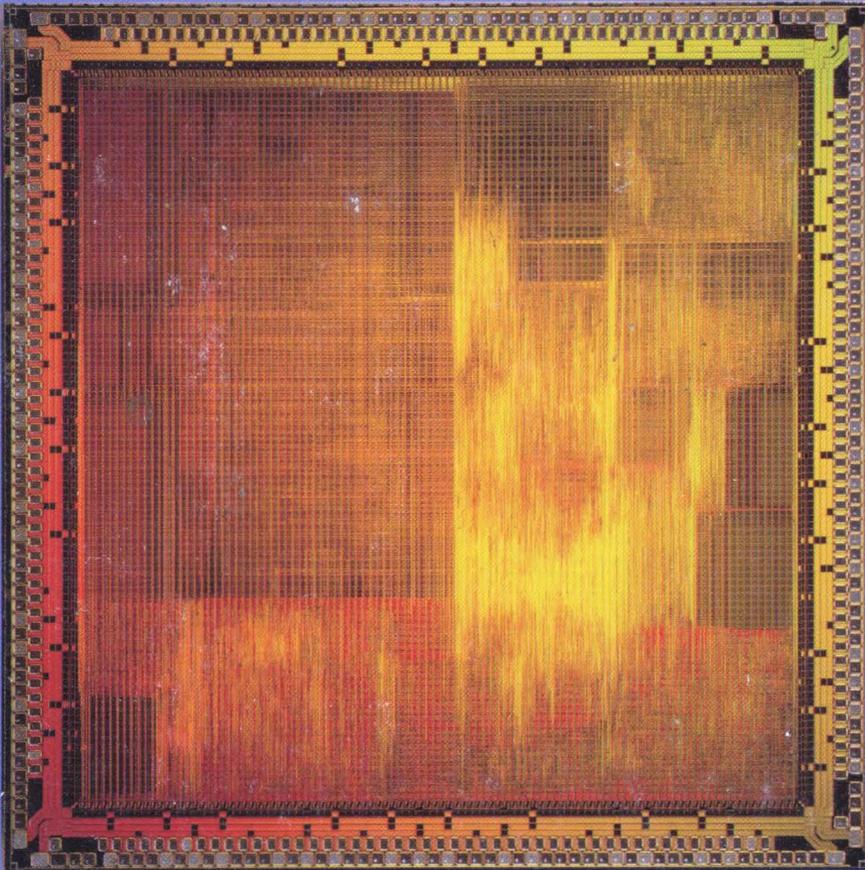


PowerPC 405GP Embedded Processor

User's Manual, Volume 2

Preliminary



IBM

PowerPC



**PowerPC 405GP
Embedded Processor
User's Manual**

Preliminary

Volume 2

PowerPC[™]

GK10-3118-03

Sixth Preliminary Edition (June 2000)

This edition of *IBM PowerPC 405GP Embedded Processor User's Manual* applies to the IBM PowerPC™ 405GP 32-bit embedded processor, until otherwise indicated in new versions or application notes.

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS MANUAL "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

IBM does not warrant that the products in this publication, whether individually or as one or more groups, will meet your requirements or that the publication or the accompanying product descriptions are error-free.

This publication could contain technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or program(s) described in this publication at any time.

It is possible that this publication may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country. Any reference to an IBM licensed program in this publication is not intended to state or imply that you can use only IBM's licensed program. You can use any functionally equivalent program instead.

While the information contained herein is believed to be accurate, such information is preliminary, and should not be relied upon for accuracy or completeness, and no representations or warranties of accuracy or completeness are made.

No part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the written permission of IBM.

Requests for copies of this publication and for technical information about IBM products should be made to your IBM Authorized Dealer or your IBM Marketing Representative.

Address technical queries about this product to ppcsupp@us.ibm.com

Address comments about this publication to:

IBM Corporation
Départment YM5A
P.O. Box 12195
Research Triangle Park, NC 27709

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1996, 2000. All rights reserved

4 3 2 1

Notice to U.S. Government Users – Documentation Related to Restricted Rights – Use, duplication, or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corporation.

Patents and Trademarks

IBM may have patents or pending patent applications covering the subject matter in this publication. The furnishing of this publication does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504, United States of America.

The following terms are trademarks of IBM Corporation:

IBM
CodePack
CoreConnect
PowerPC
PowerPC Architecture
RISCTrace
RISCWatch

Other terms which are trademarks are the property of their respective owners.

Contents

Figures	xxix
Tables	xli
About This Book	xlvii
Who Should Use This Book	xlvii
How to Use This Book	xlvii
Conventions	xlviii
Part I. Introducing the PPC405GP Embedded Processor	I-1
Chapter 1. Overview	1-1
PPC405GP Features	1-2
Bus and Peripheral Features	1-2
PowerPC Processor Core Features	1-3
PowerPC Architecture	1-4
The PPC405GP as a PowerPC Implementation	1-5
RISC Processor Core Organization	1-5
Instruction and Data Cache Controllers	1-5
Instruction Cache Unit	1-6
Data Cache Unit	1-6
Memory Management Unit	1-6
Timer Facilities	1-8
Debug	1-8
Development Tool Support	1-8
Debug Modes	1-9
Processor Core Interfaces	1-9
Processor Local Bus	1-9
Device Control Register Bus	1-9
Clock and Power Management	1-9
JTAG	1-9
Interrupts	1-9
On-Chip Memory	1-9
Processor Core Programming Model	1-9
Data Types	1-10
Processor Core Register Set Summary	1-10
General Purpose Registers	1-10
Special Purpose Registers	1-10
Machine State Register	1-10
Condition Register	1-11
Device Control Registers	1-11
Memory-Mapped I/O Registers	1-11
Addressing Modes	1-11
Chapter 2. On-Chip Buses	2-1
Processor Local Bus	2-1
PLB Features	2-2
PLB Masters and Slaves	2-2
PLB Master Assignments	2-2
PLB Transfer Protocol	2-3
Overlapped PLB Transfers	2-4
PLB Arbiter Registers	2-5
PLB Arbiter Control Register (PLB0_ACR)	2-5
PLB Error Address Register (PLB0_BEAR)	2-5

PLB Error Status Register (PLB0_BESR)	2-6
PLB to OPB Bridge Registers	2-8
Bridge Error Address Register (POB0_BEAR)	2-8
Bridge Error Status Registers (POB0_BESR0–POB0_BESR1)	2-8
On-Chip Peripheral Bus	2-11
OPB Features	2-11
OPB Master Assignments	2-12
OPB Arbiter Registers	2-12
OPB Arbiter Control Register (OPBA0_CR)	2-12
OPB Arbiter Priority Register (OPBA0_PR)	2-13
Part II. The PPC405GP RISC Processor	II-1
Chapter 3. Programming Model	3-1
User and Privileged Programming Models	3-1
Memory Organization and Addressing	3-1
Physical Address Map	3-2
Storage Attributes	3-3
Registers	3-3
General Purpose Registers (R0-R31)	3-6
Special Purpose Registers	3-6
Count Register (CTR)	3-7
Link Register (LR)	3-8
Fixed Point Exception Register (XER)	3-8
Special Purpose Register General (SPRG0–SPRG7)	3-11
Processor Version Register (PVR)	3-12
Condition Register (CR)	3-12
CR Fields after Compare Instructions	3-13
The CR0 Field	3-14
The Time Base	3-15
Machine State Register (MSR)	3-15
Device Control Registers	3-16
Directly Accessed DCRs	3-17
Indirectly Accessed DCRs	3-19
Indirect Access of SDRAM Controller DCRs	3-19
Indirect Access of External Bus Controller DCRs	3-20
Indirect Access of Decompression Controller DCRs	3-21
Memory-Mapped Input/Output Registers	3-22
Directly Accessed MMIO Registers	3-22
Indirectly Accessed MMIO Registers	3-25
Data Types and Alignment	3-26
Alignment for Storage Reference and Cache Control Instructions	3-27
Alignment and Endian Operation	3-27
Summary of Instructions Causing Alignment Exceptions	3-28
Byte Ordering	3-28
Structure Mapping Examples	3-29
Big Endian Mapping	3-29
Little Endian Mapping	3-30
Support for Little Endian Byte Ordering	3-30
Endian (E) Storage Attribute	3-30
Fetching Instructions from Little Endian Storage Regions	3-31
Accessing Data in Little Endian Storage Regions	3-31
PowerPC Byte-Reverse Instructions	3-32
Instruction Processing	3-33
Branch Processing	3-34
Unconditional Branch Target Addressing Options	3-34
Conditional Branch Target Addressing Options	3-35

Conditional Branch Condition Register Testing	3-35
BO Field on Conditional Branches	3-35
Branch Prediction	3-36
Speculative Accesses	3-37
Speculative Accesses in the PPC405GP	3-37
Prefetch Distance Down an Unresolved Branch Path	3-38
Prefetch of Branches to the CTR and Branches to the LR	3-38
Preventing Inappropriate Speculative Accesses	3-38
Fetching Past an Interrupt-Causing or Interrupt-Returning Instruction	3-39
Fetching Past tw or twi Instructions	3-39
Fetching Past an Unconditional Branch	3-40
Suggested Locations of Memory-Mapped Hardware	3-40
Summary	3-41
Privileged Mode Operation	3-41
MSR Bits and Exception Handling	3-41
Privileged Instructions	3-42
Privileged SPRs	3-42
Privileged DCRs	3-43
Synchronization	3-43
Context Synchronization	3-44
Execution Synchronization	3-46
Storage Synchronization	3-46
Instruction Set	3-47
Instructions Specific to IBM PowerPC Embedded Processors	3-48
Storage Reference Instructions	3-48
Arithmetic Instructions	3-49
Logical Instructions	3-50
Compare Instructions	3-50
Branch Instructions	3-51
CR Logical Instructions	3-51
Rotate Instructions	3-51
Shift Instructions	3-52
Cache Management Instructions	3-52
Interrupt Control Instructions	3-52
TLB Management Instructions	3-53
Processor Management Instructions	3-53
Extended Mnemonics	3-53
Chapter 4. Cache Operations	4-1
ICU Organization	4-2
ICU Operations	4-3
Instruction Cachability Control	4-4
Instruction Cache Synonyms	4-4
ICU Coherency	4-5
DCU Organization	4-5
DCU Operations	4-6
DCU Write Strategies	4-7
DCU Load and Store Strategies	4-7
Data Cachability Control	4-8
DCU Coherency	4-9
Cache Instructions	4-9
ICU Instructions	4-9
DCU Instructions	4-10
Cache Control and Debugging Features	4-11
CCR0 Programming Guidelines	4-13
ICU Debugging	4-14

DCU Debugging	4-15
DCU Performance	4-16
Pipeline Stalls	4-16
Cache Operation Priorities	4-17
Simultaneous Cache Operations	4-18
Sequential Cache Operations	4-18
Chapter 5. On-Chip Memory	5-1
OCM Addressing	5-1
OCM Programming Guidelines	5-2
Store Data Bypass Behavior and Memory Coherency	5-3
Registers	5-5
OCM Instruction-Side Address Range Compare Register (OCM0_ISARC)	5-5
OCM Instruction-Side Control Register (OCM0_ISCNTL)	5-6
OCM Data-Side Address Range Compare Register (OCM0_DSARC)	5-6
OCM Data-Side Control Register (OCM0_DSCNTL)	5-7
Chapter 6. Memory Management	6-1
MMU Overview	6-1
Address Translation	6-1
Translation Lookaside Buffer (TLB)	6-2
Unified TLB	6-2
TLB Fields	6-3
Page Identification Fields	6-3
Translation Field	6-4
Access Control Fields	6-5
Storage Attribute Fields	6-5
Shadow Instruction TLB	6-6
ITLB Accesses	6-6
Shadow Data TLB	6-7
DTLB Accesses	6-7
Shadow TLB Consistency	6-8
TLB-Related Interrupts	6-9
Data Storage Interrupt	6-9
Instruction Storage Interrupt	6-10
Data TLB Miss Interrupt	6-10
Instruction TLB Miss Interrupt	6-10
TLB Management	6-10
TLB Search Instructions (tlbsx/tlbsx.)	6-11
TLB Read/Write Instructions (tlbre/tlbwe)	6-11
TLB Invalidate Instruction (tlbia)	6-11
TLB Sync Instruction (tlbsync)	6-11
Recording Page References and Changes	6-11
Access Protection	6-12
Access Protection Mechanisms in the TLB	6-12
General Access Protection	6-12
Execute Permissions	6-13
Write Permissions	6-13
Zone Protection	6-13
Access Protection for Cache Control Instructions	6-15
Access Protection for String Instructions	6-16
Real-mode Storage Attribute Control	6-17
Storage Attribute Control Registers	6-18
Data Cache Write-through Register (DCWR)	6-18
Data Cache Cachability Register (DCCR)	6-19
Instruction Cache Cachability Register (ICCR)	6-19
Storage Guarded Register (SGR)	6-19

Storage User-defined 0 Register (SU0R)	6-19
Storage Little-Endian Register (SLER)	6-19
Part III. PPC405GP System Operations	III-1
Chapter 7. Clocking	7-1
PLL Overview	7-1
Input Reference Clock (SysClk)	7-2
External Clock Strapping Setup	7-3
Sample Clock Ratios	7-4
PCI Clocking	7-7
PCI Clocks	7-8
PCI Adapter Applications	7-8
Serial Port Clocking	7-9
Clocking Registers	7-9
PLL Mode Register (CPC0_PLLMR)	7-10
Chip Control Register 0 (CPC0_CR0)	7-11
Chapter 8. Reset and Initialization	8-1
Reset Signals	8-1
Reset Types	8-1
Core Reset	8-1
Chip Reset	8-1
System Reset	8-2
PCI Power Management Initiated Resets	8-2
Processor Initiated Resets	8-2
Processor State After Reset	8-2
Processor Register Contents After Reset	8-3
Machine State Register Contents after Reset	8-3
Contents of Special Purpose Registers after Reset	8-4
DCR Contents after Reset	8-4
MMIO Register Contents After Reset	8-8
PPC405GP Chip Initialization	8-12
OCM Initialization	8-13
Initializing Instruction-Side OCM	8-13
Initializing Data-Side OCM	8-13
UIC Initialization	8-14
UART Initialization	8-14
PPC405GP Initial Processor Sequencing	8-14
Initialization Requirements	8-15
Initialization Code Example	8-16
Chapter 9. Pin Strapping and Sharing	9-1
Pin Strapping	9-1
Chip Pin Strapping Register (CPC0_PSR)	9-1
Pin Sharing	9-3
Chapter 10. Interrupt Controller Operations	10-1
UIC Overview	10-1
UIC Features	10-1
UIC Interrupt Assignments	10-1
Interrupt Programmability	10-3
UIC Registers	10-3
UIC Status Register (UIC0_SR)	10-3
UIC Enable Register (UIC0_ER)	10-6
UIC Critical Register (UIC0_CR)	10-8
UIC Polarity Register (UIC0_PR)	10-10
UIC Trigger Register (UIC0_TR)	10-13

UIC Masked Status Register (UIC0_MSR)	10-16
UIC Vector Configuration Register (UIC0_VCR)	10-18
UIC Vector Register (UIC0_VR)	10-19
Using the Value in UIC0_VR as a Vector Address or Entry Table Lookup	10-20
Vector Generation Scenarios	10-20
Interrupt Handling in the Processor Core	10-22
Architectural Definitions and Behavior	10-22
Behavior of the PPC405GP Implementation	10-23
Interrupt Handling Priorities	10-24
Critical and Noncritical Interrupts	10-26
General Interrupt Handling Registers	10-27
Machine State Register (MSR)	10-28
Save/Restore Registers 0 and 1 (SRR0–SRR1)	10-29
Save/Restore Registers 2 and 3 (SRR2–SRR3)	10-30
Exception Vector Prefix Register (EVPR)	10-31
Exception Syndrome Register (ESR)	10-31
Data Exception Address Register (DEAR)	10-34
Critical Input Interrupts	10-34
Machine Check Interrupts	10-35
Instruction Machine Check Handling	10-35
Data Machine Check Handling	10-36
Data Storage Interrupt	10-36
Instruction Storage Interrupt	10-38
External Interrupt	10-38
External Interrupt Handling	10-39
Alignment Interrupt	10-39
Program Interrupt	10-40
System Call Interrupt	10-41
Programmable Interval Timer (PIT) Interrupt	10-41
Fixed Interval Timer (FIT) Interrupt	10-42
Watchdog Timer Interrupt	10-43
Data TLB Miss Interrupt	10-43
Instruction TLB Miss Interrupt	10-44
Debug Interrupt	10-44
Chapter 11. Timer Facilities	11-1
Time Base	11-2
Reading the Time Base	11-3
Writing the Time Base	11-3
Programmable Interval Timer (PIT)	11-4
Fixed Interval Timer (FIT)	11-5
Watchdog Timer	11-6
Timer Status Register (TSR)	11-8
Timer Control Register (TCR)	11-9
Chapter 12. Debugging	12-1
Development Tool Support	12-1
Debug Interfaces	12-1
IEEE 1149.1 Test Access Port (JTAG Debug Port)	12-1
JTAG Connector	12-2
JTAG Instructions	12-3
JTAG Boundary Scan	12-3
JTAG Implementation	12-4
JTAG ID Register (CPC0_JTAGID)	12-4
Trace Port	12-5
Debug Modes	12-6
Internal Debug Mode	12-6

External Debug Mode	12-6
Debug Wait Mode	12-7
Real-time Trace Debug Mode	12-7
Processor Control	12-8
Processor Status	12-8
Debug Registers	12-8
Debug Control Registers	12-9
Debug Control Register 0 (DBCR0)	12-9
Debug Control Register1 (DBCR1)	12-11
Debug Status Register (DBSR)	12-12
Instruction Address Compare Registers (IAC1–IAC4)	12-14
Data Address Compare Registers (DAC1–DAC2)	12-14
Data Value Compare Registers (DVC1–DVC2)	12-15
Debug Events	12-16
Instruction Complete Debug Event	12-16
Branch Taken Debug Event	12-17
Exception Taken Debug Event	12-17
Trap Taken Debug Event	12-17
Unconditional Debug Event	12-17
IAC Debug Event	12-17
IAC Exact Address Compare	12-17
IAC Range Address Compare	12-17
DAC Debug Event	12-18
DAC Exact Address Compare	12-19
DAC Range Address Compare	12-19
Data Address Compare (DAC) Applied to Cache Instructions	12-20
DAC Applied to String Instructions	12-21
Data Value Compare Debug Event	12-21
Imprecise Debug Event	12-23
Chapter 13. Clock and Power Management	13-1
CPM Registers	13-1
CPM Enable Register (CPC0_ER)	13-3
CPM Force Register (CPC0_FR)	13-3
CPM Status Register (CPC0_SR)	13-3
Chapter 14. Decompression Controller Operation	14-1
Code Compression and Decompression	14-1
Code Compression	14-1
Code Decompression	14-2
Instruction Fetches to Compressed Pages	14-2
Instruction Fetches to Uncompressed Pages	14-3
Performance	14-3
Decompression Controller Registers	14-3
Index Table Origin Registers (DCP0_ITOR0–DCP0_ITOR3)	14-4
Decompression Address Decode Definition Registers (DCP0_ADDR0–DCP0_ADDR1)	14-5
Decompression Configuration Register (DCP0_CFG)	14-5
Decompression Controller ID Register (DCP0_ID)	14-6
Decompression Controller Version Register (DCP0_VER)	14-6
Decompression Controller PLB Error Address Register (DCP0_PLBBEAR)	14-7
Decompression Controller Bus Error Address Register (DCP0_MEMBEAR)	14-7
Decompression Controller Error Status Register 0 (DCP0_ESR)	14-7
Part IV. PPC405GP External Interfaces	IV-1
Chapter 15. SDRAM Controller	15-1
Interface Signals	15-1
Accessing SDRAM Registers	15-2

