_THR)* on page 906.



Figure 32-424. UART Transmitter Holding Registers (UARTx_THR)

0:7		Data bit
Note: UARTx_THR is shown in standard PowerPC bit notation, where 0 is the MSb and 7 is the LSb.		

DCR 0x0C3 Read/Write

See *UIC0 Critical Register (UIC0_CR)* on page 382.

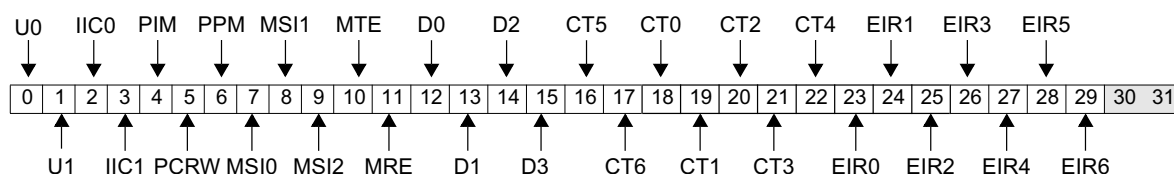


Figure 32-425. *UIC0 Critical Register (UIC0_CR)*

0	U0	UART0 Interrupt Class 0 UART0 interrupt non-critical. 1 UART0 interrupt is critical.
1	U1	UART1 Interrupt Class 0 UART1 interrupt is non-critical. 1 UART1 interrupt is critical.
2	IIC0	IIC0 Interrupt Class 0 IIC0 interrupt is non-critical. 1 IIC0 interrupt is critical.
3	IIC1	IIC1 Interrupt Class 0 IIC1 interrupt is non-critical. 1 IIC1 interrupt is critical.
4	PIM	PCI Inbound Message Interrupt Class 0 PCI inbound message interrupt is non-critical. 1 PCI inbound message interrupt is critical.
5	PCRW	PCI Command Register Write Interrupt Class 0 PCI command register write interrupt is non-critical. 1 PCI command register write interrupt is critical.
6	PPM	PCI Power Management Interrupt Class 0 PCI power management interrupt is non-critical. 1 PCI power management interrupt is critical.
7	MSI0	PCI MSI Level 0 Interrupt Class 0 PCI MSI level 0 interrupt is non-critical. 1 PCI MSI level 0 interrupt is critical.
8	MSI1	PCI MSI Level 1 Interrupt Class 0 PCI MSI level 1 interrupt is non-critical. 1 PCI MSI level 1 interrupt is critical.
9	MSI2	PCI MSI Level 2 Interrupt Class 0 PCI MSI level 2 interrupt is non-critical. 1 PCI MSI level 2 interrupt is critical.
10	MTE	MAL TX EOB Interrupt Class 0 MAL TX EOB interrupt is non-critical. 1 MAL TX EOB interrupt has is critical.
11	MRE	MAL RX EOB Interrupt Class 0 MAL RX EOB interrupt is non-critical. 1 MAL RX EOB interrupt has is critical.
12	D0	DMA Channel 0 Interrupt Class 0 DMA channel 0 interrupt is non-critical. 1 DMA channel 0 interrupt is critical.
13	D1	DMA Channel 1 Interrupt Class 0 DMA channel 1 interrupt is non-critical. 1 DMA channel 1 interrupt is critical.

14	D2	DMA Channel 2 Interrupt Class 0 DMA channel 2 interrupt is non-critical. 1 DMA channel 2 interrupt is critical.
15	D3	DMA Channel 3 Interrupt Class 0 DMA channel 3 interrupt is non-critical. 1 DMA channel 3 interrupt is critical.
16	CT5	GPT Compare Timer 5 Interrupt Class 0 Compare timer 5 interrupt is non-critical. 1 Compare timer 5 interrupt is critical.
17	CT6	GPT Compare Timer 6 Interrupt Class 0 Compare timer 6 interrupt is non-critical. 1 Compare timer 6 interrupt is critical.
18	CT0	GPT Compare Timer 0 Interrupt Class 0 Compare timer 0 interrupt is non-critical. 1 Compare timer 0 interrupt is critical.
19	CT1	GPT Compare Timer 1 Interrupt Class 0 Compare timer 1 interrupt is non-critical. 1 Compare timer 1 interrupt is critical.
20	CT2	GPT Compare Timer 2 Interrupt Class 0 Compare timer 2 interrupt is non-critical. 1 Compare timer 2 interrupt is critical.
21	CT3	GPT Compare Timer 3 Interrupt Class 0 Compare timer 3 interrupt is non-critical. 1 Compare timer 3 interrupt is critical.
22	CT4	GPT Compare Timer 4 Interrupt Class 0 Compare timer 4 interrupt is non-critical. 1 Compare timer 4 interrupt is critical.
23	EIR0	External IRQ 0 Interrupt Class 0 External IRQ 0 interrupt is non-critical. 1 External IRQ 0 interrupt is critical.
24	EIR1	External IRQ 1 Interrupt Class 0 External IRQ 1 interrupt is non-critical. 1 External IRQ 1 interrupt is critical.
25	EIR2	External IRQ 2 Interrupt Class 0 External IRQ 2 interrupt is non-critical. 1 External IRQ 2 interrupt is critical.
26	EIR3	External IRQ 3 Interrupt Class 0 External IRQ 3 interrupt is non-critical. 1 External IRQ 3 interrupt is critical.
27	EIR4	External IRQ 4 Interrupt Class 0 External IRQ 4 interrupt is non-critical. 1 External IRQ 4 interrupt is critical.
28	EIR5	External IRQ 5 Interrupt Class 0 External IRQ 5 interrupt is non-critical. 1 An external IRQ 5 interrupt is critical.
29	EIR6	External IRQ 6 Interrupt Class 0 External IRQ 6 interrupt is non-critical. 1 External IRQ 6 interrupt is critical.
30		Reserved in PPC440GX but can be programmed to use UIC 1 non-critical interrupt
31		Reserved in PPC440GX but can be programmed to use UIC 1 critical interrupt

DCR 0x0C2 Read/Write

See *UIC0 Enable Register (UIC0_ER)* on page 374.

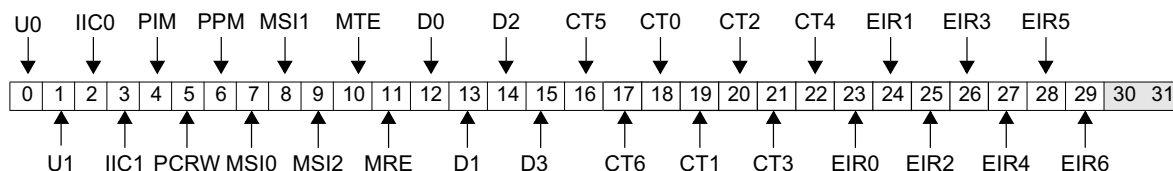


Figure 32-426. *UIC0 Enable Register (UIC0_ER)*

0	U0	UART0 Interrupt Enable 0 UART0 interrupt is disabled. 1 UART0 interrupt is enabled.
1	U1	UART1 Interrupt Enable 0 UART1 interrupt is disabled. 1 UART1 interrupt is enabled.
2	IIC0	IIC0 Interrupt Enable 0 IIC0 interrupt is disabled. 1 IIC0 interrupt is enabled.
3	IIC1	IIC1 Interrupt Enable 0 IIC1 interrupt is disabled. 1 IIC1 interrupt is enabled.
4	PIM	PCI Inbound Message Interrupt Enable 0 PCI inbound message interrupt is disabled. 1 PCI inbound message interrupt is enabled.
5	PCRW	PCI Command Register Write Interrupt Enable 0 PCI command register write interrupt is disabled. 1 PCI command register write interrupt is enabled.
6	PPM	PCI Power Management Interrupt Enable 0 PCI power management interrupt is disabled. 1 PCI power management interrupt is enabled.
7	MSI0	PCI MSI Level 0 Interrupt Enable 0 PCI MSI Level 0 interrupt is disabled. 1 PCI MSI Level 0 interrupt is enabled.
8	MSI1	PCI MSI Level 1 Interrupt Enable 0 PCI MSI Level 1 Interrupt is disabled. 1 PCI MSI Level 1 interrupt is enabled.
9	MSI2	PCI MSI Level 2 Interrupt Enable 0 PCI MSI Level 2 interrupt is disabled. 1 PCI MSI Level 2 interrupt is enabled.
10	MTE	MAL TX EOB Interrupt Enable 0 MAL TX EOB interrupt is disabled. 1 MAL TX EOB interrupt has is enabled.
11	MRE	MAL RX EOB Interrupt Enable 0 MAL RX EOB interrupt is disabled. 1 MAL RX EOB interrupt has is enabled.
12	D0	DMA Channel 0 Interrupt Enable 0 DMA channel 0 interrupt is disabled. 1 DMA channel 0 interrupt is enabled.
13	D1	DMA Channel 1 Interrupt Enable 0 DMA channel 1 interrupt is disabled. 1 DMA channel 1 interrupt is enabled.

UIC0_ER (cont.)

UIC 0 Interrupt Enable Register

Preliminary User's Manual

14	D2	DMA Channel 2 Interrupt Enable 0 DMA channel 2 interrupt is disabled. 1 DMA channel 2 interrupt is enabled.
15	D3	DMA Channel 3 Interrupt Enable 0 DMA channel 3 interrupt is disabled. 1 DMA channel 3 interrupt is enabled.
16	CT5	GPT Compare Timer 5 Interrupt Enable 0 Compare timer 5 interrupt is disabled. 1 Compare timer 5 interrupt is enabled.
17	CT6	GPT Compare Timer 6 Interrupt Enable 0 Compare timer 6 interrupt is disabled. 1 Compare timer 6 interrupt is enabled.
18	CT0	GPT Compare Timer0 Interrupt Enable 0 Compare timer 0 interrupt is disabled. 1 Compare timer 0 interrupt is enabled.
19	CT1	GPT Compare Timer 1 Interrupt Enable 0 Compare timer 1 interrupt is disabled. 1 Compare timer 1 interrupt is enabled.
20	CT2	GPT Compare Timer 2 Interrupt Enable 0 Compare timer 2 interrupt is disabled. 1 Compare timer 2 interrupt is enabled.
21	CT3	GPT Compare Timer 3 Interrupt Enable 0 Compare timer 3 interrupt is disabled. 1 Compare timer 3 interrupt is enabled.
22	CT4	GPT Compare Timer 4 Interrupt Enable 0 Compare timer 4 interrupt is disabled. 1 Compare timer 4 interrupt is enabled.
23	EIR0	External IRQ 0 Interrupt Enable 0 External IRQ 0 interrupt is disabled. 1 External IRQ 0 interrupt is enabled.
24	EIR1	External IRQ 1 Interrupt Enable 0 External IRQ 1 interrupt is disabled. 1 External IRQ 1 interrupt is enabled.
25	EIR2	External IRQ 2 Interrupt Enable 0 External IRQ 2 interrupt is disabled. 1 External IRQ 2 interrupt is enabled.
26	EIR3	External IRQ 3 Interrupt Enable 0 External IRQ 3 interrupt is disabled. 1 External IRQ 3 interrupt is enabled.
27	EIR4	External IRQ 4 Interrupt Enable 0 External IRQ 4 interrupt is disabled. 1 External IRQ 4 interrupt is enabled.
28	EIR5	External IRQ 5 Interrupt Enable 0 External IRQ 5 interrupt is disabled. 1 An external IRQ 5 interrupt is enabled.
29	EIR6	External IRQ 6 Interrupt Enable 0 External IRQ 6 interrupt is disabled. 1 External IRQ 6 interrupt is enabled.
30		Reserved in PPC440GX but can be programmed to use UIC 1 non-critical interrupt
31		Reserved in PPC440GX but can be programmed to use UIC 1 critical interrupt

DCR 0x0C6 Read-Only

See *UIC0 Masked Status Register (UIC0_MSR)* on page 403.

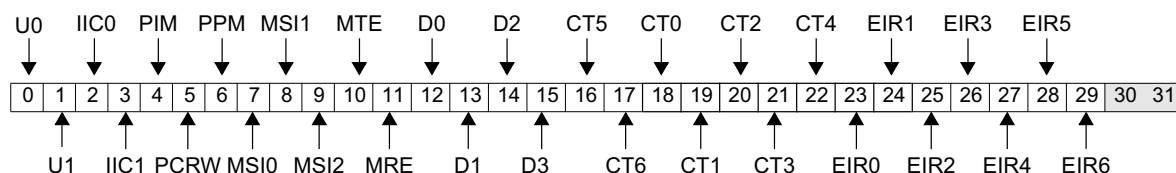


Figure 32-427. *UIC0 Masked Status Register (UIC0_MSR)*

0	U0	UART0 Masked Interrupt Status 0 UART0 masked interrupt has not occurred. 1 UART0 masked interrupt occurred.
1	U1	UART1 Masked Interrupt Status 0 UART1 masked interrupt has not occurred. 1 UART1 masked interrupt occurred.
2	IIC0	IIC0 Masked Interrupt Status 0 IIC0 masked interrupt has not occurred. 1 IIC0 masked interrupt occurred.
3	IIC1	IIC1 Masked Interrupt Status 0 IIC1 masked interrupt has not occurred. 1 IIC1 masked interrupt occurred.
4	PIM	PCI Inbound Message Masked Interrupt Status 0 PCI inbound message masked interrupt has not occurred. 1 PCI inbound message masked interrupt occurred.
5	PCRW	PCI Command Register Write Masked Interrupt Status 0 PCI command register write masked interrupt has not occurred. 1 PCI command register write masked interrupt occurred.
6	PPM	PCI Power Management Masked Interrupt Status 0 PCI power management masked interrupt has not occurred. 1 PCI power management masked interrupt occurred.
7	MSI0	PCI MSI Level 0 Masked Interrupt Status 0 PCI MSI level 0 masked interrupt has not occurred. 1 PCI MSI level 0 masked interrupt occurred.
8	MSI1	PCI MSI Level 1 Masked Interrupt Status 0 PCI MSI level 1 masked interrupt has not occurred. 1 PCI MSI level 1 masked interrupt occurred.
9	MSI2	PCI MSI Level 2 Masked Interrupt Status 0 PCI MSI level 2 masked interrupt has not occurred. 1 PCI MSI level 2 masked interrupt occurred.
10	MTE	MAL TX EOB Masked Interrupt Status 0 MAL TX EOB interrupt has not occurred. 1 MAL TX EOB interrupt has occurred.
11	MRE	MAL RX EOB Masked Interrupt Status 0 MAL RX EOB interrupt has not occurred. 1 MAL RX EOB interrupt has occurred.
12	D0	DMA Channel 0 Masked Interrupt Status 0 DMA channel 0 interrupt has not occurred. 1 DMA channel 0 interrupt occurred.

13	D1	DMA Channel 1 Masked Interrupt Status 0 DMA channel 1 interrupt has not occurred. 1 DMA channel 1 interrupt occurred.
14	D2	DMA Channel 2 Masked Interrupt Status 0 DMA channel 2 interrupt has not occurred. 1 DMA channel 2 interrupt occurred.
15	D3	DMA Channel 3 Masked Interrupt Status 0 DMA channel 3 interrupt has not occurred. 1 DMA channel 3 interrupt occurred.
16	CT5	GPT Compare Timer 5 Masked Interrupt Status 0 Compare timer 5 interrupt has not occurred. 1 Compare timer 5 interrupt occurred.
17	CT6	GPT Compare Timer 6 Masked Interrupt Status 0 Compare timer 6 interrupt has not occurred. 1 Compare timer 6 interrupt occurred.
18	CT0	GPT Compare Timer 0 Masked Interrupt Status 0 Compare timer 0 interrupt has not occurred. 1 Compare timer 0 interrupt occurred.
19	CT1	GPT Compare Timer 1 Masked Interrupt Status 0 Compare timer 1 interrupt has not occurred. 1 Compare timer 1 interrupt occurred.
20	CT2	GPT Compare Timer 2 Masked Interrupt Status 0 Compare timer 2 interrupt has not occurred. 1 Compare timer 2 interrupt occurred.
21	CT3	GPT Compare Timer 3 Masked Interrupt Status 0 Compare timer 3 interrupt has not occurred. 1 Compare timer 3 interrupt occurred.
22	CT4	GPT Compare Timer 4 Masked Interrupt Status 0 Compare timer 4 interrupt has not occurred. 1 Compare timer 4 interrupt occurred.
23	EIR0	External IRQ 0 Masked Interrupt Status 0 External IRQ 0 interrupt has not occurred. 1 External IRQ 0 interrupt occurred.
24	EIR1	External IRQ 1 Masked Interrupt Status 0 External IRQ 1 interrupt has not occurred. 1 External IRQ 1 interrupt occurred.
25	EIR2	External IRQ 2 Masked Interrupt Status 0 External IRQ 2 interrupt has not occurred. 1 External IRQ 2 interrupt occurred.
26	EIR3	External IRQ 3 Masked Interrupt Status 0 External IRQ 3 interrupt has not occurred. 1 External IRQ 3 interrupt occurred.
27	EIR4	External IRQ 4 Masked Interrupt Status 0 External IRQ 4 interrupt has not occurred. 1 External IRQ 4 interrupt occurred.
28	EIR5	External IRQ 5 Masked Interrupt Status 0 External IRQ 5 interrupt has not occurred. 1 An external IRQ 5 interrupt occurred.
29	EIR6	External IRQ 6 Masked Interrupt Status 0 External IRQ 6 interrupt has not occurred. 1 External IRQ 6 interrupt occurred.
30		Reserved in PPC440GX but can be programmed to use UIC 1 non-critical interrupt
31		Reserved in PPC440GX but can be programmed to use UIC 1 critical interrupt

DCR 0x0C4 Read/Write

See *UIC0 Polarity Register (UIC0_PR)* on page 388.

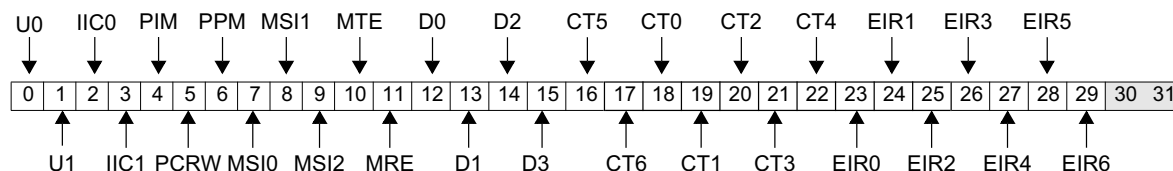


Figure 32-428. *UIC0 Polarity Register (UIC0_PR)*

0	U0	UART0 Interrupt Polarity 0 UART0 interrupt has negative polarity. 1 UART0 interrupt has positive polarity.
1	U1	UART1 Interrupt Polarity 0 UART1 interrupt has negative polarity. 1 UART1 interrupt has positive polarity.
2	IIC0	IIC0 Interrupt Polarity 0 IIC0 interrupt has negative polarity. 1 IIC0 interrupt has positive polarity.
3	IIC1	IIC1 Interrupt Polarity 0 IIC1 interrupt has negative polarity. 1 IIC1 interrupt has positive polarity.
4	PIM	PCI Inbound Message Interrupt Polarity 0 PCI inbound message interrupt has negative polarity. 1 PCI inbound message interrupt has positive polarity.
5	PCRW	PCI Command Register Write Interrupt Polarity 0 PCI command register write interrupt has negative polarity. 1 PCI command register write interrupt has positive polarity.
6	PPM	PCI Power Management Interrupt Polarity 0 PCI power management interrupt has negative polarity. 1 PCI power management interrupt has positive polarity.
7	MSI0	PCI MSI Level 0 Interrupt Polarity 0 PCI MSI level 0 interrupt has negative polarity. 1 PCI MSI level 0 interrupt has positive polarity.
8	MSI1	PCI MSI Level 1 Interrupt Polarity 0 PCI MSI level 1 interrupt has negative polarity. 1 PCI MSI level 1 interrupt has positive polarity.
9	MSI2	PCI MSI Level 2 Interrupt Polarity 0 PCI MSI level 2 interrupt has negative polarity. 1 PCI MSI level 2 interrupt has positive polarity.
10	MTE	MAL TX EOB Interrupt Polarity 0 MAL TX EOB interrupt has negative polarity. 1 MAL TX EOB interrupt has positive polarity.
11	MRE	MAL RX EOB Interrupt Polarity 0 MAL RX EOB interrupt has negative polarity. 1 MAL RX EOB interrupt has positive polarity.
12	D0	DMA Channel 0 Interrupt Polarity 0 DMA channel 0 interrupt has negative polarity. 1 DMA channel 0 interrupt has positive polarity.
13	D1	DMA Channel 1 Interrupt Polarity 0 DMA channel 1 interrupt has negative polarity. 1 DMA channel 1 interrupt has positive polarity.

14	D2	DMA Channel 2 Interrupt Polarity 0 DMA channel 2 interrupt has negative polarity. 1 DMA channel 2 interrupt has positive polarity.
15	D3	DMA Channel 3 Interrupt Polarity 0 DMA channel 3 interrupt has negative polarity. 1 DMA channel 3 interrupt has positive polarity.
16	CT5	GPT Compare Timer 5 Interrupt Polarity 0 Compare timer 5 interrupt has negative polarity. 1 Compare timer 5 interrupt has positive polarity.
17	CT6	GPT Compare Timer 6 Interrupt Polarity 0 Compare timer 6 interrupt has negative polarity. 1 Compare timer 6 interrupt has positive polarity.
18	CT0	GPT Compare Timer 0 Interrupt Polarity 0 Compare timer 0 interrupt has negative polarity. 1 Compare timer 0 interrupt has positive polarity.
19	CT1	GPT Compare Timer 1 Interrupt Polarity 0 Compare timer 1 interrupt has negative polarity. 1 Compare timer 1 interrupt has positive polarity.
20	CT2	GPT Compare Timer 2 Interrupt Polarity 0 Compare timer 2 interrupt has negative polarity. 1 Compare timer 2 interrupt has positive polarity.
21	CT3	GPT Compare Timer 3 Interrupt Polarity 0 Compare timer 3 interrupt has negative polarity. 1 Compare timer 3 interrupt has positive polarity.
22	CT4	GPT Compare Timer 4 Interrupt Polarity 0 Compare timer 4 interrupt has negative polarity. 1 Compare timer 4 interrupt has positive polarity.
23	EIR0	External IRQ 0 Interrupt Polarity 0 External IRQ 0 interrupt has negative polarity. 1 External IRQ 0 interrupt has positive polarity.
24	EIR1	External IRQ 1 Interrupt Polarity 0 External IRQ 1 interrupt has negative polarity. 1 External IRQ 1 interrupt has positive polarity.
25	EIR2	External IRQ 2 Interrupt Polarity 0 External IRQ 2 interrupt has negative polarity. 1 External IRQ 2 interrupt has positive polarity.
26	EIR3	External IRQ 3 Interrupt Polarity 0 External IRQ 3 interrupt has negative polarity. 1 External IRQ 3 interrupt has positive polarity.
27	EIR4	External IRQ 4 Interrupt Polarity 0 External IRQ 4 interrupt has negative polarity. 1 External IRQ 4 interrupt has positive polarity.
28	EIR5	External IRQ 5 Interrupt Polarity 0 External IRQ 5 interrupt has negative polarity. 1 An external IRQ 5 interrupt has positive polarity.
29	EIR6	External IRQ 6 Interrupt Polarity 0 External IRQ 6 interrupt has negative polarity. 1 External IRQ 6 interrupt has positive polarity.
30		Reserved in PPC440GX but can be programmed to use UIC 1 non-critical interrupt
31		Reserved in PPC440GX but can be programmed to use UIC 1 critical interrupt

DCR 0x0C0 Read/Write

See *UIC0 Status Register (UIC0_SR)* on page 368.

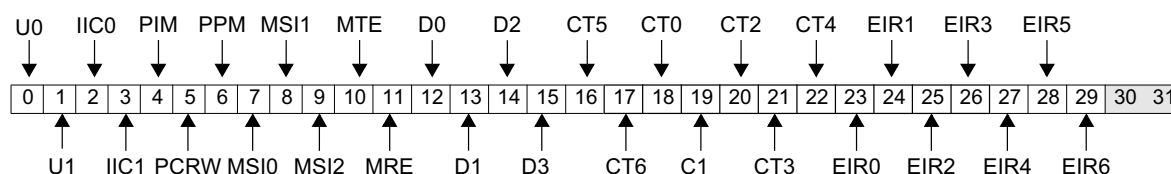


Figure 32-429. *UIC0 Status Register (UIC0_SR)*

0	U0	UART0 Interrupt Status 0 UART0 interrupt has not occurred. 1 UART0 interrupt occurred.
1	U1	UART1 Interrupt Status 0 UART1 interrupt has not occurred. 1 UART1 interrupt occurred.
2	IIC0	IIC0 Interrupt Status 0 IIC0 interrupt has not occurred. 1 IIC0 interrupt occurred.
3	IIC1	IIC1 Interrupt Status 0 IIC1 interrupt has not occurred. 1 IIC1 interrupt occurred.
4	PIM	PCI Inbound Message Interrupt Status 0 PCI inbound message interrupt has not occurred. 1 PCI inbound message interrupt occurred.
5	PCRW	PCI Command Register Write Interrupt Status 0 PCI command register write interrupt has not occurred. 1 PCI command register write interrupt occurred.
6	PPM	PCI Power Management Interrupt Status 0 PCI power management interrupt has not occurred. 1 PCI power management interrupt occurred.
7	MSI0	PCI MSI Level 0 Interrupt Status 0 PCI MSI Level 0 interrupt has not occurred. 1 PCI MSI Level 0 interrupt occurred.
8	MSI1	PCI MSI Level 1 Interrupt Status 0 PCI MSI Level 1 interrupt has not occurred. 1 PCI MSI Level 1 interrupt occurred.
9	MSI2	PCI MSI Level 2 Interrupt Status 0 PCI MSI Level 2 interrupt has not occurred. 1 PCI MSI Level 2 interrupt occurred.
10	MTE	MAL TX EOB Interrupt Status 0 MAL TX EOB interrupt has not occurred. 1 MAL TX EOB interrupt has occurred.
11	MRE	MAL RX EOB Interrupt Status 0 MAL RX EOB interrupt has not occurred. 1 MAL RX EOB interrupt has occurred.
12	D0	DMA Channel 0 Interrupt Status 0 DMA channel 0 interrupt has not occurred. 1 DMA channel 0 interrupt occurred.
13	D1	DMA Channel 1 Interrupt Status 0 DMA channel 1 interrupt has not occurred. 1 DMA channel 1 interrupt occurred.

14	D2	DMA Channel 2 Interrupt Status 0 DMA channel 2 interrupt has not occurred. 1 DMA channel 2 interrupt occurred.
15	D3	DMA Channel 3 Interrupt Status 0 DMA channel 3 interrupt has not occurred. 1 DMA channel 3 interrupt occurred.
16	CT5	GPT Compare Timer 5 Interrupt Status 0 Compare timer 5 interrupt has not occurred. 1 Compare timer 5 interrupt occurred.
17	CT6	GPT Compare Timer 6 Interrupt Status 0 Compare timer 6 interrupt has not occurred. 1 Compare timer 6 interrupt occurred.
18	CT0	GPT Compare Timer 0 Interrupt Status 0 Compare timer 0 interrupt has not occurred. 1 Compare timer 0 interrupt occurred.
19	CT1	GPT Compare Timer 1 Interrupt Status 0 Compare timer 1 interrupt has not occurred. 1 Compare timer 1 interrupt occurred.
20	CT2	GPT Compare Timer 2 Interrupt Status 0 Compare timer 2 interrupt has not occurred. 1 Compare timer 2 interrupt occurred.
21	CT3	GPT Compare Timer 3 Interrupt Status 0 Compare timer 3 interrupt has not occurred. 1 Compare timer 3 interrupt occurred.
22	CT4	GPT Compare Timer 4 Interrupt Status 0 Compare timer 4 interrupt has not occurred. 1 Compare timer 4 interrupt occurred.
23	EIR0	External IRQ 0 Interrupt Status 0 External IRQ 0 interrupt has not occurred. 1 External IRQ 0 interrupt occurred.
24	EIR1	External IRQ 1 Interrupt Status 0 External IRQ 1 interrupt has not occurred. 1 External IRQ 1 interrupt occurred.
25	EIR2	External IRQ 2 Interrupt Status 0 External IRQ 2 interrupt has not occurred. 1 External IRQ 2 interrupt occurred.
26	EIR3	External IRQ 3 Interrupt Status 0 External IRQ 3 interrupt has not occurred. 1 External IRQ 3 interrupt occurred.
27	EIR4	External IRQ 4 Interrupt Status 0 External IRQ 4 interrupt has not occurred. 1 External IRQ 4 interrupt occurred.
28	EIR5	External IRQ 5 Interrupt Status 0 External IRQ 5 interrupt has not occurred. 1 An external IRQ 5 interrupt occurred.
29	EIR6	External IRQ 6 Interrupt Status 0 External IRQ 6 interrupt has not occurred. 1 External IRQ 6 interrupt occurred.
30		Reserved in PPC440GX but can be programmed to use UIC 1 non-critical interrupt
31		Reserved in PPC440GX but can be programmed to use UIC 1 critical interrupt

DCR 0x0C5 Read/Write

See *UIC0 Trigger Register (UIC0_TR)* on page 396.

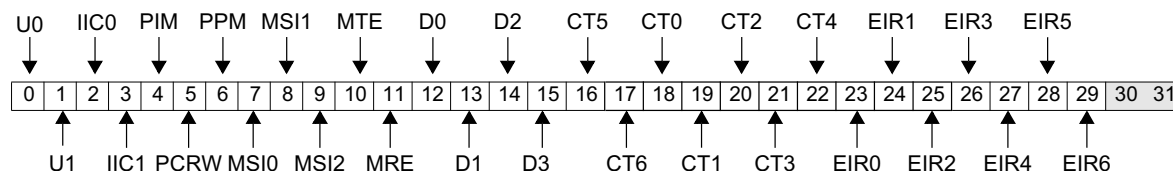


Figure 32-430. UIC0 Trigger Register (UIC0_TR)

0	U0	UART0 Interrupt Trigger 0 UART0 interrupt is level-sensitive. 1 UART0 interrupt is edge-sensitive.
1	U1	UART1 Interrupt Trigger 0 UART1 interrupt is level-sensitive. 1 UART1 interrupt is edge-sensitive.
2	IIC0	IIC0 Interrupt Trigger 0 IIC0 interrupt is level-sensitive. 1 IIC0 interrupt is edge-sensitive.
3	IIC1	IIC1 Interrupt Trigger 0 IIC1 interrupt is level-sensitive. 1 IIC1 interrupt is edge-sensitive.
4	PIM	PCI Inbound Message Interrupt Trigger 0 PCI inbound message interrupt is level-sensitive. 1 PCI inbound message interrupt is edge-sensitive.
5	PCRW	PCI Command Register Write Interrupt Trigger 0 PCI command register write interrupt is level-sensitive. 1 PCI command register write interrupt is edge-sensitive.
6	PPM	PCI Power Management Interrupt Trigger 0 PCI power management interrupt is level-sensitive. 1 PCI power management interrupt is edge-sensitive.
7	MSI0	PCI MSI Level 0 Interrupt Trigger 0 PCI MSI level 0 interrupt is level-sensitive. 1 PCI MSI level 0 interrupt is edge-sensitive.
8	MSI1	PCI MSI Level 1 Interrupt Trigger 0 PCI MSI level 1 interrupt is level-sensitive. 1 PCI MSI level 1 interrupt is edge-sensitive.
9	MSI2	PCI MSI Level 2 Interrupt Trigger 0 PCI MSI level 2 interrupt is level-sensitive. 1 PCI MSI level 2 interrupt is edge-sensitive.
10	MTE	MAL TX EOB Interrupt Trigger 0 MAL TX EOB interrupt is level-sensitive. 1 MAL TX EOB interrupt has is edge-sensitive.
11	MRE	MAL RX EOB Interrupt Trigger 0 MAL RX EOB interrupt is level-sensitive. 1 MAL RX EOB interrupt has is edge-sensitive.
12	D0	DMA Channel 0 Interrupt Trigger 0 DMA channel 0 interrupt is level-sensitive. 1 DMA channel 0 interrupt is edge-sensitive.
13	D1	DMA Channel 1 Interrupt Trigger 0 DMA channel 1 interrupt is level-sensitive. 1 DMA channel 1 interrupt is edge-sensitive.

14	D2	DMA Channel 2 Interrupt Trigger 0 DMA channel 2 interrupt is level-sensitive. 1 DMA channel 2 interrupt is edge-sensitive.
15	D3	DMA Channel 3 Interrupt Trigger 0 DMA channel 3 interrupt is level-sensitive. 1 DMA channel 3 interrupt is edge-sensitive.
16	CT5	GPT Compare Timer 5 Interrupt Trigger 0 Compare timer 5 interrupt is level-sensitive. 1 Compare timer 5 interrupt is edge-sensitive.
17	CT6	GPT Compare Timer 6 Interrupt Trigger 0 Compare timer 6 interrupt is level-sensitive. 1 Compare timer 6 interrupt is edge-sensitive.
18	CT0	GPT Compare Timer 0 Interrupt Trigger 0 Compare timer 0 interrupt is level-sensitive. 1 Compare timer 0 interrupt is edge-sensitive.
19	CT1	GPT Compare Timer 1 Interrupt Trigger 0 Compare timer 1 interrupt is level-sensitive. 1 Compare timer 1 interrupt is edge-sensitive.
20	CT2	GPT Compare Timer 2 Interrupt Trigger 0 Compare timer 2 interrupt is level-sensitive. 1 Compare timer 2 interrupt is edge-sensitive.
21	CT3	GPT Compare Timer 3 Interrupt Trigger 0 Compare timer 3 interrupt is level-sensitive. 1 Compare timer 3 interrupt is edge-sensitive.
22	CT4	GPT Compare Timer 4 Interrupt Trigger 0 Compare timer 4 interrupt is level-sensitive. 1 Compare timer 4 interrupt is edge-sensitive.
23	EIR0	External IRQ 0 Interrupt Trigger 0 External IRQ 0 interrupt is level-sensitive. 1 External IRQ 0 interrupt is edge-sensitive.
24	EIR1	External IRQ 1 Interrupt Trigger 0 External IRQ 1 interrupt is level-sensitive. 1 External IRQ 1 interrupt is edge-sensitive.
25	EIR2	External IRQ 2 Interrupt Trigger 0 External IRQ 2 interrupt is level-sensitive. 1 External IRQ 2 interrupt is edge-sensitive.
26	EIR3	External IRQ 3 Interrupt Trigger 0 External IRQ 3 interrupt is level-sensitive. 1 External IRQ 3 interrupt is edge-sensitive.
27	EIR4	External IRQ 4 Interrupt Trigger 0 External IRQ 4 interrupt is level-sensitive. 1 External IRQ 4 interrupt is edge-sensitive.
28	EIR5	External IRQ 5 Interrupt Trigger 0 External IRQ 5 interrupt is level-sensitive. 1 An external IRQ 5 interrupt is edge-sensitive.
29	EIR6	External IRQ 6 Interrupt Trigger 0 External IRQ 6 interrupt is level-sensitive. 1 External IRQ 6 interrupt is edge-sensitive.
30		Reserved in PPC440GX but can be programmed to use UIC 1 non-critical interrupt
31		Reserved in PPC440GX but can be programmed to use UIC 1 critical interrupt

DCR 0x0C8 Write-Only

See *UIC0 Vector Configuration Register (UIC0_VCR)* on page 409.

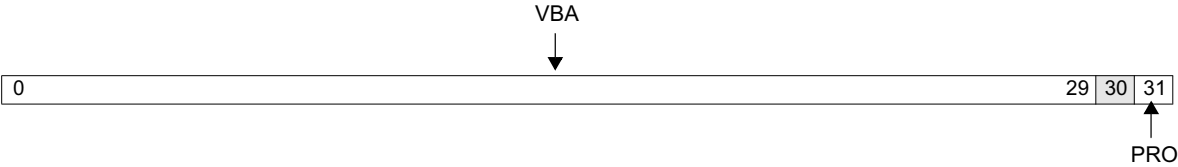


Figure 32-431. UIC0 Vector Configuration Register (UIC0_VCR)

0:29	VBA	Vector Base Address
30		Reserved
31	PRO	Priority Ordering 0 UIC0_SR[31] is the highest priority interrupt. 1 UIC0_SR[0] is the highest priority interrupt. Note: Vector generation is not performed for non-critical interrupts.

DCR 0x0C7 Read-Only

See *UIC0 Vector Register (UIC0_VR)* on page 413.

0	31
---	----

Figure 32-432. UIC Vector Register (UIC0_VR)

0:31	Interrupt Vector
------	------------------

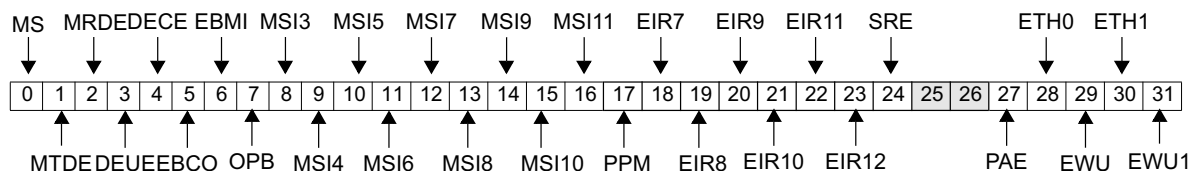
DCR 0x0D3 Read/WriteSee *UIC1 Critical Register (UIC1_CR)* on page 384.

Figure 32-433. UIC1 Critical Register (UIC1_CR)

0	MS	MAL SERR Interrupt Class 0 MAL SERR interrupt is non-critical. 1 MAL SERR interrupt is critical.
1	MTDE	MAL TXDE Interrupt Class 0 MAL TXDE interrupt is non-critical. 1 MAL TXDE interrupt is critical.
2	MRDE	MAL RXDE Interrupt Class 0 MAL RXDE interrupt is non-critical. 1 MAL RXDE interrupt is critical.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Class 0 DDRSDRAM ECC uncorrectable error interrupt is non-critical. 1 DDRSDRAM ECC uncorrectable error interrupt is critical.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Class 0 DDRSDRAM ECC correctable error interrupt is non-critical. 1 DDRSDRAM ECC correctable error interrupt is critical.
5	EBCO	EBCO Interrupt Class 0 EBCO interrupt is non-critical. 1 EBCO interrupt is critical.
6	EBMI	EBMI Interrupt Class 0 EBMI interrupt is non-critical. 1 EBMI interrupt is critical.
7	OPB	OPB to PLB Bridge Interrupt Class 0 OPB to PLB bridge interrupt is non-critical. 1 OPB to PLB bridge interrupt is critical.
8	MSI3	PCI MSI Level 3 Interrupt Class 0 PCI MSI level 3 interrupt is non-critical. 1 PCI MSI level 3 interrupt is critical.
9	MSI4	PCI MSI Level 4 Interrupt Class 0 PCI MSI level 4 interrupt is non-critical. 1 PCI MSI level 4 interrupt is critical.
10	MSI5	PCI MSI Level 5 Interrupt Class 0 PCI MSI level 5 interrupt is non-critical. 1 PCI MSI level 5 interrupt is critical.
11	MSI6	PCI MSI Level 6 Interrupt Class 0 PCI MSI level 6 interrupt is non-critical. 1 PCI MSI level 6 interrupt is critical.
12	MSI7	PCI MSI Level 7 Interrupt Class 0 PCI MSI level 7 interrupt is non-critical. 1 PCI MSI level 7 interrupt is critical.
13	MSI8	PCI MSI Level 8 Interrupt Class 0 PCI MSI level 8 interrupt is non-critical. 1 PCI MSI level 8 interrupt is critical.

UIC1_ER

UIC1 Interrupt Enable Register

Preliminary User's Manual

14	MSI9	PCI MSI Level 9 Interrupt Class 0 PCI MSI level 9 interrupt is non-critical. 1 PCI MSI level 9 interrupt is critical.
15	MSI10	PCI MSI Level 10 Interrupt Class 0 PCI MSI level 10 interrupt is non-critical. 1 PCI MSI level 10 interrupt is critical.
16	MSI11	PCI MSI Level 11 Interrupt Class 0 PCI MSI level 11 interrupt is non-critical. 1 PCI MSI level 11 interrupt is critical.
17	PPM	PPM Interrupt Class 0 PPM interrupt is non-critical. 1 PPM interrupt is critical.
18	EIR7	External IRQ 7 Interrupt Class 0 External IRQ 7 interrupt is non-critical. 1 External IRQ 7 interrupt is critical.
19	EIR8	External IRQ 8 Interrupt Class 0 External IRQ 8 interrupt is non-critical. 1 External IRQ 8 interrupt is critical.
20	EIR9	External IRQ 9 Interrupt Class 0 External IRQ 9 interrupt is non-critical. 1 External IRQ 9 interrupt is critical.
21	EIR10	External IRQ 10 Interrupt Class 0 External IRQ 10 interrupt is non-critical. 1 External IRQ 10 interrupt is critical.
22	EIR11	External IRQ 11 Interrupt Class 0 External IRQ 11 interrupt is non-critical. 1 External IRQ 11 interrupt is critical.
23	EIR12	External IRQ 12 Interrupt Class 0 External IRQ 12 interrupt is non-critical. 1 External IRQ 12 interrupt is critical.
24	SRE	Serial ROM Error Interrupt Class 0 Serial ROM error interrupt is non-critical. 1 Serial ROM error interrupt is critical.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Class 0 PCI asynchronous error interrupt is non-critical. 1 PCI asynchronous error interrupt is critical.
28	ETH0	Ethernet 0 Interrupt Class 0 Ethernet 0 interrupt is non-critical. 1 Ethernet 0 interrupt is critical.
29	EWU0	Ethernet 0 Wake-up Interrupt Class 0 Ethernet 0 wake-up interrupt is non-critical. 1 Ethernet 0 wake-up interrupt is critical.
30	ETH1	Ethernet 1 Interrupt Class 0 Ethernet 1 interrupt is non-critical. 1 Ethernet 1 interrupt is critical.
31	EWU1	Ethernet 1 Wake-up Interrupt Class 0 Ethernet 1 interrupt is non-critical. 1 Ethernet 1 interrupt is critical.

UIC1_ER

DCR 0x0D2 Read/WriteSee *UIC1 Enable Register (UIC1_ER)* on page 377.

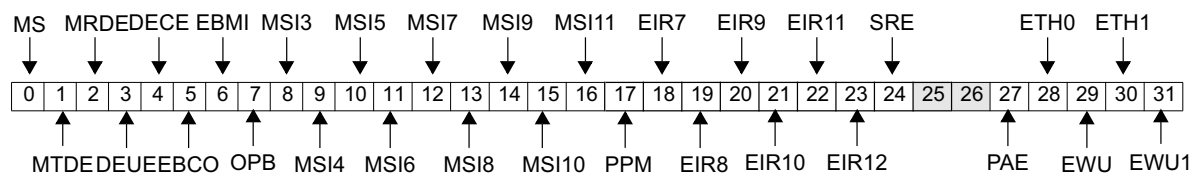


Figure 32-434. UIC1 Enable Register (UIC1_ER)

0	MS	MAL SERR Interrupt Enable 0 MAL SERR interrupt is disabled. 1 MAL SERR interrupt is enabled.
1	MTDE	MAL TXDE Interrupt Enable 0 MAL TXDE interrupt is disabled. 1 MAL TXDE interrupt is enabled.
2	MRDE	MAL RXDE Interrupt Enable 0 MAL RXDE interrupt is disabled. 1 MAL RXDE interrupt is enabled.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Enable 0 DDRSDRAM ECC uncorrectable error interrupt is disabled. 1 DDRSDRAM ECC uncorrectable error interrupt is enabled.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Enable 0 DDRSDRAM ECC correctable error interrupt is disabled. 1 DDRSDRAM ECC correctable error interrupt is enabled.
5	EBCO	EBCO Interrupt Enable 0 EBCO interrupt is disabled. 1 EBCO interrupt is enabled.
6	EBMI	EBMI Interrupt Enable 0 EBMI interrupt is disabled. 1 EBMI interrupt is enabled.
7	OPB	OPB to PLB Bridge Interrupt Enable 0 OPB to PLB bridge interrupt is disabled. 1 OPB to PLB bridge interrupt is enabled.
8	MSI3	PCI MSI Level 3 Interrupt Enable 0 PCI MSI level 3 interrupt is disabled. 1 PCI MSI level 3 interrupt is enabled.
9	MSI4	PCI MSI Level 4 Interrupt Enable 0 PCI MSI level 4 interrupt is disabled. 1 PCI MSI level 4 interrupt is enabled.
10	MSI5	PCI MSI Level 5 Interrupt Enable 0 PCI MSI level 5 interrupt is disabled. 1 PCI MSI level 5 interrupt is enabled.
11	MSI6	PCI MSI Level 6 Interrupt Enable 0 PCI MSI level 6 interrupt is disabled. 1 PCI MSI level 6 interrupt is enabled.
12	MSI7	PCI MSI Level 7 Interrupt Enable 0 PCI MSI level 7 interrupt is disabled. 1 PCI MSI level 7 interrupt is enabled.
13	MSI8	PCI MSI Level 8 Interrupt Enable 0 PCI MSI level 8 interrupt is disabled. 1 PCI MSI level 8 interrupt is enabled.
14	MSI9	PCI MSI Level 9 Interrupt Enable 0 PCI MSI level 9 interrupt is disabled. 1 PCI MSI level 9 interrupt is enabled.

UIC1_ER (cont.)

UIC1 Interrupt Enable Register

Preliminary User's Manual

15	MSI10	PCI MSI Level 10 Interrupt Enable 0 PCI MSI level 10 interrupt is disabled. 1 PCI MSI level 10 interrupt is enabled.
16	MSI11	PCI MSI Level 11 Interrupt Enable 0 PCI MSI level 11 interrupt is disabled. 1 PCI MSI level 11 interrupt is enabled.
17	PPM	PPM Interrupt Enable 0 PPM interrupt is disabled. 1 PPM interrupt is enabled.
18	EIR7	External IRQ 7 Interrupt Enable 0 External IRQ 7 interrupt is disabled. 1 External IRQ 7 interrupt is enabled.
19	EIR8	External IRQ 8 Interrupt Enable 0 External IRQ 8 interrupt is disabled. 1 External IRQ 8 interrupt is enabled.
20	EIR9	External IRQ 9 Interrupt Enable 0 External IRQ 9 interrupt is disabled. 1 External IRQ 9 interrupt is enabled.
21	EIR10	External IRQ 10 Interrupt Enable 0 External IRQ 10 interrupt is disabled. 1 External IRQ 10 interrupt is enabled.
22	EIR11	External IRQ 11 Interrupt Enable 0 External IRQ 11 interrupt is disabled. 1 External IRQ 11 interrupt is enabled.
23	EIR12	External IRQ 12 Interrupt Enable 0 External IRQ 12 interrupt is disabled. 1 External IRQ 12 interrupt is enabled.
24	SRE	Serial ROM Error Interrupt Enable 0 Serial ROM error interrupt is disabled. 1 Serial ROM error interrupt is enabled.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Enable 0 PCI asynchronous error interrupt is disabled. 1 PCI asynchronous error interrupt is enabled.
28	ETH0	Ethernet 0 Interrupt Enable 0 Ethernet 0 interrupt is disabled. 1 Ethernet 0 interrupt is enabled.
29	EWU0	Ethernet 0 Wake-up Interrupt Enable 0 Ethernet 0 wake-up interrupt is disabled. 1 Ethernet 0 wake-up interrupt is enabled.
30	ETH1	Ethernet 1 Interrupt Enable 0 Ethernet 1 interrupt is disabled. 1 Ethernet 1 interrupt is enabled.
31	EWU1	Ethernet 1 Wake-up Interrupt Enable 0 Ethernet 1 wake-up interrupt is disabled. 1 Ethernet 1 wake-up interrupt is enabled.

DCR 0x0D6 Read-Only

See *UIC0 Masked Status Register (UIC0_MSR)* on page 403.

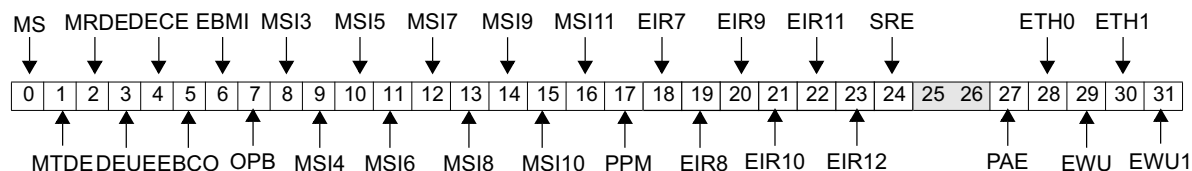


Figure 32-435. UIC1 Masked Status Register (UIC1_MSR)

0	MS	MAL SERR Masked Interrupt Status 0 MAL SERR masked interrupt has not occurred. 1 MAL SERR masked interrupt occurred.
1	MTDE	MAL TXDE Masked Interrupt Status 0 MAL TXDE masked interrupt has not occurred. 1 MAL TXDE masked interrupt occurred.
2	MRDE	MAL RXDE Masked Interrupt Status 0 MAL RXDE masked interrupt has not occurred. 1 MAL RXDE masked interrupt occurred.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Masked Interrupt Status 0 DDRSDRAM ECC uncorrectable error masked interrupt has not occurred. 1 DDRSDRAM ECC uncorrectable error masked interrupt occurred.
4	DECE	DDRSDRAM ECC Correctable Error Masked Interrupt Status 0 DDRSDRAM ECC correctable error masked interrupt has not occurred. 1 DDRSDRAM ECC correctable error masked interrupt occurred.
5	EBCO	EBCO Masked Interrupt Status 0 EBCO interrupt has not occurred. 1 EBCO interrupt occurred.
6	EBMI	EBMI Masked Interrupt Status 0 EBMI masked interrupt has not occurred. 1 EBMI masked interrupt occurred.
7	OPB	OPB to PLB Bridge Masked Interrupt Status 0 OPB to PLB bridge masked interrupt has not occurred. 1 OPB to PLB bridge masked interrupt occurred.
8	MSI3	PCI MSI Level 3 Masked Interrupt Status 0 PCI MSI level 3 masked interrupt has not occurred. 1 PCI MSI level 3 masked interrupt occurred.
9	MSI4	PCI MSI Level 4 Masked Interrupt Status 0 PCI MSI level 4 masked interrupt has not occurred. 1 PCI MSI level 4 masked interrupt occurred.
10	MSI5	PCI MSI Level 5 Masked Interrupt Status 0 PCI MSI level 5 masked interrupt has not occurred. 1 PCI MSI level 5 masked interrupt occurred.
11	MSI6	PCI MSI Level 6 Masked Interrupt Status 0 PCI MSI level 6 masked interrupt has not occurred. 1 PCI MSI level 6 masked interrupt occurred.
12	MSI7	PCI MSI Level 7 Masked Interrupt Status 0 PCI MSI level 7 masked interrupt has not occurred. 1 PCI MSI level 7 masked interrupt input occurred.