SDRAM Controller Configuration and Status	15-3
Memory Controller Configuration Register (SDRAM0_CFG)	15-3
Memory Controller Status (SDRAM0_STATUS)	15-5
Memory Bank 0–3 Configuration (SDRAM0_B0CR–SDRAM0_B3CR)	15-6
Page Management	15-7
Logical Address to Memory Address Mapping	15-8
SDRAM Timing Register (SDRAM0_TR)	15-9
Selected Timing Diagrams	15-10
Auto (CAS Before RAS) Refresh	15-13
Refresh Timer Register (SDRAM0_RTR)	15-13
Error Checking and Correction (ECC)	15-14
ECC Configuration Register (SDRAM0_ECCCFG)	15-14
Correctable ECC Errors	15-15
Uncorrectable ECC Errors	15-15
Error Locking	15-16
ECC Error Status Register (SDRAM0_ECCEsr)	15-16
Bus Error Address Register (SDRAM0_BEAR)	15-17
Bus Error Syndrome Register 0 (SDRAM0_BESR0)	15-17
Bus Error Syndrome Register 1 (SDRAM0_BESR1)	15-18
Self-Refresh	15-19
Power Management	15-20
Sleep Mode Entry	15-20
Power Management Idle Timer (SDRAM0_PMIT)	15-20
Sleep Mode Exit	15-21
Chapter 16. External Bus Controller	16-1
Interface Signals	16-1
Interfacing to Byte, Halfword and Word Devices	16-3
Multiplexed I/Os	16-4
Driver Enables	16-4
Non-Burst Peripheral Bus Transactions	16-5
Single Read Transfer	16-6
Single Write Transfer	16-7
Burst Transactions	16-8
Burst Read Transfer	16-9
Burst Write Transfer	16-10
Device-Paced Transfers	16-11
Device-Paced Single Read Transfer	16-12
Device-Paced Single Write Transfer	16-13
Device-Paced Burst Read Transfer	16-14
Device-Paced Burst Write Transfer	16-15
External Bus Master Interface	16-17
Arbitration	16-17
Transaction Overview	16-19
Single Read and Single Write Transfers	16-19
Burst Read Transfer	16-20
Burst Write Transfer	16-21
External Master Error Interrupts	16-22
EBC Registers	16-23
EBC Configuration Register (EBC0_CFG)	16-23
Peripheral Bank Configuration Registers (EBC0_BnCR)	16-25
Peripheral Bank Access Parameters (EBC0_BnAP)	16-26
Error Reporting	16-29
Error Locking	16-29
Peripheral Bus Error Address Register (EBC0_BEAR)	16-29
Peripheral Bus Error Status Register 0 (EBC0_BESR0)	16-30

Peripheral Bus Error Status Register 1 (EBC0_BESR1)	16-32
Chapter 17. PCI Interface	17-1
PCI Overview	17-1
PCI Bridge Features	17-1
PCI Bridge Block Diagram	17-2
Byte Ordering	17-2
Reference Information	17-3
PCI Bridge Functional Blocks	17-4
PLB-to-PCI Half-Bridge	17-4
PCI-to-PLB Half-Bridge	17-5
PCI Arbiter	17-5
PCI Bridge Address Mapping	17-6
PLB-to-PCI Address Mapping	17-6
PCI-to-PLB Address Mapping	17-8
PCI Target Map Configuration	17-9
PCI Bridge Transaction Handling	17-10
PLB-to-PCI Transaction Handling	17-10
PCI Master Commands	17-10
PLB Slave Read Handling	17-12
Prefetching	17-12
PLB Slave Write Handling	17-12
Aborted PLB Requests	17-13
Retried PCI Reads	17-13
PCI-to-PLB Transaction Handling	17-13
PLB Master Commands	17-14
Handling of Reads from PCI Masters	17-15
Handling Writes from PCI Masters	17-17
Miscellaneous	17-17
Completion Ordering	17-18
PCI Producer-Consumer Model	17-18
Collision Resolution	17-18
PCI Bridge Configuration Registers	17-19
PCI Bridge Register Summary	17-19
PCI Bridge Local Configuration Registers	17-21
PMM 0 Local Address Register (PCIL0_PMM0LA)	17-21
PMM 0 Mask/Attribute Register (PCIL0_PMM0MA)	17-22
PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA)	17-22
PMM 0 PCI High Address Register (PCIL0_PMM0PCIHA)	17-23
PMM 1 Local Address Register (PCIL0_PMM1LA)	17-23
PMM 1 Mask/Attribute Register (PCIL0_PMM1MA)	17-24
PMM 1 PCI Low Address Register (PCIL0_PMM1PCILA)	17-24
PMM 1 PCI High Address Register (PCIL0_PMM1PCIHA)	17-25
PMM 2 Local Address Register (PCIL0_PMM2LA)	17-25
PMM 2 Mask/Attribute Register (PCIL0_PMM2MA)	17-25
PMM 2 PCI Low Address Register (PCIL0_PMM2PCILA)	17-26
PMM 2 PCI High Address Register (PCIL0_PMM2PCIHA)	17-27
PTM 1 Memory Size/Attribute Register (PCIL0_PTM1MS)	17-27
PTM 1 Local Address Register (PCIL0_PTM1LA)	17-27
PTM 2 Memory Size/Attribute Register (PCIL0_PTM2MS)	17-28
PTM 2 Local Address Register (PCIL0_PTM2LA)	17-28
PCI Configuration Registers	17-29
PCI Configuration Address Register (PCIC0_CFGADDR)	17-29
PCI Configuration Data Register (PCIC0_CFGDATA)	17-30
PCI Vendor ID Register (PCIC0_VENDID)	17-31
PCI Device ID Register (PCIC0_DEVID)	17-31

PCI Command Register (PCIC0_CMD)	17-31
PCI Status Register (PCIC0_STATUS)	17-33
PCI Revision ID Register (PCIC0_REVID)	17-34
PCI Class Register (PCIC0_CLS)	17-35
PCI Cache Line Size Register (PCIC0_CACHELS)	17-35
PCI Latency Timer Register (PCIC0_LATTIM)	17-36
PCI Header Type Register (PCIC0_HDTYPE)	17-36
PCI Built-In Self Test (BIST) Control Register (PCIC0_BIST)	17-37
Unused PCI Base Address Register Space	17-37
PCI PTM 1 BAR (PCIC0_PTM1BAR)	17-37
PCI PTM 2 BAR (PCIC0_PTM2BAR)	17-38
PCI Subsystem Vendor ID Register (PCIC0_SBSYSVID)	17-39
PCI Subsystem ID Register (PCIC0_SBSYSID)	17-39
PCI Capabilities Pointer (PCIC0_CAP)	17-40
PCI Interrupt Line Register (PCIC0_INTLN)	17-40
PCI Interrupt Pin Register (PCIC0_INTPN)	17-41
PCI Minimum Grant Register (PCIC0_MINGNT)	17-41
PCI Maximum Latency Register (PCIC0_MAXLTNCY)	17-41
PCI Interrupt Control/Status Register (PCIC0_ICS)	17-42
Error Enable Register (PCIC0_ERREN)	17-42
Error Status Register (PCIC0_ERRSTS)	17-43
Bridge Options 1 Register (PCIC0_BRDGOPT1)	17-44
PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0)	17-45
PLB Slave Error Syndrome Register 1 (PCIC0_PLBBESR1)	17-47
PLB Slave Error Address Register (PCIC0_PLBBEAR)	17-48
Capability Identifier (PCIC0_CAPID)	17-49
Next Item Pointer (PCIC0_NEXTIPTR)	17-49
Power Management Capabilities (PCIC0_PMC)	17-50
Power Management Control/Status Register (PCIC0_PMCSR)	17-50
PMCSR PCI-to-PCI Bridge Support Extensions (PCIC0_PMCSRSE)	17-51
PCI Data Register (PCIC0_DATA)	17-52
Bridge Options 2 Register (PCIC0_BRDGOPT2)	17-52
Power Management State Change Request Register (PCIC0_PMSCRR)	17-54
Error Handling	17-55
PLB Unsupported Transfer Type	17-55
PCI Master Abort	17-55
Bridge PCI Master Receives Target Abort While PCI Bus Master	17-56
PCI Target Data Bus Parity Error Detection	17-57
PCI Master Data Bus Parity Error Detection	17-57
PCI Address Bus Parity Error While PCI Target	17-58
PLB Master Bus Error Detection	17-58
PCI Bridge Clocking Configuration	17-59
PCI Power Management Interface	17-59
Capabilities and Power Management Status and Control Registers	17-59
Power State Control	17-59
Changing Power States	17-60
PCI Bridge Reset and Initialization	17-61
Address Map Initialization	17-61
Other Configuration Register Initialization	17-63
Target Bridge Initialization	17-63
Local Processor Boot from PCI Memory	17-64
Type 0 Configuration Cycles for Other Devices	17-64
Timing Diagrams	17-64
PCI Timing Diagram Descriptions	17-65
PCI Master Burst Read From SDRAM	17-65
PCI Master Burst Write To SDRAM	17-65

CPU Read From PCI Memory Slave, Nonprefetching	17-65
CPU Read From PCI Memory Slave, Prefetching	17-65
CPU Write To PCI Memory Slave	17-65
PCI Memory To SDRAM DMA Transfer	17-66
SDRAM To PCI Memory DMA Transfer	17-66
Asynchronous	17-66
Synchronous	17-92
Chapter 18. Direct Memory Access Controller	18-1
External Interface Signals	18-1
Functional Overview	18-2
Peripheral Mode Transfers	18-2
Memory-to-Memory Transfers	18-3
Scatter/Gather Transfers	18-4
Configuration and Status Registers	18-4
DMA Polarity Configuration Register (DMA0_POL)	18-5
DMA Sleep Mode Register (DMA0_SLP)	18-6
DMA Status Register (DMA0_SR)	18-7
DMA Channel Control Registers (DMA0_CR0–DMA0_CR3)	18-8
DMA Source Address Registers (DMA0_SA0–DMA0_SA3)	18-10
DMA Destination Address Registers (DMA0_DA0–DMA0_DA3)	18-11
DMA Count Registers (DMA0_CT0–DMA0_CT3)	18-11
DMA Scatter/Gather Descriptor Address Registers (DMA0_SG0–DMA0_SG3)	18-12
DMA Scatter/Gather Command Register (DMA0_SGC)	18-13
Channel Priorities	18-13
Data Parity During DMA Peripheral Transfers	18-14
Errors	18-14
Address Alignment Error	18-14
PLB Timeout	18-15
Slave Errors	18-15
DMA Interrupts	18-15
Scatter/Gather Transfers	18-16
Programming the DMA Controller	18-17
Peripheral Mode Transfers	18-17
Memory-to-Memory Transfers	18-20
Software-Initiated Memory-to-Memory Transfers (Non-Devised Paced)	18-21
Chapter 19. Ethernet Media Access Controller	19-1
EMAC Features	19-2
EMAC Operation	19-3
MAL Slave Logic	19-4
OPB Slave Logic	19-4
Ethernet Address Match Logic	19-4
Configuration and Status Registers	19-4
Wake On LAN Logic	19-4
Ethernet MAC	19-5
EMAC Loop-Back Modes	19-5
EMAC Transmit Operation	19-5
Arbitration Between TX Channels	19-6
Independent Mode	19-6
Dependent Mode	19-6
MAL TX Descriptor Control/Status Field	19-7
Early Packet Termination in Transmit	19-9
Empty Packets	19-9
Automatic Retransmission of Collided Packets	19-9
Inter-Packet Gap (IPG) Tuning	19-9
Full-Duplex Operation	19-9

Packet Content Configuration Options	19-10
EMAC Receive Operation	19-12
EMAC – MAL RX Packet Transfer Flow	19-12
MAL RX Descriptor Status	19-12
Early Packet Termination In Receive	19-13
Discarding Packets During Receive	19-14
WOL Support	19-14
EMAC WOL Support	19-15
Flow Control	19-15
MAC Control Packet	19-15
Control Packet Transmission	19-16
Integrated Flow Control	19-16
Control Packet Reception	19-17
VLAN Support	19-18
VLAN Tagged Packet Transmission	19-19
VLAN Tagged Packet Reception	19-19
Address Match Mechanism	19-20
Non-WOL Mode	19-20
WOL Mode	19-22
EMAC Registers	19-23
Mode Register 0 (EMAC0_MR0)	19-25
Mode Register 1 (EMAC0_MR1)	19-25
Transmit Mode Register 0 (EMAC0_TMR0)	19-27
Transmit Mode Register 1 (EMAC0_TMR1)	19-28
Low-Priority Requests	19-28
Urgent-Priority Requests	19-28
Receive Mode Register (EMAC0_RMR)	19-29
Interrupt Status Register (EMAC0_ISR)	19-30
Interrupt Status Enable Register (EMAC0_ISER)	19-33
Individual Address High (EMAC0_IAHR)	19-35
Individual Address Low (EMAC0_IALR)	19-36
VLAN TPID Register (EMAC0_VTPID)	19-36
VLAN TCI Register (EMAC0_VTCI)	19-36
Pause Timer Register (EMAC0_PTR)	19-37
Individual Address Hash Tables 1–4 (EMAC0_IAHT1–EMAC0_IAHT4)	19-37
Group Address Hash Tables 1–4 (EMAC0_GAHT1–EMAC0_GAHT4)	19-37
Last Source Address High (EMAC0_LSAH)	19-38
Last Source Address Low (EMAC0_LSAL)	19-38
Inter-Packet Gap Value Register (EMAC0_IPGVR)	19-38
STA Control Register (EMAC0_STACR)	19-39
Transmit Request Threshold Register (EMAC0_TRTR)	19-40
Receive Low/High Water Mark Register (EMAC0_RWMMR)	19-41
Number of Octets Transmitted (EMAC0_OCTX)	19-42
Number of Octets Received (EMAC0_OCRX)	19-42
MII Interface	19-42
MII Station Management Interface	19-42
EMAC – MII Interface	19-43
MAL – EMAC Packet Transfer Flow	19-43
Programming Notes	19-44
Power-Up and Initialization	19-44
Reset Options	19-44
Scenario 1	19-45
Scenario 2	19-45
Scenario 3	19-46

Chapter 20. Memory Access Layer	20-1
MAL Features	20-1
MAL – Internal Structure	20-3
PLB Master	20-3
OPB Master	20-3
TX Channel Handler	20-3
RX Channel Handler	20-3
TX Channel Arbiter	20-4
RX Channel Arbiter	20-4
TX Common Channel Logic	20-4
RX Common Channel Logic	20-4
Register Map File	20-4
Transmit and Receive Operations	20-4
Buffer Descriptor Overview	20-7
Transmit Software Interface	20-9
Wrapping the BD Table for Transmit	20-10
Continuous Mode for Transmit	20-10
Back Up a Packet for Transmit	20-10
Descriptor Not Valid for Transmit	20-11
Scroll Descriptors for Transmit	20-11
Receive Software Interface	20-12
Wrapping the BD Table for Receive	20-12
Continuous Mode for Receive	20-12
Descriptor Not Valid for Receive	20-13
Buffer Length for Receive	20-13
Descriptor Buffer Status/Control Fields	20-13
Information from a Software Device Driver Directed To MAL and COMMAC	20-13
Information from MAL and COMMAC Directed to Software	20-14
Status/Control Field Handling	20-14
Status/Control Field Format	20-14
TX Status/Control Field Format	20-15
Bit 0 – R – Ready	20-15
Bit 1 – W – Wrap	20-15
Bit 2 – CM – Continuous Mode	20-15
Bit 3 – L – Last	20-15
Bit 4 – Reserved	20-16
Bit 5 – I – Interrupt	20-16
Bits 6 to 15	20-16
RX Status/Control Field Format	20-16
Bit 0 – E – Empty	20-16
Bit 1 – W – Wrap	20-17
Bit 2 – CM – Continuous Mode	20-17
Bit 3 – L – Last	20-17
Bit 4 – F – First	20-17
Bit 5 – I – Interrupt	20-17
Bits 6 to 15	20-17
MAL Programming Notes	20-18
MAL Initialization	20-18
Interrupts	20-18
Error Handling	20-19
Error Detection	20-19
Indicated Errors	20-19
Error Handling Registers	20-20
Operational Error Modes	20-21
Resolution of an Error Situation	20-21
Interrupts To Software	20-23

MAL Registers	20-24
MAL Configuration Register (MALO_CFG)	20-25
Channel Active Set and Reset Registers	20-26
End of Buffer Interrupt Status Registers	20-28
Error Registers	20-29
MAL Error Status Register (MALO_ESR)	20-29
MAL Interrupt Enable Register (MALO_IER)	20-31
Descriptor Error Interrupt Registers (MALO_TXDEIR, MALO_RXDEIR)	20-32
Channel Table Pointer Registers (MALO_TXCTP0R, MALO_RXCTP0R)	20-33
Chapter 21. Serial Port Operations	21-1
Functional Description	21-1
Serial Input Clocking	21-2
UART Registers	21-4
Receiver Buffer Registers (UARTx_RBR)	21-5
Transmitter Holding Registers (UARTx_THR)	21-5
Interrupt Enable Registers (UARTx_IER)	21-5
Interrupt Identification Registers (UARTx_IIR)	21-6
FIFO Control Registers (UARTx_FCR)	21-7
Line Control Registers (UARTx_LCR)	21-8
Modem Control Registers (UARTx_MCR)	21-10
Line Status Registers (UARTx_LSR)	21-11
Modem Status Registers (UARTx_MSR)	21-13
Scratchpad Registers (UARTx_SCR)	21-13
Divisor Latch LSB and MSB Registers (UARTx_DLL, UARTx_DLM)	21-14
FIFO Operation	21-15
Interrupt Mode	21-15
Receiver	21-15
Transmitter	21-16
Polled Mode	21-16
UART and Sleep Mode	21-16
DMA Operation	21-17
Chip Control Register 0 (CPC0_CR0)	21-17
Transmitter DMA Mode	21-18
Receiver DMA Mode	21-20
Chapter 22. IIC Bus Interface	22-1
Addressing	22-1
Addressing Modes	22-1
Seven-Bit Addresses	22-2
Ten-Bit Addresses	22-2
IIC Registers	22-2
IIC Register Descriptions	22-3
IIC0 Master Data Buffer	22-3
IIC0 Slave Data Buffer	22-4
IIC0 Low Master Address Register	22-5
IIC0 High Master Address Register	22-6
IIC0 Control Register	22-6
IIC0 Mode Control Register	22-8
IIC0 Status Register	22-10
IIC0 Extended Status Register	22-11
IIC0 Low Slave Address Register	22-14
IIC0 High Slave Address Register	22-14
IIC0 Clock Divide Register	22-15
IIC0 Interrupt Mask Register	22-16
IIC0 Transfer Count Register	22-17
IIC0 Extended Control and Slave Status Register	22-18