13	MSI8	PCI MSI Level 8 Masked Interrupt Status 0 PCI MSI level 8 masked interrupt has not occurred. 1 PCI MSI level 8 masked interrupt occurred.
14	MSI9	PCI MSI Level 9 Masked Interrupt Status 0 PCI MSI level 9 masked interrupt has not occurred. 1 PCI MSI level 9 masked interrupt occurred.
15	MSI10	PCI MSI Level 10 Masked Interrupt Status 0 PCI MSI level 10 masked interrupt has not occurred. 1 PCI MSI level 10 masked interrupt occurred.
16	MSI11	PCI MSI Level 11 Masked Interrupt Status 0 PCI MSI level 11 masked interrupt has not occurred. 1 PCI MSI level 11 masked interrupt occurred.
17	PPM	PPM Masked Interrupt Status 0 PPM masked interrupt has not occurred. 1 PPM masked interrupt occurred.
18	EIR7	External IRQ 7 Masked Interrupt Status 0 External IRQ 7 interrupt has not occurred. 1 External IRQ 7 interrupt occurred.
19	EIR8	External IRQ 8 Masked Interrupt Status 0 External IRQ 8 interrupt has not occurred. 1 External IRQ 8 interrupt occurred.
20	EIR9	External IRQ 9 Masked Interrupt Status 0 External IRQ 9 interrupt has not occurred. 1 External IRQ 9 interrupt occurred.
21	EIR10	External IRQ 10 Masked Interrupt Status 0 External IRQ 10 interrupt has not occurred. 1 External IRQ 10 interrupt occurred.
22	EIR11	External IRQ 11 Masked Interrupt Status 0 External IRQ 11 interrupt has not occurred. 1 External IRQ 11 interrupt occurred.
23	EIR12	External IRQ 12 Masked Interrupt Status 0 External IRQ 12 interrupt has not occurred. 1 External IRQ 12 interrupt occurred.
24	SRE	Serial ROM Error Interrupt Status 0 Serial ROM error interrupt has not occurred. 1 Serial ROM error interrupt occurred.
25:26		Reserved
27	PAE	PCI Asynchronous Error Masked Interrupt Status 0 PCI asynchronous error interrupt has not occurred. 1 PCI asynchronous error interrupt occurred.
28	ETH0	Ethernet 0 Masked Interrupt Status 0 Ethernet 0 interrupt has not occurred. 1 Ethernet 0 interrupt occurred.
29	EWU0	Ethernet 0 Wake-up Masked Interrupt Status 0 Ethernet 0 wake-up interrupt has not occurred. 1 Ethernet 0 wake-up interrupt occurred.
30	ETH1	Ethernet 1 Masked Interrupt Status 0 Ethernet 1 interrupt has not occurred. 1 Ethernet 1 interrupt occurred.
31	EWU1	Ethernet 1 Wake-up Masked Interrupt Status 0 Ethernet 1 wake-up interrupt has not occurred. 1 Ethernet 1 wake-up interrupt occurred.

DCR 0x0D4 Read/Write

See *UIC1 Polarity Register (UIC1_PR)* on page 391.

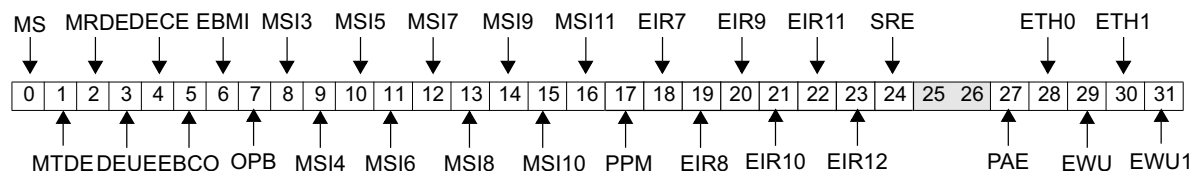


Figure 32-436. *UIC1 Polarity Register (UIC1_PR)*

0	MS	MAL SERR Interrupt Polarity 0 MAL SERR interrupt has negative polarity. 1 MAL SERR interrupt has positive polarity.
1	MTDE	MAL TXDE Interrupt Polarity 0 MAL TXDE interrupt has negative polarity. 1 MAL TXDE interrupt has positive polarity.
2	MRDE	MAL RXDE Interrupt Polarity 0 MAL RXDE interrupt has negative polarity. 1 MAL RXDE interrupt has positive polarity.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Polarity 0 DDRSDRAM ECC uncorrectable error interrupt has negative polarity. 1 DDRSDRAM ECC uncorrectable error interrupt has positive polarity.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Polarity 0 DDRSDRAM ECC correctable error interrupt has negative polarity. 1 DDRSDRAM ECC correctable error interrupt has positive polarity.
5	EBCO	EBCO Interrupt Polarity 0 EBCO interrupt has negative polarity. 1 EBCO interrupt has positive polarity.
6	EBMI	EBMI Interrupt Polarity 0 EBMI interrupt has negative polarity. 1 EBMI interrupt has positive polarity.
7	OPB	OPB to PLB Bridge Interrupt Polarity 0 OPB to PLB bridge interrupt has negative polarity. 1 OPB to PLB bridge interrupt has positive polarity.
8	MSI3	PCI MSI Level 3 Interrupt Polarity 0 PCI MSI level 3 interrupt has negative polarity. 1 PCI MSI level 3 interrupt has positive polarity.
9	MSI4	PCI MSI Level 4 Interrupt Polarity 0 PCI MSI level 4 interrupt has negative polarity. 1 PCI MSI level 4 interrupt has positive polarity.
10	MSI5	PCI MSI Level 5 Interrupt Polarity 0 PCI MSI level 5 interrupt has negative polarity. 1 PCI MSI level 5 interrupt has positive polarity.
11	MSI6	PCI MSI Level 6 Interrupt Polarity 0 PCI MSI level 6 interrupt has negative polarity. 1 PCI MSI level 6 interrupt has positive polarity.
12	MSI7	PCI MSI Level 7 Interrupt Polarity 0 PCI MSI level 7 interrupt has negative polarity. 1 PCI MSI level 7 interrupt has positive polarity.
13	MSI8	PCI MSI Level 8 Interrupt Polarity 0 PCI MSI level 8 interrupt has negative polarity. 1 PCI MSI level 8 interrupt has positive polarity.

14	MSI9	PCI MSI Level 9 Interrupt Polarity 0 PCI MSI level 9 interrupt has negative polarity. 1 PCI MSI level 9 interrupt has positive polarity.
15	MSI10	PCI MSI Level 10 Interrupt Polarity 0 PCI MSI level 10 interrupt has negative polarity. 1 PCI MSI level 10 interrupt has positive polarity.
16	MSI11	PCI MSI Level 11 Interrupt Polarity 0 PCI MSI level 11 interrupt has negative polarity. 1 PCI MSI level 11 interrupt has positive polarity.
17	PPM	PPM Interrupt Polarity 0 PPM interrupt has negative polarity. 1 PPM interrupt has positive polarity.
18	EIR7	External IRQ 7 Interrupt Polarity 0 External IRQ 7 interrupt has negative polarity. 1 External IRQ 7 interrupt has positive polarity.
19	EIR8	External IRQ 8 Interrupt Polarity 0 External IRQ 8 interrupt has negative polarity. 1 External IRQ 8 interrupt has positive polarity.
20	EIR9	External IRQ 9 Interrupt Polarity 0 External IRQ 9 interrupt has negative polarity. 1 External IRQ 9 interrupt has positive polarity.
21	EIR10	External IRQ 10 Interrupt Polarity 0 External IRQ 10 interrupt has negative polarity. 1 External IRQ 10 interrupt has positive polarity.
22	EIR11	External IRQ 11 Interrupt Polarity 0 External IRQ 11 interrupt has negative polarity. 1 External IRQ 11 interrupt has positive polarity.
23	EIR12	External IRQ 12 Interrupt Polarity 0 External IRQ 12 interrupt has negative polarity. 1 External IRQ 12 interrupt has positive polarity.
24	SRE	Serial ROM Error Interrupt Polarity 0 Serial ROM error interrupt has negative polarity. 1 Serial ROM error interrupt has positive polarity.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Polarity 0 PCI asynchronous error interrupt has negative polarity. 1 PCI asynchronous error interrupt has positive polarity.
28	ETH0	Ethernet 0 Interrupt Polarity 0 Ethernet 0 interrupt has negative polarity. 1 Ethernet 0 interrupt has positive polarity.
29	EWU0	Ethernet 0 Wake-up Interrupt Polarity 0 Ethernet 0 wake-up interrupt has negative polarity. 1 Ethernet 0 wake-up interrupt has positive polarity.
30	ETH1	Ethernet 1 Interrupt Polarity 0 Ethernet 1 interrupt has negative polarity. 1 Ethernet 1 interrupt has positive polarity.
31	EWU1	Ethernet 1 Wake-up Interrupt Polarity 0 Ethernet 1 interrupt has negative polarity. 1 Ethernet 1 interrupt has positive polarity.

DCR 0x0D0 Read/Clear

See *UIC1 Status Register (UIC1_SR)* on page 370.

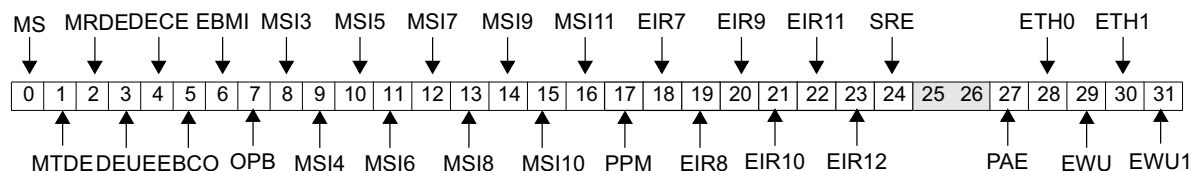


Figure 32-437. *UIC1 Status Register (UIC1_SR)*

0	MS	MAL SERR Interrupt Status 0 MAL SERR interrupt has not occurred. 1 MAL SERR interrupt occurred.
1	MTDE	MAL TXDE Interrupt Status 0 MAL TXDE interrupt has not occurred. 1 MAL TXDE interrupt occurred.
2	MRDE	MAL RXDE Interrupt Status 0 MAL RXDE interrupt has not occurred. 1 MAL RXDE interrupt occurred.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Status 0 DDRSDRAM ECC uncorrectable error interrupt has not occurred. 1 DDRSDRAM ECC uncorrectable error interrupt occurred.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Status 0 DDRSDRAM ECC correctable error interrupt has not occurred. 1 DDRSDRAM ECC correctable error interrupt occurred.
5	EBCO	EBCO Interrupt Status 0 EBCO interrupt has not occurred. 1 EBCO interrupt occurred.
6	EBMI	EBMI Interrupt Status 0 EBMI interrupt has not occurred. 1 EBMI interrupt occurred.
7	OPB	OPB to PLB Bridge Interrupt Status 0 OPB to PLB bridge interrupt has not occurred. 1 OPB to PLB bridge interrupt occurred.
8	MSI3	PCI MSI Level 3 Interrupt Status 0 PCI MSI level 3 interrupt has not occurred. 1 PCI MSI level 3 interrupt occurred.
9	MSI4	PCI MSI Level 4 Interrupt Status 0 PCI MSI level 4 interrupt has not occurred. 1 PCI MSI level 4 interrupt occurred.
10	MSI5	PCI MSI Level 5 Interrupt Status 0 PCI MSI level 5 interrupt has not occurred. 1 PCI MSI level 5 interrupt occurred.
11	MSI6	PCI MSI Level 6 Interrupt Status 0 PCI MSI level 6 interrupt has not occurred. 1 PCI MSI level 6 interrupt occurred.
12	MSI7	PCI MSI Level 7 Interrupt Status 0 PCI MSI level 7 interrupt has not occurred. 1 PCI MSI level 7 interrupt occurred.
13	MSI8	PCI MSI Level 8 Interrupt Status 0 PCI MSI level 8 interrupt has not occurred. 1 PCI MSI level 8 interrupt occurred.

14	MSI9	PCI MSI Level 9 Interrupt Status 0 PCI MSI level 9 interrupt has not occurred. 1 PCI MSI level 9 interrupt occurred.
15	MSI10	PCI MSI Level 10 Interrupt Status 0 PCI MSI level 10 interrupt has not occurred. 1 PCI MSI level 10 interrupt occurred.
16	MSI11	PCI MSI Level 11 Interrupt Status 0 PCI MSI level 11 interrupt has not occurred. 1 PCI MSI level 11 interrupt occurred.
17	PPM	PPM Interrupt Status 0 PPM interrupt has not occurred. 1 PPM interrupt occurred.
18	EIR7	External IRQ 7 Interrupt Status 0 External IRQ 7 interrupt has not occurred. 1 External IRQ 7 interrupt occurred.
19	EIR8	External IRQ 8 Interrupt Status 0 External IRQ 8 interrupt has not occurred. 1 External IRQ 8 interrupt occurred.
20	EIR9	External IRQ 9 Interrupt Status 0 External IRQ 9 interrupt has not occurred. 1 External IRQ 9 interrupt occurred.
21	EIR10	External IRQ 10 Interrupt Status 0 External IRQ 10 interrupt has not occurred. 1 External IRQ 10 interrupt occurred.
22	EIR11	External IRQ 11 Interrupt Status 0 External IRQ 11 interrupt has not occurred. 1 External IRQ 11 interrupt occurred.
23	EIR12	External IRQ 12 Interrupt Status 0 External IRQ 12 interrupt has not occurred. 1 External IRQ 12 interrupt occurred.
24	SRE	Serial ROM Error Interrupt Status 0 Serial ROM error interrupt has not occurred. 1 Serial ROM error interrupt occurred.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Status 0 PCI asynchronous error interrupt has not occurred. 1 PCI asynchronous error interrupt occurred.
28	ETH0	Ethernet 0 Interrupt Status 0 Ethernet 0 interrupt has not occurred. 1 Ethernet 0 interrupt occurred.
29	EWU0	Ethernet 0 Wake-up Interrupt Status 0 Ethernet 0 wake-up interrupt has not occurred. 1 Ethernet 0 wake-up interrupt occurred.
30	ETH1	Ethernet 1 Interrupt Status 0 Ethernet 1 interrupt has not occurred. 1 Ethernet 1 interrupt occurred.
31	EWU1	Ethernet 1 Wake-up Interrupt Status 0 Ethernet 1 wake-up interrupt has not occurred. 1 Ethernet 1 wake-up interrupt occurred.

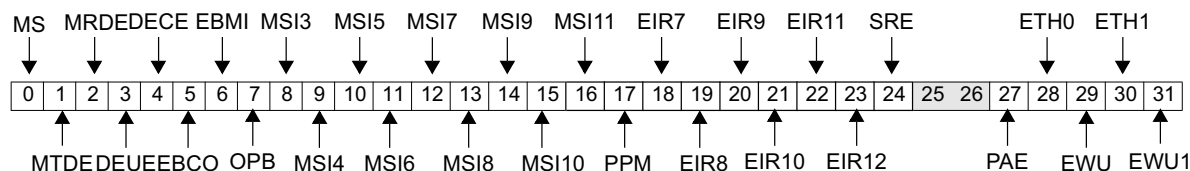
DCR 0x0D5 Read/WriteSee *UIC1 Trigger Register (UIC1_TR)* on page 398.

Figure 32-438. UIC1 Trigger Register (UIC1_TR)

0	MS	MAL SERR Interrupt Trigger 0 MAL SERR interrupt is level-sensitive. 1 MAL SERR interrupt is edge-sensitive.
1	MTDE	MAL TXDE Interrupt Trigger 0 MAL TXDE interrupt is level-sensitive. 1 MAL TXDE interrupt is edge-sensitive.
2	MRDE	MAL RXDE Interrupt Trigger 0 MAL RXDE interrupt is level-sensitive. 1 MAL RXDE interrupt is edge-sensitive.
3	DEUE	DDRSDRAM ECC Uncorrectable Error Interrupt Trigger 0 DDRSDRAM ECC uncorrectable error interrupt is level-sensitive. 1 DDRSDRAM ECC uncorrectable error interrupt is edge-sensitive.
4	DECE	DDRSDRAM ECC Correctable Error Interrupt Trigger 0 DDRSDRAM ECC correctable error interrupt is level-sensitive. 1 DDRSDRAM ECC correctable error interrupt is edge-sensitive.
5	EBCO	EBCO Interrupt Trigger 0 EBCO interrupt is level-sensitive. 1 EBCO interrupt is edge-sensitive.
6	EBMI	EBMI Interrupt Trigger 0 EBMI interrupt is level-sensitive. 1 EBMI interrupt is edge-sensitive.
7	OPB	OPB to PLB Bridge Interrupt Trigger 0 OPB to PLB bridge interrupt is level-sensitive. 1 OPB to PLB bridge interrupt is edge-sensitive.
8	MSI3	PCI MSI Level 3 Interrupt Trigger 0 PCI MSI level 3 interrupt is level-sensitive. 1 PCI MSI level 3 interrupt is edge-sensitive.
9	MSI4	PCI MSI Level 4 Interrupt Trigger 0 PCI MSI level 4 interrupt is level-sensitive. 1 PCI MSI level 4 interrupt is edge-sensitive.
10	MSI5	PCI MSI Level 5 Interrupt Trigger 0 PCI MSI level 5 interrupt is level-sensitive. 1 PCI MSI level 5 interrupt is edge-sensitive.
11	MSI6	PCI MSI Level 6 Interrupt Trigger 0 PCI MSI level 6 interrupt is level-sensitive. 1 PCI MSI level 6 interrupt is edge-sensitive.
12	MSI7	PCI MSI Level 7 Interrupt Trigger 0 PCI MSI level 7 interrupt is level-sensitive. 1 PCI MSI level 7 interrupt is edge-sensitive.
13	MSI8	PCI MSI Level 8 Interrupt Trigger 0 PCI MSI level 8 interrupt is level-sensitive. 1 PCI MSI level 8 interrupt is edge-sensitive.

14	MSI9	PCI MSI Level 9 Interrupt Trigger 0 PCI MSI level 9 interrupt is level-sensitive. 1 PCI MSI level 9 interrupt is edge-sensitive.
15	MSI10	PCI MSI Level 10 Interrupt Trigger 0 PCI MSI level 10 interrupt is level-sensitive. 1 PCI MSI level 10 interrupt is edge-sensitive.
16	MSI11	PCI MSI Level 11 Interrupt Trigger 0 PCI MSI level 11 interrupt is level-sensitive. 1 PCI MSI level 11 interrupt is edge-sensitive.
17	PPM	PPM Interrupt Trigger 0 PPM interrupt is level-sensitive. 1 PPM interrupt is edge-sensitive.
18	EIR7	External IRQ 7 Interrupt Trigger 0 External IRQ 7 interrupt is level-sensitive. 1 External IRQ 7 interrupt is edge-sensitive.
19	EIR8	External IRQ 8 Interrupt Trigger 0 External IRQ 8 interrupt is level-sensitive. 1 External IRQ 8 interrupt is edge-sensitive.
20	EIR9	External IRQ 9 Interrupt Trigger 0 External IRQ 9 interrupt is level-sensitive. 1 External IRQ 9 interrupt is edge-sensitive.
21	EIR10	External IRQ 10 Interrupt Trigger 0 External IRQ 10 interrupt is level-sensitive. 1 External IRQ 10 interrupt is edge-sensitive.
22	EIR11	External IRQ 11 Interrupt Trigger 0 External IRQ 11 interrupt is level-sensitive. 1 External IRQ 11 interrupt is edge-sensitive.
23	EIR12	External IRQ 12 Interrupt Trigger 0 External IRQ 12 interrupt is level-sensitive. 1 External IRQ 12 interrupt is edge-sensitive.
24	SRE	Serial ROM Error Interrupt Trigger 0 Serial ROM error interrupt is level-sensitive. 1 Serial ROM error interrupt is edge-sensitive.
25:26		Reserved
27	PAE	PCI Asynchronous Error Interrupt Trigger 0 PCI asynchronous error interrupt is level-sensitive. 1 PCI asynchronous error interrupt is edge-sensitive.
28	ETH0	Ethernet 0 Interrupt Trigger 0 Ethernet 0 interrupt is level-sensitive. 1 Ethernet 0 interrupt is edge-sensitive.
29	EWU0	Ethernet 0 Wake-up Interrupt Trigger 0 Ethernet 0 wake-up interrupt is level-sensitive. 1 Ethernet 0 wake-up interrupt is edge-sensitive.
30	ETH1	Ethernet 1 Interrupt Trigger 0 Ethernet 1 interrupt is level-sensitive. 1 Ethernet 1 interrupt is edge-sensitive.
31	EWU1	Ethernet 1 Wake-up Interrupt Trigger 0 Ethernet 1 interrupt is level-sensitive. 1 Ethernet 1 interrupt is edge-sensitive.

DCR 0x0D8 Write-Only

See *UIC0 Vector Configuration Register (UIC0_VCR)* on page 409.

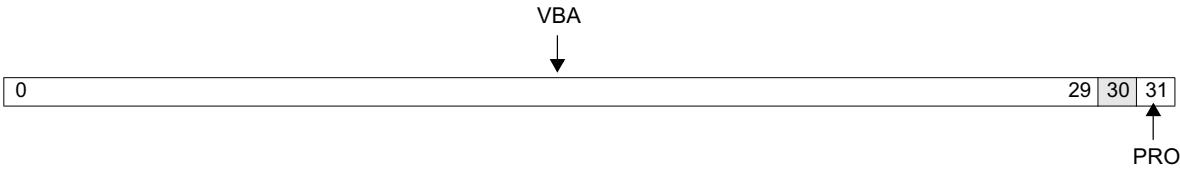


Figure 32-439. UIC1 Vector Configuration Register (UIC1_VCR)

0:29	VBA	Vector Base Address
30		Reserved
31	PRO	Priority Ordering 0 UIC1_SR[31] is the highest priority interrupt. 1 UIC1_SR[0] is the highest priority interrupt. Note: Vector generation is not performed for non-critical interrupts.

DCR 0x0D7 Read-Only

See UIC0 Vector Register (UIC0_VR) on page 413.

0	31
---	----

Figure 32-440. UIC1 Vector Register (UIC1_VR)

0:31	Interrupt Vector
------	------------------

UIC1_VR

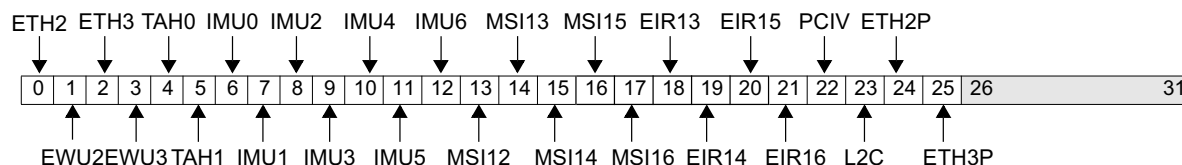
DCR 0x213 R/WSee *UIC2 Critical Register (UIC2_CR)* on page 386.

Figure 32-441. UIC2 Critical Register (UIC2_CR)

0	ETH2	Ethernet 2 Interrupt Class 0 Ethernet 2 interrupt is non-critical. 1 Ethernet 2 interrupt is critical.
1	EWU2	Ethernet 2 Wake-up Interrupt Class 0 Ethernet 2 Wake-up interrupt is non-critical. 1 Ethernet 2 Wake-up interrupt is critical.
2	ETH3	Ethernet 3 Interrupt Class 0 Ethernet 3 interrupt is non-critical. 1 Ethernet 3 interrupt is critical.
3	EWU3	Ethernet 3 Wake-up Interrupt Class 0 Ethernet 3 Wake-up interrupt is non-critical. 1 Ethernet 3 Wake-up interrupt is critical.
4	TAH0	Tah 0 Interrupt Class 0 Tah 0 interrupt is non-critical. 1 Tah 0 interrupt is critical.
5	TAH1	Tah 1 Interrupt Class 0 Tah 1 interrupt is non-critical. 1 Tah 1 interrupt is critical.
6	IMU0	IMU Level 0 Interrupt Class 0 IMU Level 0 interrupt is non-critical. 1 IMU Level 0 interrupt is critical.
7	IMU1	IMU Level 1 Interrupt Class 0 IMU Level 1 interrupt is non-critical. 1 IMU Level 1 interrupt is critical.
8	IMU2	IMU Level 2 Interrupt Class 0 IMU Level 2 interrupt is non-critical. 1 IMU Level 2 interrupt is critical.
9	IMU3	IMU Level 3 Interrupt Class 0 IMU Level 3 interrupt is non-critical. 1 IMU Level 3 interrupt is critical.
10	IMU4	IMU Level 4 Interrupt Class 0 IMU Level 4 interrupt is non-critical. 1 IMU Level 4 interrupt is critical.
11	IMU5	IMU Level 5 Interrupt Class 0 IMU Level 5 interrupt is non-critical. 1 IMU Level 5 interrupt is critical.
12	IMU6	IMU Level 6 Interrupt Class 0 IMU Level 6 interrupt is non-critical. 1 IMU Level 6 interrupt is critical.
13	MSI12	PCI MSI Level 12 Interrupt Class 0 PCI MSI level 12 interrupt is non-critical. 1 PCI MSI level 12 interrupt is critical.

14	MSI13	PCI MSI Level 13 Interrupt Class 0 PCI MSI level 13 interrupt is non-critical. 1 PCI MSI level 13 interrupt is critical.
15	MSI14	PCI MSI Level 14 Interrupt Class 0 PCI MSI level14 interrupt is non-critical. 1 PCI MSI level 14 interrupt is critical.
16	MSI15	PCI MSI Level 15 Interrupt Class 0 PCI MSI level 15 interrupt is non-critical. 1 PCI MSI level 15 interrupt is critical.
17	EIR13	External IRQ 13 Interrupt Class 0 External IRQ 13 interrupt is non-critical. 1 External IRQ 13 interrupt is critical.
18	EIR14	External IRQ 14 Interrupt Class 0 External IRQ 14 interrupt is non-critical. 1 External IRQ 14 interrupt is critical.
19	EIR15	External IRQ 15 Interrupt Class 0 External IRQ 15 interrupt is non-critical. 1 External IRQ 15 interrupt is critical.
20	EIR16	External IRQ 16 Interrupt Class 0 External IRQ 16 interrupt is non-critical. 1 External IRQ 16 interrupt is critical.
21	EIR17	External IRQ 17 Interrupt Class 0 External IRQ 17 interrupt is non-critical. 1 External IRQ 17 interrupt is critical.
22	PCIV	PCI VPD Interrupt Class 0 PCI VPD interrupt is non-critical. 1 PCI VPD interrupt is critical.
23	L2C	L2 Cache Interrupt Class 0 L2 cache interrupt is non-critical. 1 L2 cache interrupt is critical.
24	ETH2P	EMAC 2 PCS Interrupt Class 0 EMAC 2 PCS interrupt is non-critical. 1 EMAC 2 PCS interrupt is critical.
25	ETH3P	EMAC 3 PCS Interrupt Class 0 EMAC 3 PCS interrupt is non-critical. 1 EMAC 3 PCS interrupt is critical.
26:31		Reserved

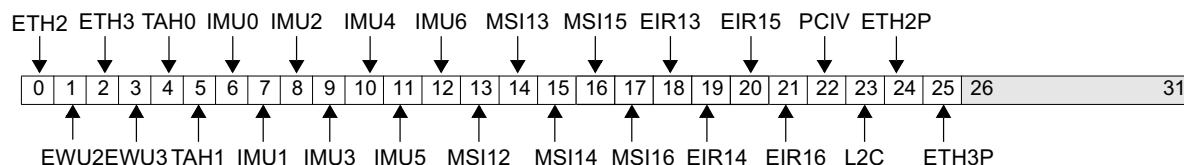
DCR 0x212 R/WSee *UIC2 Enable Register (UIC2_ER)* on page 379.

Figure 32-442. UIC2 Enable Register (UIC2_ER)

0	ETH2	Ethernet 2 Interrupt Enable 0 Ethernet 2 interrupt is disabled. 1 Ethernet 2 interrupt is enabled.
1	EWU2	Ethernet 2 Wake-up Interrupt Enable 0 Ethernet 2 Wake-up interrupt is disabled. 1 Ethernet 2 Wake-up interrupt is enabled.
2	ETH3	Ethernet 3 Interrupt Enable 0 Ethernet 3 interrupt is disabled. 1 Ethernet 3 interrupt is enabled.
3	EWU3	Ethernet 3 Wake-up Interrupt Enable 0 Ethernet 3 Wake-up interrupt is disabled. 1 Ethernet 3 Wake-up interrupt is enabled.
4	TAH0	Tah 0 Interrupt Enable 0 Tah 0 interrupt is disabled. 1 Tah 0 interrupt is enabled.
5	TAH1	Tah 1 Interrupt Enable 0 Tah 1 interrupt is disabled. 1 Tah 1 interrupt is enabled.
6	IMU0	IMU Level 0 Interrupt Enable 0 IMU Level 0 interrupt is disabled. 1 IMU Level 0 interrupt is enabled.
7	IMU1	IMU Level 1 Interrupt Enable 0 IMU Level 1 interrupt is disabled. 1 IMU Level 1 interrupt is enabled.
8	IMU2	IMU Level 2 Interrupt Enable 0 IMU Level 2 interrupt is disabled. 1 IMU Level 2 interrupt is enabled.
9	IMU3	IMU Level 3 Interrupt Enable 0 IMU Level 3 interrupt is disabled. 1 IMU Level 3 interrupt is enabled.
10	IMU4	IMU Level 4 Interrupt Enable 0 IMU Level 4 interrupt is disabled. 1 IMU Level 4 interrupt is enabled.
11	IMU5	IMU Level 5 Interrupt Enable 0 IMU Level 5 interrupt is disabled. 1 IMU Level 5 interrupt is enabled.
12	IMU6	IMU Level 6 Interrupt Enable 0 IMU Level 6 interrupt is disabled. 1 IMU Level 6 interrupt is enabled.
13	MSI12	PCI MSI Level 12 Interrupt Enable 0 PCI MSI level 12 interrupt is disabled. 1 PCI MSI level 12 interrupt is enabled.

13	MSI13	PCI MSI Level 13 Interrupt Enable 0 PCI MSI level 13 interrupt is disabled. 1 PCI MSI level 13 interrupt is enabled.
14	MSI14	PCI MSI Level 14 Interrupt Enable 0 PCI MSI level 14 interrupt is disabled. 1 PCI MSI level 14 interrupt is enabled.
15	MSI15	PCI MSI Level 15 Interrupt Enable 0 PCI MSI level 15 interrupt is disabled. 1 PCI MSI level 15 interrupt is enabled.
17	EIR13	External IRQ 13 Interrupt Enable 0 External IRQ 13 interrupt is disabled. 1 External IRQ 13 interrupt is enabled.
18	EIR14	External IRQ 14 Interrupt Enable 0 External IRQ 14 interrupt is disabled. 1 External IRQ 14 interrupt is enabled.
19	EIR15	External IRQ15 Interrupt Enable 0 External IRQ 15 interrupt is disabled. 1 External IRQ15 interrupt is enabled.
20	EIR16	External IRQ 16 Interrupt Enable 0 External IRQ 16 interrupt is disabled. 1 External IRQ 16 interrupt is enabled.
21	EIR17	External IRQ 17 Interrupt Enable 0 External IRQ 17 interrupt is disabled. 1 External IRQ 17 interrupt is enabled.
22	PCIV	PCI VPD Interrupt Enable 0 PCI VPD interrupt is enabled. 1 PCI VPD interrupt is disabled.
23	L2C	L2 Cache Interrupt Enable 0 L2 cache interrupt is enabled. 1 L2 cache interrupt is disabled.
24	ETH2P	EMAC 2 PCS Interrupt Enable 0 EMAC 2 PCS interrupt is enabled. 1 EMAC 2 PCS interrupt is disabled.
25	ETH3P	EMAC 3 PCS Interrupt Enable 0 EMAC 3 PCS interrupt is enabled. 1 EMAC 3 PCS interrupt is disabled.
26:31		Reserved

DCR 0x216 R/only

See *UIC2 Masked Status Register (UIC2_MSR)* on page 407.

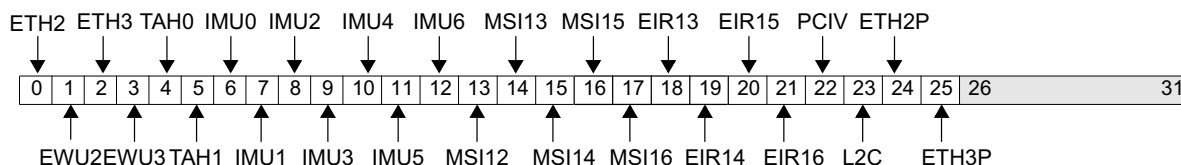


Figure 32-443. UIC2 Masked Status Register (UIC2_MSR)

0	ETH2	Ethernet 2 Masked Interrupt Status 0 Ethernet 2 masked interrupt has not occurred. 1 Ethernet 2 masked interrupt occurred.
1	EWU2	Ethernet 2 Wake-up Masked Interrupt Status 0 Ethernet 2 wake-up masked interrupt has not occurred. 1 Ethernet 2 wake-up masked interrupt occurred.
2	ETH3	Ethernet 3 Masked Interrupt Status 0 Ethernet 3 masked interrupt has not occurred. 1 Ethernet 3 masked interrupt occurred.
3	EWU3	Ethernet 3 Wake-up Masked Interrupt Status 0 Ethernet 3 wake-up masked interrupt has not occurred. 1 Ethernet 3 wake-up masked interrupt occurred.
4	TAH0	Tah 0 Masked Interrupt Status 0 Tah 0 masked interrupt has not occurred. 1 Tah 0 masked interrupt occurred.
5	TAH1	Tah 1 Masked Interrupt Status 0 Tah 1 interrupt has not occurred. 1 Tah 1 interrupt occurred.
6	IMU0	IMU Level 0 Masked Interrupt Status 0 IMU level 0 masked interrupt has not occurred. 1 IMU level 0 masked interrupt occurred.
7	IMU1	IMU Level 1 Masked Interrupt Status 0 IMU level 1 masked interrupt has not occurred. 1 IMU level 1 masked interrupt occurred.
8	IMU2	IMU Level 2 Masked Interrupt Status 0 IMU level 2 masked interrupt has not occurred. 1 IMU level 2 masked interrupt occurred.
9	IMU3	IMU Level 3 Masked Interrupt Status 0 IMU level 3 masked interrupt has not occurred. 1 IMU level 3 masked interrupt occurred.
10	IMU4	IMU Level 4 Masked Interrupt Status 0 IMU level 4 masked interrupt has not occurred. 1 IMU level 4 masked interrupt occurred.
11	IMU5	IMU Level 5 Masked Interrupt Status 0 IMU level 5 masked interrupt has not occurred. 1 IMU level 5 masked interrupt occurred.
12	IMU6	IMU Level 6 Masked Interrupt Status 0 IMU level 6 masked interrupt has not occurred. 1 IMU level 6 masked interrupt occurred.
13	MSI12	PCI MSI Level 12 Masked Interrupt Status 0 PCI MSI level 12 masked interrupt has not occurred. 1 PCI MSI level 12 masked interrupt input occurred.

14	MSI13	PCI MSI Level 13 Masked Interrupt Status 0 PCI MSI level 13 masked interrupt has not occurred. 1 PCI MSI level 13 masked interrupt occurred.
15	MSI14	PCI MSI Level 14 Masked Interrupt Status 0 PCI MSI level 14 masked interrupt has not occurred. 1 PCI MSI level 14 masked interrupt occurred.
16	MSI15	PCI MSI Level 15 Masked Interrupt Status 0 PCI MSI level 15 masked interrupt has not occurred. 1 PCI MSI level 15 masked interrupt occurred.
17	EIR13	External IRQ 13 Masked Interrupt Status 0 External IRQ 13 masked interrupt has not occurred. 1 External IRQ 13 masked interrupt occurred.
18	EIR14	External IRQ 14 Masked Interrupt Status 0 External IRQ 14 masked interrupt has not occurred. 1 External IRQ 14 masked interrupt occurred.
19	EIR15	External IRQ 15 Masked Interrupt Status 0 External IRQ15 interrupt has not occurred. 1 External IRQ15 interrupt occurred.
20	EIR16	External IRQ 16 Masked Interrupt Status 0 External IRQ 16 interrupt has not occurred. 1 External IRQ 16 interrupt occurred.
21	EIR17	External IRQ 17 Masked Interrupt Status 0 External IRQ 17 masked interrupt has not occurred. 1 External IRQ 17 masked interrupt occurred.
22	PCIV	PCI VPD Masked Interrupt Status 0 PCI VPD masked interrupt has not occurred. 1 PCI VPD masked interrupt occurred.
23	L2C	L2 Cache Masked Interrupt Status 0 L2 cache masked interrupt has not occurred. 1 L2 cache masked interrupt has occurred.
24	ETH2P	EMAC 2 PCS Masked Interrupt Status 0 EMAC 2 PCS masked interrupt has not occurred. 1 EMAC 2 PCS masked interrupt occurred.
25	ETH3P	EMAC 3 PCS Masked Interrupt Status 0 EMAC 3 PCS masked interrupt has not occurred. 1 EMAC 3 PCS masked interrupt occurred.
26:31		Reserved

DCR 0x214 R/W

See *UIC2 Polarity Register (UIC2_PR)* on page 393.

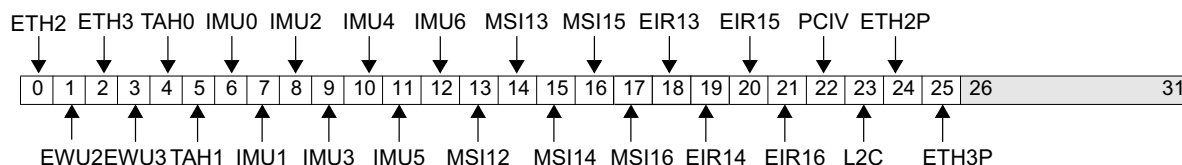
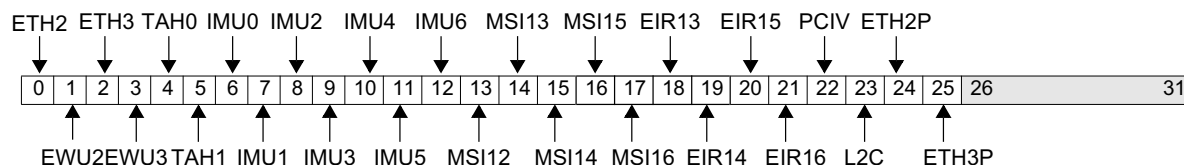


Figure 32-444. UIC2 Polarity Register (UIC2_PR)

0	ETH2	Ethernet 2 Interrupt Polarity 0 Ethernet 2 interrupt has negative polarity. 1 Ethernet 2 interrupt has positive polarity.
1	EWU2	Ethernet 2 Wake-up Interrupt Polarity 0 Ethernet 2 wake-up interrupt has negative polarity. 1 Ethernet 2 wake-up interrupt has positive polarity.
2	ETH3	Ethernet 3 Interrupt Polarity 0 Ethernet 3 interrupt has negative polarity. 1 Ethernet 3 interrupt has positive polarity.
3	EWU3	Ethernet 3 Wake-up Interrupt Polarity 0 Ethernet 3 wake-up interrupt has negative polarity. 1 Ethernet 3 wake-up interrupt has positive polarity.
4	TAH0	Tah 0 Interrupt Polarity 0 Tah 0 interrupt has negative polarity. 1 Tah 0 interrupt has positive polarity.
5	TAH1	Tah 1 Interrupt Polarity 0 Tah 1 interrupt has negative polarity. 1 Tah 1 interrupt has positive polarity.
6	IMU0	IMU Level 0 Interrupt Polarity 0 IMU level 0 interrupt has negative polarity. 1 IMU level 0 interrupt has positive polarity.
7	IMU1	IMU Level 1 Interrupt Polarity 0 IMU level 1 interrupt has negative polarity. 1 IMU level 1 interrupt has positive polarity.
8	IMU2	IMU Level 2 Interrupt Polarity 0 IMU level 2 interrupt has negative polarity. 1 IMU level 2 interrupt has positive polarity.
9	IMU3	IMU Level 3 Interrupt Polarity 0 IMU level 3 interrupt has negative polarity. 1 IMU level 3 interrupt has positive polarity.
10	IMU4	IMU Level 4 Interrupt Polarity 0 IMU level 4 interrupt has negative polarity. 1 IMU level 4 interrupt has positive polarity.
11	IMU5	IMU Level 5 Interrupt Polarity 0 IMU level 5 interrupt has negative polarity. 1 IMU level 5 interrupt has positive polarity.
12	IMU6	IMU Level 6 Interrupt Polarity 0 IMU level 6 interrupt has negative polarity. 1 IMU level 6 interrupt has positive polarity.
13	MSI12	PCI MSI Level 12 Interrupt Polarity 0 PCI MSI level 12 interrupt has negative polarity. 1 PCI MSI level 12 interrupt has positive polarity.

14	MSI13	PCI MSI Level 13 Interrupt Polarity 0 PCI MSI level 13 interrupt has negative polarity. 1 PCI MSI level 13 interrupt has positive polarity.
15	MSI14	PCI MSI Level 14 Interrupt Polarity 0 PCI MSI level 14 interrupt has negative polarity. 1 PCI MSI level 14 interrupt has positive polarity.
16	MSI15	PCI MSI Level 15 Interrupt Polarity 0 PCI MSI level 15 interrupt has negative polarity. 1 PCI MSI level 15 interrupt has positive polarity.
17	EIR13	External IRQ 13 Interrupt Polarity 0 External IRQ 13 interrupt has negative polarity. 1 External IRQ 13 interrupt has positive polarity.
18	EIR14	External IRQ 14 Interrupt Polarity 0 External IRQ 14 interrupt has negative polarity. 1 External IRQ 14 interrupt has positive polarity.
19	EIR15	External IRQ 15 Interrupt Polarity 0 External IRQ 15 interrupt has negative polarity. 1 External IRQ 15 interrupt has positive polarity.
20	EIR16	External IRQ 16 Interrupt Polarity 0 External IRQ 16 interrupt has negative polarity. 1 External IRQ 16 interrupt has positive polarity.
21	EIR17	External IRQ 17 Interrupt Polarity 0 External IRQ 17 interrupt has negative polarity. 1 External IRQ 17 interrupt has positive polarity.
22	PCIV	PCI VPD Interrupt Polarity 0 PCI VPD interrupt has negative polarity. 1 PCI VPD interrupt has positive polarity.
23	L2C	L2 Cache Interrupt Polarity 0 L2 cache interrupt has negative polarity. 1 L2 cache interrupt has positive polarity.
24	ETH2P	EMAC 2 PCS Interrupt Polarity 0 EMAC 2 PCS interrupt has negative polarity. 1 EMAC 2 PCS interrupt has positive polarity.
25	ETH3P	EMAC 3 PCS Interrupt Polarity 0 EMAC 3 PCS interrupt has negative polarity. 1 EMAC 3 PCS interrupt has positive polarity.
26:31		Reserved

DCR 0x210 R/clearSee *UIC2 Status Register (UIC2_SR)* on page 372.Figure 32-445. *UIC2 Status Register (UIC2_SR)*

0	ETH2	Ethernet 2 Interrupt Status 0 Ethernet 2 interrupt has not occurred. 1 Ethernet 2 interrupt occurred.
1	EWU2	Ethernet 2 Wake-up Interrupt Status 0 Ethernet 2 wake-up interrupt has not occurred. 1 Ethernet 2 wake-up interrupt occurred.
2	ETH3	Ethernet 3 Interrupt Status 0 Ethernet 3 interrupt has not occurred. 1 Ethernet 3 interrupt occurred.
3	EWU3	Ethernet 3 Wake-up Interrupt Status 0 Ethernet 3 wake-up interrupt has not occurred. 1 Ethernet 3 wake-up interrupt occurred.
4	TAH0	TAH 0 Interrupt Status 0 TAH 0 interrupt has not occurred. 1 TAH 0 interrupt occurred.
5	TAH1	TAH 1 Interrupt Status 0 TAH 1 interrupt has not occurred. 1 TAH 1 interrupt occurred.
6	IMU0	IMU Level 0 Interrupt Status 0 IMU level 0 interrupt has not occurred. 1 IMU level 0 interrupt occurred.
7	IMU1	IMU Level 1 Interrupt Status 0 IMU level 1 interrupt has not occurred. 1 IMU level 1 interrupt occurred.
8	IMU2	IMU Level 2 Interrupt Status 0 IMU level 2 interrupt has not occurred. 1 IMU level 2 interrupt occurred.
9	IMU3	IMU Level 3 Interrupt Status 0 IMU level 3 interrupt has not occurred. 1 IMU level 3 interrupt occurred.
10	IMU4	IMU Level 4 Interrupt Status 0 IMU level 4 interrupt has not occurred. 1 IMU level 4 interrupt occurred.
11	IMU5	IMU Level 5 Interrupt Status 0 IMU level 5 interrupt has not occurred. 1 IMU level 5 interrupt occurred.
12	IMU6	IMU Level 6 Interrupt Status 0 IMU level 6 interrupt has not occurred. 1 IMU level 6 interrupt occurred.
13	MSI12	PCI MSI Level 12 Interrupt Status 0 PCI MSI level 12 interrupt has not occurred. 1 PCI MSI level 12 interrupt occurred.

14	MSI13	PCI MSI Level 13 Interrupt Status 0 PCI MSI level 13 interrupt has not occurred. 1 PCI MSI level 13 interrupt occurred.
15	MSI14	PCI MSI Level 14 Interrupt Status 0 PCI MSI level 14 interrupt has not occurred. 1 PCI MSI level 14 interrupt occurred.
16	MSI15	PCI MSI Level 15 Interrupt Status 0 PCI MSI level 15 interrupt has not occurred. 1 PCI MSI level 15 interrupt occurred.
17	EIR13	External IRQ 13 Interrupt Status 0 External IRQ 13 interrupt has not occurred. 1 External IRQ 13 interrupt occurred.
18	EIR14	External IRQ 14 Interrupt Status 0 External IRQ 14 interrupt has not occurred. 1 External IRQ 14 interrupt occurred.
19	EIR15	External IRQ 15 Interrupt Status 0 External IRQ 15 interrupt has not occurred. 1 External IRQ 15 interrupt occurred.
20	EIR16	External IRQ 16 Interrupt Status 0 External IRQ 16 interrupt has not occurred. 1 External IRQ 16 interrupt occurred.
21	EIR17	External IRQ 17 Interrupt Status 0 External IRQ 17 interrupt has not occurred. 1 External IRQ 17 interrupt occurred.
22	PCIV	PCI VPD Interrupt Status 0 PCI VPD interrupt has not occurred. 1 PCI VPD interrupt occurred.
23	L2C	L2 Cache Interrupt Status 0 L2 cache interrupt has not occurred. 1 L2 cache interrupt occurred.
24	ETH2P	EMAC 2 PCS Interrupt Status 0 EMAC 2 PCS interrupt has not occurred. 1 EMAC 2 PCS interrupt occurred.
25	ETH3P	EMAC 3 PCS Interrupt Status 0 EMAC 3 PCS interrupt has not occurred. 1 EMAC 3 PCS interrupt occurred.
26:31		Reserved

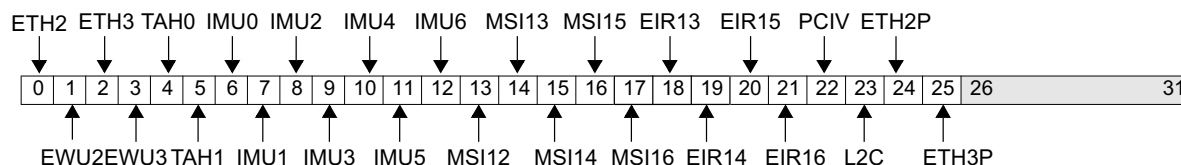
DCR 0x215 R/WSee *UIC2 Trigger Register (UIC2_TR)* on page 400.

Figure 32-446. UIC2 Trigger Register (UIC2_TR)

0	ETH2	Ethernet 2 Interrupt Trigger 0 Ethernet 2 interrupt is level-sensitive. 1 Ethernet 2 interrupt is edge-sensitive.
1	EWU2	Ethernet 2 Wake-up Interrupt Trigger 0 Ethernet 2 wake-up interrupt is level-sensitive. 1 Ethernet 2 wake-up interrupt is edge-sensitive.
2	ETH3	Ethernet 3 Interrupt Trigger 0 Ethernet 3 interrupt is level-sensitive. 1 Ethernet 3 interrupt is edge-sensitive.
3	EWU3	Ethernet 3 Wake-up Interrupt Trigger 0 Ethernet 3 wake-up interrupt is level-sensitive. 1 Ethernet 3 wake-up interrupt is edge-sensitive.
4	TAH0	Tahoe 0 Interrupt Trigger 0 Tahoe 0 interrupt is level-sensitive. 1 Tahoe 0 interrupt is edge-sensitive.
5	TAH1	Tahoe 1 Interrupt Trigger 0 Tahoe 1 interrupt is level-sensitive. 1 Tahoe 1 interrupt is edge-sensitive.
6	IMU0	IMU Level 0 Interrupt Trigger 0 IMU level 0 interrupt is level-sensitive. 1 IMU level 0 interrupt is edge-sensitive.
7	IMU1	IMU level 1 Interrupt Trigger 0 IMU level 1 interrupt is level-sensitive. 1 IMU level 1 interrupt is edge-sensitive.
8	IMU2	IMU Level 2 Interrupt Trigger 0 IMU level 2 interrupt is level-sensitive. 1 IMU level 2 interrupt is edge-sensitive.
9	IMU3	IMU Level 3 Interrupt Trigger 0 IMU level 3 interrupt is level-sensitive. 1 IMU level 3 interrupt is edge-sensitive.
10	IMU4	IMU Level 4 Interrupt Trigger 0 IMU level 4 interrupt is level-sensitive. 1 IMU level 4 interrupt is edge-sensitive.
11	IMU5	IMU Level 5 Interrupt Trigger 0 IMU level 5 interrupt is level-sensitive. 1 IMU level 5 interrupt is edge-sensitive.
12	IMU6	IMU Level 6 Interrupt Trigger 0 IMU level 6 interrupt is level-sensitive. 1 IMU level 6 interrupt is edge-sensitive.
13	MSI12	PCI MSI Level 12 Interrupt Trigger 0 PCI MSI level 12 interrupt is level-sensitive. 1 PCI MSI level 12 interrupt is edge-sensitive.

14	MSI13	PCI MSI Level 13 Interrupt Trigger 0 PCI MSI level 13 interrupt is level-sensitive. 1 PCI MSI level 13 interrupt is edge-sensitive.
15	MSI14	PCI MSI Level 14 Interrupt Trigger 0 PCI MSI level 14 interrupt is level-sensitive. 1 PCI MSI level 14 interrupt is edge-sensitive.
16	MSI15	PCI MSI Level 15 Interrupt Trigger 0 PCI MSI level 15 interrupt is level-sensitive. 1 PCI MSI level 15 interrupt is edge-sensitive.
17	EIR13	External IRQ 13 Interrupt Trigger 0 External IRQ 13 interrupt is level-sensitive. 1 External IRQ 13 interrupt is edge-sensitive.
18	EIR14	External IRQ 14 Interrupt Trigger 0 External IRQ 14 interrupt is level-sensitive. 1 External IRQ 14 interrupt is edge-sensitive.
19	EIR15	External IRQ 15 Interrupt Trigger 0 External IRQ 15 interrupt is level-sensitive. 1 External IRQ 15 interrupt is edge-sensitive.
20	EIR16	External IRQ 16 Interrupt Trigger 0 External IRQ 16 interrupt is level-sensitive. 1 External IRQ 16 interrupt is edge-sensitive.
21	EIR17	External IRQ 17 Interrupt Trigger 0 External IRQ 17 interrupt is level-sensitive. 1 External IRQ 17 interrupt is edge-sensitive.
22	PCIV	PCI VPD Interrupt Trigger 0 PCI VPD interrupt is level sensitive. 1 PCI VPD interrupt is edge sensitive.
23	L2C	L2 Cache Interrupt Trigger 0 L2 cache interrupt is level sensitive. 1 L2 cache interrupt is edge sensitive.
24	ETH2P	EMAC 2 PCS Interrupt Trigger 0 EMAC 2 PCS interrupt is level sensitive. 1 EMAC 2 PCS interrupt is edge sensitive.
25	ETH3P	EMAC 3 PCS Interrupt Trigger 0 EMAC 3 PCS interrupt is level sensitive. 1 EMAC 3 PCS interrupt is edge sensitive.
26:31		Reserved

DCR 0x217 W/only

See UIC2 Vector Configuration Register (UIC2_VCR) on page 411.