IIC0 Direct Control Register	22-20
Interrupt Handling	22-21
General Considerations	22-22
Chapter 23. GPIO Operations	23-1
GPIO Controller Overview	23-1
Features	23-1
GPIO Interface Signals	23-2
External Macro Signals	23-2
External Module Signals	23-3
Clock and Power Management	23-4
GPIO Register Overview	23-5
GPIO Register Reset Values	23-5
Detailed Register Description	23-5
GPIO Output Register (GPIO0_OR)	23-5
GPIO Three-State Control Register (GPIO0_TCR)	23-5
GPIO Open Drain Register (GPIO0_ODR)	23-6
GPIO Input Register (GPIO0_IR)	23-6
Part V. Reference	V-1
Chapter 24. Instruction Set	24-1
Instruction Set Portability	24-1
Instruction Formats	24-2
Pseudocode	24-2
Operator Precedence	24-5
Register Usage	24-5
Alphabetical Instruction Listing	24-5
add	24-6
addc	24-7
adde	24-8
addi	24-9
addic	24-10
addic.	24-11
addis	24-12
addme	24-13
addze	24-14
and	24-15
andc	24-16
andi	24-17
andis.	24-18
b	24-19
bc	24-20
bcctr	24-26
bclr	24-30
cmp	24-34
cmpl	24-35
cmpl.	24-36
cmpli	24-37
cntlzw	24-38
crand	24-39
crandc	24-40
creqv	24-41
crnand	24-42
crnor	24-43
cror	24-44

crorc	24-45
crxor	24-46
dcba	24-47
dcbf	24-49
dcbi	24-50
dcbst	24-51
dcbt	24-52
dcbtst	24-53
dcbz	24-54
dccci	24-56
dcread	24-57
divw	24-59
divwu	24-60
eieio	24-61
eqv	24-62
extsb	24-63
extsh	24-64
icbi	24-65
icbt	24-66
iccci	24-67
icread	24-68
isync	24-70
lbz	24-71
lbzu	24-72
lbzux	24-73
lbzx	24-74
lha	24-75
lhau	24-76
lhaux	24-77
lhax	24-78
lhrx	24-79
lhz	24-80
lhzu	24-81
lhzux	24-82
lhzx	24-83
lmw	24-84
lswi	24-85
lswx	24-87
lwarx	24-89
lwbrx	24-90
lwz	24-91
lwzu	24-92
lwzux	24-93
lwzx	24-94
macchw	24-95
macchws	24-96
macchwsu	24-97
macchwu	24-98
machhw	24-99
machhws	24-100
machhwsu	24-101
machhwu	24-102
maclhw	24-103
maclhws	24-104

maclhwsu	24-105
maclhwu	24-106
mcrf	24-107
mcrxr	24-108
mfcf	24-109
mfdcr	24-110
mfmsr	24-111
mfspr	24-112
mftb	24-114
mtcrf	24-116
mtdcr	24-117
mtmsr	24-118
mtspr	24-119
mulchw	24-121
mulchwu	24-122
mulhhw	24-123
mulhhwu	24-124
mulhw	24-125
mulhwu	24-126
mullhw	24-127
mullhwu	24-128
mulli	24-129
mullw	24-130
nand	24-131
neg	24-132
nmacchw	24-133
nmacchws	24-134
nmachhw	24-135
nmachhws	24-136
nmachlw	24-137
nmacchws	24-138
nor	24-139
or	24-140
orc	24-141
ori	24-142
oris	24-143
rfci	24-144
rfi	24-145
rlwimi	24-146
rlwinm	24-147
rlwnm	24-150
sc	24-151
slw	24-152
sraw	24-153
srawi	24-154
srw	24-155
stb	24-156
stbu	24-157
stbux	24-158
stbx	24-159
sth	24-160
sthbrx	24-161
sthu	24-162
sthux	24-163

sthx	24-164
stmw	24-165
stswi	24-166
stswx	24-167
stw	24-169
stwbrx	24-170
stwcx	24-171
stwu	24-173
stwux	24-174
stwx	24-175
subf	24-176
subfc	24-177
subfe	24-178
subfic	24-179
subfme	24-180
subfze	24-181
sync	24-182
tlbia	24-183
tlbre	24-184
tlbsx	24-186
tlbsync	24-187
tlbwe	24-188
tw	24-190
twi	24-193
wrtw	24-196
wrtwi	24-197
xor	24-198
xori	24-199
xoris	24-200

Chapter 25. Register Summary 25-1

Reserved Registers	25-1
Reserved Fields	25-1
General Purpose Registers	25-1
Machine State Register and Condition Register	25-1
Special Purpose Registers	25-1
Time Base Registers	25-3
Device Control Registers	25-4
Directly Accessed DCRs	25-4
Indirectly Accessed DCRs	25-6
Indirect Access of SDRAM Controller DCRs	25-7
Indirect Access of EBC DCRs	25-8
Indirect Access of Decompression Controller DCRs	25-9
MMIO Registers	25-10
Directly Accessed MMIO Registers	25-10
Indirectly Accessed MMIO Registers	25-13
Alphabetical Register Listing	25-14
CCR0	25-15
CPC0_CR0	25-17
CPC0_CR1	25-20
CPC0_ER	25-21
CPC0_FR	25-22
CPC0_JTAGID	25-23
CPC0_PLLMR	25-24
CPC0_PSR	25-26

CPC0_SR	25-28
CR	25-29
CTR	25-30
DAC1–DAC2	25-31
DBCR0	25-32
DBCR1	25-34
DBSR	25-36
DCCR	25-38
DCP0_ADDR0–DCP0_ADDR1	25-40
DCP0_CFG	25-41
DCP0_CFGADDR	25-42
DCP0_CFGDATA	25-43
DCP0_ESR	25-44
DCP0_ID	25-46
DCP0_ITOR0–DCP0_ITOR3	25-47
DCP0_MEMBEAR	25-48
DCP0_PLBBEAR	25-49
DCP0_RAM0–DCP0_RAM3FF	25-50
DCP0_VER	25-51
DCWR	25-52
DEAR	25-54
DMA0_CR0–DMA0_CR3	25-55
DMA0_CT0–DMA0_CT3	25-58
DMA0_DA0–DMA0_DA3	25-59
DMA0_POL	25-60
DMA0_SA0–DMA0_SA3	25-62
DMA0_SG0–DMA0_SG3	25-63
DMA0_SGC	25-64
DMA0_SLP	25-65
DMA0_SR	25-66
DVCR1–DVCR2	25-67
EBC0_BEAR	25-68
EBC0_BESR0	25-69
EBC0_BESR1	25-71
EBC0_BnAP	25-73
EBC0_BnCR	25-75
EBC0_CFG	25-76
EBC0_CFGADDR	25-78
EBC0_CFGDATA	25-79
EMAC0_GAHT1–EMAC0_GAHT4	25-80
EMAC0_IAHR	25-81
EMAC0_IAHT1–EMAC0_IAHT4	25-82
EMAC0_TMR0	25-83
EMAC0_TMR1	25-84
EMAC0_TRTR	25-85
EMAC0_VTCI	25-86
EMAC0_VTPID	25-87
ESR	25-88
EVPR	25-89
GPIO0_IR	25-90
GPIO0_ODR	25-91
GPIO0_OR	25-92
GPIO0_TCR	25-93
GPR0–GPR31	25-94

IAC1-IAC4	25-95
ICCR	25-96
ICDBDR	25-98
IIC0_CLKDIV	25-99
IIC0_CNTL	25-100
IIC0_DIRECTCNTL	25-101
IIC0_EXTSTS	25-102
IIC0_HMADR	25-104
IIC0_HSADR	25-105
IIC0_INTRMSK	25-106
IIC0_LMADR	25-107
IIC0_LSADR	25-108
IIC0_MDBUF	25-109
IIC0_MDCNTL	25-110
IIC0_SDBUF	25-111
IIC0_STS	25-112
IIC0_XFRCNT	25-113
IIC0_XTCNTLSS	25-114
LR	25-116
MALO_CFG	25-117
MALO_ESR	25-119
MALO_IER	25-121
MALO_RCBS0	25-122
MALO_RXCARR	25-123
MALO_RXCASR	25-124
MALO_RXCTPXR	25-125
MALO_RXDEIR	25-126
MALO_RXEOBISR	25-127
MALO_TXCARR	25-128
MALO_TXCASR	25-129
MALO_TXCTPXR	25-130
MALO_TXDEIR	25-131
MALO_TXEOBISR	25-132
MSR	25-133
OCM0_DSARC	25-135
OCM0_DSCNTL	25-136
OCM0_ISARC	25-137
OCM0_ISCNTL	25-138
OPBA0_CR	25-139
OPBA0_PR	25-140
PCI_BAR0PCI_BAR0	25-141
PCIC0_BIST	25-142
PCIC0_BRDGOPT1	25-143
PCIC0_BRDGOPT2	25-144
PCIC0_CACHELS	25-145
PCIC0_CAP	25-146
PCIC0_CAPID	25-147
PCIC0_CFGADDR	25-148
PCIC0_CFGATA	25-149
PCIC0_CLS	25-150
PCIC0_CMD	25-151
PCIC0_DATA	25-153
PCIC0_DEVID	25-154
PCIC0_ERREN	25-155

PCIC0_ERRSTS	25-156
PCIC0_HDTYPE	25-157
PCIC0_ICS	25-158
PCIC0_INTLN	25-159
PCIC0_INTPN	25-160
PCIC0_LATTIM	25-161
PCIC0_MAXLTNCY	25-162
PCIC0_MINGNT	25-163
PCIC0_NEXTIPTR	25-164
PCIC0_PLBBEAR	25-165
PCIC0_PLBBESR0	25-166
PCIC0_PLBBESR1	25-168
PCIC0_PMC	25-170
PCIC0_PMCSR	25-171
PCIC0_PMCSRBSE	25-172
PCIC0_PMSCRR	25-173
PCIC0_PTM1BAR	25-174
PCIC0_PTM2BAR	25-175
PCIC0_REVID	25-176
PCIC0_SBSYSID	25-177
PCIC0_SBSYSVID	25-178
PCIC0_STATUS	25-179
PCIC0_VENDID	25-181
PCI0_PMM0LA	25-182
PCI0_PMM0MA	25-183
PCI0_PMM0PCIHA	25-184
PCI0_PMM0PCILA	25-185
PCI0_PMM1LA	25-186
PCI0_PMM1MA	25-187
PCI0_PMM1PCIHA	25-188
PCI0_PMM1PCILA	25-189
PCI0_PMM2LA	25-190
PCI0_PMM2MA	25-191
PCI0_PMM2PCIHA	25-192
PCI0_PMM2PCILA	25-193
PCI0_PTM1LA	25-194
PCI0_PTM1MS	25-195
PCI0_PTM2LA	25-196
PCI0_PTM2MS	25-197
PID	25-198
PIT	25-199
PLB0_ACR	25-200
PLB0_BEAR	25-201
PLB0_BESR	25-202
POB0_BEAR	25-204
POB0_BESR0	25-205
POB0_BESR1	25-207
PVR	25-208
SDRAM0_B0CR--SDRAM0_B3CR	25-209
SDRAM0_BEAR	25-210
SDRAM0_BESR0	25-211
SDRAM0_BESR1	25-213
SDRAM0_CFG	25-214
SDRAM0_CFGADDR	25-216

SDRAM0_CFGDATA	25-217
SDRAM0_ECCCFG	25-218
SDRAM0_ECCESR	25-219
SDRAM0_PMIT	25-220
SDRAM0_RTR	25-221
SDRAM0_TR	25-222
SGR	25-224
SLER	25-226
SPRG0-SPRG7	25-228
SRR0	25-229
SRR1	25-230
SRR2	25-231
SRR3	25-232
SU0R	25-233
TBL	25-235
TBU	25-236
TCR	25-237
TSR	25-238
UARTx_DLL	25-239
UARTx_DLM	25-240
UARTx_FCR	25-241
UARTx_IER	25-242
UARTx_IIR	25-243
UARTx_LCR	25-244
UARTx_LSR	25-245
UARTx_MCR	25-247
UARTx_MSR	25-248
UARTx_RBR	25-249
UARTx_SCR	25-250
UARTx_THR	25-251
UIC0_CR	25-252
UIC0_ER	25-255
UIC0_MSR	25-258
UIC0_PR	25-261
UIC0_SR	25-264
UIC0_TR	25-267
UIC0_VCR	25-270
UIC0_VR	25-271
USPRG0	25-272
XER	25-273
ZPR	25-274
Chapter 26. Signal Summary	26-1
Signals Listed Alphabetically	26-1
Signal Descriptions	26-5
A. Instruction Summary	A-1
Instruction Set and Extended Mnemonics – Alphabetical	A-1
Instructions Sorted by Opcode	A-33
Instruction Formats	A-41
Instruction Fields	A-41
Instruction Format Diagrams	A-43
I-Form	A-44
B-Form	A-44
SC-Form	A-44

D-Form	A-44
X-Form	A-45
XL-Form	A-45
XFX-Form	A-46
X0-Form	A-46
M-Form	A-46
B. Instructions by Category	B-1
Implementation-Specific Instructions	B-1
Instructions in the IBM PowerPC Embedded Environment	B-5
Privileged Instructions	B-7
Assembler Extended Mnemonics	B-9
Storage Reference Instructions	B-29
Arithmetic and Logical Instructions	B-33
Condition Register Logical Instructions	B-37
Branch Instructions	B-38
Comparison Instructions	B-39
Rotate and Shift Instructions	B-40
Cache Control Instructions	B-41
Interrupt Control Instructions	B-42
Processor Management Instructions	B-42
C. Code Optimization and Instruction Timings	C-1
Code Optimization Guidelines	C-1
Condition Register Bits for Boolean Variables	C-1
CR Logical Instruction for Compound Branches	C-1
Floating-Point Emulation	C-1
Cache Usage	C-2
CR Dependencies	C-2
Branch Prediction	C-2
Alignment	C-2
Instruction Timings	C-3
General Rules	C-3
Branches	C-3
Multiplies	C-4
Scalar Load Instructions	C-5
Scalar Store Instructions	C-5
Alignment in Scalar Load and Store Instructions	C-5
String and Multiple Instructions	C-6
Loads and Store Misses	C-6
Instruction Cache Misses	C-7
Index	X-1

Figures

Figure 1-1. PPC405GP Block Diagram	1-2
Figure 2-1. Overlapped PLB Transfers	2-4
Figure 2-2. PLB Arbiter Control Register (PLB0_ACR)	2-5
Figure 2-3. PLB Error Address Register (PLB0_BEAR)	2-6
Figure 2-4. PLB Error Status Register (PLB0_BESR)	2-6
Figure 2-5. Bridge Error Address Register (POB0_BEAR)	2-8
Figure 2-6. Bridge Error Status Register 0 (POB0_BESR0)	2-9
Figure 2-7. Bridge Error Status Register 1 (POB0_BESR1)	2-10
Figure 2-8. OPB Arbiter Control Register (OPBA0_CR)	2-12
Figure 2-9. OPB Arbiter Priority Register (OPBA0_PR)	2-13
Figure 3-1. PPC405GP Programming Model—Registers	3-5
Figure 3-2. General Purpose Registers (R0-R31)	3-6
Figure 3-3. Count Register (CTR)	3-8
Figure 3-4. Link Register (LR)	3-8
Figure 3-5. Fixed Point Exception Register (XER)	3-10
Figure 3-6. Special Purpose Register General (SPRG0–SPRG7)	3-11
Figure 3-7. Processor Version Register (PVR)	3-12
Figure 3-8. Condition Register (CR)	3-13
Figure 3-9. Machine State Register (MSR)	3-15
Figure 3-10. PPC405GP Data Types	3-26
Figure 3-11. Normal Word Load or Store (Big Endian Storage Region)	3-32
Figure 3-12. Byte-Reverse Word Load or Store (Little Endian Storage Region)	3-32
Figure 3-13. Byte-Reverse Word Load or Store (Big Endian Storage Region)	3-33
Figure 3-14. Normal Word Load or Store (Little Endian Storage Region)	3-33
Figure 3-15. PPC405GP Instruction Pipeline	3-34
Figure 4-1. Instruction Flow	4-3
Figure 4-2. Core Configuration Register 0 (CCR0)	4-11
Figure 4-3. Instruction Cache Debug Data Register (ICDBDR)	4-14
Figure 5-1. OCM Address Usage	5-2
Figure 5-2. OCM Instruction-Side Address Range Compare Register (OCM0_ISARC)	5-5
Figure 5-3. OCM Instruction-Side Control Register (OCM0_ISCNTL)	5-6
Figure 5-4. OCM Data-Side Address Range Compare Register (OCM0_DSARC)	5-7
Figure 5-5. OCM Data-Side Control Register (OCM0_DSCNTL)	5-7
Figure 6-1. Effective to Real Address Translation Flow	6-2
Figure 6-2. TLB Entries	6-3
Figure 6-3. ITLB/DTLB/UTLB Address Resolution	6-8
Figure 6-4. Process ID (PID)	6-13
Figure 6-5. Zone Protection Register (ZPR)	6-14
Figure 6-6. Generic Storage Attribute Control Register	6-18
Figure 7-1. PPC405GP Clocking	7-1
Figure 7-2. PLL Mode Register (CPC0_PLLMR)	7-10
Figure 7-3. Chip Control Register 0 (CPC0_CR0)	7-12

Figure 9-1. Chip Pin Strapping Register (CPC0_PSR)	9-1
Figure 10-1. UIC Status Register (UIC0_SR)	10-3
Figure 10-2. UIC Enable Register (UIC0_ER)	10-6
Figure 10-3. UIC Critical Register (UIC0_CR)	10-8
Figure 10-4. UIC Polarity Register (UIC0_PR)	10-11
Figure 10-5. UIC Trigger Register (UIC0_TR)	10-13
Figure 10-6. UIC Masked Status Register (UIC0_MSR)	10-16
Figure 10-7. UIC Vector Configuration Register (UIC0_VCR)	10-19
Figure 10-8. UIC Vector Register (UIC0_VR)	10-20
Figure 10-9. Machine State Register (MSR)	10-28
Figure 10-10. Save/Restore Register 0 (SRR0)	10-29
Figure 10-11. Save/Restore Register 1 (SRR1)	10-30
Figure 10-12. Save/Restore Register 2 (SRR2)	10-30
Figure 10-13. Save/Restore Register 3 (SRR3)	10-31
Figure 10-14. Exception Vector Prefix Register (EVPR)	10-31
Figure 10-15. Exception Syndrome Register (ESR)	10-32
Figure 10-16. Data Exception Address Register (DEAR)	10-34
Figure 11-1. Relationship of Timer Facilities to the Time Base	11-1
Figure 11-2. Time Base Lower (TBL)	11-2
Figure 11-3. Time Base Upper (TBU)	11-3
Figure 11-4. Programmable Interval Timer (PIT)	11-5
Figure 11-5. Watchdog Timer State Machine	11-7
Figure 11-6. Timer Status Register (TSR)	11-8
Figure 11-7. Timer Control Register (TCR)	11-9
Figure 12-1. JTAG Connector Physical Layout (Top View)	12-2
Figure 12-2. JTAG ID Register (CPC0_JTAGID)	12-4
Figure 12-3. RISCTrace Header (Top View)	12-5
Figure 12-4. Debug Control Register 0 (DBCR0)	12-9
Figure 12-5. Debug Control Register 1 (DBCR1)	12-11
Figure 12-6. Debug Status Register (DBSR)	12-13
Figure 12-7. Instruction Address Compare Registers (IAC1–IAC4)	12-14
Figure 12-8. Data Address Compare Registers (DAC1–DAC2)	12-15
Figure 12-9. Data Value Compare Registers (DVC1–DVC2)	12-15
Figure 12-10. Inclusive IAC Range Address Compares	12-18
Figure 12-11. Exclusive IAC Range Address Compares	12-18
Figure 12-12. Inclusive DAC Range Address Compares	12-20
Figure 12-13. Exclusive DAC Range Address Compares	12-20
Figure 13-1. CPM Registers (CPC0_ER, CPC0_FR, CPC0_SR)	13-2
Figure 14-1. Decompression Index Table Origin Registers (DCP0_ITOR0–DCP0_ITOR3)	14-4
Figure 14-2. Decompression Address Decode Definition Registers (DCP0_ADDR0–DCP0_ADDR1)	14-5
Figure 14-3. Decompression Controller Configuration Register (DCP0_CFG)	14-6
Figure 14-4. Decompression Controller ID Register (DCP0_ID)	14-6
Figure 14-5. Decompression Controller Version Register (DCP0_VER)	14-7
Figure 14-6. Decompression Controller PLB Error Address Register (DCP0_PLBBEAR)	14-7
Figure 14-7. Decompression Controller Bus Error Address Register (DCP0_MEMBEAR)	14-7
Figure 14-8. Decompression Controller Error Status Register 0 (DCP0_ESR)	14-8

Figure 15-1. SDRAM Controller Signals	15-1
Figure 15-2. Memory Controller Configuration (SDRAM0_CFG)	15-4
Figure 15-3. Memory Controller Status (SDRAM0_STATUS)	15-5
Figure 15-4. Memory Bank 0–3 Configuration Registers (SDRAM0_B0CR–SDRAM0_B3CR)	15-6
Figure 15-5. SDRAM Timing Register (SDRAM0_TR)	15-9
Figure 15-6. Activate, Four Word Read, Precharge, Activate	15-11
Figure 15-7. Activate, Four Word Write, Precharge, Activate	15-11
Figure 15-8. Precharge All, Activate	15-12
Figure 15-9. CAS Before RAS Refresh	15-12
Figure 15-11. Refresh Timing Register (SDRAM0_RTR)	15-14
Figure 15-12. ECC Configuration Register (SDRAM0_ECCCFG)	15-15
Figure 15-13. ECC Error Status Register (SDRAM0_ECCESR)	15-16
Figure 15-14. Bus Error Address Register (SDRAM0_BEAR)	15-17
Figure 15-15. Bus Error Syndrome Register 0 (SDRAM0_BESR0)	15-17
Figure 15-16. Bus Error Status Register 1 (SDRAM0_BESR1)	15-19
Figure 15-17. Power Management Idle Timer (SDRAM0_PMIT)	15-20
Figure 16-1. External Bus Controller Signals	16-2
Figure 16-2. Attachment of Devices of Various Widths to the Peripheral Data Bus	16-4
Figure 16-3. Single Read Transfer	16-6
Figure 16-4. Single Write Transfer	16-7
Figure 16-5. Burst Read Transfer	16-9
Figure 16-6. Burst Write Transfer	16-10
Figure 16-7. Device-Paced Single Read Transfer	16-12
Figure 16-8. Device-Paced Single Write Transfer	16-13
Figure 16-9. Device-Paced Burst Read Transfer	16-14
Figure 16-10. Device-Paced Burst Write Transfer	16-16
Figure 16-11. Sample External Bus Master System	16-17
Figure 16-12. External Master Arbitration, Single Read and Single Write	16-20
Figure 16-13. External Master Burst Read	16-21
Figure 16-14. External Master Burst Write	16-22
Figure 16-15. EBC Configuration Register (EBC0_CFG)	16-24
Figure 16-16. Peripheral Bank Configuration Registers (EBC0_BnCR)	16-25
Figure 16-17. Peripheral Bank Access Parameters (EBC0_BnAP)	16-26
Figure 16-18. Peripheral Bus Error Address Register (EBC0_BEAR)	16-30
Figure 16-19. Peripheral Bus Error Status Register 0 (EBC0_BESR0)	16-30
Figure 16-20. Peripheral Bus Error Status Register 1 (EBC0_BESR1)	16-32
Figure 17-1. PCI Bridge Block Diagram	17-2
Figure 17-2. PLB-to-PCI Half-Bridge Block Diagram	17-4
Figure 17-3. PCI-to-PLB Half-Bridge Block Diagram	17-5
Figure 17-4. Arbitration Structure	17-5
Figure 17-5. PMM Register Sets Map PLB Address Space to PCI Address Space	17-8
Figure 17-6. PTM Register Sets Map PCI Address Space to PLB Address Space	17-9
Figure 17-7. PMM 0 Local Address Register (PCILO_PMM0LA)	17-21
Figure 17-8. PMM 0 Mask/Attribute Register (PCILO_PMM0MA)	17-22
Figure 17-9. PMM 0 PCI Low Address Register (PCILO_PMM0PCILA)	17-23
Figure 17-10. PMM 0 High Address Register (PCILO_PMM0PCIHA)	17-23

Figure 17-11. PMM 1 Local Address Register (PCIL0_PMM1LA)	17-23
Figure 17-12. PMM 1 Mask/Attribute Register (PCIL0_PMM1MA)	17-24
Figure 17-13. PMM 1 PCI Low Address Register (PCIL0_PMM1PCILA)	17-24
Figure 17-14. PMM 0 High Address Register (PCIL0_PMM0PCIHA)	17-25
Figure 17-15. PMM 2 Local Address Register (PCIL0_PMM2LA)	17-25
Figure 17-16. PMM 2 Mask/Attribute Register (PCIL0_PMM2MA)	17-26
Figure 17-17. PMM 2 Low Address Register (PCIL0_PMM2PCILA)	17-26
Figure 17-18. PMM 2 PCI High Address Register (PCIL0_PMM2PCIHA)	17-27
Figure 17-19. PTM 1 Memory Size/Attribute Register (PCIL0_PTM1MS)	17-27
Figure 17-20. PTM 2 Local Address Register (PCIL0_PTM1LA)	17-28
Figure 17-21. PTM 2 Memory Size/Attribute Register (PCIL0_PTM2MS)	17-28
Figure 17-22. PTM 2 Local Address Register (PCIL0_PTM2LA)	17-29
Figure 17-23. PCI Configuration Address Register (PCIC0_CFGADDR)	17-30
Figure 17-24. PCI Configuration Data Register (PCIC0_CFGDATA)	17-30
Figure 17-25. PCI Vendor ID Register (PCIC0_VENDID)	17-31
Figure 17-26. PCI Device ID Register (PCIC0_DEVID)	17-31
Figure 17-27. PCI Command Register (PCIC0_CMD)	17-32
Figure 17-28. PCI Status Register (PCIC0_STATUS)	17-33
Figure 17-29. PCI Revision ID Register (PCIC0_REVID)	17-35
Figure 17-30. PCI Class Register (PCIC0_PCICLS)	17-35
Figure 17-31. PCI Cache Line Size Register (PCIC0_CACHELS)	17-36
Figure 17-32. PCI Latency Timer Register (PCIC0_LATTIM)	17-36
Figure 17-33. PCI Header Type Register (PCIC0_HDTYPE)	17-37
Figure 17-34. PCI Built-in Self Test Control Register (PCIC0_BIST)	17-37
Figure 17-35. PCI PTM 1 BAR Register (PCIC0_PTM1BAR)	17-38
Figure 17-36. PCI PTM 2 BAR Register (PCIC0_PTM2BAR)	17-39
Figure 17-37. PCI Subsystem Vendor ID Register (PCIC0_SBSYSVID)	17-39
Figure 17-38. PCI Subsystem ID Register (PCIC0_SBSYSID)	17-40
Figure 17-39. PCI Capabilities Pointer (PCIC0_CAP)	17-40
Figure 17-40. PCI Interrupt Line Register (PCIC0_INTLN)	17-40
Figure 17-41. PCI Interrupt Pin Register (PCIC0_INTPN)	17-41
Figure 17-42. PCI Minimum Grant Register (PCIC0_MINGNT)	17-41
Figure 17-43. PCI Maximum Latency Register (PCIC0_MAXLTNCY)	17-41
Figure 17-44. PCI Interrupt Control/Status Register	17-42
Figure 17-45. Error Enable Register (PCIC0_ERREN)	17-42
Figure 17-46. Error Status Register (PCIC0_ERRSTS)	17-43
Figure 17-47. Bridge Options 1 Register (PCIC0_BRDGOPT1)	17-44
Figure 17-48. PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0)	17-46
Figure 17-49. PLB Slave Error Syndrome 1 (PCIC0_PLBBESR1)	17-47
Figure 17-50. PLB Slave Error Address Register (PCIC0_PLBBEAR)	17-49
Figure 17-51. Capability Identifier (PCIC0_CAPID)	17-49
Figure 17-52. Next Item Pointer (PCIC0_NEXTIPTR)	17-49
Figure 17-53. Power Management Capabilities Register (PCIC0_PMC)	17-50
Figure 17-54. Power Management Control/Status Register (PCIC0_PMCSR)	17-51
Figure 17-55. PMCSR PCI to PCI Bridge Support Extensions (PCIC0_PMCSRBASE)	17-51
Figure 17-56. PCI Data (PCIC0_DATA)	17-52

Figure 17-57. Bridge Options 2 Register (PCIC0_BRDGOPT2)	17-52
Figure 17-58. Power Management State Change Request Register (PCIC0_PMSCRR)	17-54
Figure 17-60. PCI Master Burst Read From SDRAM	17-68
Figure 17-61. PCI Master Burst Write To SDRAM	17-72
Figure 17-62. CPU Read From PCI Memory Slave, Nonprefetching	17-76
Figure 17-63. CPU Read From PCI Memory Slave, Prefetching	17-78
Figure 17-64. CPU Write To PCI Memory Slave	17-82
Figure 17-65. PCI Memory To SDRAM DMA Transfer	17-86
Figure 17-66. SDRAM To PCI Memory DMA Transfer	17-88
Figure 17-67. PCI Master Burst Read From SDRAM	17-92
Figure 17-68. PCI Master Burst Write To SDRAM	17-100
Figure 17-69. CPU Read From PCI Memory Slave, Nonprefetching	17-106
Figure 17-70. CPU Read From PCI Memory Slave, Prefetching	17-108
Figure 17-71. CPU Write To PCI Memory Slave	17-112
Figure 17-72. PCI Memory To SDRAM DMA Transfer	17-116
Figure 17-73. SDRAM To PCI Memory DMA Transfer	17-120
Figure 18-1. DMA Controller External Bus Control Signals	18-2
Figure 18-2. DMA Polarity Configuration Register (DMA0_POL)	18-5
Figure 18-3. DMA Sleep Mode Register (DMA0_SLP)	18-7
Figure 18-4. DMA Status Register (DMA0_SR)	18-7
Figure 18-5. DMA Channel Control Registers (DMA0_CR0-DMA0_CR3)	18-8
Figure 18-6. DMA Source Address Registers (DMA0_SA0-DMA0_SA3)	18-11
Figure 18-7. DMA Destination Address Registers (DMA0_DA0-DMA0_DA3)	18-11
Figure 18-8. DMA Count Registers (DMA0_CT0-DMA0_CT3)	18-12
Figure 18-9. DMA Scatter/Gather Descriptor Address Registers (DMA0_SG0-DMA0_SG3)	18-12
Figure 18-10. DMA Scatter/Gather Command Register (DMA0_SGC)	18-13
Figure 18-11. Peripheral-to-Memory DMA Transfer	18-18
Figure 18-12. Memory to Peripheral DMA Transfer	18-19
Figure 19-1. EMAC in a Typical Ethernet Application	19-2
Figure 19-2. Internal EMAC Structure	19-3
Figure 19-3. EMAC Loop-Back Modes	19-5
Figure 19-4. MAL TX Descriptor Control/Status Field	19-7
Figure 19-5. Transmit Packet Structure (Excluding VLAN Tagged and Control Packets)	19-10
Figure 19-6. MAL RX Descriptor Control/Status Field	19-12
Figure 19-7. Wake-Up Packet Format	19-14
Figure 19-8. Control Packet Format	19-16
Figure 19-9. Integrated Flow Control Mechanism	19-17
Figure 19-10. Pause Operation State Machine	19-18
Figure 19-11. Tagged MAC Packet Format	19-19
Figure 19-12. Tag Control Information Field Structure	19-19
Figure 19-13. Receive Address Recognition Flowchart	19-22
Figure 19-14. Ethernet Address Filter Operation	19-23
Figure 19-15. Mode Register 0 (EMAC0_MR0)	19-25
Figure 19-16. Mode Register 1 (EMAC0_MR1)	19-26
Figure 19-17. Transmit Mode Register 0 (EMAC0_TMR0)	19-27
Figure 19-18. Transmit Mode Register 1 (EMAC0_TMR1)	19-28

Figure 19-19. Receive Mode Register (EMAC0_RMR)	19-29
Figure 19-20. Interrupt Status Register (EMAC0_ISR)	19-31
Figure 19-21. Interrupt Status Register (EMAC0_ISER)	19-33
Figure 19-22. Individual Address High Register (EMAC0_IAHR)	19-35
Figure 19-23. Individual Address Low Register (EMAC0_IALR)	19-36
Figure 19-24. VLAN TPID Register (EMAC0_VTPID)	19-36
Figure 19-25. VLAN TCI Register (EMAC0_VTCI)	19-37
Figure 19-26. Pause Timer Register (EMAC0_PTR)	19-37
Figure 19-27. Individual Address Hash Tables 1–4 (EMAC0_IAHT1–EMAC0_IAHT4)	19-37
Figure 19-28. Group Address Hash Tables 1–4 (EMAC0_GAHT1–EMAC0_GAHT4)	19-38
Figure 19-29. Last Source Address High Register (EMAC0_LSAH)	19-38
Figure 19-30. Last Source Address Low Register (EMAC0_LSAH)	19-38
Figure 19-31. inter-Packet Gap Value Register (EMAC0_IPGVR)	19-39
Figure 19-32. STA Control Register (EMAC0_STACR)	19-39
Figure 19-33. Transmit Request Threshold Register (EMAC0_TRTR)	19-41
Figure 19-34. Receive Low/High Water Mark Register (EMAC0_RWMMR)	19-42
Figure 19-35. Number of Octets Transmitted (EMAC0_OCTX)	19-42
Figure 19-36. Number of Octets Received (EMAC0_OCRX)	19-42
Figure 19-37. Management Interface with PHY	19-43
Figure 19-38. EMAC-MAL Communication Phases	19-44
Figure 20-1. General PPC405GP Structure (Overview)	20-2
Figure 20-2. MAL Internal Structure	20-3
Figure 20-3. Transmit Operation	20-5
Figure 20-4. Receive Operation	20-6
Figure 20-5. Buffer Descriptor Structure	20-8
Figure 20-6. Packet Memory Structure	20-9
Figure 20-7. TX Status/Control Field	20-15
Figure 20-8. RX Status/Control Field	20-16
Figure 20-9. Error Status Register Field	20-21
Figure 20-10. MAL Error Processing	20-23
Figure 20-11. MAL Configuration Register (MAL0_CFG)	20-25
Figure 20-12. TX Channel_Active Set Register (MAL0_TXCASR)	20-27
Figure 20-13. TX Channel_Active Reset Register (MAL0_TXCARR)	20-27
Figure 20-14. RX Channel_Active Set Register (MAL0_RXCASR)	20-27
Figure 20-15. RX Channel_Active Reset Register (MAL0_RXCARR)	20-28
Figure 20-16. TX End of Buffer Interrupt Status Register (MAL0_TXEOBISR)	20-28
Figure 20-17. RX End of Buffer Interrupt Status Register (MAL0_RXEOBISR)	20-29
Figure 20-18. MAL Error Status Register (MAL0_ESR)	20-30
Figure 20-19. MAL Interrupt Enable Register (MAL0_IER)	20-31
Figure 20-20. TX Descriptor Error Interrupt Register (MAL0_TXDEIR)	20-32
Figure 20-21. RX Descriptor Error Interrupt Register (MAL0_RXDEIR)	20-32
Figure 20-22. TX Channel Table Pointer x Register (MAL0_TXCTPxR)	20-33
Figure 20-23. RX Channel Table Pointer x Register (MAL0_RXCTPxR)	20-33
Figure 21-1. UART Receiver Buffer Registers (UARTx_RBR)	21-5
Figure 21-2. UART Transmitter Holding Registers (UARTx_THR)	21-5
Figure 21-3. UART Interrupt Enable Registers (UARTx_IER)	21-5

Figure 21-4. UART Interrupt Identification Registers (UARTx_IIR)	21-7
Figure 21-5. UART FIFO Control Registers (UARTx_FCR)	21-8
Figure 21-6. UART Line Control Registers (UARTx_LCR)	21-9
Figure 21-7. UART Modem Control Registers (UARTx_MCR)	21-10
Figure 21-8. UART Line Status Registers (UARTx_LSR)	21-11
Figure 21-9. UART Modem Status Registers (UARTx_MSR)	21-13
Figure 21-10. Scratchpad Registers (UARTx_SCR)	21-14
Figure 21-11. UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM)	21-14
Figure 21-12. UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)	21-14
Figure 21-13. Chip Control Register 0 (CPC0_CR0)	21-17
Figure 22-1. 7-Bit Addressing	22-2
Figure 22-2. 10-Bit Addressing	22-2
Figure 22-3. IIC0 Master Data Buffer (IIC0_MDBUF)	22-3
Figure 22-4. FIFO Stages	22-4
Figure 22-5. IIC0 Slave Data Buffer (IIC0_SDBUF)	22-5
Figure 22-6. IIC0 Low Master Address Register (IIC0_LMADR)	22-5
Figure 22-7. IIC0 High Master Address Register (IIC0_HMADR)	22-6
Figure 22-8. IIC0 Control Register (IIC0_CNTL)	22-7
Figure 22-9. IIC0 Mode Control Register (IIC0_MDCNTL)	22-9
Figure 22-10. IIC0 Status Register (IIC0_STS)	22-10
Figure 22-11. IIC0 Extended Status Register (IIC0_EXTSTS)	22-12
Figure 22-12. IIC0 Low Slave Address Register (IIC0_LSADR)	22-14
Figure 22-13. IIC0 High Slave Address Register (IIC0_HSADR)	22-15
Figure 22-14. IIC0 Clock Divide Register (IIC0_CLKDIV)	22-15
Figure 22-15. IIC0 Interrupt Mask Register (IIC0_INTRMSK)	22-16
Figure 22-16. IIC0 Transfer Count Register (IIC0_XFRCNT)	22-17
Figure 22-17. IIC0 Extended Control and Slave Status Register (IIC0_XTCNTLSS)	22-18
Figure 22-18. IIC0 Direct Control Register (IIC0_DIRECTCNTL)	22-21
Figure 23-1. GPIO Functional Block Diagram	23-2
Figure 23-2. CPC0_CR0 Bits Controlling GPIO	23-3
Figure 23-3. GPIO Registers	23-5
Figure 25-1. Core Configuration Register 0 (CCR0)	25-15
Figure 25-2. Chip Control Register 0 (CPC0_CR0)	25-17
Figure 25-3. Chip Control Register 1 (CPC0_CR1)	25-20
Figure 25-4. CPM Enable Register (CPC0_ER)	25-21
Figure 25-5. CPM Force Register (CPC0_FR)	25-22
Figure 25-6. JTAG ID Register (CPC0_JTAGID)	25-23
Figure 25-7. PLL Mode Register (CPC0_PLLMR)	25-24
Figure 25-8. Chip Pin Strapping Register (CPC0_PSR)	25-26
Figure 25-9. CPM Status Register (CPC0_SR)	25-28
Figure 25-10. Condition Register (CR)	25-29
Figure 25-11. Count Register (CTR)	25-30
Figure 25-12. Data Address Compare Registers (DAC1–DAC2)	25-31
Figure 25-13. Debug Control Register 0 (DBC0)	25-32
Figure 25-14. Debug Control Register 1 (DBC1)	25-34
Figure 25-15. Debug Status Register (DBSR)	25-36