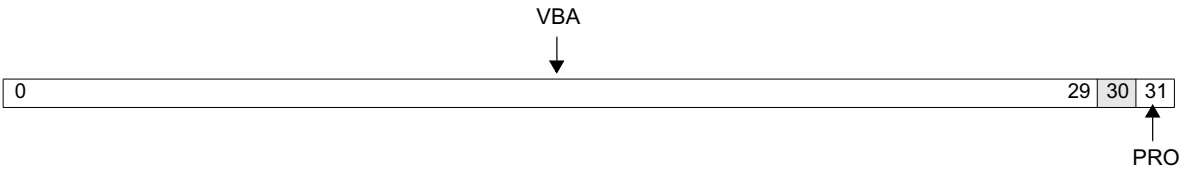


Figure 32-447. UIC2 Vector Configuration Register (UIC2_VCR)

0:29	VBA	Vector Base Address
30		Reserved
31	PRO	Priority Ordering 0 UIC2_SR[31] is the highest priority interrupt. 1 UIC2_SR[0] is the highest priority interrupt. Note: Vector generation is not performed for non-critical interrupts.

DCR 0x218 R/only

See *UIC2 Vector Configuration Register (UIC2_VCR)* on page 411.



Figure 32-448. UIC2 Vector Register (UIC2_VR)

0:31		Interrupt Vector
------	--	------------------

Preliminary User's Manual

DCR 0x203 R/W

See *UIC Base Critical Register (UICB0_CR)* on page 388.

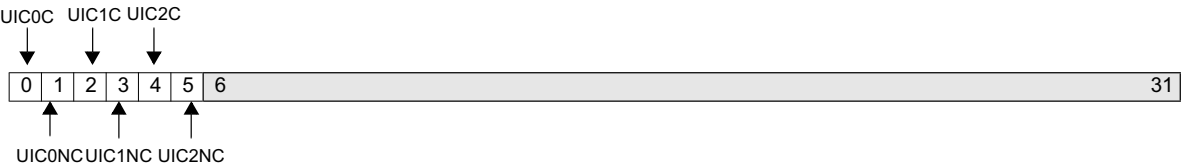


Figure 32-449. UIC Base Critical Register (UICB_CR)

0	UIC0C	UIC0 Critical Interrupt Class 0 UIC0 critical interrupt is non-critical. 1 UIC0 critical interrupt is critical.
1	UIC0NC	UIC0 Non-critical Interrupt Class 0 UIC0 non-critical interrupt is non-critical. 1 UIC0 non-critical interrupt is critical.
2	UIC1C	UIC1 Critical Interrupt Class 0 UIC1 critical interrupt is non-critical. 1 UIC1 critical interrupt is critical.
3	UIC1NC	UIC1 Non-critical Interrupt Class 0 UIC1 non-critical interrupt is non-critical. 1 UIC1 non-critical interrupt is critical.
4	UIC2C	UIC2 Critical Interrupt Class 0 UIC2 critical interrupt is non-critical. 1 UIC2 critical interrupt is critical.
5	UIC2NC	UIC2 Non-critical Interrupt Class 0 UIC2 non-critical interrupt is non-critical. 1 UIC2 non-critical interrupt is critical.
6:31		Reserved

DCR 0x202 R/W

See *UIC Base Enable Register (UICB0_ER)* on page 381.

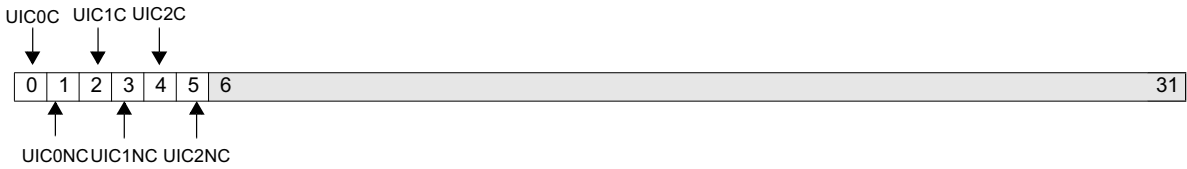


Figure 32-450. UIC Base Enable Register (UICB_ER)

0	UIC0C	UIC0 Critical Interrupt Enable 0 UIC0 critical interrupt is disabled. 1 UIC0 critical interrupt is enabled.
1	UIC0NC	UIC0 Non-critical Interrupt Enable 0 UIC0 non-critical interrupt is disabled. 1 UIC0 non-critical interrupt is enabled.
2	UIC1C	UIC1 Critical Interrupt Enable 0 UIC1 critical interrupt is disabled. 1 UIC1 critical interrupt is enabled.
3	UIC1NC	UIC1 Non-critical Interrupt Enable 0 UIC1 non-critical interrupt is disabled. 1 UIC1 non-critical interrupt is enabled.
4	UIC2C	UIC2 Critical Interrupt Enable 0 UIC2 critical interrupt is disabled. 1 UIC2 critical interrupt is enabled.
5	UIC2NC	UIC2 Non-critical Interrupt Enable 0 UIC2 non-critical interrupt is disabled. 1 UIC2 non-critical interrupt is enabled.
6:31		Reserved

DCR 0x206 R/only

See *UIC Base Masked Status Register (UICB0_MSR)* on page 409.

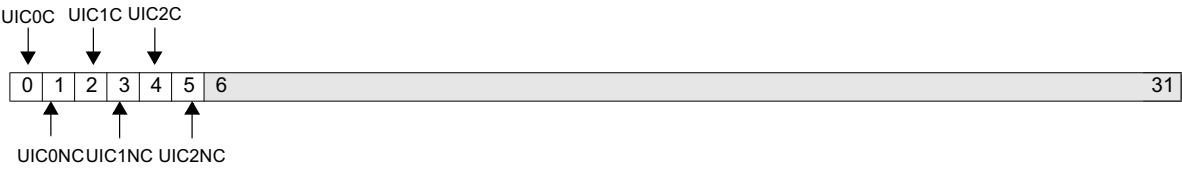


Figure 32-451. UIC Base Masked Status Register (UICB_MSR)

0	UIC0C	UIC0 Critical Masked Interrupt Status 0 UIC0 critical masked interrupt has not occurred. 1 UIC0 critical masked interrupt occurred.
1	UIC0NC	UIC0 Non-critical Masked Interrupt Status 0 UIC0 non-critical masked interrupt has not occurred. 1 UIC0 non-critical masked interrupt occurred.
2	UIC1C	UIC1 Critical Masked Interrupt Status 0 UIC1 critical masked interrupt has not occurred. 1 UIC1 critical masked interrupt occurred.
3	UIC1NC	UIC1 Non-critical Masked Interrupt Status 0 UIC1 non-critical masked interrupt has not occurred. 1 UIC1 non-critical masked interrupt occurred.
4	UIC2C	UIC2 Critical Masked Interrupt Status 0 UIC2 critical masked interrupt has not occurred. 1 UIC2 critical masked interrupt occurred.
5	UIC2NC	UIC2 Non-critical Masked Interrupt Status 0 UIC2 non-critical interrupt has not occurred. 1 UIC2 non-critical interrupt occurred.
6:31		Reserved

DCR 0x204 R/W

See *UIC Base Polarity Register (UICB0_PR)* on page 395.

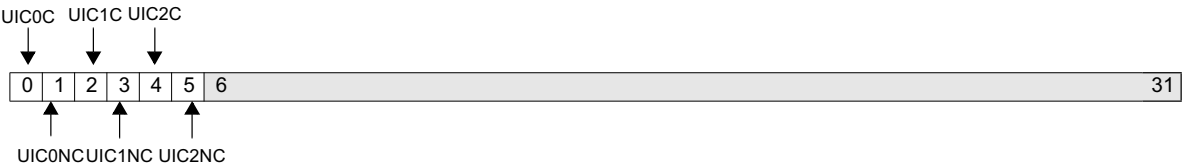


Figure 32-452. UIC Base Polarity Register (UICB_PR)

0	UIC0C	UIC0 Critical Interrupt Polarity 0 UIC0 critical interrupt has negative polarity. 1 UIC0 critical interrupt has positive polarity.
1	UIC0NC	UIC0 Non-critical Interrupt Polarity 0 UIC0 non-critical interrupt has negative polarity. 1 UIC0 non-critical interrupt has positive polarity.
2	UIC1C	UIC1 Critical Interrupt Polarity 0 UIC1 critical interrupt has negative polarity. 1 UIC1 critical interrupt has positive polarity.
3	UIC1NC	UIC1 Non-critical Interrupt Polarity 0 UIC1 non-critical interrupt has negative polarity. 1 UIC1 non-critical interrupt has positive polarity.
4	UIC2C	UIC2 Critical Interrupt Polarity 0 UIC2 critical interrupt has negative polarity. 1 UIC2 critical interrupt has positive polarity.
5	UIC2NC	UIC2 Non-critical Interrupt Polarity 0 UIC2 non-critical interrupt has negative polarity. 1 UIC2 non-critical interrupt has positive polarity.
6:31		Reserved

Preliminary User's Manual

DCR 0x200 R/clear

See UIC Base Status Register (UICB0_SR) on page 374.

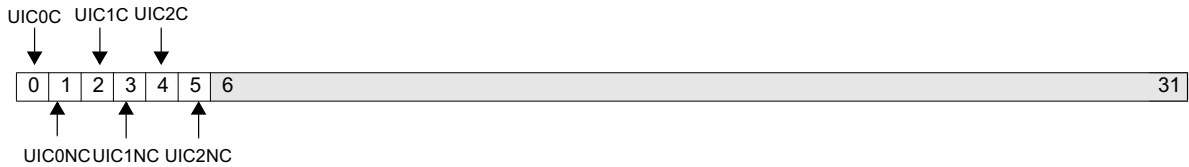


Figure 32-453. UIC Base Status Register (UICB_SR)

0	UIC0C	UIC0 Critical Interrupt Status 0 UIC0 critical interrupt has not occurred. 1 UIC0 critical interrupt occurred.
1	UIC0NC	UIC0 Non-Critical Interrupt Status 0 UIC0 non-critical interrupt has not occurred. 1 UIC0 non-critical interrupt occurred.
2	UIC1C	UIC1 Critical Interrupt Status 0 UIC1 critical interrupt has not occurred. 1 UIC1 critical interrupt occurred.
3	UIC1NC	UIC1 Non-Critical Interrupt Status 0 UIC1 non-critical interrupt has not occurred. 1 UIC1 non-critical interrupt occurred.
4	UIC2C	UIC2 Critical Interrupt Status 0 UIC2 critical interrupt has not occurred. 1 UIC2 critical interrupt occurred.
5	UIC2NC	UIC2 Non-Critical Interrupt Status 0 UIC2 non-critical interrupt has not occurred. 1 UIC2 non-critical interrupt occurred.
6:31		Reserved

DCR 0x205 R/W

See *UIC Base Trigger Register (UICB0_TR)* on page 402.

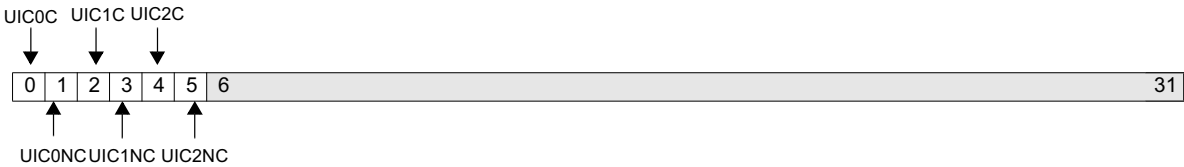


Figure 32-454. UIC Base Trigger Register (UICB_TR)

0	UIC0C	UIC0 Critical Interrupt Trigger 0 UIC0 critical interrupt is level-sensitive. 1 UIC0 critical interrupt is edge-sensitive.
1	UIC0NC	UIC0 Non-critical Interrupt Trigger 0 UIC0 non-critical interrupt is level-sensitive. 1 UIC0 non-critical interrupt is edge-sensitive.
2	UIC1C	UIC1 Critical Interrupt Trigger 0 UIC1 critical interrupt is level-sensitive. 1 UIC1 critical interrupt is edge-sensitive.
3	UIC1NC	UIC1 Non-critical Interrupt Trigger 0 UIC1 non-critical interrupt is level-sensitive. 1 UIC1 non-critical interrupt is edge-sensitive.
4	UIC2C	UIC2 Critical Interrupt Trigger 0 UIC2 critical interrupt is level-sensitive. 1 UIC2 critical interrupt is edge-sensitive.
5	UIC2NC	UIC2 Non-critical Interrupt Trigger 0 UIC2 non-critical interrupt is level-sensitive. 1 UIC2 non-critical interrupt is edge-sensitive.
6:31		Reserved

DCR 0x207 W/only

See *UIC Base Vector Configuration Register (UICB0_VCR)* on page 412.

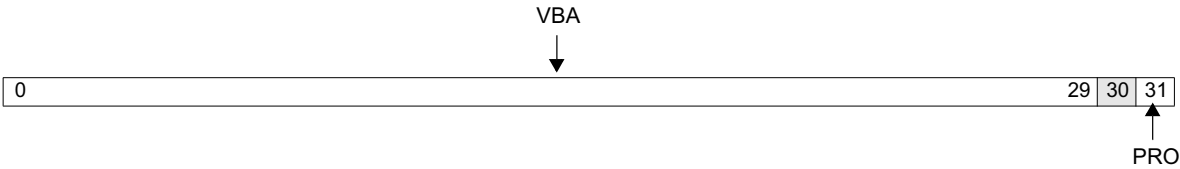


Figure 32-455. UIC Base Vector Configuration Register (UICB_VCR)

0:29	VBA	Vector Base Address
30		Reserved
31	PRO	Priority Ordering 0 UICB_SR[31] is the highest priority interrupt. 1 UICB_SR[0] is the highest priority interrupt. Note: Vector generation is not performed for non-critical interrupts.

DCR 0x208 R/only

See *UIC Base Vector Configuration Register (UICB0_VCR)* on page 412.



Figure 32-456. UIC Base Vector Register (UICB_VR)

0:31		Interrupt Vector
------	--	------------------

Preliminary User's Manual**ZMII0_FER**

Function Enable Register

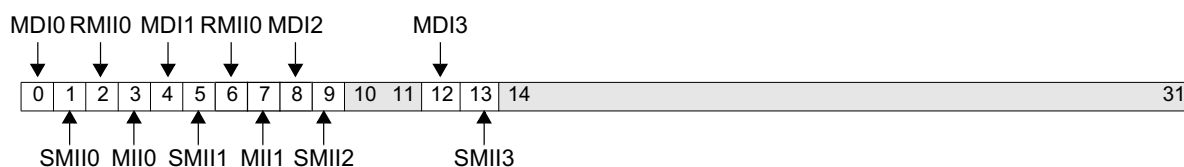
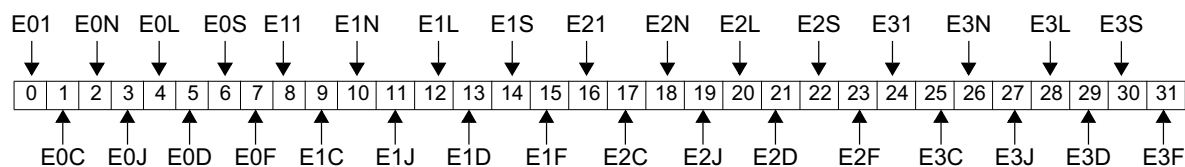
MMIO 0x014000780 Read/WriteSee *Function Enable Register (ZMII0_FER)* on page 800.

Figure 32-457. Function Enable Register (ZMII0_FER)

0	MDI0	EMAC0 MDI Enable 0 EMAC0 management data and clock are ignored. 1 EMAC0 management data and clock are driven to the PHY.	Must be 0 if MDI1, MDI2, MDI3 = 1.
1	SMII0	EMAC0 SMII Enable 0 EMAC0 SMII PHY interface is disabled. 1 EMAC0 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
2	RMII0	EMAC0 RMII Enable 0 EMAC0 RMII PHY interface is disabled. 1 EMAC0 RMII PHY interface is enabled.	Must be 0 if SMII0, SMII1, SMII2, SMII3, MII0, or MII1 is 1.
3	II0	EMAC0 MII Enable 0 EMAC0 MII PHY interface is disabled. 1 EMAC0 MII PHY interface is enabled.	Must be 0 if SMII0, SMII1, SMII2, SMII3, RMII0, RMII1, or MII1 is 1.
4	MDI1	EMAC1 MDI Enable 0 EMAC1 management data and clock are ignored. 1 EMAC1 management data and clock are driven to the PHY.	Must be 0 if MDI0, MDI2, MDI3 = 1.
5	SMII1	EMAC1 SMII Enable 0 EMAC1 SMII PHY interface is disabled. 1 EMAC1 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
6	RMII1	EMAC0 RMII Enable 0 EMAC1 RMII PHY interface is disabled. 1 EMAC1 RMII PHY interface is enabled.	Must be 0 if SMII0, SMII1, SMII2, SMII3, MII0, or MII1 is 1.
7	II1	EMAC0 MII Enable 0 EMAC1 MII PHY interface is disabled. 1 EMAC1 MII PHY interface is enabled.	Must be 0 if SMII0, SMII1, SMII2, SMII3, RMII0, MII0, or RMII1 is 1.
8	MDI2	EMAC2 MDI Enable 0 EMAC2 management data and clock are ignored. 1 EMAC2 management data and clock are driven to the PHY.	Must be 0 if MDI0, MDI1, MDI3 = 1.
9	SMII2	EMAC2 SMII Enable 0 EMAC2 SMII PHY interface is disabled. 1 EMAC2 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
10:11		Reserved	
12	MDI3	EMAC3 MDI Enable 0 EMAC3 management data and clock are ignored. 1 EMAC3 management data and clock are driven to the PHY.	Must be 0 if MDI0, MDI1, MDI2 = 1.
13	SMII3	EMAC3 SMII Enable 0 EMAC3 SMII PHY interface is disabled. 1 EMAC3 SMII PHY interface is enabled.	Must be 0 if RMII0, MII0, RMII1, or MII1 is 1.
14:31		Reserved	

ZMII0_SMIISR

SMII Status Register

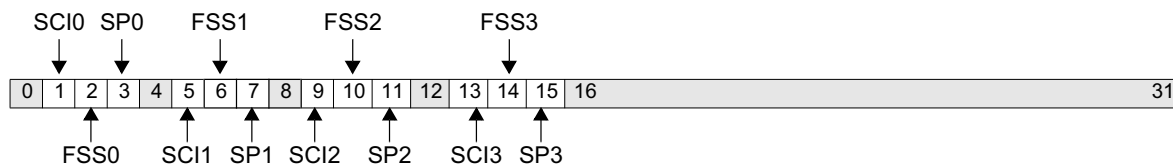
Preliminary User's Manual**MMIO 0x014000788 Read/Write**See *SMII Status Register (ZMII0_SMIISR)* on page 803.**Figure 32-458. SMII Status Register (ZMII0_SMIISR)**

0	E01	EMAC0RxD Set to 1	
1	E0C	EMAC0RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
2	E0N	EMAC0RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
3	E0J	EMAC0RxD Jabber 0 OK 1 Error	
4	E0L	EMAC0RxD Link 0 Down 1 Up	
5	E0D	EMAC0RxD Duplex 0 Half 1 Full	
6	E0S	EMAC0RxD Speed 0 10MBit 1 100MBit	
7	E0F	EMAC0RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
8	E11	EMAC1RxD Set to 1	
9	E1C	EMAC1RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
10	E1N	EMAC1RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
11	E1J	EMAC1RxD Jabber 0 OK 1 Error	
12	E1L	EMAC1RxD Link 0 Down 1 Up	
13	E1D	EMAC1RxD Duplex 0 Half 1 Full	
14	E1S	EMAC1RxD Speed 0 10MBit 1 100MBit	
15	E1F	EMAC1RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.

16	E2I	EMAC2RxD Set to 1	
17	E2C	EMAC2RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
18	E2N	EMAC2RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
19	E2J	EMAC2RxD Jabber 0 OK 1 Error	
20	E2L	EMAC2RxD Link 0 Down 1 Up	
21	E2D	EMAC2RxD Duplex 0 Half 1 Full	
22	E2S	EMAC2RxD Speed 0 10MBit 1 100MBit	
23	E2F	EMAC2RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
24	E3I	EMAC3RxD Set to 1	
25	E3C	EMAC3RxD False Carrier Detected	When asserted, this bit indicates that the PHY has detected a false carrier event.
26	E3N	EMAC3RxD Nibble 0 Invalid 1 Valid	Conveys the validity of the upper nibble of the last byte of the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.
27	E3J	EMAC3RxD Jabber 0 OK 1 Error	
28	E3L	EMAC3RxD Link 0 Down 1 Up	
29	E3D	EMAC3RxD Duplex 0 Half 1 Full	
30	E3S	EMAC3RxD Speed 0 10MBit 1 100MBit	
31	E3F	EMAC3RxD from Previous Frame	Indicates whether or not the PHY detected an error somewhere in the previous frame. Should be valid in the segment immediately following a frame, and should stay valid until the first data segment of the next frame begin.

ZMII0_SSR

Speed Selection Register

Preliminary User's Manual**MMIO 0x014000784 Read/Write**See *Speed Select Register (ZMII0_SSR)* on page 801.*Figure 32-459. Speed Selection Register (ZMII0_SSR)*

0		Reserved
1	SCI0	EMAC0 Suppress Collision Indication 0 EMAC0 collision indication signal is enabled. 1 EMAC0 collision indication signal is suppressed.
2	FSS0	EMAC0 Force Speed Selection 0 ZMII0_SSR[SP0] does not override IPG status field. 1 ZMII0_SP0 overrides IPG status field. SMII only.
3	SP0	EMAC0 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected. Must be programmed for RMII; can be programmed for SMII; ignored for MII.
4		Reserved
5	SCI1	EMAC1 Suppress Collision Indication 0 EMAC1 collision indication signal is enabled. 1 EMAC1 collision indication signal is suppressed.
6	FSS1	EMAC1 Force Speed Selection 0 ZMII0_SSR[SP1] does not override IPG status field. 1 ZMII0_SSR[SP1] overrides IPG status field. SMII only.
7	SP1	EMAC1 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected. Must be programmed for RMII; can be programmed for SMII; ignored for MII.
8		Reserved
9	SCI2	EMAC2 Suppress Collision Indication 0 EMAC2 collision indication signal is asserted. 1 EMAC2 collision indication signal is deasserted.
10	FSS2	EMAC2 Force Speed Selection 0 ZMII0_SSR[SP2] does not override IPG status field. 1 ZMII0_SSR[SP2] overrides IPG status field. SMII only.
11	SP2	EMAC3 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected. Must be programmed for RMII; can be programmed for SMII; ignored for MII.
12		Reserved
13	SCI3	EMAC3 Suppress Collision Indication 0 EMAC3 collision indication signal is asserted. 1 EMAC3 collision indication signal is deasserted.

Preliminary User's Manual

14	FSS3	EMAC3 Force Speed Selection 0 ZMII0_SSR[SP3] does not override IPG status field. 1 ZMII0_SSR[SP3] overrides IPG status field.	SMII only.
15	SP3	EMAC3 Speed Selection 0 10 Mbps selected. 1 100 Mbps selected.	Must be programmed for RMII; can be programmed for SMII; ignored for MII.
16:31		Reserved	

Preliminary User's Manual**33. Signal Summary**

This chapter provides detailed information on the PPC440GX Embedded Processor I/O signals.

33.1 Signals Listed Alphabetically

Table 33-1 lists the PPC440GX Embedded Processor signals in alphabetical order with an indication of the interface group to which it belongs to. Table 33-2 lists the PPC440GX Embedded Processor signals by interface category with a brief description.

Multiplexed signals are shown in brackets following the first signal name assigned to each multiplexed ball (for example, IRQ0:6[GPIO17:23]). Active-low signals are shown with an overbar on the signal name (for example, ExtAck).

Table 33-1. Signals Listed Alphabetically

Signal Name	Description	I/O
AGND	PLL (analog) voltage ground.	n/a
AxV _{DD}	1.5V—Filtered voltages input for PLLs (analog circuits) Note: A separate filter for each of the three voltages is recommended.	n/a
BA0:1	Bank Address supporting up to four internal banks.	O
BankSel0:3	Selects up to four external DDR SDRAM banks.	O
BusReq	Bus Request. Used when the PPC440GX needs to regain control of peripheral interface from an external master.	O
CAS	Column Address Strobe.	O
ClkEn0:3	Clock Enable. One for each bank.	O
DM0:8	Memory write data byte lane masks. MEMDM8 is the byte lane mask for the ECC byte lane.	O
DMAAck0:3	Used by the PPC440GX to indicate that data transfers have occurred.	O
DMAReq0:3	Used by slave peripherals to indicate they are prepared to transfer data.	I
DQS0:8	Byte lane data strobe. DQS8 is the data strobe for the ECC byte lane.	I/O
DrvrInh2	Driver Inhibit. Used for test purposes only. Tie up for normal operation	I
ECC0:7	ECC check bits 0:7.	I/O
EMCCD, EMC1RxErr, GMCTxCIk, GMC0TxCIk, TBITxCIk, RTBI0TxCIk	MII: Collision detection RMII 1: Receive error GMII: Transmit clock RGMII: Transmit clock TBI: Transmit clock RTBI: Transmit clock	I/O
EMCCrS, EMC0CrSDV, GMCTxD7, GMC1TxD3, TBITxD7, RTBI1TxD3	MII: Carrier sense RMII 0: Carrier sense data valid GMII: Transmit data RGMII 1: Transmit data TBI: Transmit data RTBI 1: Transmit data	I/O
EMCMDClk	MII and RMII: Management data clock	O
EMCMDIO	MII and RMII: Transfer command and status information between MII and PHY	I/O

Table 33-1. Signals Listed Alphabetically (Continued)

Signal Name	Description	I/O
EMCRxD0:3, EMC0Rx0:1, EMC1Rx0:1, EMC0Rx0, EMC1Rx0, EMC2Rx0, EMC3Rx0, GMCTxD0:1, GMC0Tx0:1, TBITxD0:1, RTBI0Tx0:1	MII: Receive data RMII 0: Receive data RMII 1: Receive data SMII 0: Receive data SMII 1: Receive data SMII 2: Receive data SMII 3: Receive data GMII: Transmit data RGMII 0: Transmit data TBI: Transmit data RTBI 0: Transmit data	I/O
EMCRxDV, EMC1CrSDV, GMCTxD4, GMC1Tx0, TBITxD4, RTBI1Tx0	MII: Receive data valid RMII 1: Carrier sense data valid GMII: Transmit data RGMII 1: Transmit data TBI: Transmit data RTBI 1: Transmit data	I/O
EMCRxCIk, GMCTxD5, GMC1Tx0, TBITxD5, RTBI1Tx0	MII: Receive clock GMII: Transmit data RGMII 1: Transmit data TBI: Transmit data RTBI 1: Transmit data	I/O
EMCRxErr, EMC0RxErr, GMCTxD6, GMC1Tx0, TBITxD6, RTBI1Tx0	MII: Receive error RMII 0: Receive error GMII: Transmit data RGMII 1: Transmit data TBI: Transmit data RTBI 1: Transmit data	I/O
EMCTxCIk, EMCRefCk	MII: Transmit clock RMII and SMII: Transmit clock	I
EMCTxD0:3, EMC0Tx0:1, EMC1Tx0:1, EMC0Tx0, EMC1Tx0, EMC2Tx0, EMC3Tx0, GMCTxD2:3, GMC0Tx0:3, TBITxD2:3, RTBI0Tx0:3	MII: Transmit data RMII 0: Transmit data RMII 1: Transmit data SMII 0: Transmit data SMII 1: Transmit data SMII 2: Transmit data SMII 3: Transmit data GMII: Transmit data RGMII 0: Transmit data TBI: Transmit data RTBI 0: Transmit data	O
EMCTxEn, EMC0TxEn, EMCSync	MII: Transmit data enabled RMII 0: Transmit data enabled SMII: Sync signal	O
EMCTxErr, EMC1TxEn, GMCRxCIk, GMC0Rx0, TBIRxCIk0, RTBI0Rx0	MII: Transmit error: RMII: Transmit data enabled GMII: Receive clock RGMII: Receive clock TBI: Receive clock RTBI: Receive clock	I/O
EOT0:3/TC0:3	End Of Transfer/Terminal Count.	I/O
ExtAck	External Acknowledgement. Used by the PPC440GX to indicate that a data transfer occurred.	O
ExtReq	External Request. Used by an external master to indicate it is prepared to transfer data.	I
ExtReset	Peripheral Reset. Used by an external master and by synchronous peripheral slaves.	O
GMCCD, GMC1Rx0, TBIRxCIk, RTBI1Rx0	GMII: Collision detection RGMII: Receive clock TBI: Receive clock RTBI: Receive clock	I

Preliminary User's Manual

Table 33-1. Signals Listed Alphabetically (Continued)

Signal Name	Description	I/O
GMCCrS, GMC1TxClk, RTBI1TxClk	GMII: Carrier sense RGMII: Transmit clock RTBI: Transmit clock	I/O
GMCTxClk	GMII, RGMII, TBI and RTBI: Gigabit reference clock	I
GMCRxD0:3, GMC0RxD0:3, TBIRxD0:3, RTBI0RxD0:3	GMII: Receive data RGMII: Receive data TBI: Receive data RTBI: Receive data	I
GMCRxD4:7, GMC1RxD0:3, TBIRxD4:7, RTBI1RxD0:3	GMII: Receive data RGMII: Receive data TBI: Receive data RTBI: Receive data	I
GMCRxDV, GMC0RxCtl, TBIRxD8, RTBI0RxCtl	GMII: Receive data valid RGMII: Receive control TBI: Receive data RTBI: Receive control	I
GMCRxEr, GMC1RxCtl, TBIRxD9, RTBI1RxCtl	GMII: Receive error RGMII: Receive control TBI: Receive data RTBI: Receive control	I
GMCTxEr, GMC0TxCtl, TBITxD8, RTBI0TxCtl	GMII: Transmit data enable RGMII: Transmit control TBI: Transmit data RTBI: Transmit control	O
GMCTxEr, GMC1TxCtl, TBITxD9, RTBI1TxCtl	GMII: Transmit error RGMII: Transmit control TBI: Transmit data RTBI: Transmit control	O
GND	Ground.	n/a
GPIO00:31	General purpose I/O 0 through 10. To access these functions, software must set DCR register bits.	I/O
Halt	Halt from external debugger.	I
HoldAck	Hold Acknowledge. Used by the PPC440GX to transfer ownership of peripheral bus to an external master.	O
HoldReq	Hold Request. Used by an external master to request ownership of the peripheral bus.	I
IIC0SClk	IIC0 Serial Clock.	I/O
IIC0SDA	IIC0 Serial Data.	I/O
IIC1SClk	IIC1 Serial Clock.	I/O
IIC1SDA	IIC1 Serial Data.	I/O
IRQ00:10	External interrupt Requests 0 through 10.	I
IRQ11:12	External interrupt Requests 11 through 12.	I
IRQ13:17	External interrupt Requests 13 through 17.	I
MemAddr00:12	Memory address bus.	O
MemClkOut0 MemClkOut0	Subsystem clock.	O
MemData00:63	Memory data bus.	I/O
MemVRef1:2	Memory reference voltage (SV _{REF}) input.	I

Table 33-1. Signals Listed Alphabetically (Continued)

Signal Name	Description	I/O
OV _{DD}	3.3V supply—I/O (except DDR SDRAM, Ethernet)	n/a
PCIXAD00:63	Address/Data bus (bidirectional).	I/O
PCIXC0:7[BE0:7]	PCI-X Command[Byte Enables].	I/O
PCIXCap	Capable of PCI-X operation.	I
PCIX133Cap	PCI-X devices are 133 MHz capable.	O
PCIXClk	Provides timing to the PCI interface for PCI transactions.	I
PCIXDevSel	Indicates the driving device has decoded its address as the target of the current access.	I/O
PCIXFrame	Driven by the current master to indicate beginning and duration of an access.	I/O
PCIXGnt0	Indicates that the specified agent is granted access to the bus. When using an external PCI/PCI-X arbiter, connect the external arbiter's Grant line to this signal.	I/O
PCIXGnt1	Indicates that the specified agent is granted access to the bus.	I/O
PCIXGnt2:5	Indicates that the specified agent is granted access to the bus.	O
PCIXIDSel	Used as a chip select during configuration read and write transactions.	I
PCIXINT	Level sensitive PCI interrupt.	O
PCIXIRDY	Indicates initiating agent's ability to complete the current data phase of the transaction.	I/O
PCIXM66En	Capable of 66MHz operation.	I
PCIXParHigh	Even parity across PCIAD32:63 and PCIXC0:3[BE4:7].	I/O
PCIXParLow	Even parity across PCIAD0:31 and PCIXC0:3[BE0:3].	I/O
PCIXPErr	Reports data parity errors during all PCI transactions except a Special Cycle.	I/O
PCIXReq0	An indication to the PCI-X arbiter that the specified agent wishes to use the bus. When using an external PCI/PCI-X arbiter, connect the external arbiter's Request line to this signal.	I/O
PCIXReq1:5	An indication to the PCI-X arbiter that the specified agent wishes to use the bus.	I
PCIXReq64	Asserted by the current bus master, indicating a 64-bit transfer.	I/O
PCIXAck64	Indicates the target can transfer data using 64 bits.	I/O
PCIXReset	Brings PCI device registers and logic to a consistent state.	O
PCIXSErr	Reports address parity errors, data parity errors on the Special Cycle command, or other catastrophic system errors.	I/O
PCIXStop	Indicates the current target is requesting the master to stop the current transaction.	I/O
PCIXTRDY	Indicates the target agent's ability to complete the current data phase of the transaction.	I/O
PerAddr00:31	Peripheral address bus used by PPC440GX when not in external master mode, otherwise used by external master. Note: PerAddr00 is the most significant bit (msb) on this bus.	I/O
PerBE0:3	External peripheral data bus byte enables.	I/O
PerBLast	Used by either the peripheral controller, DMA controller, or external master to indicates the last transfer of a memory access.	I/O
PerClk	Peripheral Clock. Used by an external master and by synchronous peripheral slaves.	O
PerCS0:7	External peripheral device select.	O
PerData00:31	Peripheral data bus used by PPC440GX when not in external master mode, otherwise used by external master. Note: PerData00 is the most significant bit (msb) on this bus.	I/O
PerErr	External Error. Used as an input used to record external master errors and external slave peripheral errors.	I/O

Preliminary User's Manual

Table 33-1. Signals Listed Alphabetically (Continued)

Signal Name	Description	I/O
$\overline{\text{PerOE}}$	Used by either peripheral controller or DMA controller depending upon the type of transfer involved. When the PPC440GX is the bus master, it enables the selected DDR SDRAMs to drive the bus.	O
PerPar0:3	External peripheral data bus byte parity.	I/O
PerReady	Used by a peripheral slave to indicate it is ready to transfer data.	I
$\overline{\text{PerR}/\overline{\text{W}}}$	Used by the PPC440GX when not in external master mode, as output by either the peripheral controller or DMA controller depending upon the type of transfer involved. High indicates a read from memory, low indicates a write to memory. Otherwise, it used by the external master as an input to indicate the direction of transfer.	I/O
$\overline{\text{PerWE}}$	Write Enable. Low when any of the four $\overline{\text{PerBE0:3}}$ signals are low.	O
UARTSerClk	Serial clock input that provides an alternative to the internally generated serial clock. Used in cases where the allowable internally generated clock rates are not satisfactory. This input can be individually connected to either or both UART0 and UART1.	I
UART0_Rx	UART0 Receive data.	I
UART0_Tx	UART0 Transmit data.	O
$\overline{\text{UART0_DCD}}$	UART0 Data Carrier Detect.	I
$\overline{\text{UART0_DSR}}$	UART0 Data Set Ready.	I
$\overline{\text{UART0_CTS}}$	UART0 Clear To Send.	I
$\overline{\text{UART0_DTR}}$	UART0 Data Terminal Ready.	O
$\overline{\text{UART0_RTS}}$	UART0 Request To Send.	O
$\overline{\text{UART0_RI}}$	UART0 Ring Indicator.	I
UART1_Rx	UART1 Receive data.	I/O
UART1_Tx	UART1 Transmit data.	I/O
$\overline{\text{UART1_DSR/CTS}}$	UART1 Data Set Ready or Clear To Send. The choice is determined by a DCR register bit setting.	I/O
$\overline{\text{UART1_RTS/DTR}}$	UART1 Request To Send or Data Terminal Ready. The choice is determined by a DCR register bit setting.	I/O
$\overline{\text{RAS}}$	Row Address Strobe.	O
RcvrInh	Receiver Inhibit. Active only when TestEn is active.	I
RefVEn	Reference Voltage Enable. Used for wafer testing. Do not connect for normal operation.	I
SysClk	Main system clock input.	Clock
SysErr	Set to 1 when a machine check is generated.	O
$\overline{\text{SysReset}}$	Main system reset. External logic can drive this bidirectional pin low (minimum of 16 cycles) to initiate a system reset. A system reset can also be initiated by software. Implemented as an open-drain output (two states; 0 or open circuit).	I/O
SV _{DD}	2.5V supply—DDR SDRAM, Ethernet	n/a
TCK	Test Clock.	I
TDI	Test Data In.	I
TDO	Test Data Out.	O
TestEn	Test Enable.	I
TmrClk	Processor timer external input clock.	I

Table 33-1. Signals Listed Alphabetically (Continued)

Signal Name	Description	I/O
TMS	Test Mode Select.	I
TrcBS0:2	Trace branch execution status.	I/O
TrcClk	Trace data capture clock, runs at 1/4 the frequency of the processor.	O
TrcES0:4	Trace Execution Status is presented every fourth processor clock cycle.	I/O
TrcTS0	Additional information on trace execution and branch status.	I/O
TrcTS1:6 (muxed with GPIO)	Additional information on trace execution and branch status.	I/O
TrcTS1:6 (muxed with EBC)	Additional information on trace execution and branch status.	I/O
$\overline{\text{TRST}}$	Test Reset.	I
V _{DD}	1.5V supply—Logic voltage.	n/a
$\overline{\text{WE}}$	Write Enable.	O

Preliminary User's Manual**33.2 Signals by Interface Category**

Table 33-2 lists the PPC440GX Embedded Processor signals by interface category with a brief description. Multiplexed signals are shown in brackets following the first signal name assigned to each multiplexed ball (for example, IRQ0:6[GPI017:23]). Active-low signals are shown with an overbar on the signal name (for example, ExtAck).

Table 33-2. Signal Descriptions by Interface Category

Signal Name	Description	I/O
PCI-X Interface		
PCIXAD00:63	Address/Data bus (bidirectional).	I/O
PCIXC0:7[BE0:7]	PCI-X Command[Byte Enables].	I/O
PCIXCap	Capable of PCI-X operation.	I
PCIX133Cap	PCI-X devices are 133 MHz capable.	O
PCIXClk	Provides timing to the PCI interface for PCI transactions.	I
PCIXDevSel	Indicates the driving device has decoded its address as the target of the current access.	I/O
PCIXFrame	Driven by the current master to indicate beginning and duration of an access.	I/O
PCIXGnt0	Indicates that the specified agent is granted access to the bus. When using an external PCI/PCI-X arbiter, connect the external arbiter's Grant line to this signal.	I/O
PCIXGnt1	Indicates that the specified agent is granted access to the bus.	I/O
PCIXGnt2:5	Indicates that the specified agent is granted access to the bus.	O
PCIXIDSel	Used as a chip select during configuration read and write transactions.	I
PCIXINT	Level sensitive PCI interrupt.	O
PCIXIRDY	Indicates initiating agent's ability to complete the current data phase of the transaction.	I/O
PCIXM66En	Capable of 66MHz operation.	I
PCIXParHigh	Even parity across PCIAD32:63 and PCIXC0:3[BE4:7].	I/O
PCIXParLow	Even parity across PCIAD0:31 and PCIXC0:3[BE0:3].	I/O
PCIXPErr	Reports data parity errors during all PCI transactions except a Special Cycle.	I/O
PCIXReq0	An indication to the PCI-X arbiter that the specified agent wishes to use the bus. When using an external PCI/PCI-X arbiter, connect the external arbiter's Request line to this signal.	I/O
PCIXReq1:5	An indication to the PCI-X arbiter that the specified agent wishes to use the bus.	I
PCIXReq64	Asserted by the current bus master, indicating a 64-bit transfer.	I/O
PCIXAck64	Indicates the target can transfer data using 64 bits.	I/O
PCIXReset	Brings PCI device registers and logic to a consistent state.	O
PCIXSErr	Reports address parity errors, data parity errors on the Special Cycle command, or other catastrophic system errors.	I/O
PCIXStop	Indicates the current target is requesting the master to stop the current transaction.	I/O
PCIXTRDY	Indicates the target agent's ability to complete the current data phase of the transaction.	I/O
DDR SDRAM Interface		
BA0:1	Bank Address supporting up to four internal banks.	O
BankSel0:3	Selects up to four external DDR SDRAM banks.	O
CAS	Column Address Strobe.	O
ClkEn0:3	Clock Enable. One for each bank.	O

Table 33-2. Signal Descriptions by Interface Category (Continued)

Signal Name	Description	I/O
DM0:8	Memory write data byte lane masks. MEMDM8 is the byte lane mask for the ECC byte lane.	O
DQS0:8	Byte lane data strobe. DQS8 is the data strobe for the ECC byte lane.	I/O
ECC0:7	ECC check bits 0:7.	I/O
MemAddr00:12	Memory address bus.	O
MemClkOut0 MemClkOut0	Subsystem clock.	O
MemData00:63	Memory data bus.	I/O
MemVRef1:2	Memory reference voltage (SV _{REF}) input.	I
$\overline{\text{RAS}}$	Row Address Strobe.	O
$\overline{\text{WE}}$	Write Enable.	O
Ethernet Interface		
EMCCD, EMC1RxErr, GMCTxCIk, GMC0TxClk, TBITxCIk, RTBI0TxClk	MII: Collision detection RMII 1: Receive error GMII: Transmit clock RGMII: Transmit clock TBI: Transmit clock RTBI: Transmit clock	I/O
EMCCrS, EMC0CrSDV, GMCTxD7, GMC1TxD3, TBITxD7, RTBI1TxD3	MII: Carrier sense RMII 0: Carrier sense data valid GMII: Transmit data RGMII 1: Transmit data TBI: Transmit data RTBI 1: Transmit data	I/O
EMCMDClk	MII and RMII: Management data clock	O
EMCMDIO	MII and RMII: Transfer command and status information between MII and PHY	I/O
EMCRxD0:3, EMC0RxD0:1, EMC1RxD0:1, EMC0RxD, EMC1RxD, EMC2RxD, EMC3RxD, GMCTxD0:1, GMC0TxD0:1, TBITxD0:1, RTBI0TxD0:1	MII: Receive data RMII 0: Receive data RMII 1: Receive data SMII 0: Receive data SMII 1: Receive data SMII 2: Receive data SMII 3: Receive data GMII: Transmit data RGMII 0: Transmit data TBI: Transmit data RTBI 0: Transmit data	I/O
EMCRxDV, EMC1CrSDV, GMCTxD4, GMC1TxD0, TBITxD4, RTBI1TxD0	MII: Receive data valid RMII 1: Carrier sense data valid GMII: Transmit data RGMII 1: Transmit data TBI: Transmit data RTBI 1: Transmit data	I/O
EMCRxCIk, GMCTxD5, GMC1TxD1, TBITxD5, RTBI1TxD1	MII: Receive clock GMII: Transmit data RGMII 1: Transmit data TBI: Transmit data RTBI 1: Transmit data	I/O
EMCRxErr, EMC0RxErr, GMCTxD6, GMC1TxD2, TBITxD6, RTBI1TxD2	MII: Receive error RMII 0: Receive error GMII: Transmit data RGMII 1: Transmit data TBI: Transmit data RTBI 1: Transmit data	I/O

Preliminary User's Manual

Table 33-2. Signal Descriptions by Interface Category (Continued)

Signal Name	Description	I/O
EMCTxCk, EMCRefCk	MII: Transmit clock RMII and SMII: Transmit clock	I
EMCTxD0:3, EMC0TxD0:1, EMC1TxD0:1, EMC0TxD, EMC1TxD, EMC2TxD, EMC3TxD, GMCTxD2:3, GMC0TxD2:3, TBITxD2:3, RTBI0TxD2:3	MII: Transmit data RMII 0: Transmit data RMII 1: Transmit data SMII 0: Transmit data SMII 1: Transmit data SMII 2: Transmit data SMII 3: Transmit data GMII: Transmit data RGMII 0: Transmit data TBI: Transmit data RTBI 0 : Transmit data	O
EMCTxEn, EMC0TxEn, EMCSync	MII: Transmit data enabled RMII 0: Transmit data enabled SMII: Sync signal	O
EMCTxErr, EMC1TxEn, GMCRxCk, GMC0RxClk, TBIRxCk0, RTBI0RxClk	MII: Transmit error: RMII : Transmit data enabled GMII: Receive clock RGMII: Receive clock TBI: Receive clock RTBI: Receive clock	I/O
GMCCD, GMC1RxClk, TBIRxCk, RTBI1RxClk	GMII: Collision detection RGMII: Receive clock TBI: Receive clock RTBI: Receive clock	I
GMCCrS, GMC1TxClk, RTBI1TxClk	GMII: Carrier sense RGMII: Transmit clock RTBI: Transmit clock	I/O
GMCRxCk	GMII, RGMII, TBI and RTBI: Gigabit reference clock	I
GMCRxD0:3, GMC0RxD0:3, TBIRxD0:3, RTBI0RxD0:3	GMII: Receive data RGMII: Receive data TBI: Receive data RTBI: Receive data	I
GMCRxD4:7, GMC1RxD0:3, TBIRxD4:7, RTBI1RxD0:3	GMII: Receive data RGMII: Receive data TBI: Receive data RTBI: Receive data	I
GMCRxDV, GMC0RxCtl, TBIRxD8, RTBI0RxCtl	GMII: Receive data valid RGMII: Receive control TBI: Receive data RTBI: Receive control	I
GMCRxEr, GMC1RxCtl, TBIRxD9, RTBI1RxCtl	GMII: Receive error RGMII: Receive control TBI: Receive data RTBI: Receive control	I
GMCTxEn, GMC0TxCtl, TBITxD8, RTBI0TxCtl	GMII: Transmit data enable RGMII: Transmit control TBI: Transmit data RTBI: Transmit control	O
GMCTxEr, GMC1TxCtl, TBITxD9, RTBI1TxCtl	GMII: Transmit error RGMII: Transmit control TBI: Transmit data RTBI: Transmit control	O
External Slave Peripheral Interface		
DMAAck0:3	Used by the PPC440GX to indicate that data transfers have occurred.	O
DMAReq0:3	Used by slave peripherals to indicate they are prepared to transfer data.	I

Table 33-2. Signal Descriptions by Interface Category (Continued)

Signal Name	Description	I/O
EOT0:3/TC0:3	End Of Transfer/Terminal Count.	I/O
PerAddr00:31	Peripheral address bus used by PPC440GX when not in external master mode, otherwise used by external master. Note: PerAddr00 is the most significant bit (msb) on this bus.	I/O
PerBE0:3	External peripheral data bus byte enables.	I/O
PerBLast	Used by either the peripheral controller, DMA controller, or external master to indicates the last transfer of a memory access.	I/O
PerCS0:7	External peripheral device select.	O
PerData00:31	Peripheral data bus used by PPC440GX when not in external master mode, otherwise used by external master. Note: PerData00 is the most significant bit (msb) on this bus.	I/O
PerOE	Used by either peripheral controller or DMA controller depending upon the type of transfer involved. When the PPC440GX is the bus master, it enables the selected DDR SDRAMs to drive the bus.	O
PerPar0:3	External peripheral data bus byte parity.	I/O
PerReady	Used by a peripheral slave to indicate it is ready to transfer data.	I
PerR/W	Used by the PPC440GX when not in external master mode, as output by either the peripheral controller or DMA controller depending upon the type of transfer involved. High indicates a read from memory, low indicates a write to memory. Otherwise, it used by the external master as an input to indicate the direction of transfer.	I/O
PerWE	Write Enable. Low when any of the four PerBE0:3 signals are low.	O
External Master Peripheral Interface		
BusReq	Bus Request. Used when the PPC440GX needs to regain control of peripheral interface from an external master.	O
ExtAck	External Acknowledgement. Used by the PPC440GX to indicate that a data transfer occurred.	O
ExtReq	External Request. Used by an external master to indicate it is prepared to transfer data.	I
ExtReset	Peripheral Reset. Used by an external master and by synchronous peripheral slaves.	O
HoldAck	Hold Acknowledge. Used by the PPC440GX to transfer ownership of peripheral bus to an external master.	O
HoldReq	Hold Request. Used by an external master to request ownership of the peripheral bus.	I
PerClk	Peripheral Clock. Used by an external master and by synchronous peripheral slaves.	O
PerErr	External Error. Used as an input used to record external master errors and external slave peripheral errors.	I/O
UART Peripheral Interface		
UARTSerClk	Serial clock input that provides an alternative to the internally generated serial clock. Used in cases where the allowable internally generated clock rates are not satisfactory. This input can be individually connected to either or both UART0 and UART1.	I
UART0_Rx	UART0 Receive data.	I
UART0_Tx	UART0 Transmit data.	O
UART0_DCD	UART0 Data Carrier Detect.	I
UART0_DSR	UART0 Data Set Ready.	I
UART0_CTS	UART0 Clear To Send.	I
UART0_DTR	UART0 Data Terminal Ready.	O
UART0_RTS	UART0 Request To Send.	O

Preliminary User's Manual

Table 33-2. Signal Descriptions by Interface Category (Continued)

Signal Name	Description	I/O
UART0_RI	UART0 Ring Indicator.	I
UART1_Rx	UART1 Receive data.	I/O
UART1_Tx	UART1 Transmit data.	I/O
UART1_DSR/CTS	UART1 Data Set Ready or Clear To Send. The choice is determined by a DCR register bit setting.	I/O
UART1_RTS/DTR	UART1 Request To Send or Data Terminal Ready. The choice is determined by a DCR register bit setting.	I/O
IIC Peripheral Interface		
IIC0SClk	IIC0 Serial Clock.	I/O
IIC0SDA	IIC0 Serial Data.	I/O
IIC1SClk	IIC1 Serial Clock.	I/O
IIC1SDA	IIC1 Serial Data.	I/O
Interrupts Interface		
IRQ00:10	External interrupt Requests 0 through 10.	I
IRQ11:12	External interrupt Requests 11 through 12.	I
IRQ13:17	External interrupt Requests 13 through 17.	I
JTAG Interface		
TCK	Test Clock.	I
TDI	Test Data In.	I
TDO	Test Data Out.	O
TMS	Test Mode Select.	I
TRST	Test Reset.	I
System Interface		
SysClk	Main system clock input.	Clock
SysErr	Set to 1 when a machine check is generated.	O
SysReset	Main system reset. External logic can drive this bidirectional pin low (minimum of 16 cycles) to initiate a system reset. A system reset can also be initiated by software. Implemented as an open-drain output (two states; 0 or open circuit).	I/O
TmrClk	Processor timer external input clock.	I
Halt	Halt from external debugger.	I
GPIO00:31	General purpose I/O 0 through 10. To access these functions, software must set DCR register bits.	I/O
TestEn	Test Enable.	I
RcvrInh	Receiver Inhibit. Active only when TestEn is active.	I
RefVEn	Reference Voltage Enable. Used for wafer testing. Do not connect for normal operation.	I
DrvrInh2	Driver Inhibit. Used for test purposes only. Tie up for normal operation	I

Table 33-2. Signal Descriptions by Interface Category (Continued)

Signal Name	Description	I/O
Trace Interface		
TrcBS0:2	Trace branch execution status.	I/O
TrcClk	Trace data capture clock, runs at 1/4 the frequency of the processor.	O
TrcES0:4	Trace Execution Status is presented every fourth processor clock cycle.	I/O
TrcTS0	Additional information on trace execution and branch status.	I/O
TrcTS1:6 (muxed with GPIO)	Additional information on trace execution and branch status.	I/O
TrcTS1:6 (muxed with EBC)	Additional information on trace execution and branch status.	I/O
Power Pins		
AGND	PLL (analog) voltage ground.	n/a
GND	Ground.	n/a
AxV _{DD}	1.5V—Filtered voltages input for PLLs (analog circuits) Note: A separate filter for each of the three voltages is recommended.	n/a
OV _{DD}	3.3V supply—I/O (except DDR SDRAM, Ethernet)	n/a
SV _{DD}	2.5V supply—DDR SDRAM, Ethernet	n/a
V _{DD}	1.5V supply—Logic voltage.	n/a

Preliminary User's Manual

Appendix A. Instruction Summary

This appendix describes the various instruction formats, and lists all of the PPC440GX Embedded Processor instructions summarized alphabetically and by opcode.