Figure 25-16.	Data Cache Cachability Register (DCCR)	25-38
Figure 25-17.	Decompression Address Decode Definition Registers (DCP0_ADDR0–DCP0_ADDR1)	25-40
Figure 25-18.	Decompression Controller Configuration Register (DCP0_CFG)	25-41
Figure 25-19.	Decompression Controller Error Status Register 0 (DCP0_ESR)	25-44
Figure 25-20.	Decompression Controller ID Register (DCP0_ID)	25-46
Figure 25-21.	Decompression Index Table Origin Registers (DCP0_ITOR0–DCP0_ITOR3)	25-47
Figure 25-22.	Decompression Controller Bus Error Address Register (DCP0_MEMBEAR)	25-48
Figure 25-23.	Decompression Controller PLB Error Address Register (DCP0_PLBBEAR)	25-49
Figure 25-24.	Decompression Controller Version Register (DCP0_VER)	25-51
Figure 25-25.	Data Cache Write-through Register (DCWR)	25-52
Figure 25-26.	Data Exception Address Register (DEAR)	25-54
Figure 25-27.	DMA Channel Control Registers (DMA0_CR0–DMA0_CR3)	25-55
Figure 25-28.	DMA Count Registers (DMA0_CT0–DMA0_CT3)	25-58
Figure 25-29.	DMA Destination Address Registers (DMA0_DA0–DMA0_DA3)	25-59
Figure 25-30.	DMA Polarity Configuration Register (DMA0_POL)	25-60
Figure 25-31.	DMA Source Address Registers (DMA0_SA0–DMA0_SA3)	25-62
Figure 25-32.	DMA Scatter/Gather Descriptor Address Registers (DMA0_SG0–DMA0_SG3)	25-63
Figure 25-33.	DMA Scatter/Gather Command Register (DMA0_SGC)	25-64
Figure 25-34.	DMA Sleep Mode Register (DMA0_SLP)	25-65
Figure 25-35.	DMA Status Register (DMA0_SR)	25-66
Figure 25-36.	Data Value Compare Registers (DVC1–DVC2)	25-67
Figure 25-37.	Peripheral Bus Error Address Register (EBC0_BEAR)	25-68
Figure 25-38.	Peripheral Bus Error Status Register 0 (EBC0_BESR0)	25-69
Figure 25-39.	Peripheral Bus Error Status Register 1 (EBC0_BESR1)	25-71
Figure 25-40.	Peripheral Bank Access Parameters (EBC0_BnAP)	25-73
Figure 25-41.	Peripheral Bank Configuration Registers (EBC0_BnCR)	25-75
Figure 25-42.	EBC Configuration Register (EBC0_CFG)	25-76
Figure 25-43.	Group Address Hash Tables 1–4 (EMAC0_GAHT1–EMAC0_GAHT4)	25-80
Figure 25-44.	Individual Address High Register (EMAC0_IAHR)	25-81
Figure 25-45.	STA Control Register (EMAC0_STACR)	25-82
Figure 25-46.	Transmit Mode Register 0 (EMAC0_TMR0)	25-83
Figure 25-47.	Transmit Mode Register 1 (EMAC0_TMR1)	25-84
Figure 25-48.	Transmit Request Threshold Register (EMAC0_TRTR)	25-85
Figure 25-49.	VLAN TCI Register (EMAC0_VTCI)	25-86
Figure 25-50.	VLAN TPID Register (EMAC0_VTPID)	25-87
Figure 25-51.	Exception Syndrome Register (ESR)	25-88
Figure 25-52.	Exception Vector Prefix Register (EVPR)	25-89
Figure 25-53.	GPIO Input Register (GPIO0_IR)	25-90
Figure 25-54.	GPIO Open Drain Register (GPIO0_ODR)	25-91
Figure 25-55.	GPIO Output Register (GPIO0_OR)	25-92
Figure 25-56.	GPIO Three-State Register (GPIO0_TCR)	25-93
Figure 25-57.	General Purpose Registers (R0–R31)	25-94
Figure 25-58.	Instruction Address Compare Registers (IAC1–IAC4)	25-95
Figure 25-59.	Instruction Cache Cachability Register (ICCR)	25-96
Figure 25-60.	Instruction Cache Debug Data Register (ICDBDR)	25-98
Figure 25-61.	IIC0 Clock Divide Register (IIC0_CLKDIV)	25-99

Figure 25-62.	IIC0 Control Register (IIC0_CNTL)	25-100
Figure 25-63.	IIC0 Direct Control Register (IIC0_DIRECTCNTL)	25-101
Figure 25-64.	IIC0 Extended Status Register (IIC0_EXTSTS)	25-102
Figure 25-65.	IIC0 High Master Address Register (IIC0_HMADR)	25-104
Figure 25-66.	IIC0 High Slave Address Register (IIC0_HSADR)	25-105
Figure 25-67.	IIC0 Interrupt Mask Register (IIC0_INTRMSK)	25-106
Figure 25-68.	IIC0 Low Master Address Register (IIC0_LMADR)	25-107
Figure 25-69.	IIC0 Low Slave Address Register (IIC0_LSADR)	25-108
Figure 25-70.	IIC0 Master Data Buffer (IIC0_MDBUF)	25-109
Figure 25-71.	IIC0 Mode Control Register (IIC0_MDCNTL)	25-110
Figure 25-72.	IIC0 Slave Data Buffer (IIC0_SDBUF)	25-111
Figure 25-73.	IIC0 Status Register (IIC0_STS)	25-112
Figure 25-74.	IIC0 Transfer Count Register (IIC0_XFRCNT)	25-113
Figure 25-75.	IIC0 Extended Control and Slave Status Register (IIC0_XTCNTLSS)	25-114
Figure 25-76.	Link Register (LR)	25-116
Figure 25-77.	MAL Configuration Register (MAL0_CFG)	25-117
Figure 25-78.	MAL Error Status Register (MAL0_ESR)	25-119
Figure 25-79.	MAL Interrupt Enable Register (MAL0_IER)	25-121
Figure 25-80.	RX Channel Buffer Size Register 0 (MAL0_RCBS0)	25-122
Figure 25-81.	RX Channel_Active Reset Register (MAL0_RXCARR)	25-123
Figure 25-82.	RX Channel_Active Set Register (MAL0_RXCASR)	25-124
Figure 25-83.	RX Channel Table Pointer x Register (MAL0_RXCTPxR)	25-125
Figure 25-84.	RX Descriptor Error Interrupt Register (MAL0_RXDEIR)	25-126
Figure 25-85.	RX End of Buffer Interrupt Status Register (MAL0_RXEOBISR)	25-127
Figure 25-86.	TX Channel_Active Reset Register (MAL0_TXCARR)	25-128
Figure 25-87.	TX Channel_Active Set Register (MAL0_TXCASR)	25-129
Figure 25-88.	TX Channel Table Pointer x Register (MAL0_TXCTPxR)	25-130
Figure 25-89.	TX Descriptor Error Interrupt Register (MAL0_TXDEIR)	25-131
Figure 25-90.	TX End of Buffer Interrupt Status Register (MAL0_TXEOBISR)	25-132
Figure 25-91.	Machine State Register (MSR)	25-133
Figure 25-92.	OCM Data-Side Address Range Compare Register (OCM0_DSARC)	25-135
Figure 25-93.	OCM Data-Side Control Register (OCM0_DSCNTL)	25-136
Figure 25-94.	OCM Instruction-Side Address Range Compare Register (OCM0_ISARC)	25-137
Figure 25-95.	OCM Instruction-Side Control Register (OCM0_ISCNTL)	25-138
Figure 25-96.	OPB Arbiter Control Register (OPBA0_CR)	25-139
Figure 25-97.	OPB Arbiter Priority Register (OPBA0_PR)	25-140
Figure 25-98.	PCI Base Address Register (PCIC0_BAR0)	25-141
Figure 25-99.	PCI Built-in Self Test Control Register (PCIC0_BIST)	25-142
Figure 25-100.	Bridge Options 1 Register (PCIC0_BRDGOPT1)	25-143
Figure 25-101.	Bridge Options 2 Register (PCIC0_BRDGOPT2)	25-144
Figure 25-102.	PCI Cache Line Size Register (PCIC0_CACHELS)	25-145
Figure 25-103.	PCI Capabilities Pointer (PCIC0_CAP)	25-146
Figure 25-104.	Capability Identifier (PCIC0_CAPID)	25-147
Figure 25-105.	PCI Configuration Address Register (PCIC0_CFGADDR)	25-148
Figure 25-106.	PCI Configuration Data Register (PCIC0_CFGDATA)	25-149
Figure 25-107.	PCI Class Register (PCIC0_PCICLS)	25-150

Figure 25-108. PCI Command Register (PCIC0_CMD)	25-151
Figure 25-109. PCI Data (PCIC0_DATA)	25-153
Figure 25-110. PCI Device ID Register (PCIC0_DEVID)	25-154
Figure 25-111. Error Enable Register (PCIC0_ERREN)	25-155
Figure 25-112. Error Status Register (PCIC0_ERRSTS)	25-156
Figure 25-113. PCI Header Type Register (PCIC0_HDTYPE)	25-157
Figure 25-114. PCI Interrupt Control/Status Register	25-158
Figure 25-115. PCI Interrupt Line Register (PCIC0_INTLN)	25-159
Figure 25-116. PCI Interrupt Pin Register (PCIC0_INTPN)	25-160
Figure 25-117. PCI Latency Timer Register (PCIC0_LATTIM)	25-161
Figure 25-118. PCI Maximum Latency Register (PCIC0_MAXLTNCY)	25-162
Figure 25-119. PCI Minimum Grant Register (PCIC0_MINGNT)	25-163
Figure 25-120. Next Item Pointer (PCIC0_NEXTIPTR)	25-164
Figure 25-121. PLB Slave Error Address Register (PCIC0_PLBBEAR)	25-165
Figure 25-122. PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0)	25-166
Figure 25-123. PLB Slave Error Syndrome 1 (PCIC0_PLBBESR1)	25-168
Figure 25-124. Power Management Capabilities Register (PCIC0_PMC)	25-170
Figure 25-125. Power Management Control/Status Register (PCIC0_PMCSR)	25-171
Figure 25-126. PMCSR PCI to PCI Bridge Support Extensions (PCIC0_PMCSRSE)	25-172
Figure 25-127. Power Management State Change Request Register (PCIC0_PMSCRR)	25-173
Figure 25-128. PCI PTM 1 BAR Register (PCIC0_PTM1BAR)	25-174
Figure 25-129. PCI PTM 2 BAR Register (PCIC0_PTM2BAR)	25-175
Figure 25-130. PCI Revision ID Register (PCIC0_REVID)	25-176
Figure 25-131. PCI Subsystem ID Register (PCIC0_SBSYSID)	25-177
Figure 25-132. PCI Subsystem Vendor ID Register (PCIC0_SBSYSVID)	25-178
Figure 25-133. PCI Status Register (PCIC0_STATUS)	25-179
Figure 25-134. PCI Vendor ID Register (PCIC0_VENDID)	25-181
Figure 25-135. PMM 0 Local Address Register (PCILO_PMM0LA)	25-182
Figure 25-136. PMM 0 Mask/Attribute Register (PCILO_PMM0MA)	25-183
Figure 25-137. PMM 1 PCI High Address Register (PCILO_PMM1PCIHA)	25-184
Figure 25-138. PMM 0 PCI Low Address Register (PCILO_PMM0PCILA)	25-185
Figure 25-139. PMM 1 Local Address Register (PCILO_PMM1LA)	25-186
Figure 25-140. PMM 1 Mask/Attribute Register (PCILO_PMM1MA)	25-187
Figure 25-141. PMM 1 PCI High Address Register (PCILO_PMM1PCIHA)	25-188
Figure 25-142. PMM 1 PCI Low Address Register (PCILO_PMM1PCILA)	25-189
Figure 25-143. PMM 2 Local Address Register (PCILO_PMM2LA)	25-190
Figure 25-144. PMM 2 Mask/Attribute Register (PCILO_PMM2MA)	25-191
Figure 25-145. PMM 2 PCI High Address Register (PCILO_PMM2PCIHA)	25-192
Figure 25-146. PMM 2 Low Address Register (PCILO_PMM2PCILA)	25-193
Figure 25-147. PTM 2 Local Address Register (PCILO_PTM1LA)	25-194
Figure 25-148. PTM 1 Memory Size/Attribute Register (PCILO_PTM1MS)	25-195
Figure 25-149. PTM 2 Local Address Register (PCILO_PTM2LA)	25-196
Figure 25-150. PTM 2 Memory Size/Attribute Register (PCILO_PTM2MS)	25-197
Figure 25-151. Process ID (PID)	25-198
Figure 25-152. Programmable Interval Timer (PIT)	25-199
Figure 25-153. PLB Arbiter Control Register (PLB0_ACR)	25-200

Figure 25-154. PLB Error Address Register (PLB0_BEAR)	25-201
Figure 25-155. PLB Error Status Register (PLB0_BESR)	25-202
Figure 25-156. Bridge Error Address Register (POB0_BEAR)	25-204
Figure 25-157. Bridge Error Status Register 0 (POB0_BESR0)	25-205
Figure 25-158. Bridge Error Status Register 1 (POB0_BESR1)	25-207
Figure 25-159. Processor Version Register (PVR)	25-208
Figure 25-160. Memory Bank 0–3 Configuration Registers (SDRAM0_B0CR–SDRAM0_B3CR)	25-209
Figure 25-161. Bus Error Address Register (SDRAM0_BEAR)	25-210
Figure 25-162. Bus Error Syndrome Register 0 (SDRAM0_BESR0)	25-211
Figure 25-163. Bus Error Status Register 1 (SDRAM0_BESR1)	25-213
Figure 25-164. Memory Controller Configuration (SDRAM0_CFG)	25-214
Figure 25-165. ECC Configuration Register (SDRAM0_ECCCFG)	25-218
Figure 25-166. ECC Error Status Register (SDRAM0_ECCEsr)	25-219
Figure 25-167. Power Management Idle Timer (SDRAM0_PMIT)	25-220
Figure 25-168. Refresh Timing Register (SDRAM0_RTR)	25-221
Figure 25-169. SDRAM Timing Register (SDRAM0_TR)	25-222
Figure 25-170. Storage Guarded Register (SGR)	25-224
Figure 25-171. Storage Little-Endian Register (SLER)	25-226
Figure 25-172. Special Purpose Registers General (SPRG0–SPRG7)	25-228
Figure 25-173. Save/Restore Register 0 (SRR0)	25-229
Figure 25-174. Save/Restore Register 1 (SRR1)	25-230
Figure 25-175. Save/Restore Register 2 (SRR2)	25-231
Figure 25-176. Save/Restore Register 3 (SRR3)	25-232
Figure 25-177. Storage User-defined 0 Register (SU0R)	25-233
Figure 25-178. Time Base Lower (TBL)	25-235
Figure 25-179. Time Base Upper (TBU)	25-236
Figure 25-180. Timer Control Register (TCR)	25-237
Figure 25-181. Timer Status Register (TSR)	25-238
Figure 25-182. UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)	25-239
Figure 25-183. UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM)	25-240
Figure 25-184. UART FIFO Control Registers (UARTx_FCR)	25-241
Figure 25-185. UART Interrupt Enable Registers (UARTx_IER)	25-242
Figure 25-186. UART Interrupt Identification Registers (UARTx_IIR)	25-243
Figure 25-187. UART Line Control Registers (UARTx_LCR)	25-244
Figure 25-188. UART Line Status Registers (UARTx_LSR)	25-245
Figure 25-189. UART Modem Control Registers (UARTx_MCR)	25-247
Figure 25-190. UART Modem Status Registers (UARTx_MSR)	25-248
Figure 25-191. UART Receiver Buffer Registers (UARTx_RBR)	25-249
Figure 25-192. Scratchpad Registers (UARTx_SCR)	25-250
Figure 25-193. UART Transmitter Holding Registers (UARTx_THR)	25-251
Figure 25-194. UIC Critical Register (UIC0_CR)	25-252
Figure 25-195. UIC Enable Register (UIC0_ER)	25-255
Figure 25-196. UIC Masked Status Register (UIC0_MSR)	25-258
Figure 25-197. UIC Polarity Register (UIC0_PR)	25-261
Figure 25-198. UIC Status Register (UIC0_SR)	25-264
Figure 25-199. UIC Trigger Register (UIC0_TR)	25-267

Figure 25-200. UIC Vector Configuration Register (UIC0_VCR)	25-270
Figure 25-201. UIC Vector Register (UIC0_VR)	25-271
Figure 25-202. User SPR General 0 (USPRG0)	25-272
Figure 25-203. Fixed Point Exception Register (XER)	25-273
Figure 25-204. Zone Protection Register (ZPR)	25-274
Figure A-1. I Instruction Format	A-44
Figure A-2. B Instruction Format	A-44
Figure A-3. SC Instruction Format	A-44
Figure A-4. D Instruction Format	A-44
Figure A-5. X Instruction Format	A-45
Figure A-6. XL Instruction Format	A-45
Figure A-7. XFX Instruction Format	A-46
Figure A-8. XO Instruction Format	A-46
Figure A-9. M Instruction Format	A-46

Tables

Table 2-1. PPC405GP PLB Agents as Masters and Slaves.....	2-2
Table 2-2. Registers Controlling PLB Master Priority Assignments.....	2-3
Table 2-3. PLB Arbiter Registers.....	2-5
Table 2-4. PLB Arbiter Registers.....	2-8
Table 2-5. PPC405GP OPB Master Assignments.....	2-12
Table 2-6. PLB Arbiter Registers.....	2-12
Table 3-1. PPC405GP Address Space.....	3-2
Table 3-2. PPC405GP SPRs.....	3-7
Table 3-3. XER[CA] Updating Instructions.....	3-10
Table 3-4. XER[SO,OV] Updating Instructions.....	3-11
Table 3-5. Time Base Registers.....	3-15
Table 3-6. Directly Accessed DCRs.....	3-17
Table 3-7. SDRAM Controller DCR Usage.....	3-19
Table 3-8. Offsets for SDRAM Controller Registers.....	3-19
Table 3-9. External Bus Controller DCR Usage.....	3-20
Table 3-10. Offsets for External Bus Controller Registers.....	3-20
Table 3-11. Decompression Controller DCR Usage.....	3-21
Table 3-12. Offsets for Decompression Controller Registers.....	3-21
Table 3-13. Directly Accessed MMIO Registers.....	3-22
Table 3-14. PCI Configuration Address and Data Registers.....	3-25
Table 3-15. PCI Configuration Registers.....	3-25
Table 3-16. Alignment Exception Summary.....	3-28
Table 3-17. Bits of the BO Field.....	3-35
Table 3-18. Conditional Branch BO Field.....	3-36
Table 3-19. Example Memory Mapping.....	3-40
Table 3-20. Privileged Instructions.....	3-42
Table 3-21. PPC405GP Instruction Set Summary.....	3-47
Table 3-22. Implementation-specific Instructions.....	3-48
Table 3-23. Storage Reference Instructions.....	3-48
Table 3-24. Arithmetic Instructions.....	3-49
Table 3-25. Multiply-Accumulate and Multiply Halfword Instructions.....	3-50
Table 3-26. Logical Instructions.....	3-50
Table 3-27. Compare Instructions.....	3-50
Table 3-28. Branch Instructions.....	3-51
Table 3-29. CR Logical Instructions.....	3-51
Table 3-30. Rotate Instructions.....	3-51
Table 3-31. Shift Instructions.....	3-52
Table 3-32. Cache Management Instructions.....	3-52
Table 3-33. Interrupt Control Instructions.....	3-52
Table 3-34. TLB Management Instructions.....	3-53
Table 3-35. Processor Management Instructions.....	3-53
Table 4-1. Instruction Cache Organization.....	4-2

Table 4-2. Data Cache Organization	4-6
Table 4-3. Priority Changes With Different Data Cache Operations	4-17
Table 5-1. Examples of Store Data Bypass	5-4
Table 5-2. OCM DCRs	5-5
Table 6-1. TLB Fields Related to Page Size	6-4
Table 6-2. Protection Applied to Cache Control Instructions	6-15
Table 7-1. Clock Strapping Values	7-3
Table 7-2. PLL Tuning Settings	7-4
Table 7-3. Possible Clocking Ratios for Reference Clock of 33.3MHz	7-5
Table 7-4. Possible Clocking Ratios for Reference Clock of 25MHz	7-6
Table 7-5. Possible Clocking Ratios for Reference Clock of 41.6MHz	7-7
Table 7-6. Example Synchronous PCI Clock Frequencies in Asynchronous Mode	7-8
Table 7-7. Clocking Control Registers	7-9
Table 8-1. MSR Contents after Reset	8-3
Table 8-2. SPR Contents After Reset	8-4
Table 8-3. DCR Contents After Reset	8-4
Table 8-4. MMIO Register Contents After Reset	8-8
Table 9-1. Multiplexed Pins	9-3
Table 10-1. UIC Interrupt Assignments	10-1
Table 10-2. UIC DCRs.....	10-3
Table 10-3. Interrupt Handling Priorities	10-25
Table 10-4. Interrupt Vector Offsets	10-27
Table 10-5. ESR Alteration by Various Interrupts	10-33
Table 10-6. ESR Alteration by Various Interrupts	10-33
Table 10-7. Register Settings during Critical Input Interrupts	10-35
Table 10-8. Register Settings during Machine Check—Instruction Interrupts	10-36
Table 10-9. Register Settings during Machine Check—Data Interrupts	10-36
Table 10-10. Register Settings during Data Storage Interrupts	10-37
Table 10-11. Register Settings during Instruction Storage Interrupts	10-38
Table 10-12. Register Settings during External Interrupts	10-39
Table 10-13. Alignment Interrupt Summary	10-39
Table 10-14. Register Settings during Alignment Interrupts	10-40
Table 10-15. ESR Usage for Program Interrupts	10-40
Table 10-16. Register Settings during Program Interrupts	10-41
Table 10-17. Register Settings during System Call Interrupts	10-41
Table 10-18. Register Settings during Programmable Interval Timer Interrupts	10-42
Table 10-19. Register Settings during Fixed Interval Timer Interrupts	10-42
Table 10-20. Register Settings during Watchdog Timer Interrupts	10-43
Table 10-21. Register Settings during Data TLB Miss Interrupts	10-43
Table 10-22. Register Settings during Instruction TLB Miss Interrupts	10-44
Table 10-23. SRR2 during Debug Interrupts.....	10-45
Table 10-24. Register Settings during Debug Interrupts	10-45
Table 11-1. Time Base Access	11-3
Table 11-2. FIT Controls	11-5
Table 11-3. Watchdog Timer Controls	11-6
Table 12-1. JTAG Connector Signals	12-2

Table 12-2. JTAG Instructions	12-3
Table 12-3. RISCTrace Header Pin Description	12-5
Table 12-4. Debug Events	12-16
Table 12-5. DAC Applied to Cache Instructions	12-20
Table 12-6. Setting of DBSR Bits for DAC and DVC Events	12-22
Table 12-7. Comparisons Based on DBCR1[DVnM]	12-23
Table 13-1. CPM Registers.....	13-1
Table 14-1. DCRs Used to Access the Decompression Controller Registers	14-4
Table 14-2. Offsets for Decompression Controller Registers	14-4
Table 15-1. SDRAM Signal Usage and State During/Following Reset.....	15-2
Table 15-2. SDRAM Controller DCR Addresses	15-2
Table 15-3. SDRAM Controller Configuration and Status Registers	15-3
Table 15-4. SDRAM Addressing Modes.....	15-7
Table 15-5. SDRAM Page Size	15-7
Table 15-6. Logical Address Bit on BA1:0 and MemAddr12:0 Versus Addressing Mode.	15-8
Table 15-7. SDRAM Memory Timing Parameters	15-10
Table 15-8. Additional Latency when using ECC.....	15-14
Table 16-1. EBC Signal Usage and State During/Following a Chip or System Reset.....	16-2
Table 16-2. Effect of Driver Enable Programming on EBC Signal States.....	16-5
Table 16-3. External Master Arbitration.	16-18
Table 16-4. Signal States During Hold Acknowledge (HoldAck=1)	16-18
Table 16-5. EBC DCR Addresses.....	16-23
Table 16-6. External Bus Configuration and Status Registers	16-23
Table 17-1. PowerPC, CoreConnect PLB, and PCI Address Bit-Naming Conventions	17-3
Table 17-2. PowerPC, CoreConnect PLB, and PCI Data Bus Bit-Naming Conventions	17-3
Table 17-3. PLB Address Map	17-6
Table 17-4. PCI Memory Address Map.....	17-8
Table 17-5. Transaction Mapping: PLB → PCI	17-10
Table 17-6. Transaction Mapping: PCI → PLB	17-14
Table 17-7. Collision Resolution	17-19
Table 17-8. Directly Accessed MMIO Registers	17-19
Table 17-9. PCI Configuration Address and Data Registers	17-20
Table 17-10. PCI Configuration Register Offsets	17-20
Table 17-11. PLB Unsupported Transfer Types	17-55
Table 17-12. Address Map Register Values	17-63
Table 18-1. DMA Controller External I/Os	18-1
Table 18-2. DMA Controller Configuration and Status Registers	18-4
Table 18-3. DMA Transfer Priorities	18-13
Table 18-4. Address Alignment Requirements	18-15
Table 18-5. Scatter/Gather Descriptor Table	18-16
Table 18-6. Bit Fields in the Scatter/Gather Descriptor Table	18-16
Table 18-7. DMA Registers Loaded from Scatter/Gather Descriptor Table.....	18-17
Table 19-1. FCS/SA Enable - Possible Configurations.....	19-11
Table 19-2. FCS/Pad Enable - Possible Configurations	19-11
Table 19-3. FCS/VLAN Tag Enable - Possible Configurations.....	19-11
Table 19-4. In Range Length Error Behavior for Various Packet Lengths.....	19-13

Table 19-5. EMAC Register Summary	19-23
Table 20-1. MAL Registers.....	20-24
Table 21-1. Baud Rate Settings	21-3
Table 21-2. UART Configuration Registers.....	21-4
Table 21-3. Interrupt Priority Level	21-6
Table 21-4. Divisor Latch Settings for Certain Baud Rates	21-15
Table 21-5. UART0 Transmitter DMA Mode Register Field Settings	21-19
Table 21-6. UART0 Receiver DMA Mode Register Field Settings	21-20
Table 22-1. IIC Registers.....	22-2
Table 22-2. IIC Response to IIC0_CNTL Field Settings	22-8
Table 22-3. IIC0_STS[ERR, PT] Decoding	22-11
Table 22-4. IIC0 Clock Divide Programming	22-16
Table 23-1. Macro I/O Interface Signals	23-2
Table 23-2. GPIO Register Summary.....	23-5
Table 23-3. GPIO0_ODR Control Logic	23-6
Table 24-1. Implementation-Specific Instructions.....	24-1
Table 24-2. Operator Precedence	24-5
Table 24-3. Extended Mnemonics for addi	24-9
Table 24-4. Extended Mnemonics for addic	24-10
Table 24-5. Extended Mnemonics for addic.	24-11
Table 24-6. Extended Mnemonics for addis	24-12
Table 24-7. Extended Mnemonics for bc, bca, bcl, bcla	24-21
Table 24-8. Extended Mnemonics for bcctr, bcctrl	24-27
Table 24-9. Extended Mnemonics for bclr, bclrl	24-30
Table 24-10. Extended Mnemonics for cmp	24-34
Table 24-11. Extended Mnemonics for cmpi	24-35
Table 24-12. Extended Mnemonics for cmpl	24-36
Table 24-13. Extended Mnemonics for cmpli	24-37
Table 24-14. Extended Mnemonics for creqv	24-41
Table 24-15. Extended Mnemonics for crnor	24-43
Table 24-16. Extended Mnemonics for cror	24-44
Table 24-17. Extended Mnemonics for crxor	24-46
Table 24-18. Transfer Bit Mnemonic Assignment	24-108
Table 24-19. Extended Mnemonics for mfspr	24-113
Table 24-20. Extended Mnemonics for mftb.....	24-114
Table 24-21. Extended Mnemonics for mftb	24-115
Table 24-22. Extended Mnemonics for mtrcf	24-116
Table 24-23. Extended Mnemonics for mtspr	24-120
Table 24-24. Extended Mnemonics for nor, nor.	24-139
Table 24-25. Extended Mnemonics for or, or.	24-140
Table 24-26. Extended Mnemonics for ori	24-142
Table 24-27. Extended Mnemonics for rlwimi, rlwimi.	24-146
Table 24-28. Extended Mnemonics for rlinm, rlinm.	24-147
Table 24-29. Extended Mnemonics for rlwnm, rlwnm.	24-150
Table 24-30. Extended Mnemonics for subf, subf., subfo, subfo.	24-176
Table 24-31. Extended Mnemonics for subfc, subfc., subfco, subfco.	24-177

Table 24-32. Extended Mnemonics for tlbre	24-185
Table 24-33. Extended Mnemonics for tlbwe	24-189
Table 24-34. Extended Mnemonics for tw	24-191
Table 24-35. Extended Mnemonics for twi	24-194
Table 25-1. PPC405GP General Purpose Registers.....	25-1
Table 25-2. Special Purpose Registers	25-2
Table 25-3. Time Base Registers	25-4
Table 25-4. Directly Accessed DCRs	25-4
Table 25-5. SDRAM Controller DCR Usage	25-7
Table 25-6. Offsets for SDRAM Controller Registers	25-7
Table 25-7. EBC DCR Usage	25-8
Table 25-8. Offsets for EBC Registers	25-8
Table 25-9. Decompression Controller DCR Usage	25-9
Table 25-10. Offsets for Decompression Controller Registers	25-9
Table 25-11. Directly Accessed MMIO Registers	25-10
Table 25-12. PCI Configuration Address and Data Registers	25-13
Table 25-13. PCI Configuration Registers	25-13
Table 26-1. Alphabetical Signal List.....	26-1
Table 26-2. Signal Descriptions	26-5
Table A-1. PPC405GP Instruction Syntax Summary	A-1
Table A-2. PPC405GP Instructions by Opcode	A-33
Table B-1. PPC405GP Instruction Set Functional Summary	B-1
Table B-2. Implementation-specific Instructions	B-1
Table B-3. Instructions in the IBM PowerPC Embedded Environment	B-5
Table B-4. Privileged Instructions	B-7
Table B-5. Extended Mnemonics for PPC405GP	B-10
Table B-6. Storage Reference Instructions	B-29
Table B-7. Arithmetic and Logical Instructions	B-33
Table B-8. Condition Register Logical Instructions	B-37
Table B-9. Branch Instructions	B-38
Table B-10. Comparison Instructions	B-39
Table B-11. Rotate and Shift Instructions	B-40
Table B-12. Cache Control Instructions	B-41
Table B-13. Interrupt Control Instructions	B-42
Table B-14. Processor Management Instructions	B-42
Table C-1. Cache Sizes, Tag Fields, and Lines.....	C-2
Table C-2. Multiply and MAC Instruction Timing.....	C-5
Table C-4. Instruction Cache Miss Penalties.....	C-7

About This Book

This user's manual provides the architectural overview, programming model, and detailed information about the registers, the instruction set, and operations of the IBM PowerPC 405GP (PPC405GP) 32-bit RISC embedded processor.

The PPC405GP RISC embedded processor features:

- PowerPC Architecture™
- Single-cycle execution for most instructions
- Instruction cache unit and data cache unit
- Support for Little Endian operation
- Interrupt interface for one critical and one non-critical interrupt signal
- JTAG interface
- Extensive development tool support

Who Should Use This Book

This book is for system hardware and software developers, and for application developers who need to understand the PPC405GP. The audience should understand embedded processor design, embedded system design, operating systems, RISC processing, and design for testability.

How to Use This Book

This book describes the PPC405GP device architecture, programming model, external interfaces, internal registers, and instruction set. This book contains the following chapters, arranged in parts:

Part I Introducing the PPC405GP Embedded Processor

Chapter 1 Overview

Chapter 2 On-Chip Buses

Part II The PPC405GP RISC Processor

Chapter 3 Programming Model

Chapter 4 Cache Operations

Chapter 5 On-Chip Memory

Chapter 6 Memory Management

Part III PPC405GP System Operations

Chapter 7 Clocking

Chapter 8 Reset and Initialization

Chapter 9 Pin Strapping and Sharing

Chapter 10 Interrupt Controller Operations

Chapter 11 Timer Facilities

Chapter 12 Debugging

Chapter 13 Clock and Power Management

Chapter 14 Decompression Controller Operation

Part IV PPC405GP External Interfaces

- Chapter 15 SDRAM Controller
- Chapter 16 External Bus Controller
- Chapter 17 PCI Interface
- Chapter 18 Direct Memory Access Controller
- Chapter 19 Ethernet Media Access Controller
- Chapter 20 Memory Access Layer
- Chapter 21 Serial Port Operations
- Chapter 22 IIC Bus Interface
- Chapter 23 GPIO Operations

Part V Reference

- Chapter 24 Instruction Set
- Chapter 25 Register Summary
- Chapter 26 Signal Summary

This book contains the following appendixes:

- Appendix A Instruction Summary
- Appendix B Instructions by Category
- Appendix C Code Optimization and Instruction Timings

To help readers find material in these chapters, the book contains:

- Contents, on page v.
- Figures, on page xxix.
- Tables, on page xli.
- Index, on page X-1.

Conventions

The following is a list of notational conventions frequently used in this manual.

<u>ActiveLow</u>	An overbar indicates an active-low signal.
<i>n</i>	A decimal number
<i>0xn</i>	A hexadecimal number
<i>0bn</i>	A binary number
=	Assignment
^	AND logical operator
¬	NOT logical operator
∨	OR logical operator
⊕	Exclusive-OR (XOR) logical operator
+	Twos complement addition
-	Twos complement subtraction, unary minus
×	Multiplication

÷	Division yielding a quotient
%	Remainder of an integer division; (33 % 32) = 1.
	Concatenation
=, ≠	Equal, not equal relations
<, >	Signed comparison relations
< ^u , > ^u	Unsigned comparison relations
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.
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.
leave	Leave innermost do loop or do loop specified in a leave statement.
FLD	An instruction or register field
FLD _b	A bit in a named instruction or register field
FLD _{b:b}	A range of bits 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
REG _b	A bit in a named register
REG _{b:b}	A range of bits in a named register
REG _{b,b,...}	A list of bits, by number or name, in a named register
REG[FLD]	A field in a named register
REG[FLD, FLD ...]	A list of fields in a named register
REG[FLD:FLD]	A range of fields in a named register
GPR(<i>r</i>)	General Purpose Register (GPR) <i>r</i> , where $0 \leq r \leq 31$.
(GPR(<i>r</i>))	The contents of GPR <i>r</i> , where $0 \leq r \leq 31$.
DCR(DCRN)	A Device Control Register (DCR) specified by the DCRF field in an mfocr or mtocr instruction
SPR(SPRN)	An SPR specified by the SPRF field in an mfspir or mtspir instruction
TBR(TBRN)	A Time Base Register (TBR) specified by the TBRF field in an mtfb instruction
GPRs	RA, RB, ...
(Rx)	The contents of a GPR, where <i>x</i> is A, B, S, or T
(RAI0)	The contents of the register RA or 0, if the RA field is 0.
CR _{FLD}	The field in the condition register pointed to by a field of an instruction.
C _{0:3}	A 4-bit object used to store condition results in compare instructions.
"b	The bit or bit value <i>b</i> is replicated <i>n</i> times.

÷	Division yielding a quotient
%	Remainder of an integer division; (33 % 32) = 1.
	Concatenation
=, ≠	Equal, not equal relations
<, >	Signed comparison relations
< ^u , > ^u	Unsigned comparison relations
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.
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.
leave	Leave innermost do loop or do loop specified in a leave statement.
FLD	An instruction or register field
FLD _b	A bit in a named instruction or register field
FLD _{b:b}	A range of bits 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
REG _b	A bit in a named register
REG _{b:b}	A range of bits in a named register
REG _{b,b,...}	A list of bits, by number or name, in a named register
REG[FLD]	A field in a named register
REG[FLD, FLD ...]	A list of fields in a named register
REG[FLD:FLD]	A range of fields in a named register
GPR(<i>r</i>)	General Purpose Register (GPR) <i>r</i> , where $0 \leq r \leq 31$.
(GPR(<i>r</i>))	The contents of GPR <i>r</i> , where $0 \leq r \leq 31$.
DCR(DCRN)	A Device Control Register (DCR) specified by the DCRF field in an mfocr or mtocr instruction
SPR(SPRN)	An SPR specified by the SPRF field in an mf spr or mt spr instruction
TBR(TBRN)	A Time Base Register (TBR) specified by the TBRF field in an mtb instruction
GPRs	RA, RB, ...
(Rx)	The contents of a GPR, where <i>x</i> is A, B, S, or T
(RAI0)	The contents of the register RA or 0, if the RA field is 0.
CR _{FLD}	The field in the condition register pointed to by a field of an instruction.
C _{0:3}	A 4-bit object used to store condition results in compare instructions.
"b	The bit or bit value <i>b</i> is replicated <i>n</i> times.

xx	Bit positions which are don't-cares.
CEIL(x)	Least integer $\geq x$.
EXTS(x)	The result of extending x on the left with sign bits.
PC	Program counter.
RESERVE	Reserve bit; indicates whether a process has reserved a block of storage.
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.
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.
MS(addr, n)	The number of bytes represented by n at the location in main storage represented by <i>addr</i> .
EA	Effective address; the 32-bit address, derived by applying indexing or indirect addressing rules to the specified operand, that specifies a location in main storage.
EA _b	A bit in an effective address.
EA _{b:b}	A range of bits in an effective address.
ROTL((RS),n)	Rotate left; the contents of RS are shifted left the number of bits specified by n .
MASK(MB,ME)	Mask having 1s in positions MB through ME (wrapping if MB > ME) and 0s elsewhere.
instruction(EA)	An instruction operating on a data or instruction cache block associated with an EA.