Appendix A.1 on page 1783 illustrates the PPC440GX Embedded Processor instruction forms (allowed arrangements of fields within instructions).

Appendix A.2 on page 1789 lists all PPC440GX Embedded Processor mnemonics, including extended mnemonics. A short functional description is included for each mnemonic.

Appendix A.3 on page 1818 identifies those opcodes which are allocated by PowerPC Book-E for implementation-dependent usage, including auxiliary processors.

Appendix A.4 on page 1818 identifies those opcodes which are identified by PowerPC Book-E as “preserved” for compatibility with previous versions of the architecture.

Appendix A.5 on page 1819 identifies those opcodes which are “reserved” for use by future versions of the architecture.

Appendix A.6 on page 1819, lists all instructions implemented within the PPC440GX, sorted by primary and secondary opcodes. Extended mnemonics are not included in the opcode list, but allocated, preserved, and reserved-nop opcodes are included.

A.1 Instruction Formats

Instructions are four bytes long. Instruction addresses are always word-aligned.

Instruction bits 0 through 5 always contain the primary opcode. Many instructions have an extended opcode in another field. Remaining instruction bits contain additional fields. All instruction fields belong to one of the following categories:

- Defined

These instructions contain values, such as opcodes, that cannot be altered. The instruction format diagrams specify the values of defined fields.

- Variable

These fields contain operands, such as GPR selectors and immediate values, that can vary from execution to execution. The instruction format diagrams specify the operands in the variable fields.

- Reserved

Bits in reserved fields should be set to 0. In the instruction format diagrams, /, //, or /// indicate reserved fields.

If any bit in a defined field does not contain the expected value, the instruction is illegal and an illegal instruction exception occurs. If any bit in a reserved field does not contain 0, the instruction form is invalid; its result is architecturally undefined. The PPC440GX executes all invalid instruction forms without causing an illegal instruction exception.

A.1.1 Instruction Fields

PPC440GX Embedded Processor instructions contain various combinations of the following fields, as indicated in the instruction format diagrams that follow the field definitions. Numbers, enclosed in parentheses, that follow the field names indicate bit positions; bit fields are indicated by starting and stopping bit positions separated by colons.

AA (30)	<p>Absolute address bit.</p> <p>0 The immediate field represents an address relative to the current instruction address (CIA). The effective address (EA) of the branch is either the sum of the LI field sign-extended to 32 bits and the branch instruction address, or the sum of the BD field sign-extended to 32 bits and the branch instruction address.</p> <p>1 The immediate field represents an absolute address. The EA of the branch is either the LI field or the BD field, sign-extended to 32 bits.</p>
BA (11:15)	Specifies a bit in the CR used as a source of a CR-logical instruction.
BB (16:20)	Specifies a bit in the CR used as a source of a CR-logical instruction.
BD (16:29)	An immediate field specifying a 14-bit signed twos complement branch displacement. This field is concatenated on the right with 0b00 and sign-extended to 32 bits.
BF (6:8)	Specifies a field in the CR used as a target in a compare or mcrf instruction.
BFA (11:13)	Specifies a field in the CR used as a source in a mcrf instruction.
BI (11:15)	Specifies a bit in the CR used as a source for the condition of a conditional branch instruction.
BO (6:10)	Specifies options for conditional branch instructions. See <i>Branch Instruction BO Field</i> on page 180.
BT (6:10)	Specifies a bit in the CR used as a target as the result of a CR-Logical instruction.
D (16:31)	Specifies a 16-bit signed two's-complement integer displacement for load/store instructions.
DCRF (11:20)	Specifies a device control register (DCR). This field represents the DCR Number (DCRN) with the upper and lower five bits reversed (that is, DCRF = DCRN[5:9] DCRN[0:4]).
FXM (12:19)	Field mask used to identify CR fields to be updated by the mtcrf instruction.
IM (16:31)	An immediate field used to specify a 16-bit value (either signed integer or unsigned).
LI (6:29)	An immediate field specifying a 24-bit signed twos complement branch displacement; this field is concatenated on the right with b'00' and sign-extended to 32 bits.
LK (31)	<p>Link bit.</p> <p>0 Do not update the link register (LR).</p> <p>1 Update the LR with the address of the next instruction.</p>
MB (21:25)	<p>Mask begin.</p> <p>Used in rotate-and-mask instructions to specify the beginning bit of a mask.</p>
ME (26:30)	<p>Mask end.</p> <p>Used in rotate-and-mask instructions to specify the ending bit of a mask.</p>
MO (6:10)	<p>Memory Ordering.</p> <p>Provides a storage ordering function for storage accesses executing prior to an mbar instruction. MO is ignored and treated as 0 in the PPC440GX CPU core.</p>
NB (16:20)	Specifies the number of bytes to move in an immediate string load or store.
OPCD (0:5)	Primary opcode. Primary opcodes, in decimal, appear in the instruction format diagrams presented with individual instructions. The OPCODE field name does not appear in instruction descriptions.
OE (21)	Enables setting the OV and SO fields in the fixed-point exception register (XER) for extended arithmetic.
RA (11:15)	A GPR used as a source or target.
RB (16:20)	A GPR used as a source.

Preliminary User's Manual

Rc (31)	Record bit. 0 Do not set the CR. 1 Set the CR to reflect the result of an operation. See <i>Condition Register (CR)</i> on page 183 for a further discussion of how the CR bits are set.
RS (6:10)	A GPR used as a source.
RT (6:10)	A GPR used as a target.
SH (16:20)	Specifies a shift amount.
SPRF (11:20)	Specifies a special purpose register (SPR). This field represents the SPR Number (SPRN) with the upper and lower five bits reversed (that is, SPRF = SPRN[5:9] SPRN[0:4]).
TO (6:10)	Specifies the conditions on which to trap, as described under tw and twi instructions.
WS (16:20)	Specifies the portion of a TLB entry to be read/written by tlbre/tlbwe .
XO (21:30)	Extended opcode for instructions without an OE field. Extended opcodes, in decimal, appear in the instruction format diagrams presented with individual instructions. The XO field name does not appear in instruction descriptions.
XO (22:30)	Extended opcode for instructions with an OE field. Extended opcodes, in decimal, appear in the instruction format diagrams presented with individual instructions. The XO field name does not appear in instruction descriptions.

A.1.2 Instruction Format Diagrams

The instruction formats (also called “forms”) illustrated in Figure A-1 through Figure A-9 are valid combinations of instruction fields. *Table A-5* on page 1820 indicates which “form” is utilized by each PPC440GX Embedded Processor opcode. Fields indicated by slashes (/, //, or ///) are reserved. The figures are adapted from the PowerPC User Instruction Set Architecture.

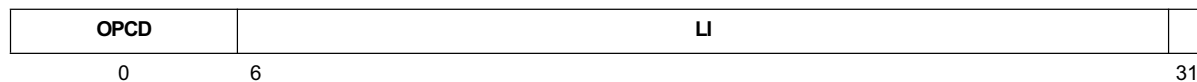
A.1.2.1 I-Form

Figure A-1. I Instruction Format

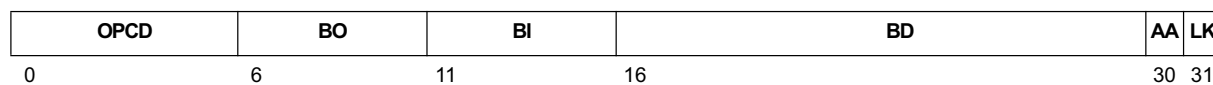
A.1.2.2 B-Form

Figure A-2. B Instruction Format

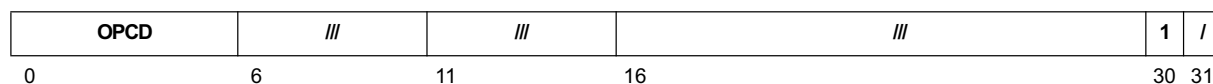
A.1.2.3 SC-Form

Figure A-3. SC Instruction Format

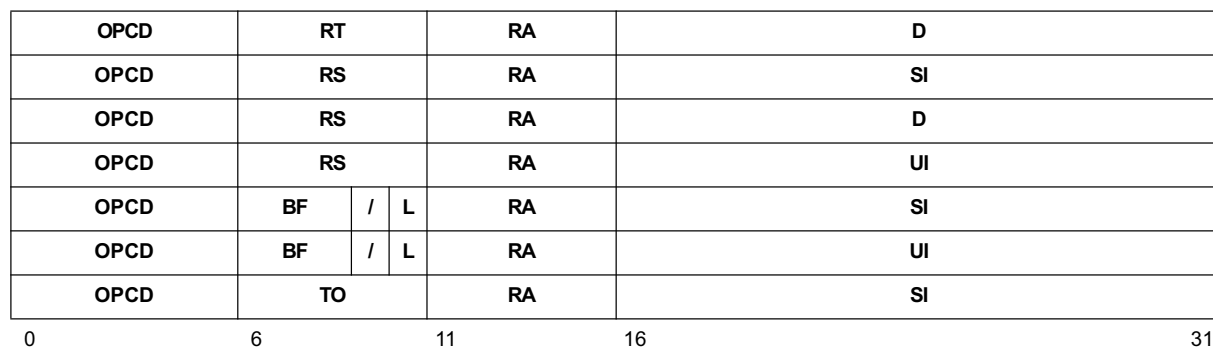
A.1.2.4 D-Form

Figure A-4. D Instruction Format

Preliminary User's Manual**A.1.2.5 X-Form**

OPCD	RT	RA	RB	XO	Rc
OPCD	RT	RA	RB	XO	/
OPCD	RT	RA	NB	XO	/
OPCD	RT	RA	WS	XO	/
OPCD	RT	///	RB	XO	/
OPCD	RT	///	///	XO	/
OPCD	RS	RA	RB	XO	Rc
OPCD	RS	RA	RB	XO	1
OPCD	RS	RA	RB	XO	/
OPCD	RS	RA	NB	XO	/
OPCD	RS	RA	WS	XO	/
OPCD	RS	RA	SH	XO	Rc
OPCD	RS	RA	///	XO	Rc
OPCD	RS	///	RB	XO	/
OPCD	RS	///	///	XO	/
OPCD	BF	/ L	RA	RB	XO
OPCD	BF	//	BFA //	///	XO
OPCD	BF	//	///	///	XO
OPCD	BF	//	///	U	XO
OPCD	BF	//	///	///	XO
OPCD	TO	RA	RB	XO	/
OPCD	BT	///	///	XO	Rc
OPCD	MO	///	///	XO	/
OPCD	///	RA	RB	XO	/
OPCD	///	///	///	XO	/
OPCD	///	///	E //	XO	/
0	6	11	16	21	31

Figure A-5. X Instruction Format

A.1.2.6 XL-Form

OPCD	BT		BA		BB	XO	/
OPCD	BC		BI		///	XO	LK
OPCD	BF	//	BFA	//	///	XO	/
OPCD	///		///		///	XO	/
0	6		11		16	21	31

Figure A-6. XL Instruction Format

A.1.2.7 XFX-Form

OPCD	RT	SPRF				XO	/
OPCD	RT	DCRF				XO	/
OPCD	RT	/	FXM			/	/
OPCD	RS	SPRF				XO	/
OPCD	RS	DCRF				XO	/
0	6	11	16	21			31

Figure A-7. XFX Instruction Format

A.1.2.8 XO-Form

OPCD	RT	RA	RB	OE	XO	Rc
OPCD	RT	RA	RB	OE	XO	Rc
OPCD	RT	RA	///	/	XO	Rc
0	6	11	16	21	22	31

Figure A-8. XO Instruction Format

A.1.2.9 M-Form

OPCD	RS	RA	RB	MB	ME	Rc
OPCD	RS	RA	SH	MB	ME	Rc
0	6	11	16	21	26	31

Figure A-9. M Instruction Format

Preliminary User's Manual**A.2 Alphabetical Summary of Implemented Instructions**

Table A-1 summarizes the PPC440GX Embedded Processor instruction set, including required extended mnemonics. All mnemonics are listed alphabetically, without regard to whether the mnemonic is realized in hardware or software. When an instruction supports multiple hardware mnemonics (for example, **b**, **ba**, **bl**, **bla** are all forms of **b**), the instruction is alphabetized under the root form. The hardware instructions are described in detail in Chapter 31, "Instruction Set," which is also alphabetized under the root form. Section 31 also describes the instruction operands and notation.

Programming Note: Bit 4 of the BO instruction field provides a hint about the most likely outcome of a conditional branch. (See *Branch Prediction* on page 181 for a detailed description of branch prediction.) Assemblers should set $BO_4 = 0$ unless a specific reason exists otherwise. In the BO field values specified in Table A-1, $BO_4 = 0$ has always been assumed. The assembler must enable the programmer to specify branch prediction. To do this, the assembler supports suffixes for the conditional branch mnemonics:

+ Predict branch to be taken.

– Predict branch not to be taken.

For example, **bc** also could be coded as **bc+** or **bc–**, and **bne** also could be coded **bne+** or **bne–**. These alternate codings set $BO_4 = 1$ only if the requested prediction differs from the standard prediction. See *Branch Prediction* on page 181 for more information.

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary

Mnemonic	Operands	Function	Other Registers Changed	Page
add	RT, RA, RB	Add (RA) to (RB). Place result in RT.		1024
add.			CR[CR0]	
addo			XER[SO, OV]	
addo.			CR[CR0] XER[SO, OV]	
addc	RT, RA, RB	Add (RA) to (RB). Place result in RT. Place carry-out in XER[CA].		1024
addc.			CR[CR0]	
addco			XER[SO, OV]	
addco.			CR[CR0] XER[SO, OV]	
adde	RT, RA, RB	Add XER[CA], (RA), (RB). Place result in RT. Place carry-out in XER[CA].		1025
adde.			CR[CR0]	
addeo			XER[SO, OV]	
addeo.			CR[CR0] XER[SO, OV]	
addi	RT, RA, IM	Add EXTS(IM) to (RA[0]). Place result in RT.		1027
addic	RT, RA, IM	Add EXTS(IM) to (RA[0]). Place result in RT. Place carry-out in XER[CA].		1028
addic.	RT, RA, IM	Add EXTS(IM) to (RA[0]). Place result in RT. Place carry-out in XER[CA].	CR[CR0]	1029
addis	RT, RA, IM	Add (IM \parallel $^{16}0$) to (RA[0]). Place result in RT.		1030

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
addme	RT, RA	Add XER[CA], (RA), (-1). Place result in RT. Place carry-out in XER[CA].		1031
addme.			CR[CR0]	
addmeo			XER[SO, OV]	
addmeo.			CR[CR0] XER[SO, OV]	
addze	RT, RA	Add XER[CA] to (RA). Place result in RT. Place carry-out in XER[CA].		1032
addze.			CR[CR0]	
addzeo			XER[SO, OV]	
addzeo.			CR[CR0] XER[SO, OV]	
and	RA, RS, RB	AND (RS) with (RB). Place result in RA.		1033
and.			CR[CR0]	
andc	RA, RS, RB	AND (RS) with \neg (RB). Place result in RA.		1034
andc.			CR[CR0]	
andi.	RA, RS, IM	AND (RS) with ($^{16}0 \parallel$ IM). Place result in RA.	CR[CR0]	1035
andis.	RA, RS, IM	AND (RS) with (IM \parallel $^{16}0$). Place result in RA.	CR[CR0]	1036
b	target	Branch unconditional relative. $LI \leftarrow (target - CIA)_{6:29}$ $NIA \leftarrow CIA + EXTS(LI \parallel ^20)$		1037
ba		Branch unconditional absolute. $LI \leftarrow target_{6:29}$ $NIA \leftarrow EXTS(LI \parallel ^20)$		
bl		Branch unconditional relative. $LI \leftarrow (target - CIA)_{6:29}$ $NIA \leftarrow CIA + EXTS(LI \parallel ^20)$	(LR) $\leftarrow CIA + 4$	
bla		Branch unconditional absolute. $LI \leftarrow target_{6:29}$ $NIA \leftarrow EXTS(LI \parallel ^20)$	(LR) $\leftarrow CIA + 4$	
bc	BO, BI, target	Branch conditional relative. $BD \leftarrow (target - CIA)_{16:29}$ $NIA \leftarrow CIA + EXTS(BD \parallel ^20)$	CTR if $BO_2 = 0$	1038
bca		Branch conditional absolute. $BD \leftarrow target_{16:29}$ $NIA \leftarrow EXTS(BD \parallel ^20)$	CTR if $BO_2 = 0$	
bcl		Branch conditional relative. $BD \leftarrow (target - CIA)_{16:29}$ $NIA \leftarrow CIA + EXTS(BD \parallel ^20)$	CTR if $BO_2 = 0$ (LR) $\leftarrow CIA + 4$	
bcla		Branch conditional absolute. $BD \leftarrow target_{16:29}$ $NIA \leftarrow EXTS(BD \parallel ^20)$	CTR if $BO_2 = 0$ (LR) $\leftarrow CIA + 4$	
bcctr	BO, BI	Branch conditional to address in CTR. Using (CTR) at exit from instruction, $NIA \leftarrow CTR_{0:29} \parallel ^20$	CTR if $BO_2 = 0$	1044
bcctrl			CTR if $BO_2 = 0$ (LR) $\leftarrow CIA + 4$	
bclr	BO, BI	Branch conditional to address in LR. Using (LR) at entry to instruction, $NIA \leftarrow LR_{0:29} \parallel ^20$	CTR if $BO_2 = 0$	1047
bclrl			CTR if $BO_2 = 0$ (LR) $\leftarrow CIA + 4$	

Preliminary User's Manual

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bctr		Branch unconditionally to address in CTR. <i>Extended mnemonic for</i> bcctr 20,0		1044
bctrl		<i>Extended mnemonic for</i> bcctrl 20,0	(LR) ← CIA + 4	
bdnz	target	Decrement CTR. Branch if CTR ≠ 0 <i>Extended mnemonic for</i> bc 16,0,target		1038
bdnza		<i>Extended mnemonic for</i> bca 16,0,target		
bdnzl		<i>Extended mnemonic for</i> bcl 16,0,target	(LR) ← CIA + 4	
bdnzla		<i>Extended mnemonic for</i> bcla 16,0,target	(LR) ← CIA + 4	
bdnzlrl		Decrement CTR. Branch if CTR ≠ 0 to address in LR. <i>Extended mnemonic for</i> bclrl 16,0		1047
bdnzlrl		<i>Extended mnemonic for</i> bclrl 16,0	(LR) ← CIA + 4	
bdnzf	cr_bit, target	Decrement CTR. Branch if CTR ≠ 0 AND CR _{cr_bit} = 0 <i>Extended mnemonic for</i> bc 0,cr_bit,target		1038
bdnzfa		<i>Extended mnemonic for</i> bca 0,cr_bit,target		
bdnzfl		<i>Extended mnemonic for</i> bcl 0,cr_bit,target	(LR) ← CIA + 4	
bdnzfla		<i>Extended mnemonic for</i> bcla 0,cr_bit,target	(LR) ← CIA + 4	
bdnzflr	cr_bit	Decrement CTR. Branch if CTR ≠ 0 AND CR _{cr_bit} = 0 to address in LR. <i>Extended mnemonic for</i> bclrl 0,cr_bit		1047
bdnzflrl		<i>Extended mnemonic for</i> bclrl 0,cr_bit	(LR) ← CIA + 4	
bdnztl	cr_bit, target	Decrement CTR. Branch if CTR ≠ 0 AND CR _{cr_bit} = 1. <i>Extended mnemonic for</i> bc 8,cr_bit,target		1038
bdnzta		<i>Extended mnemonic for</i> bca 8,cr_bit,target		
bdnztl		<i>Extended mnemonic for</i> bcl 8,cr_bit,target	(LR) ← CIA + 4	
bdnztla		<i>Extended mnemonic for</i> bcla 8,cr_bit,target	(LR) ← CIA + 4	
bdnztlr	cr_bit	Decrement CTR. Branch if CTR ≠ 0 AND CR _{cr_bit} = 1 to address in LR. <i>Extended mnemonic for</i> bclrl 8,cr_bit		1047
bdnztlrl		<i>Extended mnemonic for</i> bclrl 8,cr_bit	(LR) ← CIA + 4	

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bdz	target	Decrement CTR. Branch if CTR = 0 <i>Extended mnemonic for</i> bc 18,0,target		1038
bdza		<i>Extended mnemonic for</i> bca 18,0,target		
bdzl		<i>Extended mnemonic for</i> bcl 18,0,target	(LR) ← CIA + 4	
bdzla		<i>Extended mnemonic for</i> bcla 18,0,target	(LR) ← CIA + 4	
bdzlr		Decrement CTR. Branch if CTR = 0 to address in LR. <i>Extended mnemonic for</i> bclr 18,0		1047
bdzlrl		<i>Extended mnemonic for</i> bclrl 18,0	(LR) ← CIA + 4	
bdzf	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR _{cr_bit} = 0 <i>Extended mnemonic for</i> bc 2,cr_bit,target		1038
bdzfa		<i>Extended mnemonic for</i> bca 2,cr_bit,target		
bdzfl		<i>Extended mnemonic for</i> bcl 2,cr_bit,target	(LR) ← CIA + 4	
bdzfla		<i>Extended mnemonic for</i> bcla 2,cr_bit,target	(LR) ← CIA + 4	
bdzflr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR _{cr_bit} = 0 to address in LR. <i>Extended mnemonic for</i> bclr 2,cr_bit		1047
bdzflrl		<i>Extended mnemonic for</i> bclrl 2,cr_bit	(LR) ← CIA + 4	
bdzt	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR _{cr_bit} = 1. <i>Extended mnemonic for</i> bc 10,cr_bit,target		1038
bdzta		<i>Extended mnemonic for</i> bca 10,cr_bit,target		
bdztl		<i>Extended mnemonic for</i> bcl 10,cr_bit,target	(LR) ← CIA + 4	
bdztla		<i>Extended mnemonic for</i> bcla 10,cr_bit,target	(LR) ← CIA + 4	
bdztlr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR _{cr_bit} = 1 to address in LR. <i>Extended mnemonic for</i> bclr 10,cr_bit		1047
bdztlrl		<i>Extended mnemonic for</i> bclrl 10,cr_bit	(LR) ← CIA + 4	

Preliminary User's Manual

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
beq	[cr_field], target	Branch if equal. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bc 12,4*cr_field+2,target		1038
beqa		<i>Extended mnemonic for</i> bca 12,4*cr_field+2,target		
beql		<i>Extended mnemonic for</i> bcl 12,4*cr_field+2,target	(LR) ← CIA + 4	
beqla		<i>Extended mnemonic for</i> bcla 12,4*cr_field+2,target	(LR) ← CIA + 4	
beqctr	[cr_field]	Branch if equal to address in CTR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+2		1044
beqctrl		<i>Extended mnemonic for</i> bcctrl 12,4*cr_field+2	(LR) ← CIA + 4	
beqlr	[cr_field]	Branch if equal to address in LR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+2		1047
beqlrl		<i>Extended mnemonic for</i> bclrl 12,4*cr_field+2	(LR) ← CIA + 4	
bf	cr_bit, target	Branch if CR _{cr_bit} = 0. <i>Extended mnemonic for</i> bc 4,cr_bit,target		1038
bfa		<i>Extended mnemonic for</i> bca 4,cr_bit,target		
bfl		<i>Extended mnemonic for</i> bcl 4,cr_bit,target	(LR) ← CIA + 4	
bfla		<i>Extended mnemonic for</i> bcla 4,cr_bit,target	(LR) ← CIA + 4	
bfctr	cr_bit	Branch if CR _{cr_bit} = 0 to address in CTR. <i>Extended mnemonic for</i> bcctr 4,cr_bit		1044
bfctrl		<i>Extended mnemonic for</i> bcctrl 4,cr_bit	(LR) ← CIA + 4	
bflr	cr_bit	Branch if CR _{cr_bit} = 0 to address in LR. <i>Extended mnemonic for</i> bclr 4,cr_bit		1047
bflrl		<i>Extended mnemonic for</i> bclrl 4,cr_bit	(LR) ← CIA + 4	
bge	[cr_field], target	Branch if greater than or equal. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+0,target		1038
bgea		<i>Extended mnemonic for</i> bca 4,4*cr_field+0,target		
bgel		<i>Extended mnemonic for</i> bcl 4,4*cr_field+0,target	(LR) ← CIA + 4	
bgea		<i>Extended mnemonic for</i> bcla 4,4*cr_field+0,target	(LR) ← CIA + 4	

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bgectr	[cr_field]	Branch if greater than or equal to address in CTR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+0		1044
bgectrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+0	(LR) ← CIA + 4	
bgelr	[cr_field]	Branch if greater than or equal to address in LR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+0		1047
bgelrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+0	(LR) ← CIA + 4	
bgt	[cr_field], target	Branch if greater than. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bc 12,4*cr_field+1,target		1038
bgta		<i>Extended mnemonic for</i> bca 12,4*cr_field+1,target		
bgtl		<i>Extended mnemonic for</i> bcl 12,4*cr_field+1,target	(LR) ← CIA + 4	
bgtla		<i>Extended mnemonic for</i> bcla 12,4*cr_field+1,target	(LR) ← CIA + 4	
bgtctr	[cr_field]	Branch if greater than to address in CTR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+1		1044
bgtctrl		<i>Extended mnemonic for</i> bcctrl 12,4*cr_field+1	(LR) ← CIA + 4	
bgtlr	[cr_field]	Branch if greater than to address in LR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+1		1047
bgtlrl		<i>Extended mnemonic for</i> bclrl 12,4*cr_field+1	(LR) ← CIA + 4	
ble	[cr_field], target	Branch if less than or equal. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+1,target		1038
blea		<i>Extended mnemonic for</i> bca 4,4*cr_field+1,target		
blel		<i>Extended mnemonic for</i> bcl 4,4*cr_field+1,target	(LR) ← CIA + 4	
blela		<i>Extended mnemonic for</i> bcla 4,4*cr_field+1,target	(LR) ← CIA + 4	
blectr	[cr_field]	Branch if less than or equal to address in CTR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+1		1044
blectrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+1	(LR) ← CIA + 4	
blelr	[cr_field]	Branch if less than or equal to address in LR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+1		1047
blelrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+1	(LR) ← CIA + 4	

Preliminary User's Manual

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
blr		Branch unconditionally to address in LR. <i>Extended mnemonic for</i> bclr 20,0		1047
blrl		<i>Extended mnemonic for</i> bclrl 20,0	(LR) ← CIA + 4	
blt	[cr_field], target	Branch if less than. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bc 12,4*cr_field+0,target		1038
blta		<i>Extended mnemonic for</i> bca 12,4*cr_field+0,target		
bltl		<i>Extended mnemonic for</i> bcl 12,4*cr_field+0,target	(LR) ← CIA + 4	
bltla		<i>Extended mnemonic for</i> bcla 12,4*cr_field+0,target	(LR) ← CIA + 4	
bltctr	[cr_field]	Branch if less than to address in CTR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+0		1044
bltctrl		<i>Extended mnemonic for</i> bcctrl 12,4*cr_field+0	(LR) ← CIA + 4	
bltlr	[cr_field]	Branch if less than to address in LR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+0		1047
bltlrl		<i>Extended mnemonic for</i> bclrl 12,4*cr_field+0	(LR) ← CIA + 4	
bne	[cr_field], target	Branch if not equal. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+2,target		1038
bnea		<i>Extended mnemonic for</i> bca 4,4*cr_field+2,target		
bnel		<i>Extended mnemonic for</i> bcl 4,4*cr_field+2,target	(LR) ← CIA + 4	
bnela		<i>Extended mnemonic for</i> bcla 4,4*cr_field+2,target	(LR) ← CIA + 4	
bnctr	[cr_field]	Branch if not equal to address in CTR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+2		1044
bnctrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+2	(LR) ← CIA + 4	
bnelr	[cr_field]	Branch if not equal to address in LR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+2		1047
bnelrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+2	(LR) ← CIA + 4	

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bng	[cr_field], target	Branch if not greater than. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+1,target		1038
bnga		<i>Extended mnemonic for</i> bca 4,4*cr_field+1,target		
bngl		<i>Extended mnemonic for</i> bcl 4,4*cr_field+1,target	(LR) ← CIA + 4	
bngla		<i>Extended mnemonic for</i> bcla 4,4*cr_field+1,target	(LR) ← CIA + 4	
bngctr	[cr_field]	Branch if not greater than to address in CTR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+1		1044
bngctrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+1	(LR) ← CIA + 4	
bnglr	[cr_field]	Branch if not greater than to address in LR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+1		1047
bnglrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+1	(LR) ← CIA + 4	
bni	[cr_field], target	Branch if not less than. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+0,target		1038
bni		<i>Extended mnemonic for</i> bca 4,4*cr_field+0,target		
bnil		<i>Extended mnemonic for</i> bcl 4,4*cr_field+0,target	(LR) ← CIA + 4	
bnila		<i>Extended mnemonic for</i> bcla 4,4*cr_field+0,target	(LR) ← CIA + 4	
bnlctr	[cr_field]	Branch if not less than to address in CTR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+0		1044
bnlctrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+0	(LR) ← CIA + 4	
bnllr	[cr_field]	Branch if not less than to address in LR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+0		1047
bnllrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+0	(LR) ← CIA + 4	
bns	[cr_field], target	Branch if not summary overflow. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+3,target		1038
bnsa		<i>Extended mnemonic for</i> bca 4,4*cr_field+3,target		
bnsi		<i>Extended mnemonic for</i> bcl 4,4*cr_field+3,target	(LR) ← CIA + 4	
bnsia		<i>Extended mnemonic for</i> bcla 4,4*cr_field+3,target	(LR) ← CIA + 4	

Preliminary User's Manual

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bnsctr	[cr_field]	Branch if not summary overflow to address in CTR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+3		1044
bnsctrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+3	(LR) ← CIA + 4	
bnsldr	[cr_field]	Branch if not summary overflow to address in LR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+3		1047
bnsldr		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+3	(LR) ← CIA + 4	
bnu	[cr_field], target	Branch if not unordered. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+3,target		1038
bnu		<i>Extended mnemonic for</i> bca 4,4*cr_field+3,target		
bnul		<i>Extended mnemonic for</i> bcl 4,4*cr_field+3,target	(LR) ← CIA + 4	
bnula		<i>Extended mnemonic for</i> bcla 4,4*cr_field+3,target	(LR) ← CIA + 4	
bnuctr	[cr_field]	Branch if not unordered to address in CTR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+3		1044
bnuctrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+3	(LR) ← CIA + 4	
bnulr	[cr_field]	Branch if not unordered to address in LR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+3		1047
bnulrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+3	(LR) ← CIA + 4	
bso	[cr_field], target	Branch if summary overflow. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bc 12,4*cr_field+3,target		1038
bsoa		<i>Extended mnemonic for</i> bca 12,4*cr_field+3,target		
bsol		<i>Extended mnemonic for</i> bcl 12,4*cr_field+3,target	(LR) ← CIA + 4	
bsola		<i>Extended mnemonic for</i> bcla 12,4*cr_field+3,target	(LR) ← CIA + 4	
bsocctr	[cr_field]	Branch if summary overflow to address in CTR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+3		1044
bsoctrl		<i>Extended mnemonic for</i> bcctrl 12,4*cr_field+3	(LR) ← CIA + 4	

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bsolr	[cr_field]	Branch if summary overflow to address in LR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+3		1047
bsolrl		<i>Extended mnemonic for</i> bclrl 12,4*cr_field+3	(LR) ← CIA + 4	
bt	cr_bit, target	Branch if CR _{cr_bit} = 1. <i>Extended mnemonic for</i> bc 12,cr_bit,target		1038
bta		<i>Extended mnemonic for</i> bca 12,cr_bit,target		
btl		<i>Extended mnemonic for</i> bcl 12,cr_bit,target	(LR) ← CIA + 4	
btla		<i>Extended mnemonic for</i> bcla 12,cr_bit,target	(LR) ← CIA + 4	
btctr	cr_bit	Branch if CR _{cr_bit} = 1 to address in CTR. <i>Extended mnemonic for</i> bcctr 12,cr_bit		1044
btctrl		<i>Extended mnemonic for</i> bcctrl 12,cr_bit	(LR) ← CIA + 4	
btlr	cr_bit	Branch if CR _{cr_bit} = 1, to address in LR. <i>Extended mnemonic for</i> bclr 12,cr_bit		1047
btlrl		<i>Extended mnemonic for</i> bclrl 12,cr_bit	(LR) ← CIA + 4	
bun	[cr_field], target	Branch if unordered. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bc 12,4*cr_field+3,target		1038
buna		<i>Extended mnemonic for</i> bca 12,4*cr_field+3,target		
bunl		<i>Extended mnemonic for</i> bcl 12,4*cr_field+3,target	(LR) ← CIA + 4	
bunla		<i>Extended mnemonic for</i> bcla 12,4*cr_field+3,target	(LR) ← CIA + 4	
bunctr	[cr_field]	Branch if unordered to address in CTR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+3		1044
bunctrl		<i>Extended mnemonic for</i> bcctrl 12,4*cr_field+3	(LR) ← CIA + 4	
bunlr	[cr_field]	Branch if unordered, to address in LR. Use CR[CR0] if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+3		1047
bunlrl		<i>Extended mnemonic for</i> bclrl 12,4*cr_field+3	(LR) ← CIA + 4	
clrlwi	RA, RS, n	Clear left immediate. (n < 32) (RA) _{0:n-1} ← ⁿ 0 <i>Extended mnemonic for</i> rlwinm RA,RS,0,n,31		1168
clrlwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,0,n,31	CR[CR0]	

Preliminary User's Manual

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
clrlslwi	RA, RS, b, n	Clear left and shift left immediate. ($n \leq b < 32$) (RA) $_{b-n:31-n} \leftarrow$ (RS) $_{b:31}$ (RA) $_{32-n:31} \leftarrow$ n_0 (RA) $_{0:b-n-1} \leftarrow$ $b-n_0$ <i>Extended mnemonic for</i> rlwinm RA,RS,n,b-n,31-n		1168
clrlslwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,n,b-n,31-n	CR[CR0]	
clrrwi	RA, RS, n	Clear right immediate. ($n < 32$) (RA) $_{32-n:31} \leftarrow$ n_0 <i>Extended mnemonic for</i> rlwinm RA,RS,0,0,31-n		1168
clrrwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,0,0,31-n	CR[CR0]	
cmp	BF, 0, RA, RB	Compare (RA) to (RB), signed. Results in CR[CRn], where $n = BF$.		1051
cmpi	BF, 0, RA, IM	Compare (RA) to EXTS(IM), signed. Results in CR[CRn], where $n = BF$.		1052
cmpl	BF, 0, RA, RB	Compare (RA) to (RB), unsigned. Results in CR[CRn], where $n = BF$.		1053
cmpli	BF, 0, RA, IM	Compare (RA) to ($^{16}0 \parallel IM$), unsigned. Results in CR[CRn], where $n = BF$.		1054
cmplw	[BF,] RA, RB	Compare Logical Word. UseCR[CR0] if BF is omitted. <i>Extended mnemonic for</i> cmpl BF,0,RA,RB		1053
cmplwi	[BF,] RA, IM	Compare Logical Word Immediate. UseCR[CR0] if BF is omitted. <i>Extended mnemonic for</i> cmpli BF,0,RA,IM		1054
cmpw	[BF,] RA, RB	Compare Word. UseCR[CR0] if BF is omitted. <i>Extended mnemonic for</i> cmp BF,0,RA,RB		1051
cmpwi	[BF,] RA, IM	Compare Word Immediate. UseCR[CR0] if BF is omitted. <i>Extended mnemonic for</i> cmpi BF,0,RA,IM		1052
cntlzw	RA, RS	Count leading zeros in RS.		1055
cntlzw.		Place result in RA.	CR[CR0]	
crand	BT, BA, BB	AND bit (CR _{BA}) with (CR _{BB}). Place result in CR _{BT} .		1056
crandc	BT, BA, BB	AND bit (CR _{BA}) with \neg (CR _{BB}). Place result in CR _{BT} .		1057
crclr	bx	Condition register clear. <i>Extended mnemonic for</i> crxor bx,bx,bx		1063
creqv	BT, BA, BB	Equivalence of bit CR _{BA} with CR _{BB} . $CR_{BT} \leftarrow \neg(CR_{BA} \oplus CR_{BB})$		1058
crmove	bx, by	Condition register move. <i>Extended mnemonic for</i> cror bx,by,by		1061

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
crnand	BT, BA, BB	NAND bit (CR _{BA}) with (CR _{BB}). Place result in CR _{BT} .		1059
crnor	BT, BA, BB	NOR bit (CR _{BA}) with (CR _{BB}). Place result in CR _{BT} .		1060
crnot	bx, by	Condition register not. <i>Extended mnemonic for</i> crnor bx,by,by		1060
cror	BT, BA, BB	OR bit (CR _{BA}) with (CR _{BB}). Place result in CR _{BT} .		1061
crorc	BT, BA, BB	OR bit (CR _{BA}) with \neg (CR _{BB}). Place result in CR _{BT} .		1062
crset	bx	Condition register set. <i>Extended mnemonic for</i> creqv bx,bx,bx		1058
crxor	BT, BA, BB	XOR bit (CR _{BA}) with (CR _{BB}). Place result in CR _{BT} .		1063
dcba	RA, RB	Treated as a no-op.		1064
dcbf	RA, RB	Flush (store, then invalidate) the data cache block which contains the effective address (RA 0) + (RB).		1065
dcbi	RA, RB	Invalidate the data cache block which contains the effective address (RA 0) + (RB).		1066
dcbst	RA, RB	Store the data cache block which contains the effective address (RA 0) + (RB).		1067
dcbt	RA, RB	Load the data cache block which contains the effective address (RA 0) + (RB).		1068
dcbtst	RA, RB	Load the data cache block which contains the effective address (RA 0) + (RB).		1069
dcbz	RA, RB	Zero the data cache block which contains the effective address (RA 0) + (RB).		1070
dccci	RA, RB	Invalidate the data cache array.		1071
dcread	RT, RA, RB	Read tag and data information from the data cache line selected using effective address bits 17:26. The effective address is calculated by (RA 0) + (RB). Place the data word selected by effective address bits 27:29 in GPR RT; place the tag information in DCDBTRH and DCDBTRL.		1072
divw	RT, RA, RB	Divide (RA) by (RB), signed. Place result in RT.		1074
divw.			CR[CR0]	
divwo			XER[SO, OV]	
divwo.			CR[CR0] XER[SO, OV]	
divwu	RT, RA, RB	Divide (RA) by (RB), unsigned. Place result in RT.		1075
divwu.			CR[CR0]	
divwuo			XER[SO, OV]	
divwuo.			CR[CR0] XER[SO, OV]	

Preliminary User's Manual

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
dlimzb	RA, RS, RB	$d \leftarrow (RS) \parallel (RB)$ $i, x, y \leftarrow 0$ do while $(x < 8) \wedge (y = 0)$ $x \leftarrow x + 1$ if $d_{i:i+7} = 0$ then $y \leftarrow 1$ else $i \leftarrow i + 8$ $(RA) \leftarrow x$ $XER[TBC] \leftarrow x$ if $Rc = 1$ then $CR[CR0]_3 \leftarrow XER[SO]$ if $y = 1$ then if $x < 5$ then $CR[CR0]_{0:2} \leftarrow 0b010$ else $CR[CR0]_{0:2} \leftarrow 0b100$ else $CR[CR0]_{0:2} \leftarrow 0b001$	XER[TBC], RA	1076
dlimzb.			XER[TBC], RA, CR[CR0]	
eqv	RA, RS, RB	Equivalence of (RS) with (RB). $(RA) \leftarrow \neg((RS) \oplus (RB))$		1077
eqv.			CR[CR0]	
extlwi	RA, RS, n, b	Extract and left justify immediate. ($n > 0$) $(RA)_{0:n-1} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{n:31} \leftarrow 32^{-n}0$ Extended mnemonic for rlwinm RA,RS,b,0,n-1		1168
extlwi.		Extended mnemonic for rlwinm. RA,RS,b,0,n-1	CR[CR0]	
extrwi	RA, RS, n, b	Extract and right justify immediate. ($n > 0$) $(RA)_{32-n:31} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{0:31-n} \leftarrow 32^{-n}0$ Extended mnemonic for rlwinm RA,RS,b+n,32-n,31		1168
extrwi.		Extended mnemonic for rlwinm. RA,RS,b+n,32-n,31	CR[CR0]	
extsb	RA, RS	Extend the sign of byte (RS) _{24:31} . Place the result in RA.		1078
extsb.			CR[CR0]	
extsh	RA, RS	Extend the sign of halfword (RS) _{16:31} . Place the result in RA.		1079
extsh.			CR[CR0]	
icbi	RA, RB	Invalidate the instruction cache block which contains the effective address $(RA 0) + (RB)$.		1080
icbt	RA, RB	Load the instruction cache block which contains the effective address $(RA 0) + (RB)$.		1078
iccci	RA, RB	Invalidate the instruction cache array.		1083
icread	RA, RB	Read tag and data information from the instruction cache line selected using effective address bits 17:26. The effective address is calculated by $(RA 0) + (RB)$. Place the instruction selected by effective address bits 27:29 in ICDBDR; place the tag information in ICDBTRH and ICDBTRL.		1084
inslwi	RA, RS, n, b	Insert from left immediate. ($n > 0$) $(RA)_{b:b+n-1} \leftarrow (RS)_{0:n-1}$ Extended mnemonic for rlwimi RA,RS,32-b,b,b+n-1		1167
inslwi.		Extended mnemonic for rlwimi. RA,RS,32-b,b,b+n-1	CR[CR0]	

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
insrwi	RA, RS, n, b	Insert from right immediate. ($n > 0$) $(RA)_b:b+n-1 \leftarrow (RS)_{32-n:31}$ <i>Extended mnemonic for</i> rlwimi RA,RS,32-b-n,b,b+n-1		1167
insrwi.		<i>Extended mnemonic for</i> rlwimi. RA,RS,32-b-n,b,b+n-1	CR[CR0]	
isel	RT, RA, RB, CRb	$RT \leftarrow (RA 0)$ if CRb = 1, else $RT \leftarrow (RB)$		1086
isync		Synchronize execution context by flushing the prefetch queue.		1087
la	RT, D(RA)	Load address. ($RA \neq 0$) D is an offset from a base address that is assumed to be (RA). $(RT) \leftarrow (RA) + EXTS(D)$ <i>Extended mnemonic for</i> addi RT,RA,D		1027
lbz	RT, D(RA)	Load byte from $EA = (RA 0) + EXTS(D)$ and pad left with zeroes, $(RT) \leftarrow {}^{24}0 \parallel MS(EA,1)$.		1088
lbzu	RT, D(RA)	Load byte from $EA = (RA 0) + EXTS(D)$ and pad left with zeroes, $(RT) \leftarrow {}^{24}0 \parallel MS(EA,1)$. Update the base address, $(RA) \leftarrow EA$.		1089
lbzux	RT, RA, RB	Load byte from $EA = (RA 0) + (RB)$ and pad left with zeroes, $(RT) \leftarrow {}^{24}0 \parallel MS(EA,1)$. Update the base address, $(RA) \leftarrow EA$.		1090
lbzx	RT, RA, RB	Load byte from $EA = (RA 0) + (RB)$ and pad left with zeroes, $(RT) \leftarrow {}^{24}0 \parallel MS(EA,1)$.		1091
lha	RT, D(RA)	Load halfword from $EA = (RA 0) + EXTS(D)$ and sign extend, $(RT) \leftarrow EXTS(MS(EA,2))$.		1092
lhau	RT, D(RA)	Load halfword from $EA = (RA 0) + EXTS(D)$ and sign extend, $(RT) \leftarrow EXTS(MS(EA,2))$. Update the base address, $(RA) \leftarrow EA$.		1093
lhaux	RT, RA, RB	Load halfword from $EA = (RA 0) + (RB)$ and sign extend, $(RT) \leftarrow EXTS(MS(EA,2))$. Update the base address, $(RA) \leftarrow EA$.		1094
lhax	RT, RA, RB	Load halfword from $EA = (RA 0) + (RB)$ and sign extend, $(RT) \leftarrow EXTS(MS(EA,2))$.		1095
lhbrx	RT, RA, RB	Load halfword from $EA = (RA 0) + (RB)$, then reverse byte order and pad left with zeroes, $(RT) \leftarrow {}^{16}0 \parallel MS(EA+1,1) \parallel MS(EA,1)$.		1096
lhz	RT, D(RA)	Load halfword from $EA = (RA 0) + EXTS(D)$ and pad left with zeroes, $(RT) \leftarrow {}^{16}0 \parallel MS(EA,2)$.		1097
lhzu	RT, D(RA)	Load halfword from $EA = (RA 0) + EXTS(D)$ and pad left with zeroes, $(RT) \leftarrow {}^{16}0 \parallel MS(EA,2)$. Update the base address, $(RA) \leftarrow EA$.		1098
lhzux	RT, RA, RB	Load halfword from $EA = (RA 0) + (RB)$ and pad left with zeroes, $(RT) \leftarrow {}^{16}0 \parallel MS(EA,2)$. Update the base address, $(RA) \leftarrow EA$.		1099
lhzx	RT, RA, RB	Load halfword from $EA = (RA 0) + (RB)$ and pad left with zeroes, $(RT) \leftarrow {}^{16}0 \parallel MS(EA,2)$.		1100

Preliminary User's Manual

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
li	RT, IM	Load immediate. (RT) \leftarrow EXTS(IM) <i>Extended mnemonic for</i> addi RT,0,value		1027
lis	RT, IM	Load immediate shifted. (RT) \leftarrow (IM \parallel ¹⁶ 0) <i>Extended mnemonic for</i> addis RT,0,value		1030
lmw	RT, D(RA)	Load multiple words starting from EA = (RA 0) + EXTS(D). Place into consecutive registers RT through GPR(31). RA is not altered unless RA = GPR(31).		1101
lswi	RT, RA, NB	Load consecutive bytes from EA=(RA 0). Number of bytes n=32 if NB=0, else n=NB. Stack bytes into words in CEIL(n/4) consecutive registers starting with RT, to R _{FINAL} \leftarrow ((RT + CEIL(n/4) – 1) % 32). GPR(0) is consecutive to GPR(31). RA is not altered unless RA = R _{FINAL} .		1102
lswx	RT, RA, RB	Load consecutive bytes from EA=(RA 0)+(RB). Number of bytes n=XER[TBC]. Stack bytes into words in CEIL(n/4) consecutive registers starting with RT, to R _{FINAL} \leftarrow ((RT + CEIL(n/4) – 1) % 32). GPR(0) is consecutive to GPR(31). RA is not altered unless RA = R _{FINAL} . RB is not altered unless RB = R _{FINAL} . If n=0, content of RT is undefined.		1104
lwarx	RT, RA, RB	Load word from EA = (RA 0) + (RB) and place in RT, (RT) \leftarrow MS(EA,4). Set the Reservation bit.		1106
lwbrx	RT, RA, RB	Load word from EA = (RA 0) + (RB) then reverse byte order, (RT) \leftarrow MS(EA+3,1) \parallel MS(EA+2,1) \parallel MS(EA+1,1) \parallel MS(EA,1).		1107
lwz	RT, D(RA)	Load word from EA = (RA 0) + EXTS(D) and place in RT, (RT) \leftarrow MS(EA,4).		1108
lwzu	RT, D(RA)	Load word from EA = (RA 0) + EXTS(D) and place in RT, (RT) \leftarrow MS(EA,4). Update the base address, (RA) \leftarrow EA.		1109
lwzux	RT, RA, RB	Load word from EA = (RA 0) + (RB) and place in RT, (RT) \leftarrow MS(EA,4). Update the base address, (RA) \leftarrow EA.		1110
lwzx	RT, RA, RB	Load word from EA = (RA 0) + (RB) and place in RT, (RT) \leftarrow MS(EA,4).		1111
macchw	RT, RA, RB	prod _{0:31} \leftarrow (RA) _{16:31} \times (RB) _{0:15} temp _{0:32} \leftarrow prod _{0:31} + (RT) (RT) \leftarrow temp _{1:32}		1112
macchw.			CR[CR0]	
macchwo			XER[SO, OV]	
macchwo.			CR[CR0] XER[SO, OV]	