Part V. Reference

Chapter 24. Instruction Set

Descriptions of the PPC405GP instructions follow. Each description contains the following elements:

- Instruction names (mnemonic and full)
- Instruction syntax
- Instruction format diagram
- Pseudocode description
- Prose description
- Registers altered
- Architecture notes identifying the associated PowerPC Architecture component

Where appropriate, instruction descriptions list invalid instruction forms and exceptions, and provide programming notes.

24.1 Instruction Set Portability

To support embedded real-time applications, the instruction sets of the PPC405GP and other IBM PowerPC 400Series embedded controllers implement the IBM PowerPC Embedded Environment, which is not part of the PowerPC Architecture defined in *The PowerPC Architecture: A Specification for a New Family of RISC Processors*.

Programs using these instructions are not portable to PowerPC implementations that do not implement the IBM PowerPC Embedded Environment.

The PPC405GP implements a number of implementation-specific instructions that are not part of the PowerPC Architecture or the IBM PowerPC Embedded Environment, which are listed in Table 24-1. In the table, the syntax “[o]” indicates that an instruction has an “o” form, which updates the XER[SO,OV] fields, and a “non-o” form. The syntax “[.]” indicates that an instruction has a “record” form, which updates CR[CR0], and a “non-record” form.

Table 24-1. Implementation-Specific Instructions

dccci	macchw[o][.]	mfdcr	nmacchw[o][.]	rfci
dcread	macchws[o][.]	mtdcr	nmacchws[o][.]	tlbre
iccci	macchwsu[o][.]	mulchw[.]	nmachhw[o][.]	tlbsx[.]
icread	macchwu[o][.]	mulchwu[.]	nmachhws[o][.]	tlbwe
	machhw[o][.]	mulhhw[.]	nmachlw[o][.]	wrttee
	machhws[o][.]	mulhhwu[.]	nmaclhws[o][.]	wrteel
	machhwsu[o][.]	mullhw[.]		
	machhwu[o][.]	mullhwu[.]		
	machlw[o][.]			
	machlws[o][.]			
	machlwsu[o][.]			
	machlwu[o][.]			

24.2 Instruction Formats

For more detailed information about instruction formats, including a summary of instruction field usage and instruction format diagrams for the PPC405GP, see “Instruction Formats” on page 24-2.

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. The 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 general purpose register selectors and immediate values, that may vary from execution to execution. The instruction format diagrams specify the operands in 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 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 and its result is architecturally undefined. Unless otherwise noted, the execute all invalid instruction forms without causing an illegal instruction exception.

24.3 Pseudocode

The pseudocode that appears in the instruction descriptions provides a semi-formal language for describing instruction operations.

The pseudocode uses the following notation:

=	Assignment
^	AND logical operator
¬	NOT logical operator
∨	OR logical operator
⊕	Exclusive-OR (XOR) logical operator
+	Twos complement addition
-	Twos complement subtraction, unary minus
×	Multiplication
÷	Division yielding a quotient
%	Remainder of an integer division; (33 % 32) = 1.

	Concatenation
=, ≠	Equal, not equal relations
<, >	Signed comparison relations
\leq , \geq	Unsigned comparison relations
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.
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.
leave	Leave innermost do loop or do loop specified in a leave statement.
n	A decimal number
0xn	A hexadecimal number
0bn	A binary number
FLD	An instruction or register field
FLD _b	A bit in a named instruction or register field
FLD _{b:b}	A range of bits 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
REG _b	A bit in a named register
REG _{b:b}	A range of bits in a named register
REG _{b,b,...}	A list of bits, by number or name, in a named register
REG[FLD]	A field in a named register
REG[FLD, FLD ...]	A list of fields in a named register
REG[FLD:FLD]	A range of fields in a named register
GPR(<i>r</i>)	General Purpose Register (GPR) <i>r</i> , where $0 \leq r \leq 31$.
(GPR(<i>r</i>))	The contents of GPR <i>r</i> , where $0 \leq r \leq 31$.
DCR(DCRN)	A Device Control Register (DCR) specified by the DCRF field in an mf dcr or mt dcr instruction
SPR(SPRN)	An SPR specified by the SPRF field in an mf spr or mt spr instruction
TBR(TBRN)	A Time Base Register (TBR) specified by the TBRF field in an mf tb instruction
GPRs	RA, RB, ...
(Rx)	The contents of a GPR, where <i>x</i> is A, B, S, or T
(RAI0)	The contents of the register RA or 0, if the RA field is 0.
c _{0:3}	A four-bit object used to store condition results in compare instructions.
ⁿ b	The bit or bit value <i>b</i> is replicated <i>n</i> times.

xx	Bit positions which are don't-cares.
CEIL(x)	Least integer $\geq x$.
EXTS(x)	The result of extending x on the left with sign bits.
PC	Program counter.
RESERVE	Reserve bit; indicates whether a process has reserved a block of storage.
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.
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.
MS(addr, n)	The number of bytes represented by n at the location in main storage represented by $addr$.
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.
EA _b	A bit in an effective address.
EA _{b:b}	A range of bits in an effective address.
ROTL((RS),n)	Rotate left; the contents of RS are shifted left the number of bits specified by n .
MASK(MB,ME)	Mask having 1s in positions MB through ME (wrapping if MB > ME) and 0s elsewhere.
instruction(EA)	An instruction operating on a data or instruction cache block associated with an EA.

24.3.1 Operator Precedence

Table 24-2 lists the pseudocode operators and their associativity in descending order of precedence:

Table 24-2. Operator Precedence

Operators	Associativity
REG _b , REG[FLD], function evaluation	Left to right
ⁿ b	Right to left
¬, − (unary minus)	Right to left
×, ÷	Left to right
+, −	Left to right
	Left to right
=, ≠, <, >, < ^u , > ^u	Left to right
∧, ⊕	Left to right
∨	Left to right
←	None

24.4 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). Other registers are changed, with the details of the change not included in the instruction description. This category frequently includes the Condition Register (CR) and the Fixed-point Exception Register (XER). For discussion of the CR, see “Condition Register (CR)” on page 3-12. For discussion of XER, see “Fixed Point Exception Register (XER)” on page 3-8.

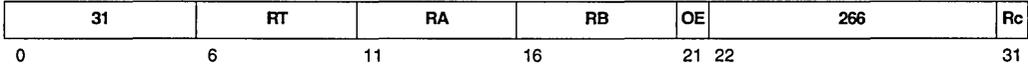
24.5 Alphabetical Instruction Listing

The following pages list the instructions available in the PPC405GP in alphabetical order.

add

Add

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



$$(RT) \leftarrow (RA) + (RB)$$

The sum of the contents of register RA and the contents of register RB is placed into register RT.

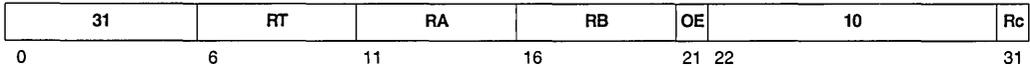
Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

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[CRO]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

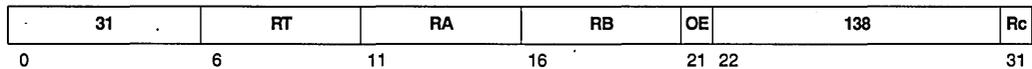
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

adde

Add Extended

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



```
(RT) ← (RA) + (RB) + XER[CA]
if (RA) + (RB) + XER[CA]  $\geq$   $2^{32} - 1$  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.

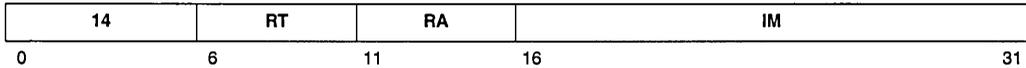
Registers Altered

- RT
- XER[CA]
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

addi RT, RA, IM



$$(RT) \leftarrow (RAI0) + EXTS(IM)$$

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.

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

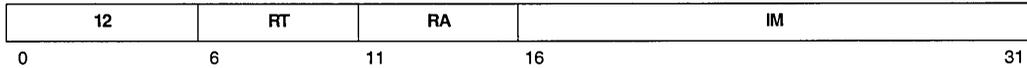
Table 24-3. Extended Mnemonics for addi

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) <i>Extended mnemonic for addi RT,RA,D</i>	
li	RT, IM	Load immediate. (RT) ← EXTS(IM) <i>Extended mnemonic for addi RT,0,IM</i>	
subi	RT, RA, IM	Subtract EXTS(IM) from (RAI0). Place result in RT. <i>Extended mnemonic for addi RT,RA,-IM</i>	

addic

Add Immediate Carrying

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]

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

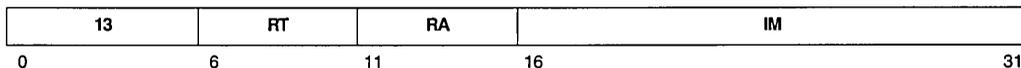
Table 24-4. Extended Mnemonics for addic

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>	

addic.

Add Immediate Carrying and Record

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]
- CR[CR0]_{LT, GT, EQ, SO}

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.**

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

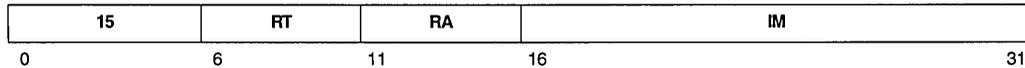
Table 24-5. Extended Mnemonics for addic.

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 addic. RT,RA,-IM</i>	CR[CR0]

addis

Add Immediate Shifted

addis RT, RA, IM



$$(RT) \leftarrow (RAI0) + (IM \parallel 160)$$

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
```

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

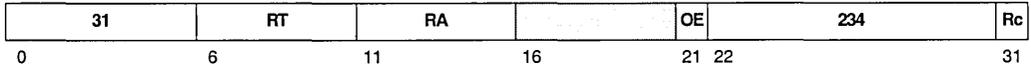
Table 24-6. Extended Mnemonics for addis

Mnemonic	Operands	Function	Other Registers Altered
lis	RT, IM	Load immediate shifted. $(RT) \leftarrow (IM \parallel 160)$ <i>Extended mnemonic for addis RT,0,IM</i>	
subis	RT, RA, IM	Subtract $(IM \parallel 160)$ from (RAI0). Place result in RT. <i>Extended mnemonic for addis RT,RA,-IM</i>	

addme

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



$(RT) \leftarrow (RA) + XER[CA] + (-1)$
if $(RA) + XER[CA] + 0xFFFF FFFF \geq 2^{32} - 1$ then
 $XER[CA] \leftarrow 1$
else
 $XER[CA] \leftarrow 0$

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]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Invalid Instruction Forms

- Reserved fields

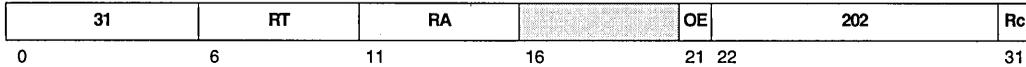
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

addze

Add to Zero Extended

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]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

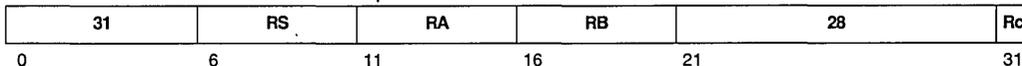
Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

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.

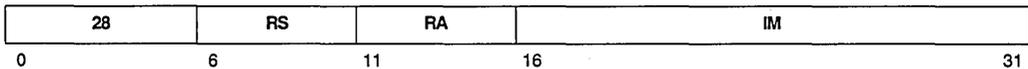
Registers Altered

- RA
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

andi. RA, RS, IM



$$(RA) \leftarrow (RS) \wedge (160 \parallel IM)$$

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]_{LT, GT, EQ, SO}

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..**

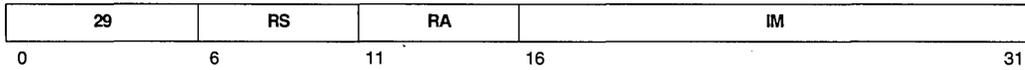
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

andis.

AND Immediate Shifted

andis. RA, RS, IM



$$(RA) \leftarrow (RS) \wedge (IM \parallel 160)$$

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]_{LT, GT, EQ, SO}

Programming Note

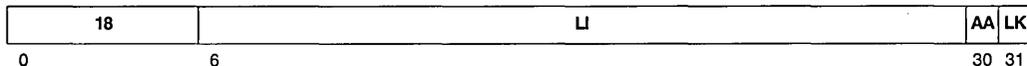
The **andis.** instruction can test whether any of the 16 most-significant bits in a GPR are 1-bits.

andis. is one of three instructions that implicitly update CR[CR0] without having an Rc field. The other instructions are **addic.** and **andi.**

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

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. 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 also the current instruction address (CIA). If the AA field contains 1, the base address is 0.

Program flow is transferred to the NIA.

If the LK field contains 1, then (CIA + 4) is placed into the LR.

Registers Altered

- LR if LK contains 1

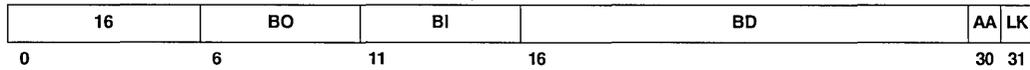
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

bc

Branch Conditional

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



```
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  $\leftarrow$  target16:29
    NIA  $\leftarrow$  EXTS(BD || 20)
  else
    BD  $\leftarrow$  (target - CIA)16:29
    NIA  $\leftarrow$  CIA + EXTS(BD || 20)
  else
    NIA  $\leftarrow$  CIA + 4
  if LK = 1 then
    (LR)  $\leftarrow$  CIA + 4
  PC  $\leftarrow$  NIA
```

If bit 2 of the BO field contains 0, the CTR decrements.

The BI field specifies a bit in the CR to be used as the condition of the branch.

The next instruction address (NIA) is the effective address of the branch. 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 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 also the current instruction address (CIA). If the AA field contains 1, the base address is 0.

The BO field controls options that determine when program flow is transferred to the NIA. The BO field also controls branch prediction, a performance-improvement feature. See "Branch Prediction" on page 3-36 for a complete discussion.

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

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-7. 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.

bc

Branch Conditional

Table 24-7. Extended Mnemonics for bc, bca, bcl, bcla (continued)

Mnemonic	Operands	Function	Other Registers Altered
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) ← CIA + 4.
bdzfla		<i>Extended mnemonic for</i> bcla 2,cr_bit,target	(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	
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

Table 24-7. Extended Mnemonics for bc, bca, bcl, bcla (continued)

Mnemonic	Operands	Function	Other Registers Altered
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
bgea		<i>Extended mnemonic for</i> bcla 4,4*cr_field+0,target	LR
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	
bgt		<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	
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
blea		<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	
blt		<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.

bc

Branch Conditional

Table 24-7. Extended Mnemonics for bc, bca, bcl, bcla (continued)

Mnemonic	Operands	Function	Other Registers Altered
bne	[cr_field,] target	Branch if not equal. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bc 4,4*cr_field+2,target</i>	
bnea		<i>Extended mnemonic for bca 4,4*cr_field+2,target</i>	
bnel		<i>Extended mnemonic for bcl 4,4*cr_field+2,target</i>	(LR) ← CIA + 4.
bnela		<i>Extended mnemonic for bcla 4,4*cr_field+2,target</i>	(LR) ← CIA + 4.
bng	[cr_field,] target	Branch if not greater than. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bc 4,4*cr_field+1,target</i>	
bnga		<i>Extended mnemonic for bca 4,4*cr_field+1,target</i>	
bngl		<i>Extended mnemonic for bcl 4,4*cr_field+1,target</i>	(LR) ← CIA + 4.
bngla		<i>Extended mnemonic for bcla 4,4*cr_field+1,target</i>	(LR) ← CIA + 4.
bnl	[cr_field,] target	Branch if not less than; use CR0 if cr_field is omitted. <i>Extended mnemonic for bc 4,4*cr_field+0,target</i>	
bnla		<i>Extended mnemonic for bca 4,4*cr_field+0,target</i>	
bnll		<i>Extended mnemonic for bcl 4,4*cr_field+0,target</i>	(LR) ← CIA + 4.
bnlla		<i>Extended mnemonic for bcla 4,4*cr_field+0,target</i>	(LR) ← CIA + 4.
bns	[cr_field,] target	Branch if not summary overflow. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bc 4,4*cr_field+3,target</i>	
bnsa		<i>Extended mnemonic for bca 4,4*cr_field+3,target</i>	
bnsi		<i>Extended mnemonic for bcl 4,4*cr_field+3,target</i>	(LR) ← CIA + 4.
bnsia		<i>Extended mnemonic for bcla 4,4*cr_field+3,target</i>	(LR) ← CIA + 4.

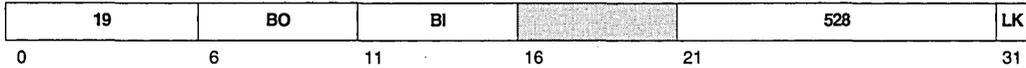
Table 24-7. Extended Mnemonics for bc, bca, bcl, bcla (continued)