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
macchw	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$ $(\text{RT}) \leftarrow \text{temp}_{1:32}$		1115
macchw.			CR[CR0]	
macchwso			XER[SO, OV]	
macchwso.			CR[CR0] XER[SO, OV]	
macchws	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$ if $((\text{prod}_0 = \text{RT}_0) \wedge (\text{RT}_0 \neq \text{temp}_1))$ then $(\text{RT}) \leftarrow (\text{RT}_0 \vee {}^{31}(\neg \text{RT}_0))$ else $(\text{RT}) \leftarrow \text{temp}_{1:32}$		1113
macchws.			CR[CR0]	
macchwso			XER[SO, OV]	
macchwso.			CR[CR0] XER[SO, OV]	
macchwsu	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$ $(\text{RT}) \leftarrow (\text{temp}_{1:32} \vee {}^{31}\text{temp}_0)$		1114
macchwsu.			CR[CR0]	
macchwsuo			XER[SO, OV]	
macchwsuo.			CR[CR0] XER[SO, OV]	
machhw	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$ $(\text{RT}) \leftarrow \text{temp}_{1:32}$		1116
machhw.			CR[CR0]	
machhwo			XER[SO, OV]	
machhwo.			CR[CR0] XER[SO, OV]	
machhwu	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$ $(\text{RT}) \leftarrow \text{temp}_{1:32}$		1119
machhwu.			CR[CR0]	
machhwsuo			XER[SO, OV]	
machhwsuo.			CR[CR0] XER[SO, OV]	
machhws	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$ if $((\text{prod}_0 = \text{RT}_0) \wedge (\text{RT}_0 \neq \text{temp}_1))$ then $(\text{RT}) \leftarrow (\text{RT}_0 \vee {}^{31}(\neg \text{RT}_0))$ else $(\text{RT}) \leftarrow \text{temp}_{1:32}$		1117
machhws.			CR[CR0]	
machhwsuo			XER[SO, OV]	
machhwsuo.			CR[CR0] XER[SO, OV]	
machhwsu	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$ $(\text{RT}) \leftarrow (\text{temp}_{1:32} \vee {}^{31}\text{temp}_0)$		1118
machhwsu.			CR[CR0]	
machhwsuo			XER[SO, OV]	
machhwsuo.			CR[CR0] XER[SO, OV]	
macihw	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$ $(\text{RT}) \leftarrow \text{temp}_{1:32}$		1120
macihw.			CR[CR0]	
macihwo			XER[SO, OV]	
macihwo.			CR[CR0] XER[SO, OV]	
macihwu	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$ $(\text{RT}) \leftarrow \text{temp}_{1:32}$		1123
macihwu.			CR[CR0]	
macihwsuo			XER[SO, OV]	
macihwsuo.			CR[CR0] XER[SO, OV]	

Preliminary User's Manual

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
maclhws	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$ if $((\text{prod}_0 = \text{RT}_0) \wedge (\text{RT}_0 \neq \text{temp}_1))$ then $(\text{RT}) \leftarrow (\text{RT}_0 \vee {}^{31}(\neg \text{RT}_0))$ else $(\text{RT}) \leftarrow \text{temp}_{1:32}$		1121
maclhws.			CR[CR0]	
maclhws0			XER[SO, OV]	
maclhws0.			CR[CR0] XER[SO, OV]	
maclhwsu	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow \text{prod}_{0:31} + (\text{RT})$ $(\text{RT}) \leftarrow (\text{temp}_{1:32} \vee {}^{31}\text{temp}_0)$		1122
maclhwsu.			CR[CR0]	
maclhwsu0			XER[SO, OV]	
maclhwsu0.			CR[CR0] XER[SO, OV]	
mbar		Storage synchronization. All loads and stores that precede the mbar instruction complete before any loads and stores that follow the instruction access main storage.		1124
mcrf	BF, BFA	Move CR field, $(\text{CR}[\text{CRn}]) \leftarrow (\text{CR}[\text{CRm}])$ where $m \leftarrow \text{BFA}$ and $n \leftarrow \text{BF}$		1125
mcrxr	BF	Move XER[0:3] into field CRn, where $n \leftarrow \text{BF}$. $\text{CR}[\text{CRn}] \leftarrow (\text{XER}[\text{SO}, \text{OV}, \text{CA}])$ $(\text{XER}[\text{SO}, \text{OV}, \text{CA}]) \leftarrow {}^30$		1126
mfcrr	RT	Move from CR to RT, $(\text{RT}) \leftarrow (\text{CR})$.		1127
mfdcr	RT, DCRN	Move from DCR to RT, $(\text{RT}) \leftarrow (\text{DCR}(\text{DCRN}))$.		1128
mfmsr	RT	Move from MSR to RT, $(\text{RT}) \leftarrow (\text{MSR})$.		1129

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mfccr0 mfccr1 mfcsrr0 mfcsrr1 mfctr mfdac1 mfdac2 mfdbcr0 mfdbcr1 mfdbcr2 mfdbdr mfdbsr mfdcdbtrh mfdcdbtrl mfdear mfdec mfdnv0 mfdnv1 mfdnv2 mfdnv3 mfdtv0 mfdtv1 mfdtv2 mfdtv3 mfdvc1 mfdvc2 mfdvlim mfesr mfiac1 mfiac2 mfiac3 mfiac4 mficdbdr mficdbtrh mficdbtrl mfinv0 mfinv1 mfinv2 mfinv3 mfity0 mfity1 mfity2 mfity3 mfivlim mfivor0 mfivor1 mfivor2 mfivor3 mfivor4 mfivor5 mfivor6 mfivor7 mfivor8 mfivor9 mfivor10 mfivor11 mfivor12 mfivor13 mfivor14 mfivor15 mfivpr mflr mfmcscr mfmcscr0 mfmcscr1 mfmmucr	RT	Move from special purpose register (SPR) SPRN. <i>Extended mnemonic for</i> mf spr RT,SPRN See Table 32-2 Special Purpose Registers Sorted by SPR Number on page 1217 for listing of valid SPRN values.		1130

Preliminary User's Manual

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mfpid mfpir mfpvr mfsprg0 mfsprg1 mfsprg2 mfsprg3 mfsprg4 mfsprg5 mfsprg6 mfsprg7 mfssrr0 mfssrr1 mftbl mftbu mftcr mftsr mfusprg0 mfxer		Move from special purpose register (SPR) SPRN. <i>Extended mnemonic for</i> mfspr RT,SPRN See Table 32-2 Special Purpose Registers Sorted by SPR Number on page 1217 for listing of valid SPRN values.		
mfspr	RT, SPRN	Move from SPR to RT, (RT) \leftarrow (SPR(SPRN)).		1130
mr	RT, RS	Move register. (RT) \leftarrow (RS) <i>Extended mnemonic for</i> or RT,RS,RS		1160
mr.		<i>Extended mnemonic for</i> or. RT,RS,RS	CR[CR0]	
msync		Synchronization. All instructions that precede msync complete before any instructions that follow msync begin. When msync completes, all storage accesses initiated prior to msync will have completed.		1134
mtcr	RS	Move to Condition Register. <i>Extended mnemonic for</i> mtcrf 0xFF,RS		1135
mtcrf	FXM, RS	Move some or all of the contents of RS into CR as specified by FXM field, $\text{mask} \leftarrow 4(\text{FXM}_0) \parallel 4(\text{FXM}_1) \parallel \dots \parallel 4(\text{FXM}_6) \parallel 4(\text{FXM}_7)$. $(\text{CR}) \leftarrow ((\text{RS}) \wedge \text{mask}) \vee (\text{CR}) \wedge \neg \text{mask}$.		1135
mtdcr	DCRN, RS	Move to DCR from RS, (DCR(DCRN)) \leftarrow (RS).		1136
mtmsr	RS	Move to MSR from RS, (MSR) \leftarrow (RS).		1137

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mtccr0 mtccr1 mtcsrr0 mtcsrr1 mtctr mtdac1 mtdac2 mtdbcr0 mtdbcr1 mtdbcr2 mtdbdr mtdbdr mtdear mtdec mtdecar mtdnv0 mtdnv1 mtdnv2 mtdnv3 mtdtv0 mtdtv1 mtdtv2 mtdtv3 mtdvc1 mtdvc2 mtdvlim mtesr mtiac1 mtiac2 mtiac3 mtiac4 mtinv0 mtinv1 mtinv2 mtinv3 mtitv0 mtitv1 mtitv2 mtitv3 mtivlim mtivor0 mtivor1 mtivor2 mtivor3 mtivor4 mtivor5 mtivor6 mtivor7 mtivor8 mtivor9 mtivor10 mtivor11 mtivor12 mtivor13 mtivor14 mtivor15 mtivpr mtr mtmcsr mtmcsrr0 mtmcsrr1 mtmmucr mtpid	RS	<p>Move to SPR SPRN. Extended mnemonic for mtspr SPRN,RS</p> <p>See Table 32-2 <i>Special Purpose Registers Sorted by SPR Number</i> on page 1217 for listing of valid SPRN values.</p>		1138

Preliminary User's Manual

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mtsprg0 mtsprg1 mtsprg2 mtsprg3 mtsprg4 mtsprg5 mtsprg6 mtsprg7 mtsrr0 mtsrr1 mttbl mttbu mttcr mttcr mtusprg0 mtixer				
mtspr	SPRN, RS	Move to SPR from RS. (SPR(SPRN)) \leftarrow (RS).		1138
mulchw	RT, RA, RB	(RT) _{0:31} \leftarrow (RA) _{16:31} \times (RB) _{0:15} (signed)		1141
mulchw.			CR[CR0]	
mulchwu	RT, RA, RB	(RT) _{0:31} \leftarrow (RA) _{16:31} \times (RB) _{0:15} (unsigned)		1142
mulchwu.			CR[CR0]	
mulhhw	RT, RA, RB	(RT) _{0:31} \leftarrow (RA) _{0:15} \times (RB) _{0:15} (signed)		1143
mulhhw.			CR[CR0]	
mulhhwu	RT, RA, RB	(RT) _{0:31} \leftarrow (RA) _{0:15} \times (RB) _{0:15} (unsigned)		1144
mulhhwu.			CR[CR0]	
mulhw	RT, RA, RB	Multiply (RA) and (RB), signed. Place high-order result in RT. prod _{0:63} \leftarrow (RA) \times (RB) (signed). (RT) \leftarrow prod _{0:31} .		1145
mulhw.			CR[CR0]	
mulhwu	RT, RA, RB	Multiply (RA) and (RB), unsigned. Place high-order result in RT. prod _{0:63} \leftarrow (RA) \times (RB) (unsigned). (RT) \leftarrow prod _{0:31} .		1146
mulhwu.			CR[CR0]	
mullhw	RT, RA, RB	(RT) _{0:31} \leftarrow (RA) _{16:31} \times (RB) _{16:31} (signed)		1147
mullhw.			CR[CR0]	
mullhwu	RT, RA, RB	(RT) _{16:31} \leftarrow (RA) _{0:15} \times (RB) _{16:31} (unsigned)		1148
mullhwu.			CR[CR0]	
mulli	RT, RA, IM	Multiply (RA) and IM, signed. Place low-order result in RT. prod _{0:47} \leftarrow (RA) \times IM (signed) (RT) \leftarrow prod _{16:47}		1149
mullw	RT, RA, RB	Multiply (RA) and (RB), signed. Place low-order result in RT. prod _{0:63} \leftarrow (RA) \times (RB) (signed). (RT) \leftarrow prod _{32:63} .		1150
mullw.			CR[CR0]	
mullwo			XER[SO, OV]	
mullwo.			CR[CR0] XER[SO, OV]	
nand	RA, RS, RB	NAND (RS) with (RB). Place result in RA.		1151
nand.			CR[CR0]	
neg	RT, RA	Negative (two's complement) of RA. (RT) \leftarrow \neg (RA) + 1		1152
neg.			CR[CR0]	
nego			XER[SO, OV]	
nego.			CR[CR0] XER[SO, OV]	

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
nmacchw	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow -\text{prod}_{0:31} + (\text{RT})$ $(\text{RT}) \leftarrow \text{temp}_{1:32}$		1153
nmacchw.			CR[CR0]	
nmacchwo			XER[SO, OV]	
nmacchwo.			CR[CR0] XER[SO, OV]	
nmacchws	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow -\text{prod}_{0:31} + (\text{RT})$ if $((\text{prod}_0 = \text{RT}_0) \wedge (\text{RT}_0 \neq \text{temp}_1))$ then $(\text{RT}) \leftarrow (\text{RT}_0 \vee {}^{31}(\neg \text{RT}_0))$ else $(\text{RT}) \leftarrow \text{temp}_{1:32}$		1154
nmacchws.			CR[CR0]	
nmacchwso			XER[SO, OV]	
nmacchwso.			CR[CR0] XER[SO, OV]	
nmachhw	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{0:15} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow -\text{prod}_{0:31} + (\text{RT})$ $(\text{RT}) \leftarrow \text{temp}_{1:32}$		1155
nmachhw.			CR[CR0]	
nmachhwo			XER[SO, OV]	
nmachhwo.			CR[CR0] XER[SO, OV]	
nmachhws	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{0:15} \times (\text{RB})_{0:15}$ $\text{temp}_{0:32} \leftarrow -\text{prod}_{0:31} + (\text{RT})$ if $((\text{prod}_0 = \text{RT}_0) \wedge (\text{RT}_0 \neq \text{temp}_1))$ then $(\text{RT}) \leftarrow (\text{RT}_0 \vee {}^{31}(\neg \text{RT}_0))$ else $(\text{RT}) \leftarrow \text{temp}_{1:32}$		1156
nmachhws.			CR[CR0]	
nmachhwso			XER[SO, OV]	
nmachhwso.			CR[CR0] XER[SO, OV]	
nmacihw	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{16:31}$ $\text{temp}_{0:32} \leftarrow -\text{prod}_{0:31} + (\text{RT})$ $(\text{RT}) \leftarrow \text{temp}_{1:32}$		1157
nmacihw.			CR[CR0]	
nmaciho			XER[SO, OV]	
nmaciho.			CR[CR0] XER[SO, OV]	
nmacihws	RT, RA, RB	$\text{prod}_{0:31} \leftarrow (\text{RA})_{16:31} \times (\text{RB})_{16:31}$ $\text{temp}_{0:32} \leftarrow -\text{prod}_{0:31} + (\text{RT})$ if $((\text{prod}_0 = \text{RT}_0) \wedge (\text{RT}_0 \neq \text{temp}_1))$ then $(\text{RT}) \leftarrow (\text{RT}_0 \vee {}^{31}(\neg \text{RT}_0))$ else $(\text{RT}) \leftarrow \text{temp}_{1:32}$		1158
nmacihws.			CR[CR0]	
nmaciwso			XER[SO, OV]	
nmaciwso.			CR[CR0] XER[SO, OV]	
nop		Preferred no-op, triggers optimizations based on no-ops. <i>Extended mnemonic for</i> ori 0,0,0		1162
nor	RA, RS, RB	NOR (RS) with (RB). Place result in RA.		1159
nor.			CR[CR0]	
not	RA, RS	Complement register. $(\text{RA}) \leftarrow \neg(\text{RS})$ <i>Extended mnemonic for</i> nor RA,RS,RS		1159
not.		<i>Extended mnemonic for</i> nor. RA,RS,RS	CR[CR0]	
or	RA, RS, RB	OR (RS) with (RB). Place result in RA.		1160
or.			CR[CR0]	
orc	RA, RS, RB	OR (RS) with $\neg(\text{RB})$. Place result in RA.		1161
orc.			CR[CR0]	
ori	RA, RS, IM	OR (RS) with $({}^{16}0 \parallel \text{IM})$. Place result in RA.		1162
oris	RA, RS, IM	OR (RS) with $(\text{IM} \parallel {}^{16}0)$. Place result in RA.		1163

Preliminary User's Manual

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
rfci		Return from critical interrupt (PC) \leftarrow (CSRR0). (MSR) \leftarrow (CSRR1).		1164
rfi		Return from interrupt. (PC) \leftarrow (SRR0). (MSR) \leftarrow (SRR1).		1165
rfmci		Return from machine check interrupt (PC) \leftarrow (MCSRR0). (MSR) \leftarrow (MCSRR1).		1166
rlwimi	RA, RS, SH, MB, ME	Rotate left word immediate, then insert according to mask. $r \leftarrow \text{ROTL}((RS), SH)$ $m \leftarrow \text{MASK}(MB, ME)$ $(RA) \leftarrow (r \wedge m) \vee ((RA) \wedge \neg m)$	CR[CR0]	1167
rlwimi.				
rlwinm	RA, RS, SH, MB, ME	Rotate left word immediate, then AND with mask. $r \leftarrow \text{ROTL}((RS), SH)$ $m \leftarrow \text{MASK}(MB, ME)$ $(RA) \leftarrow (r \wedge m)$	CR[CR0]	1168
rlwinm.				
rlwnm	RA, RS, RB, MB, ME	Rotate left word, then AND with mask. $r \leftarrow \text{ROTL}((RS), (RB)_{27:31})$ $m \leftarrow \text{MASK}(MB, ME)$ $(RA) \leftarrow (r \wedge m)$	CR[CR0]	1170
rlwnm.				
rotlw	RA, RS, RB	Rotate left. $(RA) \leftarrow \text{ROTL}((RS), (RB)_{27:31})$ <i>Extended mnemonic for</i> rlwnm RA,RS,RB,0,31	CR[CR0]	1170
rotlw.		<i>Extended mnemonic for</i> rlwnm. RA,RS,RB,0,31		
rotlwi	RA, RS, n	Rotate left immediate. $(RA) \leftarrow \text{ROTL}((RS), n)$ <i>Extended mnemonic for</i> rlwinm RA,RS,n,0,31	CR[CR0]	1168
rotlwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,n,0,31		
rotrwi	RA, RS, n	Rotate right immediate. $(RA) \leftarrow \text{ROTL}((RS), 32-n)$ <i>Extended mnemonic for</i> rlwinm RA,RS,32-n,0,31	CR[CR0]	1168
rotrwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,32-n,0,31		
sc		System call exception is generated. (SRR1) \leftarrow (MSR) (SRR0) \leftarrow (PC) $PC \leftarrow \text{EVPR}_{0:15} \parallel 0x0C00$ (MSR[WE, PR, EE, PE, DR, IR]) \leftarrow 0		1171
slw	RA, RS, RB	Shift left (RS) by (RB) _{27:31} . $n \leftarrow (RB)_{27:31}$ $r \leftarrow \text{ROTL}((RS), n)$ if $(RB)_{26} = 0$ then $m \leftarrow \text{MASK}(0, 31-n)$ else $m \leftarrow 320$ $(RA) \leftarrow r \wedge m$.	CR[CR0]	1172
slw.				
slwi	RA, RS, n	Shift left immediate. ($n < 32$) $(RA)_{0:31-n} \leftarrow (RS)_{n:31}$ $(RA)_{32-n:31} \leftarrow n0$ <i>Extended mnemonic for</i> rlwinm RA,RS,n,0,31-n	CR[CR0]	1168
slwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,n,0,31-n		

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
sraw	RA, RS, RB	Shift right algebraic (RS) by (RB) _{27:31} . $n \leftarrow (RB)_{27:31}$. $r \leftarrow \text{ROTL}((RS), 32 - n)$. if $(RB)_{26} = 0$ then $m \leftarrow \text{MASK}(n, 31)$ else $m \leftarrow 32_0$ $s \leftarrow (RS)_0$ $(RA) \leftarrow (r \wedge m) \vee (32_s \wedge \neg m)$. $\text{XER}[CA] \leftarrow s \wedge ((r \wedge \neg m) \neq 0)$.	CR[CR0]	1173
sraw.				
srawi	RA, RS, SH	Shift right algebraic (RS) by SH. $n \leftarrow SH$. $r \leftarrow \text{ROTL}((RS), 32 - n)$. $m \leftarrow \text{MASK}(n, 31)$. $s \leftarrow (RS)_0$ $(RA) \leftarrow (r \wedge m) \vee (32_s \wedge \neg m)$. $\text{XER}[CA] \leftarrow s \wedge ((r \wedge \neg m) \neq 0)$.	CR[CR0]	1174
srawi.				
srw	RA, RS, RB	Shift right (RS) by (RB) _{27:31} . $n \leftarrow (RB)_{27:31}$. $r \leftarrow \text{ROTL}((RS), 32 - n)$. if $(RB)_{26} = 0$ then $m \leftarrow \text{MASK}(n, 31)$ else $m \leftarrow 32_0$ $(RA) \leftarrow r \wedge m$.	CR[CR0]	1175
srw.				
srwi	RA, RS, n	Shift right immediate. ($n < 32$) $(RA)_{n:31} \leftarrow (RS)_{0:31-n}$ $(RA)_{0:n-1} \leftarrow 0$ <i>Extended mnemonic for</i> rlwinm RA,RS,32-n,n,31	CR[CR0]	1168
srwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,32-n,n,31		
stb	RS, D(RA)	Store byte (RS) _{24:31} in memory at $EA = (RA 0) + \text{EXTS}(D)$.		1176
stbu	RS, D(RA)	Store byte (RS) _{24:31} in memory at $EA = (RA 0) + \text{EXTS}(D)$. Update the base address, $(RA) \leftarrow EA$.		1177
stbux	RS, RA, RB	Store byte (RS) _{24:31} in memory at $EA = (RA 0) + (RB)$. Update the base address, $(RA) \leftarrow EA$.		1178
stbx	RS, RA, RB	Store byte (RS) _{24:31} in memory at $EA = (RA 0) + (RB)$.		1179
sth	RS, D(RA)	Store halfword (RS) _{16:31} in memory at $EA = (RA 0) + \text{EXTS}(D)$.		1180
sthbrx	RS, RA, RB	Store halfword (RS) _{16:31} byte-reversed in memory at $EA = (RA 0) + (RB)$. $\text{MS}(EA, 2) \leftarrow (RS)_{24:31} \parallel (RS)_{16:23}$		1181
sthu	RS, D(RA)	Store halfword (RS) _{16:31} in memory at $EA = (RA 0) + \text{EXTS}(D)$. Update the base address, $(RA) \leftarrow EA$.		1182
sthux	RS, RA, RB	Store halfword (RS) _{16:31} in memory at $EA = (RA 0) + (RB)$. Update the base address, $(RA) \leftarrow EA$.		1183
sthx	RS, RA, RB	Store halfword (RS) _{16:31} in memory at $EA = (RA 0) + (RB)$.		1184

Preliminary User's Manual

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
stmw	RS, D(RA)	Store consecutive words from RS through GPR(31) in memory starting at EA = (RA 0) + EXTS(D).		1185
stswi	RS, RA, NB	Store consecutive bytes in memory starting at EA=(RA 0). Number of bytes n=32 if NB=0, else n=NB. Bytes are unstacked from CEIL(n/4) consecutive registers starting with RS. GPR(0) is consecutive to GPR(31).		1185
stswx	RS, RA, RB	Store consecutive bytes in memory starting at EA=(RA 0)+(RB). Number of bytes n=XER[TBC]. Bytes are unstacked from CEIL(n/4) consecutive registers starting with RS. GPR(0) is consecutive to GPR(31).		1187
stw	RS, D(RA)	Store word (RS) in memory at EA = (RA 0) + EXTS(D).		1188
stwbrx	RS, RA, RB	Store word (RS) byte-reversed in memory at EA = (RA 0) + (RB). $MS(EA, 4) \leftarrow (RS)_{24:31} \parallel (RS)_{16:23} \parallel (RS)_{8:15} \parallel (RS)_{0:7}$		1189
stwcx.	RS, RA, RB	Store word (RS) in memory at EA = (RA 0) + (RB) only if reservation bit is set. if RESERVE = 1 then $MS(EA, 4) \leftarrow (RS)$ RESERVE \leftarrow 0 $(CR[CR0]) \leftarrow {}^2_0 \parallel 1 \parallel XER_{SO}$ else $(CR[CR0]) \leftarrow {}^2_0 \parallel 0 \parallel XER_{SO}$.		1190
stwu	RS, D(RA)	Store word (RS) in memory at EA = (RA 0) + EXTS(D). Update the base address, (RA) \leftarrow EA.		1192
stwux	RS, RA, RB	Store word (RS) in memory at EA = (RA 0) + (RB). Update the base address, (RA) \leftarrow EA.		1193
stwx	RS, RA, RB	Store word (RS) in memory at EA = (RA 0) + (RB).		1194
sub	RT, RA, RB	Subtract (RB) from (RA). (RT) \leftarrow $\neg(RB) + (RA) + 1$. <i>Extended mnemonic for</i> subf RT,RB,RA		1195
sub.		<i>Extended mnemonic for</i> subf. RT,RB,RA	CR[CR0]	
subo		<i>Extended mnemonic for</i> subfo RT,RB,RA	XER[SO, OV]	
subo.		<i>Extended mnemonic for</i> subfo. RT,RB,RA	CR[CR0] XER[SO, OV]	

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
subc	RT, RA, RB	Subtract (RB) from (RA). $(RT) \leftarrow \neg(RB) + (RA) + 1$. Place carry-out in XER[CA]. <i>Extended mnemonic for</i> subfc RT,RB,RA		1196
subc.		<i>Extended mnemonic for</i> subfc. RT,RB,RA	CR[CR0]	
subco		<i>Extended mnemonic for</i> subfco RT,RB,RA	XER[SO, OV]	
subco.		<i>Extended mnemonic for</i> subfco. RT,RB,RA	CR[CR0] XER[SO, OV]	
subf	RT, RA, RB	Subtract (RA) from (RB). $(RT) \leftarrow \neg(RA) + (RB) + 1$.		1195
subf.			CR[CR0]	
subfo			XER[SO, OV]	
subfo.			CR[CR0] XER[SO, OV]	
subfc	RT, RA, RB	Subtract (RA) from (RB). $(RT) \leftarrow \neg(RA) + (RB) + 1$. Place carry-out in XER[CA].		1196
subfc.			CR[CR0]	
subfco			XER[SO, OV]	
subfco.			CR[CR0] XER[SO, OV]	
subfe	RT, RA, RB	Subtract (RA) from (RB) with carry-in. $(RT) \leftarrow \neg(RA) + (RB) + XER[CA]$. Place carry-out in XER[CA].		1197
subfe.			CR[CR0]	
subfeo			XER[SO, OV]	
subfeo.			CR[CR0] XER[SO, OV]	
subfic	RT, RA, IM	Subtract (RA) from EXTS(IM). $(RT) \leftarrow \neg(RA) + EXTS(IM) + 1$. Place carry-out in XER[CA].		1198
subfme	RT, RA, RB	Subtract (RA) from (–1) with carry-in. $(RT) \leftarrow \neg(RA) + (-1) + XER[CA]$. Place carry-out in XER[CA].		1199
subfme.			CR[CR0]	
subfmeo			XER[SO, OV]	
subfmeo.			CR[CR0] XER[SO, OV]	
subfze	RT, RA, RB	Subtract (RA) from zero with carry-in. $(RT) \leftarrow \neg(RA) + XER[CA]$. Place carry-out in XER[CA].		1200
subfze.			CR[CR0]	
subfzeo			XER[SO, OV]	
subfzeo.			CR[CR0] XER[SO, OV]	
subi	RT, RA, IM	Subtract EXTS(IM) from (RA)0. Place result in RT. <i>Extended mnemonic for</i> addi RT,RA,–IM		1027
subic	RT, RA, IM	Subtract EXTS(IM) from (RA). Place result in RT. Place carry-out in XER[CA]. <i>Extended mnemonic for</i> addic RT,RA,–IM		1028

Preliminary User's Manual

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
subic.	RT, RA, IM	Subtract EXTS(IM) from (RA). Place result in RT. Place carry-out in XER[CA]. <i>Extended mnemonic for</i> addic. RT,RA,-IM	CR[CR0]	1029
subis	RT, RA, IM	Subtract (IM ¹⁶ 0) from (RA 0). Place result in RT. <i>Extended mnemonic for</i> addis RT,RA,-IM		1030
tlbre	RT, RA, WS	$tlbentry \leftarrow TLB[(RA)_{26:31}]$ if WS = 0 $(RT)_{0:27} \leftarrow tlbentry[EPN,V,TS,SIZE]$ $(RT)_{28:31} \leftarrow 40$ $MMUCR[STID] \leftarrow tlbentry[TID]$ else if WS = 1 $(RT)_{0:21} \leftarrow tlbentry[RPN]$ $(RT)_{22:27} \leftarrow 60$ $(RT)_{28:31} \leftarrow tlbentry[ERPN]$ else if WS = 2 $(RT)_{0:15} \leftarrow 160$ $(RT)_{16:24} \leftarrow tlbentry[U0,U1,U2,U3,W,I,M,G,E]$ $(RT)_{25} \leftarrow 0$ $(RT)_{26:31} \leftarrow tlbentry[UX,UW,UR,SX,SW,SR]$ else (RT), MMUCR[STID] \leftarrow undefined		1201
tlbsx	RT,RA,RB	Search the TLB for a valid entry that translates the EA. EA = (RA 0) + (RB)	CR[CR0]	1203
tlbsx.		if Rc = 1 $CR[CR0]_0 \leftarrow 0$ $CR[CR0]_1 \leftarrow 0$ $CR[CR0]_3 \leftarrow XER[SO]$ if Valid TLB entry matching EA and MMUCR[STID,STS] is in the TLB then (RT) \leftarrow Index of matching TLB Entry if Rc = 1 $CR[CR0]_2 \leftarrow 1$ else (RT) \leftarrow Undefined if Rc = 1 $CR[CR0]_2 \leftarrow 0$		
tlbsync		tlbsync does not complete until all previous TLB-update instructions executed by this processor have been received and completed by all other processors. For the PPC440GX, tlbsync is a no-op.		1204
tlbwe	RS, RA, WS	$tlbentry \leftarrow TLB[(RA)_{26:31}]$ if WS = 0 $tlbentry[EPN,V,TS,SIZE] \leftarrow (RS)_{0:27}$ $tlbentry[TID] \leftarrow MMUCR[STID]$ else if WS = 1 $tlbentry[RPN] \leftarrow (RS)_{0:21}$ $tlbentry[ERPN] \leftarrow (RS)_{28:31}$ else if WS = 2 $tlbentry[U0,U1,U2,U3,W,I,M,G,E] \leftarrow (RS)_{16:24}$ $tlbentry[UX,UW,UR,SX,SW,SR] \leftarrow (RS)_{26:31}$ else tlbentry \leftarrow undefined		1205

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
trap		Trap unconditionally. <i>Extended mnemonic for</i> tw 31,0,0		
tw eq	RA, RB	Trap if (RA) equal to (RB). <i>Extended mnemonic for</i> tw 4,RA,RB		1206
tw ge		Trap if (RA) greater than or equal to (RB). <i>Extended mnemonic for</i> tw 12,RA,RB		
tw gt		Trap if (RA) greater than (RB). <i>Extended mnemonic for</i> tw 8,RA,RB		
tw le		Trap if (RA) less than or equal to (RB). <i>Extended mnemonic for</i> tw 20,RA,RB		
tw lge		Trap if (RA) logically greater than or equal to (RB). <i>Extended mnemonic for</i> tw 5,RA,RB		
tw lgt		Trap if (RA) logically greater than (RB). <i>Extended mnemonic for</i> tw 1,RA,RB		
tw lle		Trap if (RA) logically less than or equal to (RB). <i>Extended mnemonic for</i> tw 6,RA,RB		
tw llt		Trap if (RA) logically less than (RB). <i>Extended mnemonic for</i> tw 2,RA,RB		
tw lng		Trap if (RA) logically not greater than (RB). <i>Extended mnemonic for</i> tw 6,RA,RB		
tw lnl		Trap if (RA) logically not less than (RB). <i>Extended mnemonic for</i> tw 5,RA,RB		
tw lt		Trap if (RA) less than (RB). <i>Extended mnemonic for</i> tw 16,RA,RB		
tw ne		Trap if (RA) not equal to (RB). <i>Extended mnemonic for</i> tw 24,RA,RB		
tw ng		Trap if (RA) not greater than (RB). <i>Extended mnemonic for</i> tw 20,RA,RB		
tw nl		Trap if (RA) not less than (RB). <i>Extended mnemonic for</i> tw 12,RA,RB		
tw	TO, RA, RB	Trap exception is generated if, comparing (RA) with (RB), any condition specified by TO is true.		1206

Preliminary User's Manual

Table A-1. PPC440GX Embedded Processor Instruction Syntax Summary (Continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
tweqi	RA, IM	Trap if (RA) equal to EXTS(IM). <i>Extended mnemonic for</i> wi 4,RA,IM		1208
twgei		Trap if (RA) greater than or equal to EXTS(IM). <i>Extended mnemonic for</i> twi 12,RA,IM		
twgti		Trap if (RA) greater than EXTS(IM). <i>Extended mnemonic for</i> twi 8,RA,IM		
twlei		Trap if (RA) less than or equal to EXTS(IM). <i>Extended mnemonic for</i> twi 20,RA,IM		
twlgei		Trap if (RA) logically greater than or equal to EXTS(IM). <i>Extended mnemonic for</i> wi 5,RA,IM		
twlgti		Trap if (RA) logically greater than EXTS(IM). <i>Extended mnemonic for</i> twi 1,RA,IM		
twllei		Trap if (RA) logically less than or equal to EXTS(IM). <i>Extended mnemonic for</i> twi 6,RA,IM		
twllti		Trap if (RA) logically less than EXTS(IM). <i>Extended mnemonic for</i> twi 2,RA,IM		
twlngi		Trap if (RA) logically not greater than EXTS(IM). <i>Extended mnemonic for</i> twi 6,RA,IM		
twlnli		Trap if (RA) logically not less than EXTS(IM). <i>Extended mnemonic for</i> twi 5,RA,IM		
twlti		Trap if (RA) less than EXTS(IM). <i>Extended mnemonic for</i> twi 16,RA,IM		
twnei		Trap if (RA) not equal to EXTS(IM). <i>Extended mnemonic for</i> twi 24,RA,IM		
twngi		Trap if (RA) not greater than EXTS(IM). <i>Extended mnemonic for</i> twi 20,RA,IM		
twnli		Trap if (RA) not less than EXTS(IM). <i>Extended mnemonic for</i> twi 12,RA,IM		
twi	TO, RA, IM	Trap exception is generated if, comparing (RA) with EXTS(IM), any condition specified by TO is true.		1208
wrtee	RS	Write value of RS ₁₆ to MSR[EE].		1210
wrteei	E	Write value of E to MSR[EE].		1211
xor	RA, RS, RB	XOR (RS) with (RB). Place result in RA.		1212
xor.			CR[CR0]	
xori	RA, RS, IM	XOR (RS) with (¹⁶ 0 IM). Place result in RA.		1213
xoris	RA, RS, IM	XOR (RS) with (IM ¹⁶ 0). Place result in RA.		1214

A.3 Allocated Instruction Opcodes

Allocated instructions are provided for purposes that are outside the scope of PowerPC Book-E architecture, and are for implementation-dependent and application-specific use, including use within auxiliary processors.

Table A-2 lists the blocks of opcodes which have been allocated by PowerPC Book-E for these purposes. In the table, the character “u” designates a secondary opcode bit which can be set to any value. In some cases, the decimal value of a secondary opcode is shown in parentheses after the binary value.

Table A-2. Allocated Opcodes

Primary Opcode	Extended Opcodes	PPC440GX Embedded Processor Usage
0	All instruction encodings (bits 6:31) except 0x00000000 (the instruction encoding of 0x00000000 is and always will be reserved-illegal)	None
4	All instruction encodings (bits 6:31)	Various (see <i>Table A-5</i> on page 1820)
19	Secondary opcodes (bits 21:30) = 0b000000u11u	None
31	Secondary opcodes (bits 21:30) = 0b000000011u Secondary opcodes (bits 21:30) = 0b000000u110 Secondary opcode (bits 21:30) = 0b0101010110 (342) Secondary opcode (bits 21:30) = 0b0101110110 (374) Secondary opcode (bits 21:30) = 0b1100110110 (822)	Various (see <i>Table A-5</i> on page 1820)
59	Secondary opcodes (bits 21:30) = 0b000000u10u	None
63	Secondary opcodes (bits 21:30) = 0b000000u10u (except secondary opcode decimal 12, which is the fsrp defined instruction)	None

All of the allocated opcodes listed in the table above are available for use by auxiliary processors attached to the PPC440GX Embedded Processor, except for those which have already been implemented within the PPC440GX for certain implementation-specific purposes. As indicated in the table above, this is the case for certain secondary opcodes within primary opcodes 4 and 31. These opcodes are identified in *Table A-5* on page 1820, along with all of the defined, preserved, and reserved-nop class opcodes which are implemented within the PPC440GX.

A.4 Preserved Instruction Opcodes

The preserved instruction class is provided to support backward compatibility with the PowerPC Architecture, and/or earlier versions of the PowerPC Book-E architecture. This instruction class includes opcodes which were defined for these previous architectures, but which are no longer defined for PowerPC Book-E.

Table A-3 lists the reserved opcodes designated by PowerPC Book-E. The decimal value of the secondary opcode is shown in parentheses after the binary value.

Table A-3. Preserved Opcodes

Primary Opcode	Extended Opcode	Preserved Mnemonic	PPC440GX Embedded Processor Usage
31	0b0011010010 (210)	mtsr	
31	0b0011110010 (242)	mtsrin	
31	0b0101110010 (370)	tlbia	
31	0b0100110010 (306)	tlbie	
31	0b0101110011 (371)	mftb	Yes
31	0b1001010011 (595)	mfsr	
31	0b1010010011 (659)	mfsrin	
31	0b0100110110 (310)	eciwx	

Preliminary User's Manual

Table A-3. Preserved Opcodes (Continued)

Primary Opcode	Extended Opcode	Preserved Mnemonic	PPC440GX Embedded Processor Usage
31	0b0110110110 (438)	ecowx	

As indicated in the table above, the only preserved opcode which is implemented within the PPC440GX is the **mftb** instruction. See *Preserved Instruction Class* on page 170 for more information on PPC440GX Embedded Processor support for this instruction. All other preserved instructions are treated as reserved by PPC440GX Embedded Processor and will cause Illegal Instruction exception type Program interrupts if their execution is attempted.

The preserved opcode for **mftb** is included in Table A-5 on page 1820, along with all of the defined, allocated, and reserved-nop class opcodes which are implemented within the PPC440GX.

A.5 Reserved Instruction Opcodes

This class of instructions consists of all instruction primary opcodes (and associated extended opcodes, if applicable) which do not belong to either the defined, allocated, or preserved instruction classes.

Reserved instructions are available for future versions of PowerPC Book-E architecture. That is, future versions of PowerPC Book-E may define any of these instructions to perform new functions or make them available for implementation-dependent use as allocated instructions. There are two types of reserved instructions: reserved-illegal and reserved-nop.

Table A-4 lists the reserved-nop opcodes designated by PowerPC Book-E. In the table, the character “u” designates a secondary opcode bit which can be set to any value. All other reserved opcodes are in the reserved-illegal class.

Table A-4. Reserved-nop Opcodes

Primary Opcode	Extended Opcode
31	0b10uuu10010

As shown in the table, there are a total of eight (8) secondary opcodes in the reserved-nop class. The PPC440GX Embedded Processor implements all of the reserved-nop instruction opcodes as true no-ops. These opcodes are included in Table A-5 on page 1820, along with all of the defined, allocated, and preserved class opcodes which are implemented within the PPC440GX.

A.6 Implemented Instructions Sorted by Opcode

Table A-5 on page 1820 lists all of the instructions which have been implemented within the PPC440GX, sorted by primary and secondary opcode. These include defined, allocated, preserved, and reserved-nop class instructions (see *Instruction Classes* on page 168 for a more detailed description of each of these instruction classes).

Opcodes which are *not* implemented in the PPC440GX are *not* shown in the table, and consist of the following:

- Defined instructions

These include the floating-point operations (which may be implemented in an auxiliary processor and executed via the AP interface), as well as the 64-bit operations and the **tlbiva** and **mfapidi** instructions, all of which are handled as reserved-illegal instructions by the PPC440GX Embedded Processor.

- Allocated instructions

These include all of the allocated opcodes identified in *Table A-2* on page 1818 which are not already implemented within the PPC440GX. If not implemented within an attached auxiliary processor, these instructions will be handled as reserved-illegal by the PPC440GX Embedded Processor.

- Preserved instructions

These include all of the preserved opcodes identified in *Table A-3* on page 1818 except for the **mftb** opcode (which is implemented and thus included in *Table A-5*). These instructions will be handled as reserved-illegal by the PPC440GX Embedded Processor.

- Reserved instructions

These include all of the reserved opcodes as defined by *Appendix A.5* on page 1819, except for the reserved-nop opcodes identified in *Table A-4* on page 1819. These instructions by definition are all in the reserved-illegal class and will be handled as such by the PPC440GX Embedded Processor.

All PowerPC Book-E instructions are four bytes long and word aligned. All instructions have a primary opcode field (shown as field OPCODE in Figure A-1 through Figure A-9, beginning on page 1786) in bits 0:5. Some instructions also have a secondary opcode field (shown as field XO in Figure A-1 through Figure A-9).

The “Form” indicated in the table refers to the arrangement of valid field combinations within the four-byte instruction. See *Appendix A.1* on page 1783, for the field layouts of each form.

Form X has a 10-bit secondary opcode field, while form XO uses only the low-order 9-bits of that field. Form XO uses the high-order secondary opcode bit (the tenth bit) as a variable; therefore, every form XO instruction really consumes two secondary opcodes from the 10-bit secondary-opcode space. The implicitly consumed secondary opcode is listed in parentheses for form XO instructions in the table below.

Table A-5. PPC440GX Embedded Processor Instructions by Opcode

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
3		D	twi	TO, RA, IM	1208
4	8	X	mulhhuw mulhhuw.	RT, RA, RB	1144
4	12 (524)	XO	machhuw machhuw. machhuwo machhuwo.	RT, RA, RB	1119
4	40	X	mulhhw mulhhw.	RT, RA, RB	1143
4	44 (556)	XO	machhw machhw. machhwo machhwo.	RT, RA, RB	1116
4	46 (558)	XO	nmachhw nmachhw. nmachhwo nmachhwo.	RT, RA, RB	1155
4	76 (588)	XO	machhwsu machhwsu. machhwsuo machhwsuo.	RT, RA, RB	1118

Preliminary User's Manual

Table A-5. PPC440GX Embedded Processor Instructions by Opcode (Continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
4	108 (620)	XO	machhws	RT, RA, RB	1117
			machhws.		
			machhwso		
			machhwso.		
4	110 (622)	XO	nmachhws	RT, RA, RB	1156
			nmachhws.		
			nmachhwso		
			nmachhwso.		
4	136	X	mulchwu	RT, RA, RB	1142
			mulchwu.		
4	140 (652)	XO	macchwu	RT, RA, RB	1115
			macchwu.		
			macchwuo		
			machhwuo.		
4	168	X	mulchw	RT, RA, RB	1141
			mulchw.		
4	172 (684)	XO	macchw	RT, RA, RB	1112
			macchw.		
			macchwo		
			macchwo.		
4	174 (686)	XO	nmacchw	RT, RA, RB	1153
			nmacchw.		
			nmacchwo		
			nmacchwo.		
4	204 (716)	XO	macchwsu	RT, RA, RB	1114
			macchwsu.		
			macchwsuo		
			macchwsuo.		
4	236 (748)	XO	macchws	RT, RA, RB	1113
			macchws.		
			macchwso		
			macchwso.		
4	238 (750)	XO	nmacchws	RT, RA, RB	1154
			nmacchws.		
			nmacchwso		
			nmacchwso.		
4	392	X	mullhwu	RT, RA, RB	1148
			mullhwu.		
4	396 (908)	XO	maclhwu	RT, RA, RB	1123
			maclhwu.		
			maclhwuo		
			maclhwuo.		
4	424	X	mullhw	RT, RA, RB	1147
			mullhw.		

Table A-5. PPC440GX Embedded Processor Instructions by Opcode (Continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
4	428 (940)	XO	macldw	RT, RA, RB	1120
			macldw.		
			macldwo		
			macldwo.		
4	430 (942)	XO	nmacldw	RT, RA, RB	1157
			nmacldw.		
			nmacldwo		
			nmacldwo.		
4	460 (972)	XO	macldwsu	RT, RA, RB	1122
			macldwsu.		
			macldwsuo		
			macldwsuo.		
4	492 (1004)	XO	macldws	RT, RA, RB	1121
			macldws.		
			macldwso		
			macldwso.		
4	494 (1006)	XO	nmacldws	RT, RA, RB	1158
			nmacldws.		
			nmacldwso		
			nmacldwso.		
7		D	mulli	RT, RA, IM	1149
8		D	subfic	RT, RA, IM	1198
10		D	cmpli	BF, 0, RA, IM	1054
11		D	cmpi	BF, 0, RA, IM	1052
12		D	addic	RT, RA, IM	1028
13		D	addic.	RT, RA, IM	1029
14		D	addi	RT, RA, IM	1027
15		D	addis	RT, RA, IM	1030
16		B	bc	BO, BI, target	1038
			bca		
			bcl		
			bcla		
17		SC	sc		1171
18		I	b	target	1037
			ba		
			bl		
			bla		
19	0	XL	mcrf	BF, BFA	1125
19	16	XL	bclr	BO, BI	1047
			bclrl		
19	33	XL	crnor	BT, BA, BB	1060
19	38	XL	rfmci		1166
19	50	XL	rfi		1165
19	51	XL	rfci		1164

Preliminary User's Manual

Table A-5. PPC440GX Embedded Processor Instructions by Opcode (Continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
19	129	XL	crandc	BT, BA, BB	1057
19	150	XL	isync		1087
19	193	XL	crxor	BT, BA, BB	1063
19	225	XL	crnand	BT, BA, BB	1059
19	257	XL	crand	BT, BA, BB	1056
19	289	XL	creqv	BT, BA, BB	1058
19	417	XL	crorc	BT, BA, BB	1062
19	449	XL	cror	BT, BA, BB	1061
19	528	XL	bcctr	BO, BI	1044
			bcctrl		
20		M	rlwimi	RA, RS, SH, MB, ME	1167
			rlwimi.		
21		M	rlwinm	RA, RS, SH, MB, ME	1168
			rlwinm.		
23		M	rlwnm	RA, RS, RB, MB, ME	1170
			rlwnm.		
24		D	ori	RA, RS, IM	1162
25		D	oris	RA, RS, IM	1163
26		D	xori	RA, RS, IM	1213
27		D	xoris	RA, RS, IM	1214
28		D	andi.	RA, RS, IM	1035
29		D	andis.	RA, RS, IM	1036
31	0	X	cmp	BF, 0, RA, RB	1051
31	4	X	tw	TO, RA, RB	1206
31	8 (520)	XO	subfc	RT, RA, RB	1196
			subfc.		
			subfco		
			subfco.		
31	10 (522)	XO	addc	RT, RA, RB	1024
			addc.		
			addco		
			addco.		
31	11 (523)	XO	mulhwu	RT, RA, RB	1146
			mulhwu.		
31	15	XO	isel	RT, RA, RB	1086
31	19	X	mfcrr	RT	1127
31	20	X	lwarx	RT, RA, RB	1106
31	22	X	icbt	RA, RB	1081
31	23	X	lwzx	RT, RA, RB	1111
31	24	X	slw	RA, RS, RB	1172
			slw.		

Table A-5. PPC440GX Embedded Processor Instructions by Opcode (Continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	26	X	cntlzw	RA, RS	1055
			cntlzw.		
31	28	X	and	RA, RS, RB	1033
			and.		
31	32	X	cmpl	BF, 0, RA, RB	1053
31	40 (552)	XO	subf	RT, RA, RB	1195
			subf.		
			subfo		
			subfo.		
31	54	X	dcbst	RA, RB	1069
31	55	X	lwzux	RT, RA, RB	1110
31	60	X	andc	RA, RS, RB	1034
			andc.		
31	75 (587)	XO	mulhw	RT, RA, RB	1145
			mulhw.		
31	78	X	dlmzb	RA, RS, RB	1076
			dlmzb.		
31	83	X	mfmsr	RT	1129
31	86	X	dcbf	RA, RB	1065
31	87	X	lbzx	RT, RA, RB	1091
31	104 (616)	XO	neg	RT, RA	1152
			neg.		
			nego		
			nego.		
31	119	X	lbzux	RT, RA, RB	1090
31	124	X	nor	RA, RS, RB	1159
			nor.		
31	131	X	wrttee	RS	1210
31	136 (648)	XO	subfe	RT, RA, RB	1197
			subfe.		
			subfeo		
			subfeo.		
31	138 (650)	XO	adde	RT, RA, RB	1025
			adde.		
			addeo		
			addeo.		
31	144	XFX	mtcrf	FXM, RS	1135
31	146	X	mtmsr	RS	1137
31	150	X	stwcx.	RS, RA, RB	1190
31	151	X	stwx	RS, RA, RB	1194
31	163	X	wrteei	E	1211
31	183	X	stwux	RS, RA, RB	1193

Preliminary User's Manual

Table A-5. PPC440GX Embedded Processor Instructions by Opcode (Continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	200 (712)	XO	subfze	RT, RA, RB	1200
			subfze.		
			subfzeo		
			subfzeo.		
31	202 (714)	XO	addze	RT, RA	1032
			addze.		
			addzeo		
			addzeo.		
31	215	X	stbx	RS, RA, RB	1179
31	232 (744)	XO	subfme	RT, RA, RB	1199
			subfme.		
			subfmeo		
			subfmeo.		
31	234 (746)	XO	addme	RT, RA	1031
			addme.		
			addmeo		
			addmeo.		
31	235 (747)	XO	mullw	RT, RA, RB	1150
			mullw.		
			mullwo		
			mullwo.		
31	246	X	dcbtst	RA, RB	1069
31	247	X	stbux	RS, RA, RB	1177
31	262	X	icbt	RA, RB	1081
31	266 (778)	XO	add	RT, RA, RB	1024
			add.		
			addo		
			addo.		
31	278	X	dcbt	RA, RB	1067
31	279	X	lhzx	RT, RA, RB	1100
31	284	X	eqv	RA, RS, RB	1077
			eqv.		
31	311	X	lhzux	RT, RA, RB	1099
31	316	X	xor	RA, RS, RB	1212
			xor.		
31	323	XFX	mfdcr	RT, DCRN	1128
31	339	XFX	mf spr	RT, SPRN	1130
31	343	X	lhax	RT, RA, RB	1095
31	371	XFX	mftb	RT, SPRN	1818
31	375	X	lhaux	RT, RA, RB	1094
31	407	X	sthx	RS, RA, RB	1184

Table A-5. PPC440GX Embedded Processor Instructions by Opcode (Continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	412	X	orc	RA, RS, RB	1161
			orc.		
31	439	X	sthux	RS, RA, RB	1183
31	444	X	or	RA, RS, RB	1160
			or.		
31	451	AFX	mtdcr	DCRN, RS	1136
31	454	X	dccci	RA, RB	1071
31	459 (971)	XO	divwu	RT, RA, RB	1075
			divwu.		
			divwuo		
			divwuo.		
31	467	AFX	mtspr	SPRN, RS	1138
31	470	X	dcbi	RA, RB	1066
31	476	X	nand	RA, RS, RB	1151
			nand.		
31	486	X	dcread	RT, RA, RB	1072
31	491 (1003)	XO	divw	RT, RA, RB	1074
			divw.		
			divwo		
			divwo.		
31	512	X	mcrxr	BF	1126
31	530		Reserved-nop		1819
31	533	X	lswx	RT, RA, RB	1104
31	534	X	lwbrx	RT, RA, RB	1107
31	536	X	srw	RA, RS, RB	1175
			srw.		
31	562		Reserved-nop		1819
31	566	X	tlbsync		1204
31	594		Reserved-nop		1819
31	597	X	lswi	RT, RA, NB	1102
31	598	X	msync		1134
31	626		Reserved-nop		1819
31	658		Reserved-nop		1819
31	661	X	stswx	RS, RA, RB	1187
31	662	X	stwbrx	RS, RA, RB	1189
31	690		Reserved-nop		1819
31	722		Reserved-nop		1819
31	725	X	stswi	RS, RA, NB	1185
31	754		Reserved-nop		1819
31	758	X	dcba	RA, RB	1064
31	790	X	lhbrx	RT, RA, RB	1096
31	792	X	sraw	RA, RS, RB	1173
			sraw.		

Preliminary User's Manual

Table A-5. PPC440GX Embedded Processor Instructions by Opcode (Continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	824	X	srawi	RA, RS, SH	1174
			srawi.		
31	854	X	mbar	MO	1124
31	914	X	tlbsx	RT, RA, RB	1203
			tlbsx.		
31	918	X	sthbrx	RS, RA, RB	1181
31	922	X	extsh	RA, RS	1079
			extsh.		
31	946	X	tlbre	RT, RA, WS	1201
31	954	X	extsb	RA, RS	1078
			extsb.		
31	966	X	iccci	RA, RB	1083
31	978	X	tlbwe	RS, RA, WS	1205
31	982	X	icbi	RA, RB	1080
31	998	X	icread	RA, RB	1084
31	1014	X	dcbz	RA, RB	1070
32		D	lwz	RT, D(RA)	1108
33		D	lwzu	RT, D(RA)	1109
34		D	lbz	RT, D(RA)	1088
35		D	lbzu	RT, D(RA)	1089
36		D	stw	RS, D(RA)	1188
37		D	stwu	RS, D(RA)	1192
38		D	stb	RS, D(RA)	1176
39		D	stbu	RS, D(RA)	1177
40		D	lhz	RT, D(RA)	1097
41		D	lhzu	RT, D(RA)	1098
42		D	lha	RT, D(RA)	1092
43		D	lhau	RT, D(RA)	1093
44		D	sth	RS, D(RA)	1180
45		D	sthu	RS, D(RA)	1182
46		D	lmw	RT, D(RA)	1101
47		D	stmw	RS, D(RA)	1185

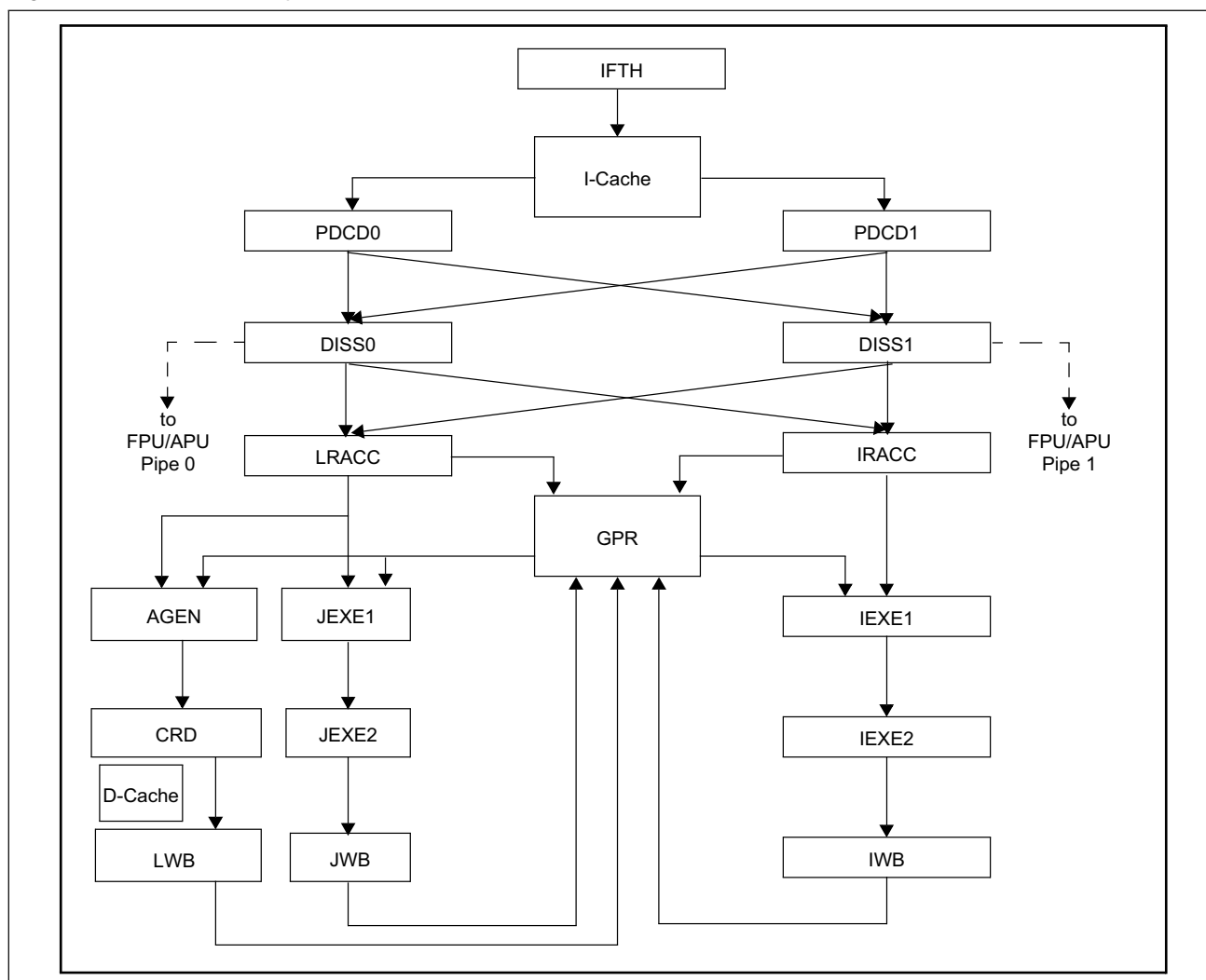
Preliminary User's Manual**Appendix B. Instruction Execution Performance and Code Optimizations**

The instruction timing information and code optimization guidelines provided in this appendix can help compiler developers and application programmers produce high-performance code and accurately analyze instruction execution performance. While this appendix does not comprehensively identify every micro-architectural characteristic that could have a potential impact on instruction execution time within the PPC440GX, it does provide a high-level overview of basic instruction operation and pipeline performance. The information provided is sufficient to analyze the performance of code sequences to a high degree of accuracy.

B.1 PPC440GX Pipeline Overview

PPC440GX processor core is a superscalar processor core capable of issuing two instructions per cycle to any two of the core's three execution pipelines (complex integer, simple integer, and load/store). *Figure B-1* provides an illustration of the 7-stage pipeline of the PPC440GX.

Figure B-1. PPC440GX Pipeline Structure



As illustrated in *Figure B-1*, the first three pipeline stages are common and provide the fetch (IFTH), decode (PDCD0/PDCD1), and issue (DISS0/DISS1) of up to two instructions per cycle. The next pipeline stage (LRACC/IRACC) provides register access and dispatch of up to two instructions per cycle, with the IRACC stage being dedicated to the complex integer (I-pipe) pipeline, and the LRACC stage being shared between the simple integer (J-pipe) and load/store (L-pipe) pipelines. The final three pipeline stages are replicated for each pipeline, and provide for the execution of the instructions.

In the IFTH pipeline stage, the next two instructions are fetched from the I-cache, unless the instruction fetch address has reached the last instruction in the cache line (32 bytes), in which case only one instruction can be fetched and the first instruction of the next cache line can be fetched in the next cycle. The Branch Target Address Cache (BTAC) and the Branch History Table (BHT) are also accessed in the IFTH stage.

In the PDCD pipeline stage, the two instructions which were just fetched are decoded and sent on to the DISS stage.

The DISS pipeline stage is actually a queue with four positions (DISS0 - DISS3), although only the oldest two positions (DISS0 and DISS1) are eligible for issue to the RACC stage, and thus only these positions are illustrated in the pipeline figure. During the DISS stage, a determination is made as to which pipeline is required by each of the two instructions in DISS0 and DISS1, and the instructions then *issue* to the RACC stage accordingly (the term "issue" refers specifically to the action of moving from either DISS0 or DISS1 to either the LRACC or the IRACC stage). An instruction in DISS1 (which by design is guaranteed to be *newer* than an instruction in DISS0) may issue *out-of-order* with respect to an instruction in DISS0, in the event that the instruction in DISS0 requires a RACC stage which is not currently available while the RACC stage required by the instruction in DISS1 is available. See *Section B.3 Instruction Issue Operation* on page 1832 for more information on how instructions issue to the RACC stage.

The LRACC and IRACC pipeline stages are generally where instructions hold waiting for their source operands to become ready. This is the case for all source operands except the data operand for store instructions (for which the store instruction will hold in the AGEN stage) and the accumulate operand for integer multiply-accumulate instructions (for which the integer multiply-accumulate instruction will hold in the IEXE1 stage). Once its source operands become ready, the instruction will *dispatch* to the first execution stage of the corresponding pipeline (the term "dispatch" refers specifically to the action of moving from one of the RACC stages to the first execution stage of a pipeline -- either IEXE1, JEXE1, or AGEN). Instructions in the RACC stages may dispatch out-of-order with respect to each other, in the event that the pipeline required by the newer instruction becomes available prior to the pipeline required by the older instruction. Due to the out-of-order issue capability from the DISS stage, there is no guaranteed relative order between the two instructions in the RACC stage. However, the sequence of instructions contained *within a given execution pipeline* are always guaranteed to be in-order with respect to each other, since they have to pass through the same RACC stage in order to enter the execution pipeline.

The last three stages of the pipeline are unique for each of the three pipelines. In the L-pipe, the AGEN stage is where addresses are generated, and also where store instructions hold waiting for the store data operand to become ready. The CRD stage is where the D-cache is accessed to determine whether the target location exists in the cache, and to obtain load data. The LWB stage is where load hit data is written back to the GPR file, and where store hit data is written back to the D-cache.

In the I-pipe, the IEXE1 stage is the first cycle of instruction execution, and for most operations, the result is available to be forwarded from the end of this stage to subsequent instructions requiring the result as a source operand. Such operations which calculate their result completely in IEXE1 simply proceed down through IEXE2 and into IWB, from which they write their result back to the GPR file. IEXE1 is also where integer multiply-accumulate instructions hold waiting for the accumulate source operand to become ready. Some operations (such as multiply and divide instructions) must continue to execute in IEXE2 and IWB in order to fully calculate their results. Divide instructions actually reside in IWB for 33 cycles as they iteratively calculate their result, at which point they write the result back to the GPR file.

Preliminary User's Manual

In the J-pipe, all instructions which can utilize the J-pipe are also capable of calculating their final result in the first execution stage JEXE1. These instructions then simply proceed down through JEXE2 and into JWB, from which they write their result back to the GPR file.

The subsequent sections of this appendix provide additional details regarding the performance of various instruction sequences, including the simultaneous issue capabilities and latencies of various instruction pairs.

B.2 Instruction Execution Latency and Penalty

The term *latency* refers to the number of cycles of execution required for a given instruction to produce its “result”, typically the value to be written to the target general-purpose register (GPR) specified as part of the instruction. Most integer instructions (such as the standard arithmetic and logical instructions) have *1-cycle latency*, meaning that their results are “ready” at the end of the first execution stage of the pipeline, and thus available to be *forwarded* (delivered) to any subsequent instruction which may require that result as one of the subsequent instruction’s source operands. One significant exception to this is the *load* instruction category. These instructions have *3-cycle latency* (assuming the target memory location is found in the data cache), with their results becoming available at the end of the third execution stage of the pipeline. Various other special cases exist, and are described in more detail in *Section B.4 Instruction Pair Execution Performance Rules* on page 1834

The term *penalty* refers to the number of processor cycles for which a given instruction cannot proceed down the processor pipeline, due to a *dependency* between itself and an immediately preceding instruction. In other words, if a source operand for a given instruction is the same as the target operand for the preceding instruction, then the given instruction may have to “hold” in the operand access pipeline stage for some number of cycles waiting for its source operand to become “ready”, depending on the latency of the preceding instruction. For example, if a source operand for a given instruction is the same as the target operand of an immediately preceding load instruction, which has three-cycle latency, then there would be a two-cycle *penalty* associated with the given instruction, as it “waits” at the operand access stage of the pipeline for two extra cycles, in order for the load instruction to reach the third execution stage and forward its result to the given instruction. In contrast, if the earlier instruction has one-cycle latency, there would be a zero-cycle penalty (no penalty) associated with the dependent instruction, as the dependent instruction would be able to proceed down the pipeline immediately after the earlier instruction, which would forward its result from the first execution stage of the pipeline.

Generally speaking, if a given instruction immediately follows the instruction on which it is dependent, then the number of penalty cycles will be one less than the number of cycles of latency associated with the earlier instruction. There are exceptions to this rule, however, and these are summarized in more detail later in this appendix.

Note that the concept of penalty associated with the instructions in a given code sequence is generally defined to be relative to the performance of a code sequence consisting entirely of non-dependent and/or 1-cycle latency instructions, and executing on a *single-issue* processor micro-architecture. In other words, the “base” performance level is one in which an instruction sequence executes with one instruction completing every cycle. If there are no dependencies between instructions, or every instruction has only 1-cycle latency, then a single-issue micro-architecture will be able to complete one instruction every cycle.

Since the PPC440GX is a dual-issue micro-architecture, it is possible for certain instruction sequences to execute at a rate of more than one instruction per cycle, up to a maximum of two instructions per cycle. Thus, the penalty associated with such an instruction stream could be viewed as being “less than zero”, relative to the single-issue micro-architecture. To illustrate with a couple of examples, consider a sequence of two instructions, an add followed by a subtract, which have no dependency between them (that is, neither of the source operands of the subtract are the same as the target operand of the add). These two instructions could issue and dispatch at the same time to two different pipelines, giving a performance rate of two instructions per cycle, whereby the “penalty” associated with the subtract instruction could be considered to be “negative one cycle” in that it was able to issue *at the same time* as the add instruction, rather than one cycle later.

Now consider the same example, but with the subtract instruction dependent on the add. Now although the two instructions could still issue at the same time to the two RACC stages, they can no longer dispatch at the same time, since the subtract instruction will have to wait in the RACC stage for one extra cycle, while the add instruction dispatches to the first execute cycle and makes its result available as a source operand for the subtract instruction. Since the two instructions must dispatch in two separate but consecutive cycles, the performance is equivalent to a single-issue micro-architecture, and thus exhibits a relative penalty of zero cycles.

Lastly, consider a modified sequence where the add instruction is replaced by a load, with the subtract instruction still dependent on the load. In this case, again the instructions can still issue to the two different RACC stages in the same cycle (with the load necessarily going to LRACC while the subtract goes to IRACC), and the subtract instruction must hold for one cycle in IRACC while the load instruction dispatches to AGEN. But in this case however, the subtract instruction must also hold in IRACC for two additional cycles (the penalty cycles) before it dispatches, three cycles after the load instruction dispatched, in order for the load instruction to have reached the LWB stage and made its result available (assuming a D-cache hit) as a source operand to the subtract instruction. This gives a total of two instructions dispatched in four cycles, or two cycles worse than the baseline non-dependent single-issue performance, and thus a penalty of two cycles.

B.3 Instruction Issue Operation

The PPC440GX can generally issue two instructions in any given cycle to the RACC stage of the pipeline. The two oldest instructions in the issue queue stage of the pipeline (DISS0 and DISS1) are examined to determine which RACC stage they require (LRACC for the L-pipe or the J-pipe, and IRACC for the I-pipe). If they require (or can use) different RACC stages, then they may issue together. Conversely, if both instructions require the same RACC stage then they must issue in separate cycles, with the older instruction issuing first. Certain instruction types must use LRACC and the L-pipe (such as storage access instructions), certain instructions must use IRACC and the I-pipe (such as branch and multiply instructions), and the rest of the instructions can use either LRACC/J-pipe or IRACC/I-pipe. This section summarizes the pipelines which may or must be used by each of the instruction categories, and the rules regarding the simultaneous issuing of instructions.

See *Section 4.4 Implemented Instruction Set Summary* on page 171 for a summary of the instructions in the various instruction categories.

B.3.1 L-Pipe Instructions

The following categories of instructions must utilize the LRACC dispatch stage and be executed by the L-pipe:

- Storage access

Note that “update” forms of load/store instructions, which update the base address register used for the load/store operation, are executed simultaneously by both the L-pipe and the J-pipe. For such instructions, the storage access operation utilizes the L-pipe, while the base address update operation uses the J-pipe.

- Cache management
- Storage synchronization
- Allocated cache management
- Allocated cache debug

B.3.2 I-Pipe Instructions

The following categories of instructions must utilize the IRACC dispatch stage and be executed by the I-pipe:

- Integer multiply
- Integer divide

Preliminary User's Manual

- Allocated arithmetic (includes multiply-accumulate, negative multiply-accumulate, and multiply halfword)
- Integer trap
- Integer count leading zeros
- Allocated logical (**dlimzb**)
- TLB management
- Branch
- Processor control (includes processor synchronization, register management, system linkage, and condition register logical instructions)
- All instructions which use and/or update the CR (see *Table 4-43 CR Updating Instructions* on page 184)

Note: the **stwcx.** instruction is both a storage access and a CR-updating instruction, and hence it is executed by *both* the L-pipe and the I-pipe. It simultaneously issues from DISS0 to both LRACC and IRACC.

- All instructions which use and/or update the XER (see *Table 4-44 XER[SO,OV] Updating Instructions* on page 187 and *Table 4-45 XER[CA] Updating Instructions* on page 188)

Note: the load/store string indexed (**lswx** and **stswx**) instructions are exceptions to this rule, in that they use the XER[TBC] field but are executed by the L-pipe.

B.3.3 I-Pipe/J-Pipe Instructions

All other integer instructions may utilize either the IRACC or the LRACC dispatch stage and be executed by either the I-pipe or the J-pipe, respectively. These include:

- Integer arithmetic instructions which do *not* utilize the CR or the XER, except for multiply and divide instructions (which must be executed by the I-pipe)
- Integer logical instructions which do *not* update the CR, except for count leading zeros instructions (which must be executed by the I-pipe)
- Integer rotate instructions which do *not* update the CR
- Integer shift instructions which do *not* update the CR or the XER

B.3.4 Instruction Issue Rules

When determining to which RACC stage(s) the two instructions in DISS0 or DISS1 should be issued, the following rules apply:

- If both instructions require the same RACC stage, then only the DISS0 instruction will issue, with the DISS1 instruction moving to DISS0 and the next instruction moving into DISS1
- If one of the two DISS instructions requires a particular RACC stage, and the other instruction can use either RACC stage, then the instruction requiring the particular RACC stage will issue to it, and the other instruction will issue to the other RACC stage.
- If neither instruction requires a particular RACC stage, then DISS0 will by default issue to LRACC and DISS1 will by default issue to IRACC.
- The **stwcx.** instruction requires both the LRACC and the IRACC dispatch stages, as it both accesses storage and updates the CR. Therefore, this instruction cannot issue from DISS1, but must wait until it enters DISS0 and then can simultaneously issue to both LRACC and IRACC.
- Due to various architectural requirements regarding context synchronization and interrupt ordering, the following instructions are treated uniquely with regards to issuing: **isync**, **mtmsr**, **rfi**, **rfci**, **rfmci**, **sc**, **wrtee**, and **wrteei**. These instructions all must wait until they occupy DISS0 before issuing to the IRACC stage. Further-

more, these special instructions each block any subsequent instructions from issuing until the special instruction has completed execution, at which time all subsequent instructions are flushed from the pipelines and re-fetched, at the appropriate address according to the functional definition of the particular instruction. This behavior effectively increases the latency associated with these instructions, and the penalty associated with this extra latency is described in *Section B.4.15 Processor Control Instruction Operation* on page 1840.

B.4 Instruction Pair Execution Performance Rules

In general, most sequences of two non-dependent instructions which do not require the same RACC stage can be simultaneously issued, dispatched, executed, and completed, resulting in a net execution performance of two instructions in one cycle (corresponding to a *penalty* of “negative one” cycle, relative to the single-issue micro-architecture model of two instructions in two cycles; see the definition of “penalty” in *Section B.2 Instruction Execution Latency and Penalty* on page 1831). There are, however, many instruction sequence scenarios where such parallel instruction processing is not possible, due to various factors such as dependencies between the instructions, contention for the same RACC stage and/or execution pipeline, and so on.

This section summarizes the exception cases, identifying those instruction sequences for which simultaneous instruction processing of one form or another is not possible, thereby leading to an increase in the number of cycles required to process the instructions (a decrease in the instructions per cycle metric). These penalty cycles are generally due to one or the other of the two instructions having to hold in a given pipeline stage for more than a cycle, while a dependency or pipeline resource conflict is resolved. For any given sequence of two instructions, if the sequence is not covered by one of the rules listed in this section, then it may be assumed that the two instructions can be processed simultaneously at the rate of two instructions per cycle.

It is important to note that calculating the total number of cycles to execute a sequence of greater than two instructions is not simply a matter of adding up the number of cycles identified in these rules for each of the consecutive instruction pairs in the sequence. Rather, the out-of-order issue, dispatch, execution, and completion capabilities of the PPC440GX make it possible in most cases for the cycles associated with any given instruction pair to be overlapped to varying degrees with the cycles associated with another instruction pair. To illustrate with a simple example, consider a sequence of two non-dependent load instructions followed by two non-dependent branch instructions, for which each pair of instructions is subject to exception rule #1 in the preceding list, in that both instructions in each pair require the same execution pipeline, and thus each pair effectively requires two cycles to complete. However, since the first branch instruction can be issued together with the second load instruction, the net throughput for these four instructions would be three cycles, not four. Similarly, the first load instruction *might* issue together with the immediately preceding instruction, and the second branch instruction *might* issue together with the immediately following instruction, potentially maintaining the theoretical maximum execution rate of two instructions per cycle, when considered in the context of the overall instruction sequence.

Also note that when considering the penalty associated with the execution of any given pair of instructions, where the second instruction has some form of dependency on the immediately preceding instruction, this penalty can generally be reduced or avoided altogether by inserting other, non-dependent instructions between the pair. For example, consider the previously mentioned case of a load instruction followed immediately by an instruction dependent on the load result. These two instructions take four cycles to execute, for a penalty of two cycles. As described above, if the load instruction is able to issue together with the instruction preceding it, and the second (dependent) instruction is able to issue together the instruction following it, then the net execution time is four cycles for the *four* instructions, or a zero cycle penalty (net of two cycles per two instructions, the same as the single-issue micro-architecture default). However, if the software sequence were able to be changed such that four more non-dependent instructions were to be inserted between the load and the dependent instruction, then the net execution performance could be increased back to eight instructions for the four cycles (the load and its preceding instruction, the four instructions after the load, the dependent instruction and its successor). This corresponds to the default “negative one” cycle penalty for the two-issue micro-architecture model, or one cycle per two instruc-

Preliminary User's Manual

tions. Generally speaking, compilers should whenever possible attempt to eliminate the penalties associated with the instruction pairings described in the following sections by inserting non-dependent but still useful instructions between the penalty-inducing pair.

Note: Some of the execution performance rules described in the following subsections are related to CR dependencies. When considering these dependencies, there is a special case that should be noted. Specifically, condition register logical instructions specify two bits of the CR as source operands, and specify a third bit of the CR as the target operand. However, since the PPC440GX updates the CR on a *field* (as opposed to a *bit*) basis, the field containing the *target* bit operand is actually considered a *source* operand for the sake of any of the CR-related dependency rules described in the following subsections. This is necessary in order to source the “old” value of the three bits of the target field which are *not* being updated by the condition register logical instruction.

Note: Some of the execution performance rules described in the following subsections are related to XER dependencies. Specifically, various “o” form instructions update XER[SO,OV], and various other instructions read XER[SO] and/or XER[OV] as a source operand. These instructions that use XER[SO] and/or XER[OV] as a source operand are: **mfspr** (with the XER specified as the source SPR), **mcrxr**, compare instructions (which copy XER[SO] into CR[CR0]₃), and all “record” form instructions (which copy XER[SO] into CR[CR0]₃).

B.4.1 Contention for the Same RACC Stage

If the two instructions require the same RACC stage, then they must issue in separate cycles and thus their effective throughput is two cycles for the two instructions. This corresponds to a “penalty” of zero cycles, one cycle worse than the “negative one” cycle penalty for the default, non-contention case where the two instructions can issue together.

B.4.2 General GPR Operand Dependency

If the second instruction has a GPR source operand that is the same as one of the first instruction's GPR target operands (that is, a GPR read-after-write (RaW) hazard), then in general the second instruction must execute at least one cycle after the first instruction, thereby requiring at least two cycles to execute the two instructions (zero cycles of penalty). Depending on the instruction type of the first instruction (and the pipeline stage at which it finishes calculating its result), the second instruction might have to wait more than one cycle for its source operand to become ready, thereby increasing the penalty for the two instruction sequence even further. The circumstances that result in such additional delay are described in rules listed later in this section.

Two exceptions to this general rule exist. These exceptions are for integer store instructions and the allocated integer multiply-accumulate (MAC) instructions. Specifically, when the second instruction of the sequence is either a store instruction or a MAC instruction, and the source GPR operand of the second instruction which matches the target GPR operand of the first instruction is specifically the store *data* operand (that is, the RS operand shown in the store instruction description) or the MAC *accumulate* operand (that is, the RT operand shown in the MAC instruction description, which is both a source and a target for MAC instructions), respectively, then the two instructions will still generally be able to execute and complete in parallel, such that the effective throughput will still generally be one cycle for the two instructions, which is equivalent to the non-dependent case.

This parallel execution is generally possible because the store data operand and the MAC accumulate operand are each accessed one cycle later in the execution pipeline than are other GPR source operands (see *Section B.1 PPC440GX Pipeline Overview* on page 1829). As is the case for the general GPR operand dependency rule described in the preceding paragraph, however, and depending on the instruction type of the first instruction, there may be additional delay in the calculation of the first instruction's result and hence additional penalty in the execution of the two-instruction sequence in such cases, even for the special case of the second instruction being one of these two types. Again, the circumstances under which this additional penalty applies are described in rules listed later in this section. In general though, the GPR dependency-related penalty for the special case of the dependency being for the store data or MAC accumulate operand will be one cycle less than the standard GPR dependency-related penalty.

B.4.3 General CR Operand Dependency

There is no need for a separate general rule for execution penalty associated with CR operand dependencies, corresponding to the general rule for GPR dependencies. This is because all instructions which utilize the CR (either as a source or as a target operand) must issue to the IRACC pipeline stage and be executed by the I-pipe. Therefore, the RACC contention rule described in *Section B.4.1* applies to all instruction sequences involving such CR dependencies, leading to a “default” base execution rate of two cycles for the two instructions with such a dependency. For example, the sequence of a compare instruction (which writes a field of the CR as a target) followed by a conditional branch (which reads a bit of the CR as a source) takes two cycles to execute. This is true whether or not the branch instruction is actually conditional upon a CR bit which was updated by the compare instruction (or indeed, whether or not the branch is even conditional). Of course for branch instructions there are other considerations, related to the predicted outcome of the branch and the latency with which the instructions *subsequent* to the branch may be executed. Also, as is the case with GPR dependencies, there are other special cases involving instructions which do not calculate their CR results in the first cycle of execution (IEXE1 pipeline stage), and hence which introduce additional cycles of penalty when the subsequent instruction is dependent on those CR results. These special cases are covered in the rules listed later in this section.

B.4.4 Multiply Dependency

Multiply instructions (including the PowerPC-architected 32-bit x 32-bit multiply instructions and the allocated 16-bit x 16-bit multiply-halfword instructions) calculate their results in the IWB pipeline stage (including the GPR result, and the CR result for “record” forms of multiply which update the CR, and the XER result for “o” forms of multiply which update XER[SO,OV]). Therefore, instruction sequences consisting of a multiply followed immediately by an instruction which uses the multiply result (either the GPR, CR, or XER result) as an input operand will take four cycles to complete, which corresponds to a two-cycle penalty, or two cycles *more* than the penalty for the general GPR dependency rule described in *Section B.4.2* on page 1835. Also note that as described in that section, if the dependency involved in the sequence is specifically a store *data* GPR operand, then the penalty is one cycle less, or a total execution time of three cycles, not four. The same is true if the first instruction is a PowerPC-architected 32-bit x 32-bit multiply instruction and the second instruction is a MAC instruction, with the only dependency between the two being the MAC accumulate GPR operand (the total execution time is three cycles).

However, unlike what is described in *Section B.4.2*, if the first instruction in the sequence is specifically a multiply-halfword instruction and the second instruction is a MAC instruction using the GPR result of the multiply-halfword instruction as the accumulate operand, then the penalty associated with the sequence is zero cycles, or a total execution time of two cycles for the two instructions, not four.

B.4.5 Multiply-Accumulate (MAC) Dependency

MAC instructions calculate their results in the IWB pipeline stage (including the GPR result, and the CR result for “record” forms of MAC which update the CR, and the XER result for “o” forms of MAC which update XER[SO,OV]). Therefore, instruction sequences consisting of a MAC instruction followed immediately by an instruction which uses the MAC result (either the GPR, CR, or XER result) as an input operand will generally take four cycles to complete, which corresponds to a two-cycle penalty, or two cycles *more* than the penalty for the general GPR dependency rule described in *Section B.4.2* on page 1835. Also note that as described in that section, if the dependency involved in the sequence is specifically a store *data* GPR operand, then the penalty is one cycle less, or a total execution time of three cycles, not four.

However, unlike what is described in *Section B.4.2*, if the second instruction in the sequence is another MAC instruction using the same GPR accumulate operand (and there is no XER[SO] dependency between the instructions), then the penalty associated with the sequence is zero cycles, or a total execution time of two cycles for the two instructions, not four. In other words, MAC instructions with the only dependency between them being the GPR accumulate operand can be executed with single-cycle throughput, due to a special forwarding path within the execution pipeline.

Preliminary User's Manual

Lastly, due to a *write-after-read* (WaR) hazard, instruction sequences consisting of a MAC instruction *preceded* immediately by an instruction which updates the same GPR as the MAC instruction updates will generally take two cycles to complete, which corresponds to a zero-cycle penalty.

B.4.6 Divide Dependency and Pipeline Stall

Divide instructions iteratively calculate their results (including the GPR result, and the CR result for “record” forms of divide which update the CR, and the XER result for “o” forms of divide which update XER[SO,OV]) while remaining in the IWB stage for a total of 33 cycles. Therefore, instruction sequences consisting of a divide followed immediately by an instruction which uses the divide result (either the GPR, CR, or XER result) as an input operand will take 36 cycles to complete, which corresponds to a 34-cycle penalty, or 34 cycles *more* than the penalty for the general GPR dependency rule described in *Section B.4.2* on page 1835. Also note that as described in that section, if the dependency involved in the sequence is specifically a store *data* or MAC *accumulate* GPR operand (and there is no XER[SO] dependency between the instructions), then the penalty is one cycle less, or a total execution time of 35 cycles, not 36.

Furthermore, since divide instructions occupy the IWB pipeline stage for a total of 33 cycles (instead of the standard one cycle), they impose an additional 32-cycle penalty on *any* immediately succeeding instruction that also uses the I-pipe, regardless of any dependency that may exist. That is, any instruction sequence involving a divide instruction followed immediately by another instruction that uses the I-pipe will take a minimum of 34 cycles to execute, or a total penalty of 32 cycles. This includes the zero-cycle penalty associated with contention for the same RACC stage (the rule described in *Section B.4.1* on page 1835), plus the 32 additional cycles of penalty caused by the divide occupying the IWB stage for 33 cycles instead of just one cycle.

On the other hand, instructions subsequent to the divide which utilize the L-pipe or J-pipe and are not dependent on the result of the divide can be executed and completed while the divide is iterating in the IWB pipeline stage.

B.4.7 Move To Condition Register Fields (mtcrf) Instruction Dependency

Due to the nature of the **mtcrf** instruction, which can update any combination of the eight, 4-bit CR fields at once, subsequent instructions which utilize *any* bit or field of the CR as a source must wait for the preceding **mtcrf** instruction to complete before dispatching from the IRACC stage. Therefore, the total execution time for a **mtcrf** instruction followed by an instruction using the CR as a source operand is five cycles, or a penalty of three cycles. Note that this penalty applies whether or not the **mtcrf** instruction is actually updating any of the CR bits or fields being used as source operands by the subsequent instruction.

The following instructions use the CR as a source operand and hence are subject to this three-cycle penalty when they immediately follow a **mtcrf** instruction:

- **bc**, **bclr**, **bcctr** (with BO[0]=0)
- **mfcrr**
- **mcrf**
- Condition register logical instructions (**crand**, **cror**, **crnand**, **crnor**, **crandc**, **crorc**, **crxor**, **creqv**)
- **isel**

B.4.8 Store Word Conditional Indexed (stwcx.) Instruction Dependency

Due to the nature of the **stwcx.** instruction, which conditionally performs a storage access in addition to updating CR[CR0], subsequent instructions which utilize any bit of CR[CR0] as a source operand must wait for the preceding **stwcx.** instruction to complete before dispatching from the IRACC stage. Therefore, the total execution time for a **stwcx.** instruction followed by an instruction using any bit of CR[CR0] as a source operand is five cycles, or a penalty of three cycles.

The following instructions potentially use CR[CR0] (either the whole field or a single bit of the field) as a source operand and if so are subject to this three-cycle penalty when they immediately follow a **stwcx.** instruction:

- **bc, bclr, bcctr** (with BO[0]=0)
- **mfcrr**
- **mcrf**
- Condition register logical instructions (**crand, cror, crnand, crnor, crandc, crorc, crxor, creqv**)
- **isel**

B.4.9 Move From Condition Register (mfcrr) Instruction Dependency

Since the **mfcrr** instruction reads all eight CR fields at once, and since there can be multiple CR-updating instructions in execution at one time, the **mfcrr** instruction must wait until all preceding CR updates have completed before beginning execution. Therefore, any two-instruction sequence involving a CR-updating instruction followed immediately by a **mfcrr** instruction will take four cycles to execute, or a penalty of two cycles.

See *Table 4-43 CR Updating Instructions* on page 184 for a list of instructions that update the CR. Note that although the **mtcrr** instruction is included in this table, the actual penalty for the sequence of **mtcrr** followed immediately by **mfcrr** is three cycles not two, as described in *Section B.4.7* above. Similarly, although the **stwcx.** instruction is included in the table as well, the actual penalty for the sequence of **stwcx.** followed immediately by **mfcrr** is three cycles not two, as described in *Section B.4.8* above.

B.4.10 Move From Special Purpose Register (mfspr) Dependency

The **mfspr** instruction provides its result in the IEXE2 pipeline stage. Therefore, instruction sequences consisting of a **mfspr** followed immediately by an instruction which uses the target GPR of the **mfspr** instruction as an input operand will generally take three cycles to complete, which corresponds to a one-cycle penalty, or one cycle *more* than the penalty for the general GPR dependency rule described in *Section B.4.2* on page 1835. Also note that as described in that section, if the dependency involved in the sequence is specifically a store *data* or MAC *accumulate* operand, then the penalty is one cycle less, or a total execution time of two cycles, not three.

However, this rule applies only to SPRs other than the LR, CTR, or XER. For these three SPRs, the result of **mfspr** is available in the IEXE1 stage and therefore the general GPR dependency rule of *Section B.4.2* on page 1835 applies.

B.4.11 Move From Machine State Register (mfmsr) Dependency

The **mfmsr** instruction provides its result in the IEXE2 pipeline stage. Therefore, the same rule described in *Section B.4.10* above for **mfspr** applies to the **mfmsr** instruction as well.

B.4.12 Move To Special Purpose Register (mtspr) Dependency and Pipeline Stall

Mtspr instructions occupy the IWB stage for a total of three cycles, and do not perform the write of the target SPR until this third cycle, in order to enforce various architectural rules regarding instruction ordering. Therefore, instruction sequences consisting of a **mtspr** followed immediately by a **mfspr** that references the *same* SPR will take six cycles to complete, which corresponds to a four-cycle penalty. However, this penalty does not apply to the LR, CTR, or XER registers. Special handling within the execution pipeline allows a **mtspr/mfspr** sequence that involves one of these three registers to operate in two cycles, thereby incurring only the zero-cycle penalty due to both instructions requiring the I-pipe.

Similarly, when a **mtspr** instruction that specifically targets the MMUCR is followed immediately by a **tlbsx** instruction (which uses some fields of the MMUCR as input operands), the sequence will also take six cycles to complete.

Preliminary User's Manual

Furthermore, since **mtspr** instructions occupy the IWB pipeline stage for a total of three cycles (instead of the standard one cycle), they impose an additional two-cycle penalty on *any* immediately succeeding instruction that also uses the I-pipe, regardless of any dependency that may exist. That is, any instruction sequence involving a **mtspr** instruction followed immediately by another instruction that uses the I-pipe will take a minimum of four cycles to execute, or a total penalty of two cycles. However, this penalty again does not apply to the LR, CTR, or XER, nor does it apply to the SPRG registers (SPRG0 - SPRG7 and USPRG0). Special handling within the execution pipeline allows a **mtspr** instruction which targets one of these registers to move through the pipeline in the normal fashion, occupying the IWB stage for only one cycle.

Also, instructions subsequent to the **mtspr** which utilize the L-pipe or J-pipe can be executed and completed while the **mtspr** is continuing to occupy the IWB pipeline stage.

B.4.13 TLB Management Instruction Dependencies

In addition to the dependency between a **mtspr** that targets the MMUCR and a subsequent **tlbsx** instruction, for which the penalty is described in *Section B.4.12* on page 1838, there are four other special case dependencies involving TLB management instructions that lead to execution penalties.

First, the **tlbwe** instruction occupies the IWB pipeline stage for a total of three cycles (similar to the **mtspr** instruction). Therefore, any instruction sequence involving a **tlbwe** instruction followed immediately by another instruction that uses the I-pipe will take a minimum of four cycles to execute, or a total penalty of two cycles. However, instructions subsequent to the **tlbwe** which utilize the L-pipe or J-pipe can be executed and completed while the **tlbwe** is continuing to occupy the IWB pipeline stage.

Second, instruction sequences involving a **tlbre** or **tlbsx** instruction followed immediately by a **mfspr** instruction (that targets any SPR *except* the LR, CTR, or XER) take four cycles to complete, corresponding to a penalty of two cycles. This penalty is due to conflicting usage of pipeline resources between the two instructions.

Third, instruction sequences involving a **tlbwe** instruction followed immediately by a **tlbre** or **tlbsx** instruction also take four cycles to complete, corresponding to a penalty of two cycles. This penalty is due to conflicting usage of the TLB array between the two instructions.

Fourth, instruction sequences involving a **tlbre** or **tlbsx** instruction followed immediately by a load, store, cache management (except **dcbz**, which performs no operation on the PPC440GX), cache debug, or storage synchronization instruction, take four cycles to complete, corresponding to a penalty of two cycles. Similarly, if the first instruction is instead a **tlbwe** then the two-instruction sequence takes six cycles to complete, since the **tlbwe** instruction holds in the IWB pipeline stage for two extra cycles. Conversely, if the order of the two instructions is reversed, with the TLB management instruction coming immediately *after* a load, store, cache management, cache debug, or storage synchronization instruction, the two-instruction sequence takes either three or eight cycles to complete (it takes eight cycles if the first instruction is **icbi**, **icbt**, **iccci**, or **icread**, and three cycles otherwise). These penalties are all due to the potential for conflicting usage of the TLB array or other pipeline resources between the two instructions.

B.4.14 DCR Register Management Instruction Dependency and Pipeline Stall

Since the DCR register management instructions (**mtdcr** and **mfdcr**) must interact with the asynchronous DCR interface of the PPC440GX, they stall temporarily within the I-pipe. Specifically, these instructions hold in the IEXE1 pipeline stage as they participate in the asynchronous handshake protocol of the DCR interface. The number of cycles for which these instructions remain in the IEXE1 pipeline stage depends upon the speed with which the DCR device responds to the transaction. In general, a DCR register management instruction will occupy the IEXE1 pipeline stage for two cycles, plus the number of CPU clock cycles associated with the DCR interface clock synchronization and the transaction itself. The number of these extra cycles, beyond the base two cycles, depends on the relative clock frequencies of the CPU clock and the DCR interface clock, and on the number of cycles of the DCR transaction itself. Assume a CPU:DCR clock ratio of $R = C/D$, where C and D are both positive

integers, with $C > D$. Also assume a number of DCR interface transaction cycles (of the DCR clock) M , where M includes all cycles beginning with the one in which the PPC440GX first drives the DCR operation signal active, and ending with the one in which the DCR acknowledge signal is asserted into the PPC440GX by the DCR device. Given these assumptions, the average total number of CPU clock cycles N associated with the clock synchronization and the DCR transaction is given by the equation:

$$N = \text{FLOOR} \left(\sum_{i=1}^C \frac{i}{CD} + MR + 1 \right)$$

Note that as specified above, N is an *average* number of cycles; the actual number can vary slightly according to the specific timing of the DCR request relative to the CPU and DCR clock domains. Accordingly, the DCR register management instructions will occupy the IEXE1 pipeline stage for $N+2$ cycles, thereby leading to a penalty of $N+1$ cycles for any immediately subsequent instruction which must utilize the I-pipe. On the other hand, instructions subsequent to a DCR register management instruction which utilize the L-pipe or J-pipe can be executed and completed while the DCR register management instruction is continuing to occupy the IEXE1 pipeline stage.

Furthermore, the **mfdcr** instruction can not forward its GPR result to a subsequent instruction until the IEXE2 pipeline stage. Therefore, instruction sequences consisting of a **mfdcr** followed immediately by an instruction which uses the **mfdcr** target register as an input operand will generally take $N+4$ cycles to complete, which corresponds to an $N+2$ -cycle penalty. Also note that as described in *Section B.4.2 General GPR Operand Dependency*, if the dependency involved in the sequence is specifically a store *data* or MAC *accumulate* operand, then the penalty is one cycle less, or a total execution time of $N+3$ cycles, not $N+4$.

B.4.15 Processor Control Instruction Operation

Various processor control instructions require special handling within the PPC440GX due to the context synchronization requirements of the PowerPC Book-E architecture. These instructions include:

- **sc**
- **mtmsr**
- **wrtee**
- **wrteei**
- **isync**
- **rfi**
- **rfci**
- **rfmci**

Each of these instructions requires that the instruction stream be flushed and re-fetched immediately after the instruction's execution, either at the next sequential address (for **mtmsr**, **wrtee**, **wrteei**, and **isync**), or at the System Call interrupt vector location (for **sc**), or at the interrupt return address (for **rfi**, **rfci**, and **rfmci**). Due to the instruction re-fetching requirement and other instruction processing requirements, the minimum execution time for a two-instruction sequence involving one of these instructions as the *first* instruction is as follows:

- eight cycles (for **sc**, **wrteei**, **rfi**, **rfci**, and **rfmci**)
- eleven cycles (for **mtmsr**, **wrtee**, and **isync**)

Preliminary User's Manual

Furthermore, none of these instructions can be issued together with *any* preceding instruction, which means that the minimum execution time is two cycles (zero-cycle penalty) for any two-instruction sequence in which the *second* instruction is one of these instructions.

B.4.16 Load Instruction Dependency

Load instructions that obtain their data from the data cache generally provide their result in the LWB pipeline stage. Therefore, instruction sequences consisting of a load instruction followed immediately by an instruction which uses the target GPR of the load instruction as an input operand will generally take four cycles to complete, which corresponds to a two-cycle penalty, or one cycle *more* than the penalty for the general GPR dependency rule described in *Section B.4.2* on page 1835. Also note that as described in that section, if the dependency involved in the sequence is specifically a store *data* or MAC *accumulate* operand, then the penalty is one cycle less, or a total execution time of three cycles, not four. The dependency described by this section applies only to the target *data* operand of a load instruction, and not to the target *address* operand of a load *with update* instruction, for which the result is available from the JEXE1 pipeline stage and hence only the general GRP dependency rule applies.

Note that there are many other factors that affect the performance of load and other storage access instructions (such as whether or not their target location is in the data cache).

Preliminary User's Manual**Index****A**

add, 1024
 add., 1024
 addc, 1024
 addc., 1024
 addco, 1024
 addco., 1024
 adde, 1025
 adde., 1025
 addeo, 1025
 addeo., 1025
 addi, 1027
 addic, 1028
 addic., 1029
 addis, 1030
 addme, 1031
 addme., 1031
 addmeo, 1031
 addmeo., 1031
 addo, 1024
 addo., 1024
 address map
 illustrated, 131
 addressing, 131
 addressing modes, 70, 134
 data storage, 134
 instruction storage, 134
 addze, 1032
 addze., 1032
 addzeo, 1032
 addzeo., 1032
 alignment
 load and store, 220
 Alignment interrupt, 442
 alignment interrupts, 442
 allocated instruction summary, 178
 allocation
 data cache line on store miss, 221
 alphabetical summary of implemented instructions, 1789
 and, 1033
 and., 1033
 andc, 1034
 andc., 1034
 andi., 1035
 andis., 1036
 arithmetic compare, 185
 arrays, shadow TLB, 253
 asynchronous interrupt class, 419
 attributes, storage, 246
 bc, 1038
 bca, 1038
 bcctr, 1044
 bcctrl, 1044
 bcl, 1038
 bcla, 1038
 bclr, 1047
 bclrl, 1047
 bctr, 1044
 bctrl, 1044
 bdnz, 1039
 bdnza, 1039
 bdnzf, 1039
 bdnzfa, 1039
 bdnzfl, 1039
 bdnzfla, 1039
 bdnzflr, 1048
 bdnzflrl, 1048
 bdnzl, 1039
 bdnzla, 1039
 bdnzlr, 1048
 bdnzlrl, 1048
 bdnzt, 1039
 bdnzta, 1039
 bdnztl, 1039
 bdnztla, 1039
 bdnztlr, 1048
 bdnztlrl, 1048
 bdz, 1039
 bdza, 1039
 bdzf, 1039
 bdzfa, 1039
 bdzfl, 1039
 bdzfla, 1039
 bdzflr, 1048
 bdzflrl, 1048
 bdzl, 1039
 bdzla, 1039
 bdzlr, 1048
 bdzlrl, 1048
 bdzt, 1040
 bdzta, 1040
 bdztl, 1040
 bdztla, 1040
 bdztlr, 1048
 bdztlrl, 1048
 beq, 1040
 beqa, 1040
 beqctr, 1045
 beqctrl, 1045
 beql, 1040
 beqlr, 1048
 beqlrl, 1048
 bf, 1040
 bfa, 1040
 bfctr, 1045
 bfctrl, 1045
 bfl, 1040
 bfla, 1040

B

b, 1037
 ba, 1037

- bflr, 1048
- bflrl, 1048
- bge, 1040
- bgea, 1040
- bgectrl, 1045
- bgei, 1040
- bgela, 1040
- bgelr, 1049
- bgelrl, 1049
- bgrctr, 1045
- bgt, 1041
- bgta, 1041
- bgtctr, 1045
- bgtctrl, 1045
- bgtl, 1041
- bgtla, 1041
- bgtlr, 1049
- bgtlrl, 1049
- BI field on conditional branches, 180
- big endian
 - defined, 135
 - structure mapping, 137
- big endian mapping, 136
- bl, 1037
- bla, 1037
- ble, 1041
- blea, 1041
- blectr, 1045
- blectrl, 1045
- blel, 1041
- blela, 1041
- blelr, 1049
- blelrl, 1049
- blr, 1047
- blrl, 1047
- blt, 1041
- blta, 1041
- bltctr, 1045
- bltctrl, 1045
- bltl, 1041
- bltla, 1041
- bltlr, 1049
- bltlrl, 1049
- bne, 1041
- bnea, 1041
- bnctr, 1045
- bnctrl, 1045
- bnel, 1041
- bnela, 1041
- bnelr, 1049
- bnelrl, 1049
- bng, 1042
- bnga, 1042
- bngctr, 1045
- bngctrl, 1045
- bngl, 1042
- bngla, 1042
- bnglr, 1049
- bnglrl, 1049
- bnl, 1042
- bnla, 1042
- bnlctr, 1046
- bnlctrl, 1046
- bnll, 1042
- bnlla, 1042
- bnllr, 1049
- bnllrl, 1049
- bns, 1042
- bnsa, 1042
- bnsctr, 1046
- bnsctrl, 1046
- bnsi, 1042
- bnsia, 1042
- bnsir, 1049
- bnsirl, 1049
- bnu, 1042
- bnua, 1042
- bnuctr, 1046
- bnuctrl, 1046
- bnul, 1042
- bnula, 1042
- bnulr, 1050
- bnulrl, 1050
- BO field on conditional branches, 180
- bootstrap controller, 65
- boundary scan, 504
- Boundary Scan Description Language (BSDL), 504
- branch instruction summary, 175
- branch instructions, exception priorities for, 456
- branch prediction, 181, 1789
- branch processing, 179
- branch taken (BRT) debug events, 519
- branching control
 - BI field on conditional branches, 180
 - BO field on conditional branches, 180
 - branch addressing, 179
 - branch prediction, 181
 - registers, 182
- BSDL, 504
- bso, 1043
- bsoa, 1043
- bsoctr, 1046
- bsoctrl, 1046
- bsol, 1043
- bsola, 1043
- bsolr, 1050
- bsolrl, 1050
- bt, 1043
- bta, 1043
- btctr, 1046
- btctrl, 1046
- bti, 1043
- btla, 1043
- btlr, 1050
- btlrl, 1050
- bun, 1043
- buna, 1043
- bunctr, 1046

Preliminary User's Manual

bunctrl, 1046
 bunl, 1043
 bunla, 1043
 bunlr, 1050
 bunlrl, 1050
 byte ordering, 135
 big endian, defined, 135
 instructions, 137, 138
 little endian, defined, 136
 structure mapping
 big-endian mapping, 136
 little endian mapping, 137

C

cache block, defined, 211
 cache line
 See also cache block
 cache line locking, 203
 cache line replacement policy, 202
 cache locking transient mechanism, 203
 cache management instructions
 summary
 data cache, 225
 instruction cache, 211
 caching inhibited, 247
 CCR0, 191, 212, 213, 227, 1244
 CCR1, 1246
 change status management, 255
 chip reset results, 261
 CIX0_PLBBESR, 659
 CIX0_POM1LAL, 663
 clock and power management, 497
 registers, 497
 clocking, 477
 PCI, 488
 registers, 489
 system, 478
 bypass PLL, 482
 choosing system clock ratios, 483
 clocks for offchip use, 481
 CPU feedback example, 484, 485
 CPU PLB frequency, 482
 feedback selection, 479
 IIC bootstrap controller clocking, 481
 input SysClk, 479
 M value for SYS PLL, 480
 PerClk feedback example, 486
 SYS PLL strapping, 482
 SYS PLL TUNE setting, 481
 system clock ratio examples, 484
 VCO frequency, 480
 clrlslwi, 1168
 clrlslwi., 1168
 clrlwi, 1168
 clrlwi., 1168
 clrrwi, 1168
 clrrwi., 1168
 cmp, 1051
 cmpi, 1052
 cmpl, 1053
 cmpli, 1054
 cmplw, 1053
 cmplwi, 1054
 cmpw, 1051
 cmpwi, 1052, 1124
 cntlzw, 1055
 cntlzw., 1055
 code
 self-modifying, 210
 coherence
 data cache, 225
 coherency
 instruction cache, 209
 compare
 arithmetic, 185
 logical, 185
 Condition Register. *See also* CR
 context synchronization, 196
 control
 data cache, 225
 instruction cache, 211
 conventions
 notational, 47
 core reset results, 261
 CPM0_ER, 498, 1351
 CPM0_FR, 499, 1352
 CPM0_SR, 501, 1353
 CPR0 registers, 286
 CPR0_CFGADDR, 489
 CPR0_CFGDATA, 490
 CPR0_CLKUPD, 287, 490, 1301
 CPR0_ICFG, 286, 1302
 CPR0_MALD, 288, 495, 1303
 CPR0_OPBD, 288, 494, 1304
 CPR0_PERD, 289, 495, 1305
 CPR0_PLLC, 289, 491, 1306
 CPR0_PLLD, 290, 492, 1307
 CPR0_PRIMAD, 291, 493, 1309
 CPR0_PRIMBD, 292, 494, 1310
 CR, 183, 1247
 defined
 CR updating instructions, 184
 instructions
 integer
 CR, 185
 crand, 1056
 crandc, 1057
 crclr, 1063
 creqv, 1058
 Critical Input interrupt, 436
 critical interrupts, 421
 Critical Save/Restore Register 0, 427, 428
 Critical Save/Restore Register 1, 428, 429
 crmove, 1061
 crnand, 1059
 crnor, 1060

crnot, 1060
 cror, 1061
 crorc, 1062
 crset, 1058
 crxor, 1063
 CSRR0, 427, 428, 1248
 CSRR1, 428, 429, 1249
 CTR, 183, 554, 555, 556, 559, 560, 561, 562, 571, 573,
 578, 579, 1250, 1663, 1664, 1671, 1672, 1673,
 1674, 1676, 1677, 1678, 1679, 1681, 1682,
 1685, 1686

D**DAC**

debug events
 applied to instructions that result in multiple storage
 accesses, 516
 applied to various instruction types, 516
 fields, 512
 overview, 512
 processing, 514
 registers
 DAC1–DAC2, 530
 DAC1–DAC2, 530, 1251
 Data Address Compare Register (DAC1), 530
 data address compare *See also* DAC, 512
 data addressing modes, 134
 data cache
 coherency, 225
 data cache array organization and operation, 201
 data cache controller. *See* DCC
 data cache line allocation on store miss, 221
 data read PLB interface requests
 PLB interface, 223
 data read requests, 223
 data storage addressing modes, 134
 Data Storage interrupt, 439
 data storage interrupts, 439
 Data TLB Error interrupt, 446
 data TLB error interrupts, 446
 data value compare *See also* DVC, 517
 data write PLB interface requests
 PLB interface, 223
 data write requests, 223
 DBCR0, 524, 527, 1252
 DBCR1, 526, 1254
 DBCR2, 1256
 DBDR, 531
 DBSR, 528
 dcba
 operation summary, 225
 dcbf, 1065
 operation summary, 226
 dcbi, 1066
 operation summary, 226
 dcbst, 1067
 operation summary, 226

dcbt
 formal description, 1068
 functional description, 227
 operation summary, 226
 dcbt and dcbtst operation, 227
 dcbtst
 formal description, 1069
 functional description, 227
 operation summary, 226
 dcbz, 1070
 operation summary, 226
 DCC (data cache controller)
 control, 225
 debug, 225
 features, 218
 operations, 218
 dccci, 1071
 operation summary, 226
 DCDBTRH, 228, 1261
 DCDBTRL, 228, 1262
 dcread
 functional description, 228, 1072
 operation summary, 226
 DCRs
 defined, 145
 DDR_SDRAM, 545
 commands and operations, 584
 device configuration, 548
 DIMM support, 591
 error checking and correction, 592
 initialization, 580
 interface signals, 545
 page management, 581
 PLB slave interface options, 580
 PLB to memory address code, 580
 power management, 597
 registers
 address map, 547
 DEAR, 1263
 debug
 debug cache, 225
 instruction cache, 211
 debug events
 BRT, 519
 DAC, 512
 DAC fields, 512
 DVC, 517
 DVC fields, 518
 IAC, 508, 520
 IAC fields, 508
 ICMP, 521
 IPRT, 522
 overview, 507
 RET, 520
 summary, 523
 TRAP, 520
 UDE, 522
 Debug Interrupt, 448
 debug interrupts, 448

Preliminary User's Manual

- debug modes
 - debug wait, 507
 - external, 506
 - internal, 506
 - overview, 505
 - trace, 507
 - debug wait mode, 507
 - debugging
 - boundary scan chain, 504
 - debug events, 507
 - debug interfaces, 503
 - JTAG
 - debug port, 503
 - JTAG connector, 503
 - trace status interface, 505
 - debug modes, 505
 - development tool support, 503
 - registers
 - DAC1–DAC2, 530
 - DBCR0, 524, 527
 - DBCR1, 526
 - DBDR, 531
 - DBSR, 528
 - DVC1–DVC2, 531
 - IAC1–IAC4, 530
 - overview, 524
 - reset, 523
 - timer freeze, 523
 - trace port, 505
 - DEC, 461, 1264
 - DECAR, 462, 1265
 - decompression controller
 - access procedures, overview, 150, 1224
 - Decrementer Interrupt, 445
 - decrementer interrupts, 445
 - device control registers, 145
 - Device Control Registers. *See also* DCRs
 - direct write to memory, 221
 - divw, 1074
 - divw., 1074
 - divwo, 1074
 - divwo., 1074
 - divwu, 1075
 - divwu., 1075
 - divwuo, 1075
 - divwuo., 1075
 - dlnmb, 1076
 - dlnmb., 1076
 - DMA controller, 695, 723
 - configuration and status registers, 699
 - external interface signals, 695
 - transfers, 697
 - DMA operations
 - channel priorities, 712
 - data parity, 712
 - errors, 713
 - interrupts, 714, 730
 - peripheral and device paced memory bursts, 712
 - PLB bus timeout, 730
 - programming, 716
 - scatter gather transfers, 715
 - DMA0_CR0–DMA0_CR3, 701, 1354, 1367
 - DMA0_CT0–DMA0_CT3, 703, 1356, 1369
 - DMA0_DAH0–DMA0_DAH3, 705, 1357, 1370
 - DMA0_DAL0–DMA0_DAL3, 1358, 1371
 - DMA0_DAL3–DMA0_DAL3, 705
 - DMA0_POL, 711, 1359, 1372
 - DMA0_SAH0–DMA0_SAH3, 704
 - DMA0_SAH0–DMA0_SAH3, 1360, 1373
 - DMA0_SAH0–DMA0_SAL3, 1361, 1374
 - DMA0_SAL0–DMA0_SAL3, 704
 - DMA0_SGC, 709, 1362, 1375
 - DMA0_SGH0–DMA0_SGH3, 707
 - DMA0_SGH0–DMA0_SGH3, 1363, 1376
 - DMA0_SGL0–DMA0_SGL3, 707
 - DMA0_SGL0–DMA0_SGL3, 1364, 1377
 - DMA0_SLP, 710, 1365, 1378
 - DMA0_SR, 708, 1366, 1379
 - DNV0–DNV3, 1266
 - DTV0–DTV3, 1267
 - duration counter logic, 106
 - DVC
 - debug events
 - applied to instructions that result in multiple storage accesses, 519
 - applied to various instruction types, 519
 - fields, 518
 - overview, 517
 - processing, 519
 - registers
 - DVC1–DVC2, 531
 - DVC1–DVC2, 531
 - DVLIM, 1269
- ## E
- E storage attribute, 136, 248
 - EBC, 983
 - EBC (external bus controller)
 - DCRs
 - indirect access, 153, 154, 156, 1227, 1228, 1230
 - offsets, 153, 154, 155, 1227, 1228, 1229
 - EBC0_B0AP–EBC0_B7AP, 1009, 1380
 - EBC0_B0CR–EBC0_B7CR, 1007, 1382
 - EBC0_BEAR, 1013, 1383
 - EBC0_BESR, 1014, 1384
 - EBC0_CFG, 1015, 1385
 - EBC0_CFGADDR, 1007, 1387
 - EBC0_CFGDATA, 1007, 1388, 1394
 - EBC0_CFGDATA (Peripheral Controller Data Register)
 - accessing, 154, 1228
 - EBC0_CID, 1016, 1389, 1395
 - EBM0_BEAR, 975, 1390
 - EBM0_BEMR, 977, 1391
 - EBM0_BESR, 976, 1392
 - EBM0_CFGADDR, 972, 1393
 - EBM0_CFGDATA, 973

- EBM0_CID, 980
- EBM0_CTL, 973, 1396
- EBM0_FAIR, 981, 1398
- EBM0_LCNT, 975, 1399
- EBM0_SLPMD, 979, 1400
- EBM0_UAM, 978, 1401
- EBM0_UAR, 977, 1402
- effective address
 - calculation, 134
- EMAC to PHY bridge, 793
- EMACx_GAHT1-EMACx_GAHT4, 1403
- EMACx_GAHT1-GAHT4, 854
- EMACx_IAHR, 851, 1404
- EMACx_IAHT1-EMACx_IAHT4, 1405
- EMACx_IAHT1-IAHT4, 854
- EMACx_IALR, 852, 1406
- EMACx_IPCR, 859, 1407
- EMACx_IPGVR, 855, 1408
- EMACx_ISER, 849, 1409, 1411
- EMACx_ISR, 847
- EMACx_LSAH, 855, 1413
- EMACx_LSAL, 855, 1414
- EMACx_MR0, 841, 1415
- EMACx_MR1, 842, 1416
- EMACx_OCRX, 859, 1418
- EMACx_OCTX, 859, 1419
- EMACx_PTR, 853, 1420
- EMACx_RMR, 846, 1421
- EMACx_RWMMR, 858, 1423
- EMACx_STACR, 856, 1424
- EMACx_TMR0, 844, 1425
- EMACx_TMR1, 845, 1426
- EMACx_TRTR, 857, 1427
- EMACx_VTCI, 853, 1428
- EMACx_VTPID, 852, 1429
- endianness, 135, 248
- eqv, 1077
- eqv., 1077
- ESR, 431, 1270
- Ethernet MAC, 815
 - features, 817
 - flow control, 829
 - MAL- MAC packet transfer flow, 874
 - MII, 874
 - operations, 818
 - receive operation, 826
 - registers, 837
 - transmit operation, 821
 - VLAN support, 833
- exception
 - alignment exception, 442
 - critical input exception, 436
 - data storage exception, 439
 - external input exception, 442
 - illegal instruction exception, 443
 - instruction storage exception, 441
 - instruction TLB miss exception, 447
 - machine check exception, 436
 - privileged instruction exception, 444
 - program exception, 443
 - system call exception, 444
- exception priorities, 454
- exception priorities for
 - all other instructions, 458
 - allocated load and store instructions, 455
 - branch instructions, 456
 - integer load, store, and cache management instructions, 454
 - other allocated instructions, 455
 - preserved instructions, 457
 - privileged instructions, 455
 - reserved instructions, 457
 - return from interrupt instructions, 457
 - system call instruction, 456
 - trap instructions, 456
- Exception Syndrome Register, 431
- exception syndrome register, 431
- Exceptions, 419
- execution pipelines, 61
- execution synchronization, 198
- extended mnemonics
 - bctr, 1044
 - bctrl, 1044
 - bdnz, 1039
 - bdnza, 1039
 - bdnzf, 1039
 - bdnzfa, 1039
 - bdnzfkr, 1048
 - bdnzfl, 1039
 - bdnzfla, 1039
 - bdnzflrl, 1048
 - bdnzl, 1039
 - bdnzla, 1039
 - bdnzlr, 1048
 - bdnzlrl, 1048
 - bdnzt, 1039
 - bdnzta, 1039
 - bdnztl, 1039
 - bdnztla, 1039
 - bdnztlr, 1048
 - bdnztlrl, 1048
 - bdz, 1039
 - bdza, 1039
 - bdzf, 1039
 - bdzfa, 1039
 - bdzfl, 1039
 - bdzfla, 1039
 - bdzflr, 1048
 - bdzflrl, 1048
 - bdzl, 1039
 - bdzla, 1039
 - bdzlr, 1048
 - bdzlrl, 1048
 - bdzt, 1040
 - bdzta, 1040
 - bdztl, 1040
 - bdztla, 1040
 - bdztlr, 1048

Preliminary User's Manual

bdztlrl, 1048	bnga, 1042
beq, 1040	bngctr, 1045
beqa, 1040	bngctrl, 1045
beqctr, 1045	bngl, 1042
beqctrl, 1045	bngla, 1042
beql, 1040	bnglr, 1049
beqlr, 1048	bnglrl, 1049
beqlrl, 1048	bnl, 1042
bf, 1040	bnla, 1042
bfa, 1040	bnlctr, 1046
bfctr, 1045	bnlctrl, 1046
bfctrl, 1045	bnll, 1042
bfl, 1040	bnlla, 1042
bfla, 1040	bnllr, 1049
bflr, 1048	bnllrl, 1049
bflrl, 1048	bns, 1042
bge, 1040	bnsa, 1042
bgea, 1040	bnsctr, 1046
bgctr, 1045	bnsctrl, 1046
bgctrl, 1045	bnsi, 1042
bgel, 1040	bnsia, 1042
bgela, 1040	bnsir, 1049
bgelr, 1049	bnsirl, 1049
bgelrl, 1049	bnu, 1042
bgt, 1041	bnua, 1042
bgta, 1041	bnuctr, 1046
bgtctr, 1045	bnuctrl, 1046
bgtctrl, 1045	bnul, 1042
bgtl, 1041	bnula, 1042
bgtla, 1041	bnulr, 1050
bgtlr, 1049	bnulrl, 1050
bgtlrl, 1049	bsalr, 1050
ble, 1041	bso, 1043
blea, 1041	bsoa, 1043
blectr, 1045	bsoctr, 1046
blectrl, 1045	bsoctrl, 1046
blel, 1041	bsol, 1043
blela, 1041	bsola, 1043
blelr, 1049	bsolrl, 1050
blelrl, 1049	bt, 1043
blr, 1047	bta, 1043
blrl, 1047	btctr, 1046
blt, 1041	btctrl, 1046
blta, 1041	btl, 1043
bltctr, 1045	btla, 1043
bltctrl, 1045	btlr, 1050
bltl, 1041	btlrl, 1050
bltla, 1041	bun, 1043
bltlr, 1049	buna, 1043
bltlrl, 1049	bunctr, 1046
bne, 1041	bunctrl, 1046
bnea, 1041	bunl, 1043
bnectr, 1045	bunla, 1043
bnectrl, 1045	bunlr, 1050
bnel, 1041	bunlrl, 1050
bnela, 1041	clrlslwi, 1168
bnelr, 1049	clrlslwi., 1168
bnelrl, 1049	clrlwi, 1168
bng, 1042	clrlwi., 1168

clrrwi, 1168
 clrrwi., 1168
 cmplw, 1053
 cmplwi, 1054
 cmpw, 1051
 cmpwi, 1052, 1124
 crclr, 1063
 crmove, 1061
 crnot, 1060
 crset, 1058
 extlwi, 1169
 extlwi., 1169
 extrwi, 1169
 extrwi., 1169
 for addi, 1027
 for addic, 1028
 for addic., 1029
 for addis, 1030
 for bc, bca, bcl, bcla, 1039
 for bcctr, bcctrl, 1044
 for bclr, bclrl, 1047
 for cmp, 1051
 for cmpi, 1052
 for cmpl, 1053
 for cmpli, 1054
 for creqv, 1058
 for crnor, 1060
 for cror, 1061
 for crxor, 1063
 for mbar 0, 1124
 for mfspr, 1132, 1139
 for mtcfr, 1135
 for nor, nor., 1159
 for or, or., 1160
 for ori, 1162
 for rlwimi, rlwimi., 1167
 for rlwinm, rlwinm., 1168
 for rlwnm, rlwnm., 1170
 for subf, subf., subfo, subfo., 1195
 for subfc, subfc., subfco, subfco., 1196
 for tw, 1207
 for twi, 1209
 inslwi, 1167
 inslwi., 1167
 insrwi, 1167
 insrwi., 1167
 li, 1027
 lis, 1030
 mr, 1160
 mr., 1160
 mtc, 1135
 nop, 1162
 not, 1159
 not., 1159
 rotlw, 1170
 rotlw., 1170
 rotlwi, 1169
 rotlwi., 1169
 rotzwi, 1169

rotzwi., 1169
 slwi, 1169
 slwi., 1169
 srwi, 1169
 srwi., 1169
 sub, 1195
 sub., 1195
 subc, 1196
 subc., 1196
 subco, 1196
 subco., 1196
 subi, 1027
 subic, 1028
 subic., 1029
 subis, 1030
 subo, 1195
 subo., 1195
 trap, 1207
 treq, 1207
 treqi, 1209
 twge, 1207
 twgei, 1209
 twgle, 1207
 twgt, 1207
 twgti, 1209
 twle, 1207
 twlei, 1209
 twlgei, 1209
 twlgt, 1207
 twlgti, 1209
 twlle, 1207
 twllei, 1209
 twllt, 1207
 twllti, 1209
 twlng, 1207
 twlngi, 1209
 twlnl, 1207
 twlnli, 1209
 twlt, 1207
 twlti, 1209
 twne, 1207
 twnei, 1209
 twng, 1207
 twngi, 1209
 twnl, 1207
 twnli, 1209
 external bus controller, 983
 burst transactions, 994
 device paced transfers, 998
 error reporting, 1012
 non-burst peripheral transactions, 991
 registers, 1006
 signals, 983
 external bus master interface
 buffer management, 970
 registers, 971
 external bus master interface (ebmi), 959
 external debug mode, 506
 External Input interrupt, 442

Preliminary User's Manual

external input interrupts, 442
 extlwi, 1169
 extlwi., 1169
 extrwi, 1169
 extrwi., 1169
 extsb, 1078
 extsb., 1078

F

features
 DCC, 218
 ICC, 207
 FIT, 462
 fixed interval timer, 462
 fixed interval timer interrupt, 445
 Fixed-Interval Timer interrupt, 445
 freezing the timer facilities, 467
 functional block diagrams, 103

G

G storage attribute, 247
 General Purpose Registers. *See also* GPRs
 general purpose timers, 469
 registers, 471
 generic pipeline event counter logic, 105
 gigabit mode physical coding sublayer, 860
 GMII, 874
 GPCS, 860
 GPCS registers, 860
 GPCSx_ANAR, 864, 1430
 GPCSx_ANER, 867, 1432
 GPCSx_ANLNPR, 869, 1433
 GPCSx_ANLR, 866, 1434
 GPCSx_ANPTR, 868, 1436
 GPCSx_CFG, 873, 1437
 GPCSx_CR, 861, 1438
 GPCSx_ESR, 870, 1439
 GPCSx_ID0, 863, 1440, 1442
 GPCSx_ID1, 864, 1441
 GPCSx_ISER, 872
 GPCSx_ISR, 871, 1443
 GPCSx_RAR, 870, 1444
 GPCSx_SR, 862, 1445
 GPIO controller, 951
 interface signals, 952
 registers, 955
 GPIO0_TCR, 956
 GPIO0_IR, 958, 1446
 GPIO0_ODR, 957, 1447
 GPIO0_OR, 956, 1448
 GPIO0_TCR, 1449
 GPR0-GPR31, 1271
 GPRs
 defined, 144
 GPRs, illustrated, 186

GPT0_COMP0-GPT0_COMP6, 476, 1450
 GPT0_IE, 475, 1451
 GPT0_IM, 472, 1452
 GPT0_IS, 474
 GPT0_ISC, 1453
 GPT0_ISS, 1453
 GPT0_MASK0-GPT0_MASK6, 476, 1454
 GPT0_TBC, 472, 1455
 guarded, 247

H, I, J, K

I storage attribute, 247
 I2O messaging unit, 723
 IAC
 debug events
 fields, 508
 overview, 508, 520
 processing, 511
 registers
 IAC1-IAC4, 530
 IAC1-IAC4, 1268, 1272
 IAC1-IAC4, 530
 icbi, 1080
 operation summary, 211
 icbt
 formal description, 1081
 functional description, 214
 operation summary, 211
 ICC (instruction cache controller)
 control, 211
 debug, 211
 features, 207
 operations, 208
 iccci, 1083
 operation summary, 211
 ICDBDR, 1273
 ICDBTRH, 1274
 ICDBTRL, 1275
 icread, 1084
 functional description, 215
 operation summary, 211
 IIC bootstrap controller, 339
 configuration, 339
 IIC bus interface, 925
 addressing modes, 925
 interrupt handling, 948
 registers, 927
 IICx_CLKDIV, 941, 1485
 IICx_CNTL, 932, 1486
 IICx_DIRECTCNTL, 947, 1487
 IICx_EXTSTS, 937, 1488
 IICx_HMADR, 931, 1490
 IICx_HSADR, 940, 1491
 IICx_INTRMSK, 943, 1492
 IICx_LMADR, 930, 1493
 IICx_LSADR, 939, 1494
 IICx_MDBUF, 928, 1495

IICx_MDCNTL, 933, 1496
 IICx_SDBUF, 929, 1497
 IICx_STS, 935, 1498
 IICx_XFRCNT, 944, 1499
 IICx_XTCNTLSS, 945, 1500
 implemented instruction set summary, 171
 implicit update, 185
 imprecise interrupts, 420
 IMU registers, 732
 IMU0_BEAR, 747, 1456
 IMU0_BEMR, 747, 1457
 IMU0_BESR, 746, 1458
 IMU0_CFG, 739, 1459
 IMU0_IDR, 735, 1460
 IMU0_IFHPR, 742, 1461
 IMU0_IFTPR, 742, 1462
 IMU0_IIMR, 736, 1463
 IMU0_IISR, 735, 1464
 IMU0_IMR0registers
 IMU0_IMR1, 734, 1465
 IMU0_IMR1, 734, 1465
 IMU0_IPHPR, 743, 1466
 IMU0_IPTPR, 743, 1467
 IMU0_IQPR, 740, 1468
 IMU0_MIRQ, 748, 1469
 IMU0_ODR, 737, 1470
 IMU0_OFHPR, 744, 1471, 1472
 IMU0_OFTPR, 744
 IMU0_OIMR, 738, 1473
 IMU0_OISR, 737, 1474, 1475
 IMU0_OMR0registers
 IMU0_OMR1, 734
 IMU0_OMR1, 734
 IMU0_OPHPR, 745, 1476
 IMU0_OPTPR, 745, 1477
 IMU0_OQPR, 741, 1478
 IMU0_PPR, 748, 1479
 IMU0_QBAHR, 739, 1480
 IMU0_QBALR, 740, 1481
 IMU0_REVID, 749, 1482
 IMU0_SLP, 749, 1483
 IMU0_SR, 750, 1484
 initialization, 286
 inslwi, 1167
 inslwi., 1167
 insrwi, 1167
 insrwi., 1167
 instruction
 add, 1024
 add., 1024
 addc, 1024
 addc., 1024
 addco, 1024
 addco., 1024
 adde, 1025
 adde., 1025
 addeo, 1025
 addeo., 1025
 addi, 1027
 addic, 1028
 addic., 1029
 addis, 1030
 addme, 1031
 addme., 1031
 addmeo, 1031
 addmeo., 1031
 addo, 1024
 addo., 1024
 addze, 1032
 addze., 1032
 addzeo, 1032
 addzeo., 1032
 and, 1033
 and., 1033
 andc, 1034
 andc., 1034
 andi, 1035
 andis, 1036
 b, 1037
 ba, 1037
 bc, 1038
 bca, 1038
 bcctr, 1044
 bcctrl, 1044
 bcl, 1038
 bcla, 1038
 bclr, 1047
 bclrl, 1047
 bl, 1037
 bla, 1037
 cmp, 1051
 cmpi, 1052
 cmpl, 1053
 cmpli, 1054
 cntlzw, 1055
 cntlzw., 1055
 crand, 1056
 crandc, 1057
 creqv, 1058
 crnand, 1059
 crnor, 1060
 cror, 1061
 crorc, 1062
 crxor, 1063
 dcbf, 1065
 dcbi, 1066
 dcbst, 1067
 dcbt, 1068
 dcbtst, 1069
 dcbz, 1070
 dcci, 1071
 dcread, 1072
 divw, 1074
 divw., 1074
 divwo, 1074
 divwo., 1074
 divwu, 1075
 divwu., 1075

Preliminary User's Manual

divwuo, 1075
 divwuo., 1075
 dlmzb, 1076
 dlmzb., 1076
 eqv, 1077
 eqv., 1077
 extsb, 1078
 extsb., 1078
 icbi, 1080
 icbt, 1081
 iccci, 1083
 icread, 1084
 isel, 1086
 isync, 1087
 lbz, 1088
 lbzu, 1089
 lbzx, 1091
 lha, 1092
 lhau, 1093
 lhax, 1095
 lhbrx, 1096
 lhz, 1097
 lhzu, 1098
 lhzux, 1099
 lhzx, 1100
 lmw, 1101
 lswi, 1102
 lswx, 1104
 lwarx, 1106
 lwz, 1108
 lwzu, 1109
 lwzux, 1110
 lwzx, 1111
 macchw, 1112
 macchws, 1113
 macchwsu, 1114
 macchwu, 1115
 machhw, 1116
 machhwsu, 1118
 machhwu, 1119
 maclhw, 1120
 maclhws, 1121, 1158
 maclhwu, 1123
 mbar, 1124
 mcrf, 1125
 mcrxr, 1126
 mfcrr, 1127
 mfdcr, 1128
 mfmsr, 1129
 mfspr, 1130
 msync, 1134
 mtrf, 1135
 mtdcr, 1136
 mtspr, 1138
 mulchw, 1141
 mulchwu, 1142
 mulhhw, 1143
 mulhhwu, 1144
 mulhwu, 1146
 mulhwu., 1146
 mullhw, 1147
 mullhwu, 1148
 mulli, 1149
 mullw, 1150
 mullw., 1150
 mullwo, 1150
 mullwo., 1150
 nand, 1151
 nand., 1151
 neg, 1152
 neg., 1152
 nego, 1152
 nego., 1152
 nmacchw, 1153
 nmacchws, 1154
 nmachhw, 1155
 nmachhws, 1156
 nmaclhw, 1157
 nmaclhws, 1158
 nor, 1159
 nor., 1159
 or, 1160
 or., 1160
 orc, 1161
 orc., 1161
 ori, 1162
 oris, 1163
 partially executed, 423
 rfc, 1164
 rfi, 1165
 rfmci, 1166
 rlwimi, 1167
 rlwimi., 1167
 rlwinm, 1168
 rlwinm., 1168
 rlwnm, 1170
 rlwnm., 1170
 sc, 1171
 slw, 1172
 slw., 1172
 saw, 1173
 saw., 1173
 sawi, 1174
 sawi., 1174
 srw, 1175
 srw., 1175
 stb, 1176
 stbu, 1177
 stbux, 1178
 stbx, 1179
 sth, 1180
 sthbrx, 1181
 sthu, 1182
 sthux, 1183
 sthx, 1184
 stmw, 1185
 stswi, 1185
 stw, 1188

- stwbrx, 1189
- stwcx., 1190
- stwu, 1192
- stwux, 1193
- stwx, 1194
- subf, 1195
- subf., 1195
- subfc, 1196
- subfc., 1196
- subfco, 1196
- subfco., 1196
- subfe, 1197
- subfe., 1197
- subfeo, 1197
- subfeo., 1197
- subfic, 1198
- subfme, 1199
- subfme., 1199
- subfmeo, 1199
- subfmeo., 1199
- subfo, 1195
- subfo., 1195
- subfze, 1200
- subfze., 1200
- subfzeo, 1200
- subfzeo., 1200
- tlbre, 1201
- tlbsx, 1203
- tlbsx., 1203
- tlbsync, 1204
- tlbwe, 1205
- tw, 1206
- twi, 1208
- wrtee, 1210
- wrteei, 1211
- xor, 1212
- xori, 1213
- instruction address compare *See also* IAC, 508, 520
- instruction addressing modes, 134
- instruction cache array organization and operation, 201
- instruction cache coherency, 209
- instruction cache controller. *See* ICC
- instruction cache synonyms, 210
- instruction classes, 168
- instruction complete (ICMP) debug events, 521
- instruction fields, 1783
- instruction formats, 1020, 1783
 - diagrams, 1785
- instruction forms, 1783, 1785
 - B-form, 1786
 - D-form, 1786
 - I-form, 1786
 - M-form, 1788
 - SC-form, 1786
 - X-form, 1787
 - XFX-form, 1788
 - XL-form, 1788
 - XO-form, 1788
- instruction set
 - classes, 168
 - summary
 - allocated instructions, 178
 - branch, 175
 - cache management, 178
 - CR logical, 176
 - integer arithmetic, 172
 - integer compare, 174
 - integer logical, 173
 - integer rotate, 174
 - integer shift, 174
 - integer storage access, 172
 - integer trap, 174
 - processor synchronization, 177
 - register management, 176
 - system linkage, 176
 - TLB management, 178
- instruction set portability, 1020
- instruction set summary, 171
- instruction storage addressing modes, 134
- Instruction Storage interrupt, 441
- instruction storage interrupts, 441
- Instruction TLB Error Interrupt, 447
- instruction TLB error interrupts, 447
- Instructions
 - classes
 - allocated, 169
- instructions
 - all other, exception priorities for, 458
 - allocated (other), exception priorities for, 455
 - allocated instruction opcodes, 1818
 - allocated load and store, exception priorities for, 455
 - alphabetical listing, 1024
 - alphabetical summary, 1789
 - branch, exception priorities for, 456
 - byte ordering, 137, 138
 - byte-reverse, 139
 - categories, 1019
 - allocated instruction summary, 178
 - branch, 175
 - integer, 172
 - processor control, 176
 - storage control, 177
 - storage synchronization, 178
 - classes
 - defined, 168, 170
 - preserved, 170
 - CR updating, 184
 - DAC debug events applied to
 - cache management, 516
 - instructions that result in multiple storage accesses, 516
 - lswx, stswx, 516
 - special cases, 516
 - stwcx., 516
 - various, 516
 - data cache management instruction summary, 225
 - DVC debug events applied to
 - cache management, 519

Preliminary User's Manual

- instructions that result in multiple storage accesses, 519
- lswx, stswx, 519
- special cases, 519
- stwcx., 519
- various, 519
- format diagrams, 1785
- formats, 1783
- forms, 1783, 1785
- implemented instruction set summary, 171
- instruction cache management instruction summary, 211
- integer compare
 - CR update, 185
- integer load, store, and cache management, exception priorities for, 454
- mfmsr, 424
- mtmsr, 424
- opcodes, 1819
- partially executed, 423
- preserved instruction opcodes, 1818
- preserved, exception priorities for, 457
- privileged, 195
- privileged instructions, exception priorities for, 455
- pseudocode operator precedence, 1023
- register usage, 1023
- reserved instruction opcodes, 1819
- reserved, exception priorities for, 457
- reserved-illegal, 1819
- reserved-nop, 1819
- return from interrupt, exception priorities for, 457
- rfi, 426
- sorted by opcode, 1819
- syntax summary, 1789
- system call, exception priorities for, 456
- trap, exception priorities for, 456
- integer instructions
 - arithmetic, 172
 - compare, 174
 - logical, 173
 - rotate, 174
 - shift, 174
 - storage access, 172
 - trap, 174
- integer load, store, and cache management instructions, exception priorities for, 454
- integer processing, 186
- interfaces, 64
 - DDR_SDRAM, 66
 - DMA, 66
 - EBC, 69
 - EBMI, 69
 - EMAC, 68
 - GPIO, 69
 - GPT, 68
 - IIC, 69
 - JTAG, 64
 - MAL, 67
 - PCIX, 66
 - trace, 64
- UART, 69
- UIC, 68
- ZMII, 67
- internal buses, 64
- internal debug mode, 506
- internal sram controller, 535
 - errors, 544
 - registers, 535
- interrupt
 - alignment interrupt, 442
 - data storage interrupt, 439
 - external input interrupt, 442
 - instruction
 - partially executed, 423
 - Instruction Storage, 441
 - instruction storage interrupt, 441
 - instruction TLB miss interrupt, 447
 - machine check interrupt, 436
 - masking, 451
 - guidelines for system software, 453
 - ordering, 451, 453
 - guidelines for system software, 453
 - program interrupt, 443
 - illegal instruction exception, 443
 - privileged instruction exception, 444
 - system call interrupt, 444
 - type
 - Alignment, 442
 - Critical Input, 436
 - Data Storage, 439
 - Data TLB Error, 446
 - Debug, 448
 - Decrementer, 445
 - External Input, 442
 - Fixed-Interval Timer, 445
 - Instruction TLB Error, 447
 - Machine Check, 436
 - Program interrupt, 443
 - System Call, 444
 - Watchdog Timer, 446
- interrupt (IRPT) debug events, 522
- interrupt and exception handling registers
 - ESR, 431
- interrupt classes
 - asynchronous, 419
 - critical and non-critical, 421
 - machine check, 421
 - synchronous, 419
- interrupt processing, 422
 - interrupt vector, 422
- interrupt vector, 422
- Interrupts, 419
- interrupts
 - imprecise, 420
 - order, 453
 - ordering and masking, 451
 - ordering and software, 452
 - partially executed instructions, 423
 - precise, 420

registers, processing, 424
 synchronous and imprecise, 420
 synchronous and precise, 420
 types

- alignment, 442
- data storage, 439
- data TLB error, 446
- debug, 448
- decrementer, 445
- external inputs, 442
- fixed interval timer, 445
- instruction storage, 441
- instruction TLB error, 447
- machine check, 436
- program, 443
- watchdog timer, 446

vectors, 422

INV0–INV3, 1276

isel, 1086

isync, 1087

ITV0–ITV3, 1277

IVLIM, 1278

IVOR0–IVOR15, 1279

IVPR, 1280

JTAG

- boundary scan, 504
- clock requirements, 503
- connector, 503
- instructions, 504
- JTAG ID Register (SDR0_JTAGID), 505
- reset requirements, 503
- signals, 503
- test access port (TAP), 503

L

L2 cache, 323

- disable, 338
- enable, 337
- reset and initialization, 337

L2 cache and DCR, 325

L2 cache and PLB, 325

L2 cache and processor core, 325

L2 cache operations, 325

L2C0_ADDR, 333

L2C0_CFG, 329, 1502

L2C0_CMD, 331, 1504

L2C0_DATA, 334, 1505

L2C0_REVID, 335, 1506

L2C0_SNP0, 1507

L2C0_SNP0/

- L2C0_SNP3, 336

L2C0_SR, 335, 1508

lbz, 1088

lbzu, 1089

lbzx, 1091

lha, 1092

lhau, 1093

lhax, 1095

lhbrx, 1096

lhz, 1097

lhzu, 1098

lhzux, 1099

lhzx, 1100

li, 1027

lis, 1030

little endian

- structure mapping, 137

little endian mapping, 137

little endian, defined, 136

lmw, 1101

load and store alignment, 220

load operations, 220

locking, cache lines, 203

logical compare, 185

LR, 182, 1281

lswi, 1102

lswx, 1104

lwarx, 1106

lwz, 1108

lwzu, 1109

lwzux, 1110

lwzx, 1111

M

M storage attribute, 247

macchw, 1112

macchws, 1113

macchwsu, 1114

macchwu, 1115

machhw, 1116

machhwsu, 1118

machhwu, 1119

Machine Check, 421

Machine Check interrupt, 436

machine check interrupts, 421, 436

Machine State Register. *See also* MSR

maclhw, 1120

maclhws, 1121, 1158

maclhwu, 1123

MAL0_CFG, 776, 1509

MAL0_ESR, 780, 1511

MAL0_IER, 783, 1513

MAL0_RCBSx, 1514

MAL0_RXBADDR, 1515

MAL0_RXCARR, 1516

MAL0_RXCASR, 1517

MAL0_RXCTPxR, 787, 1518

MAL0_RXDEIR, 783, 785, 1519

MAL0_RXEOBISR, 779, 1520

MAL0_RXTATRR, 785

MAL0_TXCASR, 1524

MAL0_TXCTPxR, 787

MAL0_TXDEIR, 783, 785, 1526

MAL0_TXEOBISR, 779, 1527

Preliminary User's Manual

MAL0_TXTATTR, 785, 1528
 masking and ordering interrupts, 451
 master event counter logic, 103
 mbar, 1124
 mcrf, 1125
 mcrxr, 1126
 MCSR, 1282
 MCSRR0, 1283
 MCSRR1, 1284
 media access layer
 programming, 766
 registers, 775
 memory access layer, 751
 buffer descriptor, 756
 descriptor buffer status fields, descriptor buffer control fields, 762
 interfaces and channel assignments, 755
 receive software interface, 761
 transmit and receive operations, 771
 transmit software interface, 758
 memory coherence required, 247
 Memory Controller Address Register. *See* SDRAM0_CFGADDR
 memory management unit, 62
 memory management. *See also* MMU
 memory map, 131
 address space usage, 131
 memory organization, 131
 memory-mapped input/output registers. *See also* MMIO registers
 memory-mapped input/output registers. *See* MMIO registers
 mfcrr, 1127
 mfdcr, 1128
 mfmsr, 424, 1129
 mfspr, 1130
 MII, 874
 MMIO (memory-mapped input/output) registers
 directly accessed, 157, 1231
 MMU
 change status management, 255
 overview, 235
 page reference, 255
 PowerPC Book-E MMU Architecture, nonsupported features, 235
 support for Power PC Book-E MMU architecture, 235
 TLB management instructions
 overview, 253
 read/write (tlbre, tlbre), 254
 search (tlbsx), 254
 MMUCR, 249, 1285
 monitoring PLB events, 107
 mr, 1160
 mr., 1160
 MSR, 424, 1286
 defined, 144
 msync, 1134
 mtcrr, 1135
 mtcrrf, 1135
 mtdcr, 1136

mtmsr, 424
 mtspr, 1138
 mulchwr, 1141
 mulchwu, 1142
 mulhhr, 1143
 mulhhrwu, 1144
 mulhwu, 1146
 mulhwu., 1146
 mullhr, 1147
 mullhrwu, 1148
 mulli, 1149
 mullw, 1150
 mullw., 1150
 mullwo, 1150
 mullwo., 1150

N

nand, 1151
 nand., 1151
 neg, 1152
 neg., 1152
 nego, 1152
 nego., 1152
 nmacchw, 1153
 nmacchws, 1154
 nmachwr, 1155
 nmachws, 1156
 nmaclwr, 1157
 nmaclws, 1158
 non-critical interrupts, 421
 nop, 1162
 nor, 1159
 nor., 1159
 not, 1159
 not., 1159
 notation, 47, 1021, 1783
 notational conventions, 47

O

on-chip buses, 75
 dcr bus, 100
 OPB arbiter registers, 96
 OPB bus, 95
 OPB master assignments, 95
 OPB to PLB registers, 98
 PLB arbiter registers, 83
 PLB to OPB Bridge registers, 88
 processor local bus, 75
 master and slave assignments, 76
 master priority assignments, 77
 OPB0_BEARH, 100, 1532
 OPB0_BEARL, 99, 1533
 OPB0_BSTAT, 1534
 OPB0_CTRL, 98, 1531
 OPB0_REVID, 100, 1535

Preliminary User's Manual

OPB0_STAT, 99
 OPBA0_CR, 97, 1529
 OPBA0_PR, 96, 1530
 opcodes, 1819
 allocated instruction, 1818
 preserved instruction, 1818
 operands
 storage, 133
 operations
 DCC, 218
 ICC, 208
 line flush, 222
 load, 220
 store, 220
 or, 1160
 or., 1160
 orc, 1161
 orc., 1161
 ordering
 storage access, 224
 ordering and masking interrupts, 451
 ori, 1162
 oris, 1163

P

page management, 255
 partially executed instructions, 423
 PCIX bridge controller, 599
 address mapping and translation, 606
 boot modes, 691
 data flow and buffer, 611
 error handling, 623
 external configuration, 605
 features, 599
 inbound transaction, 602
 initialization, 622
 message passing, 615
 message signal interrupts, 618
 outbound transaction, 604
 PCI arbiter, 621
 power management, 619
 software initialization, 693
 PCIX0_BAR0H, 646, 1543
 PCIX0_BAR0L, 645, 1544
 PCIX0_BAR1, 647, 1545
 PCIX0_BAR2H, 649, 1546
 PCIX0_BAR2L, 648, 1547
 PCIX0_BIST, 644, 1548
 PCIX0_BRDGOPT1, 653
 PCIX0_BRDGOPT2, 654
 PCIX0_BRGDGOPT1, 1549
 PCIX0_BRGDGOPT2, 1550
 PCIX0_CACHELS, 642, 1552
 PCIX0_CAP, 651, 1553
 PCIX0_CLS, 642, 1554
 PCIX0_CMD, 639, 1555
 PCIX0_DEVID, 638, 1556

PCIX0_EROMBA, 650, 1557
 PCIX0_ERREN, 656, 1558
 PCIX0_ERRSTS, 657, 1560
 PCIX0_HDTYPE, 643, 1561
 PCIX0_IM, 690, 1562
 PCIX0_INTLN, 651, 1563
 PCIX0_INTPN, 651, 1564
 PCIX0_LATTIM, 643
 PCIX0_MAXLTNCY, 652, 1566
 PCIX0_MINGNT, 652, 1567
 PCIX0_MSGIH, 689, 1568
 PCIX0_MSGIL, 689, 1569
 PCIX0_MSGOH, 690, 1570
 PCIX0_MSGOL, 690, 1571
 PCIX0_OMCAPID, 676, 1572
 PCIX0_OMMA, 677, 1573
 PCIX0_OMMC, 677, 1574
 PCIX0_OMMDATA, 678, 1575
 PCIX0_OMMEOI, 679, 1576
 PCIX0_OMMUA, 678, 1577
 PCIX0_OMNIPTR, 676, 1578
 PCIX0_PCIXCAPID, 683, 687, 1579
 PCIX0_PCIXCID, 686, 1580
 PCIX0_PCIXCMD, 684, 1581
 PCIX0_PCIXIDR, 686, 1582
 PCIX0_PCIXNIPTR, 683, 687, 688, 1583
 PCIX0_PCIXRID, 686, 1584
 PCIX0_PCIXSTS, 684, 1585
 PCIX0_PIM0LAH, 669, 1586
 PCIX0_PIM0LAL, 669, 1587
 PCIX0_PIM0SA, 667, 668, 673
 PCIX0_PIM0SAH, 1589
 PCIX0_PIM0SAL, 1588
 PCIX0_PIM1LAH, 671, 1590
 PCIX0_PIM1LAL, 671, 1591
 PCIX0_PIM1SA, 670, 1592
 PCIX0_PIM2LAH, 675, 1593
 PCIX0_PIM2LAL, 674, 1594
 PCIX0_PIM2SA, 672
 PCIX0_PIM2SAH, 1596
 PCIX0_PIM2SAL, 1595
 PCIX0_PLBBEARH, 660, 1597
 PCIX0_PLBBEARL, 659, 1598
 PCIX0_PLBBESR, 1599
 PCIX0_PMC, 680, 1600
 PCIX0_PMCAPID, 679, 1601
 PCIX0_PMCSR, 681, 1603
 PCIX0_PMCSEBSE, 681, 1602
 PCIX0_PMDATA, 682, 1604
 PCIX0_PMNIPTR, 680, 1605
 PCIX0_PMSCRR, 682, 1606
 PCIX0_POM0LAH, 661, 1607
 PCIX0_POM0LAL, 660, 1608
 PCIX0_POM0MA, 661
 PCIX0_POM0PCIAH, 662, 1609
 PCIX0_POM0PCIAL, 662, 1610
 PCIX0_POM0SA, 1611
 PCIX0_POM1LAH, 663, 1612
 PCIX0_POM1LAL, 1613

Preliminary User's Manual

- PCIX0_POM1PCIAH, 665, 1615
- PCIX0_POM1PCIAL, 665, 1616
- PCIX0_POM1SA, 664, 1614
- PCIX0_POM2SA, 666, 1617
- PCIX0_REVID, 641, 1618
- PCIX0_SBSYSID, 650, 1619
- PCIX0_SBSYSVID, 649, 1620
- PCIX0_STATUS, 640, 1621
- PCIX0_VENDID, 638, 1622
- PCIX0_VPDADR, 1623
- PCIX0_VPDCAPID, 1624
- PCIX0_VPDDATA, 1625
- PCIX0_VPDNIPTR, 1626
- PCIX0_XCR, 632
- PCIX0_XPLLC, 634
- PCIX0_XPLLD, 634
- Peripheral Controller Data Register. See EBC0_CFGDATA
- physical address map, 131
- PID, 252, 1287
- PIR, 191, 1288
- PLB performance monitor, 64, 101
 - features, 101
 - monitoring PLB events, 107
 - registers, 109
- PLB0_ACR, 84, 1536
- PLB0_BEARH, 88, 1537
- PLB0_BEARL, 87, 1538
- PLB0_BESR, 85, 1539
- PLB0_REVID, 84, 1542
- POB0_BEARH, 91, 1627
- POB0_BEARL, 91, 1628
- POB0_BESR0, 89, 1629
- POB0_BESR1, 92, 1631
- POB0_CONFIG, 94, 1633
- POB0_LATENCY, 94, 1634
- POB0_REVID, 95, 1635
- portability, instruction set, 1020
- PPC440GP
 - addressing modes, 70
 - block diagram, 52
 - CoreConnect features, 58
 - development tool support, 73
 - execution pipelines, 61
 - features, 53
 - interfaces, 64
 - DDR_SDRAM, 66
 - DMA, 66
 - EBC, 69
 - EBMI, 69
 - EMAC, 68
 - GPIO, 69
 - GPT, 68
 - IIC, 69
 - JTAG, 64
 - MAL, 67
 - PCIX, 66
 - trace, 64
 - UART, 69
 - UIC, 68
 - ZMII, 67
 - internal buses, 64
 - memory management unit, 62
 - peripheral features, 55
 - PowerPC implementation, 60
 - processor core features, 53
 - processor core functions, 61
 - registers, 71
 - condition registers, 72
 - device control registers, 72
 - general purpose registers, 71
 - machine state registers, 72
 - memory mapped registers, 72
 - special purpose registers, 71
 - time base registers, 71
 - signals, 72, 1771
 - superscalar instruction unit, 61
 - supported configurability, 70
- PPM (PLB performance monitor)
 - DCRs
 - indirect access, 154, 1228
 - PPM0_CCR, 116, 1636
 - PPM0_CFGADDR, 111, 1637
 - PPM0_CFGDATA, 112, 1638
 - PPM0_CR, 114, 1639
 - PPM0_DCMNR0-PPM0_DCMNR1, 127, 1641
 - PPM0_DCMXR0-PPM0_DCMXR1, 126, 1642, 1643
 - PPM0_DCOTR0-PPM0_DCOTR1, 128
 - PPM0_DCSR0-PPM0_DCSR1, 125, 1644
 - PPM0_DCTVR0-PPM0_DCTVR1, 127, 1645
 - PPM0_GCR0-PPM0_GCR3, 125, 1646
 - PPM0_GCSR0-PPM0_GCSR3, 123, 1647
 - PPM0_ISR, 112, 1648
 - PPM0_LAMR, 118, 1650
 - PPM0_LAR, 117, 1651
 - PPM0_MCR0-PPM0_MCR3, 123, 1652
 - PPM0_MCSR0-PPM0_MCSR3, 119
 - PPM0_MCSR0-PPM0_MCSR7, 1653
 - PPM0_RIDR, 118, 1655
 - PPM0_SCR0-PPM0_SCR3, 124, 1656
 - PPM0_SCSR0-PPM0_SCSR3, 1657
 - PPM0_SCSR0-PPM0_SCSR7, 121
 - PPM0_UAMR, 117, 1659
 - PPM0_UAR, 116, 1660
 - PPMx_CFGDATA, 112
 - PPMx_ISR, 112
 - precise interrupts, 420
 - prefetch mechanism, speculative, 209
 - preserved instructions, exception priorities for, 457
 - primary opcodes, 1819
 - priorities, exception, 454
 - privileged instructions, 195
 - privileged mode, 195
 - privileged operation, 195
 - privileged SPRs, 196
 - problem state, 195
 - processor control instruction summary, 176
 - processor control instructions
 - CR logical, 176

register management, 176
 synchronization, 177
 system linkage, 176
 processor control registers, 189
 Program interrupt, 443
 program interrupts, 443
 pseudocode, 1021
 PVR, 190, 1289
 PXIC0_LATTIM, 1565

R

R0-R31, 1271
 reading the time base, 460
 register
 CSRR0, 427, 428
 CSRR1, 428, 429
 ESR, 431
 PID, 252
 SRR0, 426
 SRR1, 427
 registers, 139
 branching control, 182
 CCR0, 191, 212, 213, 227, 1244
 CCR1, 1246
 clock and power management, 497
 clocking, 489
 condition registers, 72
 CPM0_ER, 498, 1351
 CPM0_FR, 499, 1352
 CPM0_SR, 501, 1353
 CPR0_CFGADDR, 489
 CPR0_CFGDATA, 490
 CPR0_CLKUPD, 287, 490, 1301
 CPR0_ICFG, 286, 1302
 CPR0_MALD, 288, 495, 1303
 CPR0_OPBD, 288, 494, 1304
 CPR0_PERD, 289, 495, 1305
 CPR0_PLLC, 289, 491, 1306
 CPR0_PLLD, 290, 492, 1307
 CPR0_PRIMAD, 291, 493, 1309
 CPR0_PRIMBD, 292, 494, 1310
 CR, 144, 183, 1247
 CSRR0, 1248
 CSRR1, 1249
 CTR, 183, 554, 555, 556, 559, 560, 561, 562, 571,
 573, 578, 579, 1250, 1663, 1664, 1671,
 1672, 1673, 1674, 1676, 1677, 1678, 1679,
 1681, 1682, 1685, 1686
 DAC1–DAC2, 530, 1251
 DBCR0, 524, 527, 1252
 DBCR1, 526, 1254
 DBCR2, 1256
 DBDR, 531
 DBSR, 528
 DCDBTRH, 228, 1261
 DCDBTRL, 228, 1262
 DCR numbering, 1219

DEAR, 1263
 DEC, 461, 1264
 DECAR, 462, 1265
 device control registers, 72
 DMA0_CR0–DMA0_CR3, 701, 1354, 1367
 DMA0_CT0–DMA0_CT3, 703, 1356, 1369
 DMA0_DAH0–DMA0_DAH3, 705, 1357, 1370
 DMA0_DAL0–DMA0_DAL3, 705, 1358, 1371
 DMA0_POL, 711, 1359, 1372
 DMA0_SAH0–DMA0_SAH3, 704
 DMA0_SAH0–DMA0_SAH3, 1360, 1373
 DMA0_SAH0–DMA0_SAL3, 1361, 1374
 DMA0_SAL0–DMA0_SAL3, 704
 DMA0_SGC, 709, 1362, 1375
 DMA0_SGH0–DMA0_SGH3, 707
 DMA0_SGH0–DMA0_SGH3, 1363, 1376
 DMA0_SGL0–DMA0_SGL3, 707
 DMA0_SGL0–DMA0_SGL3, 1364, 1377
 DMA0_SLP, 710, 1365, 1378
 DMA0_SR, 708, 1366, 1379
 DNV0–DNV3, 1266
 DTV0–DTV3, 1267
 DVC1–DVC2, 531
 DVLIM, 1269
 EBC0_B0AP–EBC0_B7AP, 1009, 1380
 EBC0_B0CR–EBC0_B7CR, 1007, 1382
 EBC0_BEAR, 1013, 1383
 EBC0_BESR, 1014, 1384
 EBC0_CFG, 1015, 1385
 EBC0_CFGADDR, 1007, 1387
 EBC0_CFGDATA, 1007, 1388, 1394
 EBC0_CID, 1016, 1389, 1395
 EBM0_BEAR, 975, 1390
 EBM0_BEMR, 977, 1391
 EBM0_BESR, 976, 1392
 EBM0_CFGADDR, 972, 1393
 EBM0_CFGDATA, 973
 EBM0_CID, 980
 EBM0_CTL, 973, 1396
 EBM0_FAIR, 981, 1398
 EBM0_LCNT, 975, 1399
 EBM0_SLPMD, 979, 1400
 EBM0_UAM, 978, 1401
 EBM0_UAR, 977, 1402
 EMAC0xGAHT1–EMACx_GAHT4, 1403
 EMACx_GAHT1–GAHT4, 854
 EMACx_IAHR, 851, 1404
 EMACx_IAHT1–EMACx_IAHT4, 1405
 EMACx_IAHT1–IAHT4, 854
 EMACx_IALR, 852, 1406
 EMACx_IPCR, 859, 1407
 EMACx_IPGVR, 855, 1408
 EMACx_ISER, 849, 1409, 1411
 EMACx_ISR, 847
 EMACx_LSAH, 855, 1413
 EMACx_LSAL, 855, 1414
 EMACx_MR0, 841, 1415
 EMACx_MR1, 842, 1416
 EMACx_OCRX, 859, 1418

Preliminary User's Manual

EMACx_OCTX, 859, 1419
 EMACx_PTR, 853, 1420
 EMACx_RMR, 846, 1421
 EMACx_RWMR, 858, 1423
 EMACx_STACR, 856, 1424
 EMACx_TMR0, 844, 1425
 EMACx_TMR1, 845, 1426
 EMACx_TRTR, 857, 1427
 EMACx_VTCI, 853, 1428
 EMACx_VTPID, 852, 1429
 ESR, 431, 1270
 Ethernet MAC, 837
 external bus controller, 1006
 external bus master interface, 971
 general purpose registers, 71
 GPCS, 860
 GPCSx_ANAR, 864, 1430
 GPCSx_ANER, 867, 1432
 GPCSx_ANLNPR, 869, 1433
 GPCSx_ANLR, 866, 1434
 GPCSx_ANPTR, 868, 1436
 GPCSx_CFG, 873, 1437
 GPCSx_CR, 861, 1438
 GPCSx_ESR, 870, 1439
 GPCSx_ID0, 863, 1440, 1442
 GPCSx_ID1, 864, 1441
 GPCSx_ISER, 872
 GPCSx_ISR, 871, 1443
 GPCSx_RAR, 870, 1444
 GPCSx_SR, 862, 1445
 GPIO controller, 955
 GPIO0_IR, 958, 1446
 GPIO0_ODR, 957, 1447
 GPIO0_OR, 956, 1448
 GPIO0_TCR, 1449
 GPR0-GPR31, 1271
 GPRs, 186
 GPT0_COMP0-GPT0_COMP6, 476, 1450
 GPT0_IE, 475, 1451
 GPT0_IM, 472, 1452
 GPT0_IS, 474
 GPT0_ISS GPT0_ISC, 1453
 GPT0_MASK0-GPT0_MASK6, 476, 1454
 GPT0_TBC, 472, 1455
 IAC1-IAC4, 1268, 1272
 IAC1-IAC4, 530
 ICDBDR, 1273
 ICDBTRH, 1274
 ICDBTRL, 1275
 IIC bus interface, 927
 IICx_CLKDIV, 941, 1485
 IICx_CNTL, 932, 1486
 IICx_DIRECTCNTL, 947, 1487
 IICx_EXTSTS, 937, 1488
 IICx_HMADR, 931, 1490
 IICx_HSADR, 940, 1491
 IICx_INTRMSK, 943, 1492
 IICx_LMADR, 930, 1493
 IICx_LSADR, 939, 1494
 IICx_MDBUF, 928, 1495
 IICx_MDCNTL, 933, 1496
 IICx_SDBUF, 929, 1497
 IICx_STS, 935, 1498
 IICx_XFRCNT, 944, 1499
 IICx_XTCNTLSS, 945, 1500
 IMU, 732
 IMU0_BEAR, 747, 1456
 IMU0_BEMR, 747, 1457
 IMU0_BESR, 746, 1458
 IMU0_CFG, 739, 1459
 IMU0_IDR, 735, 1460
 IMU0_IFHPR, 742, 1461
 IMU0_IFTP, 742, 1462
 IMU0_IIMR, 736, 1463
 IMU0_IISR, 735, 1464
 IMU0_IMR0, 734, 1465
 IMU0_IPHPR, 743, 1466
 IMU0_IPTPR, 743, 1467
 IMU0_IQPR, 740, 1468
 IMU0_MIRQ, 748, 1469
 IMU0_ODR, 737, 1470
 IMU0_OFHPR, 744, 1471, 1472
 IMU0_OFTP, 744
 IMU0_OIMR, 738, 1473
 IMU0_OISR, 737, 1474, 1475
 IMU0_OMR0, 734
 IMU0_OPHPR, 745, 1476
 IMU0_OPTPR, 745, 1477
 IMU0_OQPR, 741, 1478
 IMU0_PPR, 748, 1479
 IMU0_QBAHR, 739, 1480
 IMU0_QBALR, 740, 1481
 IMU0_REVID, 749, 1482
 IMU0_SLP, 749, 1483
 IMU0_SR, 750, 1484
 internal sram controller, 535
 interrupt processing, 424
 INV0-INV3, 1276
 ITV0-ITV3, 1277
 IVLIM, 1278
 IVOR0-IVOR15, 1279
 IVPR, 1280
 L2 cache, 329
 L2C0_ADDR, 333
 L2C0_CFG, 329, 1502
 L2C0_CMD, 331, 1504
 L2C0_DATA, 334, 1505
 L2C0_REVID, 335, 1506
 L2C0_SNP0, 1507
 L2C0_SNP0/
 L2C0_SNP3, 336
 L2C0_SR, 335, 1508
 LR, 182, 1281
 machine state registers, 72
 MAL0_CFG, 776, 1509
 MAL0_ESR, 780, 1511
 MAL0_IER, 783, 1513
 MAL0_RCBSx, 1514

Preliminary User's Manual

MAL0_RXBADDR, 1515
 MAL0_RXCARR, 1516
 MAL0_RXCASR, 1517
 MAL0_RXCTPxR, 787, 1518
 MAL0_RXDEIR, 783, 785, 1519
 MAL0_RXEOBISR, 779, 1520
 MAL0_RXTATTRR, 785
 MAL0_TXCASR, 1524
 MAL0_TXCTPxR, 787, 1525
 MAL0_TXDEIR, 783, 785, 1526
 MAL0_TXEOBISR, 779, 1527
 MAL0_TXTATTRR, 785, 1528
 malrbl, 789
 malrbs, 790
 maltbl, 790
 maltbs, 791
 MCSR, 1282
 MCSRR0, 1283
 MCSRR1, 1284
 media access layer, 775
 memory mapped registers, 72
 MMIO registers
 directly accessed, 157, 1231
 MMUCR, 249, 1285
 MSR, 144, 424, 1286
 OPB0_BEARH, 100, 1532
 OPB0_BEARL, 99, 1533
 OPB0_BSTAT, 1534
 OPB0_CTRL, 98, 1531
 OPB0_REVID, 100, 1535
 OPB0_STAT, 99
 OPBA0_CR, 97, 1529
 OPBA0_PR, 96, 1530
 PCIX0_BAR0H, 646, 1543
 PCIX0_BAR0L, 645, 1544
 PCIX0_BAR1, 647, 1545
 PCIX0_BAR2H, 649, 1546
 PCIX0_BAR2L, 648, 1547
 PCIX0_BIST, 644, 1548
 PCIX0_BRDGOPT1, 653
 PCIX0_BRDGOPT2, 654
 PCIX0_BRDGOPT1, 1549
 PCIX0_BRDGOPT2, 1550
 PCIX0_CACHELS, 642, 1552
 PCIX0_CAP, 651, 1553
 PCIX0_CLS, 642, 1554
 PCIX0_CMD, 639, 1555
 PCIX0_DEVID, 638, 1556
 PCIX0_EROMBA, 650, 1557
 PCIX0_ERREN, 656, 1558
 PCIX0_ERRSTS, 657, 1560
 PCIX0_HDTYPE, 643, 1561
 PCIX0_IM, 690, 1562
 PCIX0_INTLN, 651, 1563
 PCIX0_INTPN, 651, 1564
 PCIX0_LATTIM, 643, 1565
 PCIX0_MAXLTNCY, 652, 1566
 PCIX0_MINGNT, 652, 1567
 PCIX0_MSGIH, 689, 1568
 PCIX0_MSGIL, 689, 1569
 PCIX0_MSGOH, 690, 1570
 PCIX0_MSGOL, 690, 1571
 PCIX0_OMCAPID, 676, 1572
 PCIX0_OMMA, 677, 1573
 PCIX0_OMMC, 677, 1574
 PCIX0_OMMDATA, 678, 1575
 PCIX0_OMMEOI, 679, 1576
 PCIX0_OMMUA, 678, 1577
 PCIX0_OMNIPTR, 676, 1578
 PCIX0_PCIXCAPID, 683, 687, 1579
 PCIX0_PCIXCID, 686, 1580
 PCIX0_PCIXCMD, 684, 1581
 PCIX0_PCIXIDR, 686, 1582
 PCIX0_PCIXNIPTR, 683, 687, 688, 1583
 PCIX0_PCIXRID, 686, 1584
 PCIX0_PCIXSTS, 684, 1585
 PCIX0_PIM0LAH, 669, 1586
 PCIX0_PIM0LAL, 669, 1587
 PCIX0_PIM0SA, 667, 668, 673
 PCIX0_PIM0SAH, 1589
 PCIX0_PIM0SAL, 1588
 PCIX0_PIM1LAH, 671, 1590
 PCIX0_PIM1LAL, 671, 1591
 PCIX0_PIM1SA, 670, 1592
 PCIX0_PIM2LAH, 675, 1593
 PCIX0_PIM2LAL, 674, 1594
 PCIX0_PIM2SA, 672
 PCIX0_PIM2SAH, 1596
 PCIX0_PIM2SAL, 1595
 PCIX0_PLBBEARH, 660, 1597
 PCIX0_PLBBEARL, 659, 1598
 PCIX0_PLBBESR, 659, 1599
 PCIX0_PMC, 680, 1600
 PCIX0_PMCAPID, 679, 1601
 PCIX0_PMCSR, 681, 1603
 PCIX0_PMCSRBSE, 681, 1602
 PCIX0_PMDATA, 682, 1604
 PCIX0_PMNIPTR, 680, 1605
 PCIX0_PMSCRR, 682, 1606
 PCIX0_POM0LAH, 661, 1607
 PCIX0_POM0LAL, 660, 1608
 PCIX0_POM0MA, 661
 PCIX0_POM0PCIAH, 662, 1609
 PCIX0_POM0PCIAL, 662, 1610
 PCIX0_POM0SA, 1611
 PCIX0_POM1LAH, 663, 1612
 PCIX0_POM1LAL, 663, 1613
 PCIX0_POM1PCIAH, 665, 1615
 PCIX0_POM1PCIAL, 665, 1616
 PCIX0_POM1SA, 664, 1614
 PCIX0_POM2SA, 666, 1617
 PCIX0_REVID, 641, 1618
 PCIX0_SBSYSID, 650, 1619
 PCIX0_SBSYSVID, 649, 1620
 PCIX0_STATUS, 640, 1621
 PCIX0_VENDID, 638, 1622
 PCIX0_VPDADR, 1623
 PCIX0_VPDCAPID, 1624

Preliminary User's Manual

PCIX0_VPDDATA, 1625
 PCIX0_VPDNIPTR, 1626
 PCIX0_XCR, 632
 PCIX0_XPLLC, 634
 PCIX0_XPLLD, 634
 PID, 1287
 PIR, 191, 1288
 PLB performance monitor, 109
 PLB0_ACR, 84, 1536
 PLB0_BEARH, 88, 1537
 PLB0_BEARL, 87, 1538
 PLB0_BESR, 85, 1539
 PLB0_REVID, 84, 1542
 POB0_BEARH, 91, 1627
 POB0_BEARL, 91, 1628
 POB0_BESR0, 89, 1629
 POB0_BESR1, 92, 1631
 POB0_CONFIG, 94, 1633
 POB0_LATENCY, 94, 1634
 POB0_REVID, 95, 1635
 PPM0_CCR, 116, 1636
 PPM0_CFGADDR, 111, 1637
 PPM0_CFGDATA, 112, 1638
 PPM0_CR, 114, 1639
 PPM0_DCMNR0-PPM0_DCMNR1, 127, 1641
 PPM0_DCMXR0-PPM0_DCMXR1, 126, 1642, 1643
 PPM0_DCOTR0-PPM0_DCOTR1, 128
 PPM0_DCSR0-PPM0_DCSR1, 125, 1644
 PPM0_DCTVR0-PPM0_DCTVR1, 127, 1645
 PPM0_GCR0-PPM0_GCR3, 125, 1646
 PPM0_GCSR0-PPM0_GCSR3, 123, 1647
 PPM0_ISR, 112, 1648
 PPM0_LAMR, 118, 1650
 PPM0_LAR, 117, 1651
 PPM0_MCR0-PPM0_MCR3, 123, 1652
 PPM0_MCSR0-PPM0_MCSR3, 119
 PPM0_MCSR0-PPM0_MCSR7, 1653
 PPM0_RIDR, 118, 1655
 PPM0_SCR0-PPM0_SCR3, 124, 1656
 PPM0_SCSR0-PPM0_SCSR3, 1657
 PPM0_SCSR0-PPM0_SCSR7, 121
 PPM0_UAMR, 117, 1659
 PPM0_UAR, 116, 1660
 PPMx_CFGDATA, 112
 PPMx_ISR, 112
 processor control, 189
 PVR, 190, 1289
 R0-R31, 1271
 RGMII0_FER, 1501, 1661
 RGMII0_SSR, 1662
 RSTCFG, 194, 1290
 SDR0_AMP, 78, 292, 1311
 SDR0_CP440, 80, 294, 1313
 SDR0_CUST0, 295, 356, 1315
 SDR0_CUST1, 296, 357, 1316
 SDR0_DDRDL, 296, 574, 1317
 SDR0_EBC, 296, 1318
 SDR0_ECID0, 297, 1319
 SDR0_ECID1, 297, 1320
 SDR0_ECID2, 297, 1321
 SDR0_JTAG, 298, 1322
 SDR0_JTAGID, 505
 SDR0_MALRB, 298
 SDR0_MALRBL, 1323, 1325, 1326
 SDR0_MALRBS, 299, 1324
 SDR0_MALTB, 299
 SDR0_MALTB, 300
 SDR0_MFR, 300, 1327
 SDR0_MIRQ0, 81, 302, 1329
 SDR0_MIRQ1, 82, 303, 1330
 SDR0_PFC0, 303, 953, 1331
 SDR0_PFC1, 305, 1333
 SDR0_PINSTP, 306, 351, 1334
 SDR0_SDCS, 307, 351, 1335
 SDR0_SDSTP0, 307, 1336
 SDR0_SDSTP1, 309, 353, 1338
 SDR0_SDSTP2, 311, 355, 1340
 SDR0_SDSTP3, 311, 356, 1341
 SDR0_SLPIPE, 83, 311, 1342
 SDR0_SRST, 312, 1343
 SDR0_STDSTP0, 352
 SDR0_UART0, 313, 918, 1345
 SDR0_UART1, 314, 1346
 SDR0_XCR, 315, 1347
 SDR0_XPLLC, 316, 1349
 SDR0_XPLLD, 317, 1350
 SDRAM0_B0CR-SDRAM0_B3CR, 562, 1663
 SDRAM0_BEAR, 554, 1664
 SDRAM0_BESR0, 551, 1665
 SDRAM0_BESR1, 553, 1667
 SDRAM0_CFG0, 556
 SDRAM0_CFG1, 559, 1669
 SDRAM0_CFG2, 1671
 SDRAM0_CID, 579, 1672
 SDRAM0_CLKTR, 571, 1673
 SDRAM0_DEVOPT, 559, 1674
 SDRAM0_DLYCAL, 577, 1675
 SDRAM0_ECCESR, 578, 1676
 SDRAM0_MCSTS, 560, 1677
 SDRAM0_MIRQ, 555
 SDRAM0_PMIT, 561, 1678, 1679
 SDRAM0_RID, 579, 1680
 SDRAM0_RTR, 560, 1681
 SDRAM0_SLIO, 556, 1682
 SDRAM0_TR0, 563, 1683
 SDRAM0_TR1, 565, 1685
 SDRAM0_UABBA, 561, 1686
 SDRAM0_WDDCTR, 1687
 special purpose registers, 71
 SPRG0 SPRG7, 190
 SPRG0-SPRG3, 190
 SPRG0-SPRG7, 1291
 SRAM0_BEAR, 537, 1688
 SRAM0_BESR0, 538, 1689
 SRAM0_BESR1, 540, 1691
 SRAM0_CID, 1693
 SRAM0_DPC, 543
 SRAM0_PMEG, 542, 1694, 1696

SRAM0_RID, 1695
 SRAM0_SB3CR, 536
 SRAM0_SBxCR, 1697
 SRR0, 1292
 SRR1, 1293
 storage control, 249
 TAHx, 894
 TAHx_MR, 895, 1698
 TAHx_REVID, 895, 1699
 TAHx_SSR0-TAHx_SSR5, 897, 1700
 TAHx_TSR, 898, 1701
 TBL, 1294
 TBU, 1295
 TCR, 462, 463, 466, 1296
 time base registers, 71
 TSR, 464, 467, 1297
 types, 144
 CR, 144
 DCR, 145
 GPR, 144
 MSR, 144
 SPR, 144
 UART controller, 905
 UARTx_DLL, 914, 1702
 UARTx_DLM, 914, 1703
 UARTx_FCR, 909, 1704
 UARTx_IER, 906, 1705
 UARTx_IIR, 907, 1706
 UARTx_LCR, 910, 1707
 UARTx_LSR, 912, 1708
 UARTx_MCR, 911, 1710
 UARTx_MSR, 913, 1711
 UARTx_RBR, 906, 1712
 UARTx_SCR, 914, 1713
 UARTx_THR, 906, 1714
 UIC controller, 367
 UIC0_CR, 382, 1715
 UIC0_ER, 374, 1717
 UIC0_MSR, 403, 1719
 UIC0_PR, 388, 1721
 UIC0_SR, 368, 1723
 UIC0_TR, 396, 1725
 UIC0_VCR, 409, 1727
 UIC0_VR, 413, 1728
 UIC1_CR, 384, 386, 388, 1729
 UIC1_ER, 377, 379, 381, 1730
 UIC1_MSR, 405, 407, 409, 1733
 UIC1_PR, 391, 393, 395, 1735
 UIC1_SR, 370, 372, 374, 1737
 UIC1_TR, 398, 400, 402, 1739
 UIC1_VCR, 410, 411, 412, 1741
 UIC1_VR, 414, 415, 417, 1742
 UIC2_CR, 1743
 UIC2_ER, 1745
 UIC2_MSR, 1747
 UIC2_PR, 1749
 UIC2_SR, 1751
 UIC2_TR, 1753
 UIC2_VCR, 1755

UIC2_VR, 1756
 UICB0_CR, 1757
 UICB0_ER, 1758
 UICB0_MSR, 1759
 UICB0_PR, 1760
 UICB0_SR, 1761
 UICB0_TR, 1762
 UICB0_VCR, 1763
 UICB0_VR, 1764
 USPRG0, 190, 1298
 XER, 187, 1299
 ZMII bridge, 799, 812
 ZMII0_FER, 800, 812, 1765
 ZMII0_SMIISR, 803, 1766
 ZMII0_SSR, 801, 813, 1768
 registers, device control, 145
 registers, summary, 139
 replacement policy, cache line, 202
 requirements
 software
 interrupt ordering, 452
 reservation bit, 1106, 1190
 reserved instructions, exception priorities for, 457
 reserved-illegal instructions, 1819
 reserved-nop instructions, 1819
 reset
 debug, 523
 reset types
 chip, 261
 core, 261
 system, 262
 resets
 signals, 261
 types, 261
 return (RET) debug events, 520
 return from interrupt instructions, exception priorities for, 457
 rfc1, 1164
 rfi, 426, 1165
 rfmc1, 1166
 RGMII0_FER, 1501, 1661
 RGMII0_SSR, 1662
 rlwimi, 1167
 rlwimi., 1167
 rlwinm, 1168
 rlwinm., 1168
 rlwnm, 1170
 rlwnm., 1170
 rotlw, 1170
 rotlw., 1170
 rotlwi, 1169
 rotlwi., 1169
 rotrwi, 1169
 rotrwi., 1169
 RSTCFG, 194, 1290

S

Save/Restore Register 0, 426

Preliminary User's Manual

- Save/Restore Register 1, 427
- sc, 1171
- SDR0 registers, 286
- SDR0_AMP, 78, 292, 1311
- SDR0_CP440, 80, 294, 1313
- SDR0_CUST0, 295, 356, 1315
- SDR0_CUST1, 296, 357, 1316
- SDR0_DDRDL, 296, 574, 1317
- SDR0_EBC, 296, 1318
- SDR0_ECID0, 297, 1319
- SDR0_ECID1, 1320
- SDR0_ECID2, 297, 1321
- SDR0_JTAG, 298, 1322
- SDR0_JTAGID, 505
- SDR0_MALRBL, 298, 1323, 1325, 1326
- sdr0_malrbl, 789
- SDR0_MALRBS, 299, 1324
- sdr0_malrbs, 790
- SDR0_MALTBL, 299
- sdr0_maltbl, 790
- SDR0_MALTBS, 300
- sdr0_maltbs, 791
- SDR0_MFR, 300, 1327
- SDR0_MIRQ0, 81, 302, 1329
- SDR0_MIRQ1, 82, 303, 1330
- SDR0_PFC0, 303, 953, 1331
- SDR0_PFC1, 305, 1333
- SDR0_PINSTP, 306, 351, 1334
- SDR0_SCID1, 297
- SDR0_SDSCS, 307, 351, 1335
- SDR0_SDSTP0, 307, 352, 1336
- SDR0_SDSTP1, 309, 353, 1338
- SDR0_SDSTP2, 311, 355, 1340
- SDR0_SDSTP3, 311, 356, 1341
- SDR0_SLPIPE, 83, 311, 1342
- SDR0_SRST, 312, 1343
- SDR0_UART0, 313, 1345
- SDR0_UART0registers
 - SDR0_UART1, 918
- SDR0_UART1, 314, 918, 1346
- SDR0_XCR, 315, 1347
- SDR0_XPLLC, 316, 1349
- SDR0_XPLLD, 317, 1350
- SDRAM controller
 - DCRs
 - access procedures, overview, 150, 1224
 - indirect access, 150, 152, 1224, 1226
 - offsets, 150, 151, 152, 1224, 1225, 1226
- SDRAM0_B0CR-SDRAM0_B3CR, 562, 1663
- SDRAM0_BEAR, 554, 1664
- SDRAM0_BESR0, 551, 1665
- SDRAM0_BESR1, 553, 1667
- SDRAM0_CFG0, 556
- SDRAM0_CFG1, 559, 1669
- SDRAM0_CFG2, 1671
- SDRAM0_CFGADDR (Memory Controller Address Register)
 - accessing, 150, 152, 164, 1224, 1226, 1238
- SDRAM0_CID, 579, 1672
- SDRAM0_CLKTR, 571, 1673
- SDRAM0_DEVOPT, 559, 1674
- SDRAM0_DLYCAL, 577, 1675
- SDRAM0_ECCEsr, 578, 1676
- SDRAM0_MCSTS, 560, 1677
- SDRAM0_MIRQ, 555
- SDRAM0_PMIT, 561, 1678, 1679
- SDRAM0_RID, 579, 1680
- SDRAM0_RTR, 560, 1681
- SDRAM0_SLIO, 556, 1682
- SDRAM0_TR0, 563, 1683
- SDRAM0_TR1, 565, 1685
- SDRAM0_UABBA, 561, 1686
- SDRAM0_WDDCTR, 1687
- secondary opcodes, 1819
- self-modifying code, 210
- shadow TLB arrays, 253
- signals, 1771
 - ExtReset, 261
 - resets, 261
 - SysReset, 261
- slave event counter logic, 104
- slw, 1172
- slw., 1172
- slwi, 1169
- slwi., 1169
- software
 - interrupt ordering requirements, 452
- Special Purpose Registers. *See also* SPRs
- speculative fetching, 196
- speculative prefetch mechanism, 209
- SPRG0-SPRG7, 190
- SPRG0-SPRG3, 190
- SPRG0-SPRG7, 1291
- SPRs
 - defined, 144
- SRAM, 65
- SRAM memory, 338
- SRAM0_BEAR, 537, 1688
- SRAM0_BESR0, 538, 1689
- SRAM0_BESR1, 540, 1691
- SRAM0_CID, 1693
- SRAM0_DPC, 543
- SRAM0_PMEG, 542, 1694, 1696
- SRAM0_RID, 1695
- SRAM0_SB3CR, 536
- SRAM0_SBxCR, 1697
- sraw, 1173
- sraw., 1173
- srawi, 1174
- srawi., 1174
- SRR0, 426, 1292
- SRR1, 427, 1293
- srw, 1175
- srw., 1175
- srwi, 1169
- srwi., 1169
- stb, 1176
- stbu, 1177
- stbux, 1178

- stbx, 1179
- sth, 1180
- sthbrx, 1181
- sthu, 1182
- sthux, 1183
- sthx, 1184
- stmw, 1185
- storage access ordering, 224
- storage attributes
 - caching inhibited, 247
 - endian, 248
 - guarded, 247
 - Memory Coherence Required, 247
 - supported combinations, 248
 - user-definable (U0–U3), 248
 - write-through required, 247
- storage control instruction summary, 177
- storage control instructions
 - cache management, 178
 - TLB management, 178
- storage operands, 133
- storage synchronization, 198
- storage synchronization instruction summary, 178
- store gathering, 221
- store miss
 - allocation of data cache line, 221
- store operations, 220
- structure mapping
 - big endian, 137
 - little endian, 137
- stswi, 1185
- stw, 1188
- stwbrx, 1189
- stwcx., 1190
- stwu, 1192
- stwux, 1193
- stwx, 1194
- sub, 1195
- sub., 1195
- subc, 1196
- subc., 1196
- subco, 1196
- subco., 1196
- subf, 1195
- subf., 1195
- subfc, 1196
- subfc., 1196
- subfco, 1196
- subfco., 1196
- subfe, 1197
- subfe., 1197
- subfeo, 1197
- subfeo., 1197
- subfic, 1198
- subfme, 1199
- subfme., 1199
- subfmeo, 1199
- subfmeo., 1199
- subfo, 1195

- subfo., 1195
- subfze, 1200
- subfze., 1200
- subfzeo, 1200
- subfzeo., 1200
- subi, 1027
- subic, 1028
- subic., 1029
- subis, 1030
- subo, 1195
- subo., 1195
- superscalar instruction unit, 61
- supervisor state, 195
- supported configurability, 70
- synchronization
 - architectural references, 196
 - context, 196
 - execution, 198
 - storage, 198
- synchronous interrupt class, 419
- synonyms, instruction cache, 210
- system call instruction, exception priorities for, 456
- System Call interrupt, 444
- system reset results, 262

T

- TAH, 879
- TAHx
 - operations, 882
 - receive, 891
 - transmit, 883
 - power-up and initialization, 899
 - registers, 894
- TAHx_MR, 895, 1698
- TAHx_REVID, 895, 1699
- TAHx_SSR0-TAHx_SSR5, 897, 1700
- TAHx_TSR, 898, 1701
- TAP, 503
- TBL, 1294
- TBU, 1295
- TCPIP, 68
- TCPIP acceleration hardware on ethernet, 879
- TCR, 463, 466, 1296
- test access port *See also* TAP, 503
- time base
 - defined, 460
 - reading, 460
 - writing, 461
- timer freeze (debug), 523
- timers
 - DEC, 461
 - DECAR, 462
 - decrementer, 461, 462
 - FIT, 462
 - fixed interval timer, 462
 - freezing the timer facilities, 467
 - TCR, 466

Preliminary User's Manual

TSR, 467
 watchdog timer, 463
 watchdog timer state machine, 465

TLB
 entry fields
 E, 238
 EPN, 237
 ERPN, 237
 G, 238
 I, 238
 M, 238
 RPN, 237
 SIZE, 237
 TID, 237
 TS, 237
 U0, 238
 U1, 238
 U2, 238
 U3, 238
 UR, 239
 UW, 239
 UX, 238
 V, 237
 W, 238
 overview, 236
 shadow arrays, 253

TLB management instructions
 overview, 253

tlbre, 1201
 tlbsx, 1203
 tlbsx., 1203
 tlbsync, 1204
 tlbwe, 1205
 trace debug mode, 507
 trace port, 505
 transient mechanism, cache, 203
 translation lookaside buffer. *See also* TLB
 trap, 1207
 trap (TRAP) debug events, 520
 trap instructions
 exception priorities for, 456
 TSR, 464, 467, 1297
 tw, 1206
 treq, 1207
 treqi, 1209
 twge, 1207
 twgei, 1209
 twgle, 1207
 twgt, 1207
 twgti, 1209
 twi, 1208
 twle, 1207
 twlei, 1209
 twlgei, 1209
 twlgt, 1207
 twlgti, 1209
 twlle, 1207
 twllei, 1209
 twllt, 1207

twllti, 1209
 twlng, 1207
 twlngi, 1209
 twlnl, 1207
 twlnli, 1209
 twlt, 1207
 twlti, 1209
 twne, 1207
 twnei, 1209
 twng, 1207
 twngi, 1209
 twnl, 1207
 twnli, 1209

U, V, W

U0–U3 storage attributes, 248
 UART controller, 901
 clocking, 902
 DMA operation, 918
 features, 901
 FIFO, 916
 registers, 905
 sleep mode, 917
 UARTx_DLL, 914, 1702
 UARTx_DLM, 914, 1703
 UARTx_FCR, 909, 1704
 UARTx_IER, 906, 1705
 UARTx_IIR, 907, 1706
 UARTx_LCR, 910, 1707
 UARTx_LSR, 912, 1708
 UARTx_MCR, 911, 1710
 UARTx_MSR, 913, 1711
 UARTx_RBR, 906, 1712
 UARTx_SCR, 914, 1713
 UARTx_THR, 906, 1714
 UIC controller, 361
 features, 361
 interrupt assignments, 363
 programming, 366
 registers, 367
 UIC0_CR, 382, 1715
 UIC0_ER, 374, 1717
 UIC0_MSR, 403, 1719
 UIC0_PR, 388, 1721
 UIC0_SR, 368, 1723
 UIC0_TR, 396, 1725
 UIC0_VCR, 409, 1727
 UIC0_VR, 413, 1728
 UIC1_CR, 384, 386, 388, 1729
 UIC1_ER, 377, 379, 381, 1730
 UIC1_MSR, 405, 407, 409, 1733
 UIC1_PR, 391, 393, 395, 1735
 UIC1_SR, 370, 372, 374, 1737
 UIC1_TR, 398, 400, 402, 1739
 UIC1_VCR, 410, 411, 412, 1741
 UIC1_VR, 414, 415, 417, 1742
 UIC2_CR, 1743

UIC2_ER, 1745
UIC2_MSR, 1747
UIC2_PR, 1749
UIC2_SR, 1751
UIC2_TR, 1753
UIC2_VCR, 1755
UIC2_VR, 1756
UICB0_CR, 1757
UICB0_ER, 1758
UICB0_MSR, 1759
UICB0_PR, 1760
UICB0_SR, 1761
UICB0_TR, 1762
UICB0_VCR, 1763
UICB0_VR, 1764
unconditional (UDE) debug events, 522
user mode, 195
USPRG0, 190, 1298
W storage attribute, 247
Watchdog Timer interrupt, 446
watchdog timer interrupts, 446
write-through required, 247
writing the time base, 461
wrtee, 1210
wrteei, 1211

X

XER, 187, 1299
 carry (CA) field, 189
 overflow (OV) field, 188
 summary overflow (SO) field, 188
 transfer byte count (TBC) field, 189
xor, 1212
xori, 1213

Z

ZMII bridge
 features, 795
 interface signals, 795
 interfaces, 796, 811
 registers, 799, 812
ZMII0_FER, 800, 812, 1765
ZMII0_SMIISR, 803, 1766
ZMII0_SSR, 801, 813, 1768

Preliminary User's Manual**Revision Log**

Revision Date	Revision	Contents of Modification
10/01/07	1.22	Pages 526 and 527, Figure 16-3, Updated comment of bitfields IAC12M and IAC34M Page 902, Section 26.2 Serial Port Clocking, replaced first paragraph
09/20/07	1.21	Page 470, Figure 13-1, replaced "comparison result (1/5)" with "comparison result (1/7)". Page 527, Figure 16-3, Bitfield IAC34M, replaced description with "Match if DAC1 & address < DAC2 Match if address < DAC1 OR address & DAC2" Page 797, Figure 23-2, 23.4.1, replace "ZMII0_FER[MDI0,MDI1,MDI2,MDI2]" with "ZMII0_FER[MDI0,MDI1,MDI2,MDI3]" Page 801, Figure 23-6, Figure 23-6, Bitfield FSS0, replaced "ZMII0_SP0" with "ZMII0_SSR[SP0]" Page 902, Section 26.2, updated last sentence first paragraph. From: The internally generated serial clock is derived from the PLB clock, and may only be an integer (n) fraction of that clock where n is in the range from 1 to 32. To: The internally generated serial clock is derived from the PLB clock, and may only be an integer (n) fraction of that clock where n is in the range from 1 to 256. Page 917, Section 26.4.1.2, updated last sentence second paragraph. From: One to 16 characters may be written to the transmitter FIFO while servicing this interrupt. To: One to 64 characters may be written to the transmitter FIFO while servicing this interrupt. Page 935, Figure 27-9, Bitfield SLPR, replace "SDR0_ER" with "CPR0_ER" Page 947, Section 27.3.15, replaced "IICx_DIRECRCNTL" with "IICx_DIRECTCNTL" Page 948, Section 27.4, replaced "IICx_XTCTLSS" with "IICx_XTCNTLSS" (3rd and 4th paragraph) Page 954, Figure 28-2, Bitfield G11E, replaced "GNCTxCik" with "GMCTxCik"
09/10/07	1.20	Page 815, Second paragraph in section 24.7.17, Inter-Packet Gap Value Register (EMACx_IPGVR), is incomplete: IS: The resolution of each bit is 8-bit times. The minimum value in the register is four, causing a minimum. CHANGE TO: The resolution of each bit in the EMACx_IPGVR register is 8-bit times. The minimum value in the register is four, causing the minimum Inter Frame Gap period to be equal to 96-bit times. Page 958, Replace SysClk with PerClk in Figure 30-16. Page 969, Updated 7th paragraph 3rd sentence. Changed EBC0_BnAP[BCE] = 1 to EBC0_BnAP[BCE] = 0.
04/05/07	1.19	Pages 256 - 257, Updated section 6.11.1 Reading TLB Parity Bits with tlbre Page 421, Updated section 11.2.4 Machine Check Interrupts Page 433, Updated Figure 11-12 Machine Check Status Register (MCSR) Pages 436 - 438, Updated section 11.5.2 Machine Check Interrupt Page 761, Updated Figure 23-6 Speed Selection Register (ZMII0_SSR) Page 1218, Updated Table 32-2. Special Purpose Registers Sorted by SPR Number Page 1282, Updated Figure 32-34. Machine Check Status Register (MCSR) Page 1700, Updated Figure 32-459, Speed Selection Register (ZMII0_SSR)
04/03/07	1.18	Page 801 - 802, Figures 23-6, Speed Selection Register bit 10 and 14 definition Page 1768 - 1769, Figures 32-459, Speed Selection Register bit 10 and 14 definition
03/26/07	1.17	Page 229, Updated Figure 5-11 Page 231, Added Section 5.3.3.7 D-Cache Parity Error Recovery Algorithm Pages 352 - 353, Updated Figure 9-5 Pages 355 - 356, Updated Figure 9-10 Page 1337, Updated Figure 32-83 Page 1345, Updated Figure 32-90
03/05/07	1.16	Update ZMII0_FER register definition on pages 794 and 1758.
02/23/07	1.15	Table 4-9 and 32-7 column 3 incorrectly defines SDR0_CPC440 as "R" (read only). Column 3 should show "R/W" (read/write). The MAL chapter has been revised for usability. No register values are affected and as such, no code changes are necessary.

Revision Date	Revision	Contents of Modification
02/07/07	1.14	<p>Page 268, Table 7-2 Reset Value entries for SDR0_MALRBL and SDR0_MALTBL are incorrect. SDR0_MALRBL Reset Value is: 0x00000000 SDR0_MALRBL Reset Value should be: 0x55550000 SDR0_MALTBL Reset Value is: 0x00000000 SDR0_MALTBL Reset Value should be: 0x55550000</p> <p>First line of PCIX)CMD register on page 1548 of the 440GX user manual shows read only from PCI-X. Should be R/W. Is: MMIO 0x2 0EC80004 R/W (PLB) 0x05-0x04 Read-Only (PCI-X) Change to: MMIO 0x2 0EC80004 R/W (PLB) 0x05-0x04 R/W (PCI-X)</p>
01/30/07	1.13	<p>The following statement is reversed in the first paragraph of UM sections 10.5.9, 10.5.10, 10.5.11, and 10.5.12: Is: The processor handles critical interrupts when MSR[EE] = 1 and non-critical interrupts when MSR[CE]=1. Should be: The processor handles non-critical interrupts when MSR[EE] = 1 and critical interrupts when MSR[CE]=1.</p>
11/09/06	1.12	<p>1. Change description of ZMII0_SSR[SCIO] in Figure 23-6 and Figure 32-459 to: EMAC0 Suppress Collision Indication 0 EMAC0 collision indication signal is enabled. 1 EMAC0 collision indication signal is suppressed.</p> <p>2. Change description of ZMII0_SSR[SCI1] in Figure 23-6 and Figure 32-459 to: EMAC0 Suppress Collision Indication 0 EMAC1 collision indication signal is enabled. 1 EMAC1 collision indication signal is suppressed.</p> <p>3. Change two occurrences of ZMII_SSR to ZMII0_SSR in section 23.5.2.</p>
06/26/06	1.11	<p>Page 778, Updated Figure 22-22 MAL0_ESR[CIDT] field description is incorrect. Page 1504, Updated Figure 32-239 MAL0_ESR[CIDT] field description is inconsistent.</p>
05/17/06	1.10	<p>Page 304, Updated Figure 7-32, EPS pin selection Page 308, Updated Figure 7-35, EPS pin selection Page 353, Updated Figure 9-6, EPS pin selection Page 787, Updated Table 23-1 Page 1327, Updated Figure 32-79, EPS pin selection Page 1333, Updated Figure 32-83, EPS pin selection</p>
05/10/06	1.09	<p>Page 67, Updated second paragraph in Section 1.5.11 Memory Access Layer Page 562, Updated Figure 18-15, RFTA row Page 789, Updated Table 23-2</p>
02/28/06	1.08	Page 267-274, Updated Table 7-2 DCR Contents After Reset
01/24/06	1.07	Page 1002, Updated Figure 30-24 Peripheral Bank Access Parameters (EBC0_B0AP-EBC0_B2AP)
01/03/06	1.06	Page 556, Updated Figure 18-11 Refresh Timing Register (SDRAM0_RTR)
12/13/05	1.05	<p>Page 368, Updated register in section "UIC2 Status Register (UIC2_SR)" Page 375, Updated register in section "UIC2 Enable Register (UIC2_ER)" Page 382, Updated register in section "UIC2 Critical Register (UIC2_CR)" Page 389, Updated register in section "UIC2 Polarity Register (UIC2_PR)" Page 396, Updated register in section "UIC2 Trigger Register (UIC2_TR)" Page 403, Updated register in section "UIC2 Masked Status Register (UIC2_MSR)" Page 556, Replaced values of 0x5F0/0x7E0 to 0x05F0/0x07E0 Page 773, Updated Interrupt in Figure 22-11 Page 774, Updated Interrupt in Figure 22-12 Page 799, Updated Table 23-5. Ethernet Muxing for MII, RMII, SMII and RGMII Interfaces Page 803, Updated Table 23-6. Ethernet Muxing for GMII, TBI, and RTBI Interfaces</p>

Preliminary User's Manual

Revision Date	Revision	Contents of Modification
10/03/05	1.04	Pages. 189-190, added DBTAC bit field to Core Configuration Register 0 (CCR0) section. Page 1502, states that register MAL0_ESR bit1 CIDT has Channel ID Type : 0 channel ID represents RX channel 1 channel ID represents TX channel CHANGE: Channel ID(1)='0' - TX Channels Channel ID(1)='1' - RX Channels
8/26/05	1.03	Page 350, In section 9.5.3 Serial Device Strap Register 0 (SDR0_SDSTP0) Change last sentence in the first paragraph from: "If the IIC Bootstrap controller is unable to read the bootstraps from a serial ROM device, strapping option A is used as the default configuration.." To: " If the IIC Bootstrap controller is unable to read the bootstraps from a serial ROM device, strapping option B is used as the default configuration."

Revision Date	Revision	Contents of Modification
6/29/05	1.02	<p>Page 263, SDR0_DDRDL reset value is defined as 0x010010. Should be 0b010010 or 0x00000012.</p> <p>Page 263, Table 7.2 shows SDR0_SRST bits 0:31 Reset according to values received from serial ROM logic. This register gets reset to 0x00000000</p> <p>1. The register reset value in table 7-2 indicates 0x00000000.</p> <p>Reset value for both registers should be defined as 0xF0000000</p> <p>2. Register detail for both these registers in 3 different locations in the user manual indicate that there are 8 bits defined.</p> <p>Change 0000 0000 to: 0b0000</p> <p>Change 1111 1111 to: 0b1111</p> <p>Change description from:</p> <p>"Indicates the OPB slave interface width for the EMAC attached to this channel"</p> <p>To:</p> <p>"Indicates the OPB slave interface width for all four EMAC channels is 128 bits"</p> <p>1. The register reset value in table 7-2 indicates 0x00000000.</p> <p>Reset value for BOTH registers should be defined as 0x55550000</p> <p>2. Detail register definitions for these registers needs to be changed in 3 sections of the user manual (7.7.22, 22.3.3.3, and 32.6).</p> <p>For the SDR0_MALRLB[RBL0] field description change:</p> <p>"Indicates the requested amount of data to be received between MAL and EMAC0 in a single transaction."</p> <p>To:</p> <p>"Defines the maximum burst length transferred between EMAC0 and MAL. Reset default value = 0b101 (64 words)."</p> <p>Repeat above for RBL1, RBL2, and RBL3 fields replacing EMAC0 with EMAC1, EMAC2, and EMAC3 as appropriate.</p> <p>For SDR0_MALTLB[RBL0] field description change:</p> <p>"Indicates the requested amount of data to be transferred between MAL and EMAC0 in a single transaction."</p> <p>To:</p> <p>"Defines the maximum burst length transferred between MAL and EMAC0. Reset default value = 0b101 (64 words)."</p> <p>Repeat above for TBL1, TBL2, and TBL3 fields replacing EMAC0 with EMAC1, EMAC2, and EMAC3 as appropriate.</p> <p>New Content on Page 335:</p> <p>If the IIC bootstrap controller is disabled, the PPC440GX is not limited to the default hardwired bootstrap values. The following algorithm list the steps for programming the boot configuration without reading the bootstrap values from an IIC serial ROM.</p> <p>1. Configure the following CPR0 and SDR0 registers with the desired configuration:</p> <p>CPR0_PLLC</p> <p>CPR0_PLLD</p> <p>CPR0_PRIMBD</p> <p>CPR0_OPBD</p> <p>CPR0_PERD</p> <p>CPR0_MALD</p> <p>SDR0_EBC</p> <p>SDR0_CP440</p> <p>SDR0_XCR</p> <p>SDR0_PFC0</p> <p>SDR0_PFC1</p> <p>SDR0_CUST0</p> <p>SDR0_CUST1</p> <p>2. Set CPR0_ICFG[RLI]=1. The Reload Inhibit bit preserves the configuration set in step 1 through a chip reset. The preserved register contents are used as the boot configuration after the chip reset.</p> <p>3. Set DBCR0[RST]=0b10 to generate a chip reset.</p>

Preliminary User's Manual

Revision Date	Revision	Contents of Modification
6/29/05 (Cont.)	1.02	<p>Section 13.4.1 - Remove this section.</p> <p>Add the note (shown below) under Table 14-6 CPR0_PLLC[TUNE] Bit Settings on page 475 of 440GX User's Manual:</p> <p>Note: 440GX evaluation board factory default PLL Tune Bit Setting may vary from value(s) indicated in Table 14-6. Page 790, Figure 23-5. Function Enable Register (ZMII0_FER) contains typo in bit 6 and 7 description</p> <p>Original:</p> <p>EMAC0 RMII Enable</p> <p>0 EMAC1 RMII PHY interface is disabled.</p> <p>1 EMAC1 RMII PHY interface is enabled.</p> <p>EMAC0 MII Enable</p> <p>0 EMAC1 MII PHY interface is disabled.</p> <p>1 EMAC1 MII PHY interface is enabled.</p> <p>Suggested Change:</p> <p>EMAC1 RMII Enable</p> <p>0 EMAC1 RMII PHY interface is disabled.</p> <p>1 EMAC1 RMII PHY interface is enabled.</p> <p>EMAC1 MII Enable</p> <p>0 EMAC1 MII PHY interface is disabled.</p> <p>1 EMAC1 MII PHY interface is enabled.</p> <p>Page 893, In "PLB Clock =133 MHz" section of the table, "PLB:UART divide ratio" (column 2) "28" should be "14" for the first 5 entries of that section.</p> <p>Table 33-1 on page 1762 and table 33-2 on page 1765 incorrectly lists PCIX133CAP as an input. This signal is an output.</p>
1/21/05	1.02	Re-released preliminary AMCC version for external distribution.
11/21/03		Released updated preliminary version (SA14-2687-05) for external distribution.
11/03/03 to 11/17/03		<p>Updated the following chapters: programming model, instruction and data caches, debug facilities, reset and initialization, clocking, ddr sdram, GPIO operations, IIC interface and corresponding registers. Also updated the appendix on instruction execution performance and code optimizations.</p> <p>Note in clocking chapter to the effect that the value of PLLOUTA and PLLOUTB must not exceed the rated speed of the processor core. Updated corresponding tables accordingly.</p> <p>CPR0_CLKUPD register has been noted as reserved</p> <p>UICB and UIC2 address map updated</p> <p>Added table note for SDR0_MALTBL, SDR0_MALTBS, SDR0_MALRBL, SDR0_MALRBS registers about using default values for all the the bits described.</p> <p>Clarified the use of SDR0_MFR register bit 2.</p> <p>Updated SDR0_PFC0 and SDR0_PFC1 register bits</p> <p>Debug chapter: Included more information to the DVC debug event description regarding special handling for guarded loads</p> <p>Cache chapter: CCR0[ICSLT] field removed as it no longer applicable. Updated description of the dcread instruction operation, to point out that software must prevent interrupts from separating the msync and the dcread.</p> <p>Prgmodel chapter: added the isel instruction to the list of instructions that access the CR. Also made minor corrections to some table headers regarding allocated instructions, and the text for instructions that access XER[SO].</p>
07/21/03		This updated preliminary version (SA14-2687-04) reflects changes and updates made since the previous release dated 02/11/03. Updates made to this release affected multiple sections of this manual as a result of extensive feedback and review/comments received from hardware, software and application design team. Detailed information on changes and updates requested and reviewed can be found in individual emails and pdf attachments. This column lists overall functional and template related changes and updates made.

Preliminary User's Manual

Revision Date	Revision	Contents of Modification
06/16/03 to 07/15/03		<p>Overview chapter updates include rewrite of some sections and additional editing.</p> <p>On-chip peripheral bus chapter updates include modifications to some registers and sections.</p> <p>PLB performance monitor chapter has additional information included on its functionality.</p> <p>Programming model chapter includes minor updates to tables containing dcrs and mmios.</p> <p>Reset and initialization chapter updates include rewrite of some sections as well as inclusion of new power-on and system reset diagrams.</p> <p>L2 cache chapter updates include some minor fixes and editing.</p> <p>Bootstrap controller updates included rewrite to some sections and updates to bootstrap configuration and strapping tables and registers.</p> <p>Universal interrupt controller chapter updates include switching of UIC2 and UICB DCR address map.</p> <p>Interrupts and exceptions chapter updates include rewrite and contoning of information pertaining to floating point unit and auxilliary processor unit enabled interrupts.</p> <p>General purpose timers chapter updates include rewrite and inclusion of a block diagram showing comparison between timebase counter and compare registers. Output enable registers removed.</p> <p>Clocking chapter updates include updates to system PLL configuration tables and PLL tune bit setting. Section on dynamically changing clocking modes is conditioned out.</p> <p>Minor updates to SRAM, DDR SDRAM, PCIX, DMA, and IMU chapters made.</p> <p>Media access layer chapter updates include some rewrite and fixes.</p> <p>Ethernet to Phy bridge chapter updates include minor rewrite and modifications to ethernet combinations table and ZMII and RGMII bridge configurations diagram.</p> <p>EMAC chapter updates include numerous edits and inclusion of 2 GBCS units (with registers) one for each gigabit ethernet.</p> <p>Minor updates to TCP/IP hardware acceleration chapter.</p> <p>Serial port operations chapter updates include new baud rate settings table.</p> <p>Minor updates to ebmi and ebco chapters.</p> <p>Multiple cpr0 and sdr0 registers reviewed and updated accordingly.</p> <p>Register summaries reviewed and updated accordingly.</p> <p>Implemented additional index markers.</p>
02/11/03		This preliminary version (SA14-2687-03) reflects all changes and updates made since the previous release dated 11/21/02.
02/07/03		Imported registers in interrupt chapter. Minor changes to all UIC2 register figures. Updated EMAC chapter where MAL RX descriptor status bit 5 (parity error) is now handled by MAL. GPT chapter updated. Updated RGMII_FER and RGMII_SSR registers. EMAC2 and EMAC3 channels switched based on RGMII core spec.
01/31/03		Updated signal summary chapter, bootstrap controller chapter. Added addresses for all EMAC 2 and EMAC3 registers in EMAC register summary chapter. Updated UIC interrupt assignment table for EMAC2 and EMAC3 with PCS interrupt status.
01/22/03		Updated initialization software requirements section in reset and initialization chapter. Made minor clarification to L2C0_SNP0:L2C0_SNP1 register. Minor updates to SDR0_PFC1 register. ZMII0_SMIISR register has been updated to reflect correct mapping of bits.
01/15/03		SDR0_SRST register bit 29 is now reserved. Bit 3 of CPM0_SR, CPM0_FR and CPM0_ER is now reserved. Removed requests from PCIX from SDR0_MIRQ0 and SDR0_MIRQ1 registers. Updated CCR1 register address to 0x378. Updated default base address and SRAM0_SB0CR-SRAM0_SB3CR register in SRAM chapter. Fixed figure in TCP/IP chapter. Updated SDR0_SDSTP1[EARV] register bit. SDR0_CP440 register bits have been updated with more information. Updated SDR0_AMP register bits in on-chip bus chapter. CPR0_CLKUPD register has been updated to reverse CUI and CUD bits. Updated CPR0_CFGDATA and CPR0_CFGADDR register. Minor updates to L2 cache chapter. Updated IMU chapter and registers. Minor updates made to GPT chapter.
01/10/03		Updated reset types with a block diagram in reset and initialization chapter. Also updated JTAG instruction table in debug chapter.
11/21/02		This preliminary version (SA14-2687-02) reflects all changes and updates made to each and every chapter that underwent design and application engineer review. Each chapter has been sourced and reviewed to meet individual chapter content model requirements.
09/30/02		This is the second draft of the manual (SA14-2687-01) with individual chapters sourced and updated based on preliminary selective review and number of TBDs completed. New chapters have been added based on additional information requirements.
06/25/02		This is the first draft of the manual (SA14-2687-00) which includes advance information, definitions for information requirements, placeholders, and various TBDs