Mnemonic	Operands	Function	Other Registers Altered
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	
bnua		<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.
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	
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.
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 BO2 = 0 then
    CTR ← CTR - 1
if (BO2 = 1 ∨ ((CTR = 0) = BO3)) ∧ (BO0 = 1 ∨ (CRBI = BO1)) then
    NIA ← CTR0:29 || 200
else
    NIA ← CIA + 4
if LK = 1 then
    (LR) ← CIA + 4
PC ← NIA
```

The BI field specifies a bit in the CR to be used as the condition of the branch.

The next instruction address (NIA) is the target address of the branch. The NIA is formed by concatenating the 30 most significant bits of the CTR with two 0-bits on the right.

The BO field controls options that determine when program flow is transferred to the NIA. The BO field also controls branch prediction, a performance-improvement feature. See “Branch Prediction” on page 3-36 for a complete discussion.

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
- If bit 2 of the BO field contains 0, the instruction form is invalid, but the pseudocode applies. If the branch condition is true, the branch is taken; the NIA is the contents of the CTR after it is decremented.

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-8. Extended Mnemonics for bcctr, bcctrl

Mnemonic	Operands	Function	Other Registers Altered
bctr		Branch unconditionally to address in CTR. <i>Extended mnemonic for bcctr 20,0</i>	
bcctrl		<i>Extended mnemonic for bcctrl 20,0</i>	(LR) ← CIA + 4.
beqctr	[cr_field]	Branch, if equal, to address in CTR Use CR0 if cr_field is omitted. <i>Extended mnemonic for bcctr 12,4*cr_field+2</i>	
beqctrl		<i>Extended mnemonic for bcctrl 12,4*cr_field+2</i>	(LR) ← CIA + 4.
bfctr	cr_bit	Branch, if CR _{cr_bit} = 0, to address in CTR. <i>Extended mnemonic for bcctr 4,cr_bit</i>	
bfctrl		<i>Extended mnemonic for bcctrl 4,cr_bit</i>	(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 bcctr 4,4*cr_field+0</i>	
bgectrl		<i>Extended mnemonic for bcctrl 4,4*cr_field+0</i>	(LR) ← CIA + 4.
bgctr	[cr_field]	Branch, if greater than, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bcctr 12,4*cr_field+1</i>	
bgctrl		<i>Extended mnemonic for bcctrl 12,4*cr_field+1</i>	(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 bcctr 4,4*cr_field+1</i>	
blectrl		<i>Extended mnemonic for bcctrl 4,4*cr_field+1</i>	(LR) ← CIA + 4.
blctr	[cr_field]	Branch, if less than, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bcctr 12,4*cr_field+0</i>	
blctrl		<i>Extended mnemonic for bcctrl 12,4*cr_field+0</i>	(LR) ← CIA + 4.

bcctr

Branch Conditional to Count Register

Table 24-8. Extended Mnemonics for bcctr, bcctrl (continued)

Mnemonic	Operands	Function	Other Registers Altered
bnctr	[cr_field]	Branch, if not equal, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bcctr 4,4*cr_field+2</i>	
bnctrl		<i>Extended mnemonic for bcctrl 4,4*cr_field+2</i>	(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 bcctr 4,4*cr_field+1</i>	
bngctrl		<i>Extended mnemonic for bcctrl 4,4*cr_field+1</i>	(LR) ← CIA + 4.
bnlctr	[cr_field]	Branch, if not less than, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bcctr 4,4*cr_field+0</i>	
bnlctrl		<i>Extended mnemonic for bcctrl 4,4*cr_field+0</i>	(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 bcctr 4,4*cr_field+3</i>	
bnsctrl		<i>Extended mnemonic for bcctrl 4,4*cr_field+3</i>	(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 bcctr 4,4*cr_field+3</i>	
bnuctrl		<i>Extended mnemonic for bcctrl 4,4*cr_field+3</i>	(LR) ← CIA + 4.
bsoctr	[cr_field]	Branch, if summary overflow, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bcctr 12,4*cr_field+3</i>	
bsoctrl		<i>Extended mnemonic for bcctrl 12,4*cr_field+3</i>	(LR) ← CIA + 4.
btctr	cr_bit	Branch if CR _{cr_bit} = 1 to address in CTR. <i>Extended mnemonic for bcctr 12,cr_bit</i>	
btctrl		<i>Extended mnemonic for bcctrl 12,cr_bit</i>	(LR) ← CIA + 4.

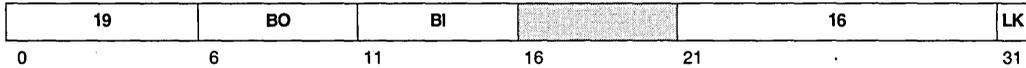
Table 24-8. Extended Mnemonics for bcctr, bcctrl (continued)

Mnemonic	Operands	Function	Other Registers Altered
buncctr	[cr_field]	Branch if unordered to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bcctr 12,4*cr_field+3</i>	
buncctrl		<i>Extended mnemonic for bcctrl 12,4*cr_field+3</i>	(LR) ← CIA + 4.

bclr

Branch Conditional to Link Register

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 bit 2 of the BO field contains 0, the CTR is decremented.

The BI field specifies a bit in the CR to be used as the condition of the branch.

The next instruction address (NIA) is the target address of the branch. The NIA is formed by concatenating the 30 most significant bits of the LR with two 0-bits on the right.

The BO field controls options that determine when program flow is transferred to the NIA. The BO field also controls branch prediction, a performance-improvement feature. See “Branch Prediction” on page 3-36 for a complete discussion.

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

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-9. Extended Mnemonics for bclr, bclrl

Mnemonic	Operands	Function	Other Registers Altered
blr		Branch unconditionally to address in LR. <i>Extended mnemonic for bclr 20,0</i>	
bclrl		<i>Extended mnemonic for bclrl 20,0</i>	(LR) ← CIA + 4.

Table 24-9. Extended Mnemonics for bclr, bclrl (continued)

Mnemonic	Operands	Function	Other Registers Altered
bdnzlr		Decrement CTR. Branch if CTR \neq 0 to address in LR. <i>Extended mnemonic for bclr 16,0</i>	
bdnzlrl		<i>Extended mnemonic for bclrl 16,0</i>	(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 bclr 0,cr_bit</i>	
bdnzflrl		<i>Extended mnemonic for bclrl 0,cr_bit</i>	(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 bclr 8,cr_bit</i>	
bdnztlrl		<i>Extended mnemonic for bclrl 8,cr_bit</i>	(LR) \leftarrow CIA + 4.
bdzlr		Decrement CTR. Branch if CTR = 0 to address in LR. <i>Extended mnemonic for bclr 18,0</i>	
bdzlrl		<i>Extended mnemonic for bclrl 18,0</i>	(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 bclr 2,cr_bit</i>	
bdzflrl		<i>Extended mnemonic for bclrl 2,cr_bit</i>	(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 bclr 10,cr_bit</i>	
bdztlrl		<i>Extended mnemonic for bclrl 10,cr_bit</i>	(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 bclr 12,4*cr_field+2</i>	
beqlrl		<i>Extended mnemonic for bclrl 12,4*cr_field+2</i>	(LR) \leftarrow CIA + 4.

bclr

Branch Conditional to Link Register

Table 24-9. Extended Mnemonics for bclr, bclrl (continued)

Mnemonic	Operands	Function	Other Registers Altered
bflr	cr_bit	Branch if CR _{cr_bit} = 0 to address in LR. <i>Extended mnemonic for bclr 4,cr_bit</i>	
bflrl		<i>Extended mnemonic for bclrl 4,cr_bit</i>	(LR) ← CIA + 4.
bgelr	[cr_field]	Branch, if greater than or equal, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bclr 4,4*cr_field+0</i>	
bgelrl		<i>Extended mnemonic for bclrl 4,4*cr_field+0</i>	(LR) ← CIA + 4.
bgtlr	[cr_field]	Branch, if greater than, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bclr 12,4*cr_field+1</i>	
bgtlrl		<i>Extended mnemonic for bclrl 12,4*cr_field+1</i>	(LR) ← CIA + 4.
blelr	[cr_field]	Branch, if less than or equal, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bclr 4,4*cr_field+1</i>	
blelrl		<i>Extended mnemonic for bclrl 4,4*cr_field+1</i>	(LR) ← CIA + 4.
bltlr	[cr_field]	Branch, if less than, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bclr 12,4*cr_field+0</i>	
bltlrl		<i>Extended mnemonic for bclrl 12,4*cr_field+0</i>	(LR) ← CIA + 4.
bnelr	[cr_field]	Branch, if not equal, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bclr 4,4*cr_field+2</i>	
bnelrl		<i>Extended mnemonic for bclrl 4,4*cr_field+2</i>	(LR) ← CIA + 4.
bnglr	[cr_field]	Branch, if not greater than, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bclr 4,4*cr_field+1</i>	
bnglrl		<i>Extended mnemonic for bclrl 4,4*cr_field+1</i>	(LR) ← CIA + 4.

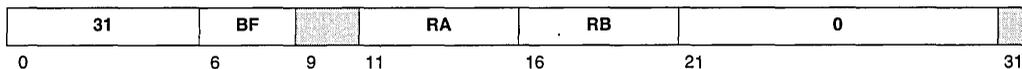
Table 24-9. Extended Mnemonics for bclr, bclrl (continued)

Mnemonic	Operands	Function	Other Registers Altered
bnlr	[cr_field]	Branch, if not less than, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bclr 4,4*cr_field+0</i>	
bnlrl		<i>Extended mnemonic for bclrl 4,4*cr_field+0</i>	(LR) ← CIA + 4.
bnsr	[cr_field]	Branch if not summary overflow to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bclr 4,4*cr_field+3</i>	
bnsrl		<i>Extended mnemonic for bclrl 4,4*cr_field+3</i>	(LR) ← CIA + 4.
bnulr	[cr_field]	Branch if not unordered to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bclr 4,4*cr_field+3</i>	
bnulrl		<i>Extended mnemonic for bclrl 4,4*cr_field+3</i>	(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 bclr 12,4*cr_field+3</i>	
bsolrl		<i>Extended mnemonic for bclrl 12,4*cr_field+3</i>	(LR) ← CIA + 4.
btlr	cr_bit	Branch if CR _{cr_bit} = 1 to address in LR. <i>Extended mnemonic for bclr 12,cr_bit</i>	
btlrl		<i>Extended mnemonic for bclrl 12,cr_bit</i>	(LR) ← CIA + 4.
bunlr	[cr_field]	Branch if unordered to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bclr 12,4*cr_field+3</i>	
bunlrl		<i>Extended mnemonic for bclrl 12,4*cr_field+3</i>	(LR) ← CIA + 4.

cmp

Compare

cmp BF, 0, RA, RB



```
c0:3 ← '0
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

The PowerPC Architecture defines this instruction as **cmp BF,L,RA,RB**, where L selects operand size for 64-bit PowerPC implementations. For all 32-bit PowerPC implementations, L = 0 is required (L = 1 is an invalid form); hence for PPC405GP, use of the extended mnemonic **cmpw BF,RA,RB** is recommended.

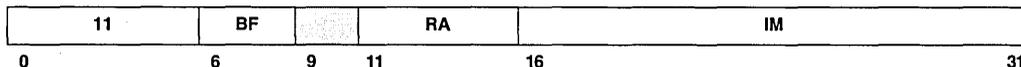
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-10. 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 cmp BF,0,RA,RB</i>	

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[CRn] where n is specified by the BF field

Invalid Instruction Forms

- Reserved fields

Programming Note

The PowerPC Architecture defines this instruction as **cmpi BF,L,RA,IM**, where L selects operand size for 64-bit PowerPC implementations. For all 32-bit PowerPC implementations, L = 0 is required (L = 1 is an invalid form); hence for the PPC405GP, use of the extended mnemonic **cmpwi BF,RA,IM** is recommended.

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

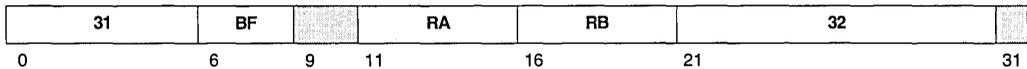
Table 24-11. Extended Mnemonics for cmpi

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

Compare Logical

cmpl 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 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

The PowerPC Architecture defines this instruction as **cmpl BF,L,RA,RB**, where L selects operand size for 64-bit PowerPC implementations. For all 32-bit PowerPC implementations, L = 0 is required (L = 1 is an invalid form); hence for PPC405GP, use of the extended mnemonic **cmplw BF,RA,RB** is recommended.

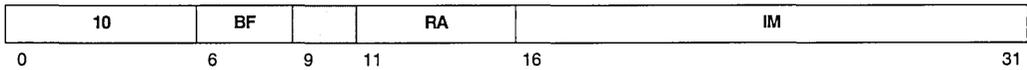
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-12. 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 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) = u (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[CRn] where n is specified by the BF field

Invalid Instruction Forms

- Reserved fields

Programming Note

The PowerPC Architecture defines this instruction as **cmpli BF,L,RA,IM**, where L selects operand size for 64-bit PowerPC implementations. For all 32-bit PowerPC implementations, L = 0 is required (L = 1 is an invalid form); hence for the PPC405GP, use of the extended mnemonic **cmplwi BF,RA,IM** is recommended.

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

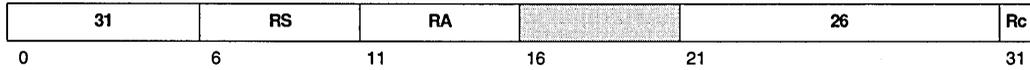
Table 24-13. Extended Mnemonics for cmpli

Mnemonic	Operands	Function	Other Registers Changed
cmplwi	[BF,] RA, IM	Compare Logical Word Immediate. Use CR0 if BF is omitted. <i>Extended mnemonic for cmpli BF,0,RA,IM</i>	

cntlzw

Count Leading Zeros Word

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]_{LT, GT, EQ, SO} if Rc contains 1

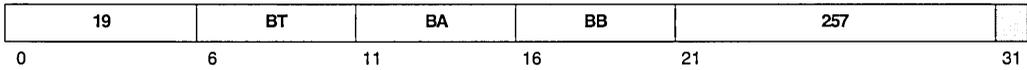
Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

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.

Registers Altered

- CR

Invalid Instruction Forms

- Reserved fields

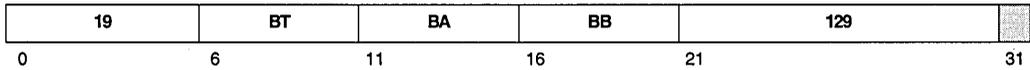
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

crandc

Condition Register AND with Complement

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.

Registers Altered

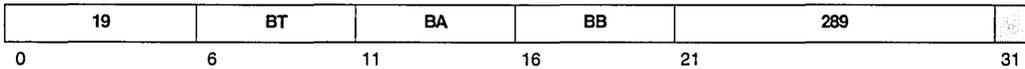
- CR

Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

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.

Registers Altered

- CR

Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

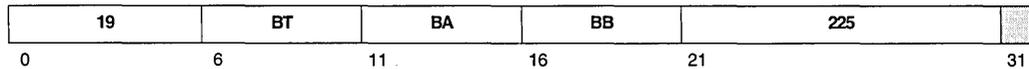
Table 24-14. Extended Mnemonics for creqv

Mnemonic	Operands	Function	Other Registers Altered
crset	bx	CR set. <i>Extended mnemonic for creqv bx,bx,bx</i>	

crnand

Condition Register NAND

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.

Registers Altered

- CR

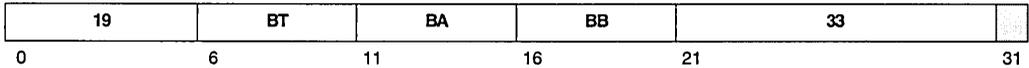
Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

crnor 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.

Registers Altered

- CR

Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

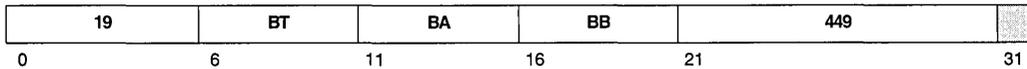
Table 24-15. Extended Mnemonics for crnor

Mnemonic	Operands	Function	Other Registers Altered
crnot	bx, by	CR not. <i>Extended mnemonic for crnor bx,by,by</i>	

cror

Condition Register OR

cror 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.

Registers Altered

- CR

Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

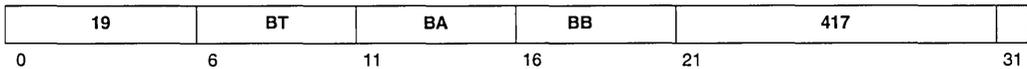
Table 24-16. Extended Mnemonics for cror

Mnemonic	Operands	Function	Other Registers Altered
crmove	bx, by	CR move. <i>Extended mnemonic for cror bx,by,by</i>	

crorc

Condition Register OR with Complement

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.

Registers Altered

- CR

Invalid Instruction Forms

- Reserved fields

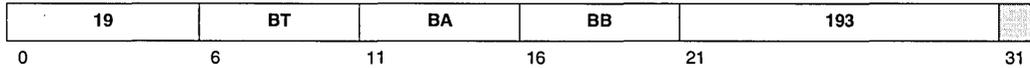
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

crxor

Condition Register XOR

crxor BT, 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.

Registers Altered

- CR

Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-17. Extended Mnemonics for crxor

Mnemonic	Operands	Function	Other Registers Altered
crclr	bx	Condition register clear. <i>Extended mnemonic for crxor bx,bx,bx</i>	

dcba RA, RB



$$EA \leftarrow (RA \ll 10) + (RB)$$

$$DCBA(EA)$$

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 EA is marked as cachable and non-write-through, the data in the cache block is architecturally undefined. For the PPC405GP, the cache data block is set to 0.

If the data block at the EA is not in the data cache and the EA is marked as cachable and not marked as write-through, a cache block is established and set to an architecturally-undefined value. Note that no data is read from main storage, as described in the programming note.

If the data block at the EA is marked as non-cachable, a no-op occurs.

If the data block at the EA is in the data cache and marked as write-through, architecturally the data in the cache block can be left unmodified. Alternatively, the data block at the EA can be undefined in the data cache and in main storage. For the PPC405GP, a no-op occurs.

If the data block at the EA is not in the data cache and marked as write-through, architecturally the instruction can establish a cache block and set the block to 0, or a no-op can occur. For the PPC405GP, a no-op 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 **dcba** can establish an address in the data cache without copying the contents of that address from main storage, the address established can be invalid with respect to main storage. 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 machine check exception could occur under these circumstances.

dcba provides a hint that a block of storage will soon be stored to or no longer needed; there is no need to retain the data in the block. Establishing the line in the cache, without reading from main storage, improves performance.

dcba

Data Cache Block Allocate

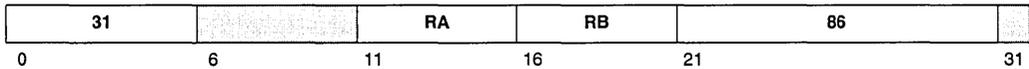
Exceptions

This instruction is considered a “store” with respect to data storage exceptions. However, this instruction does not cause data storage exceptions or data TLB-miss exceptions. If conditions occur that would otherwise cause such exceptions, **dcba** is treated as a no-op.

This instruction is considered a “store” with respect to data address compare (DAC) debug exceptions. See “Data Storage Interrupt” on page 10-36.

Architecture Note

This instruction is part of the IBM PowerPC Embedded Virtual Environment.

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 EA is marked as cachable.

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 10-36.

This instruction is considered a “store” with respect to data address compare (DAC) debug exceptions. See “Debug Interrupt” on page 10-44.

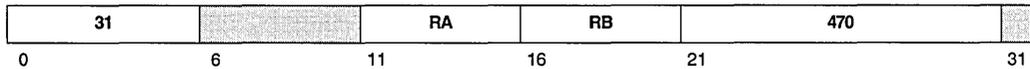
Architecture Note

This instruction is part of the IBM PowerPC Embedded Virtual Environment.

dcbi

Data Cache Block Invalidate

dcbi RA, RB



$EA \leftarrow (RA10) + (RB)$
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 EA is marked as cachable. 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 10-36.

This instruction is considered a “store” with respect to data address compare (DAC) debug exceptions. See “Debug Interrupt” on page 10-44.

Architecture Note

This instruction is part of the IBM PowerPC Embedded Operating Environment.

dcbst RA, RB



$$EA \leftarrow (RA10) + (RB)$$

$$DCBST(EA)$$

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 EA is marked as cachable.

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 10-36.

This instruction is considered a “store” with respect to data address compare (DAC) debug exceptions. See “Debug Interrupt” on page 10-44.

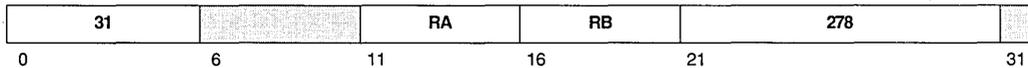
Architecture Note

This instruction is part of the IBM PowerPC Embedded Virtual Environment.

dcbt

Data Cache Block Touch

dcbt RA, RB



$EA \leftarrow (RA \neq 0) + (RB)$
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 EA is marked as cachable, 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 EA is marked as non-cachable, no operation is performed.

This instruction is not allowed to cause data storage exceptions or data TLB miss exceptions. If execution of the instruction would cause such an exception, then no operation is performed, and no 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

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.

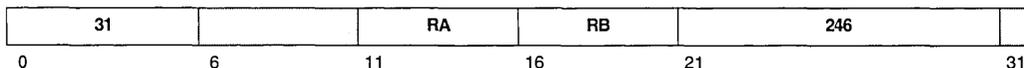
Exceptions

This instruction is considered a “load” with respect to data storage exceptions. See “Data Storage Interrupt” on page 10-36.

This instruction is considered a “load” with respect to data address compare (DAC) debug exceptions. See “Debug Interrupt” on page 10-44.

Architecture Note

This instruction is part of the IBM PowerPC Embedded Virtual Environment.

dcbtst RA, RB

$$EA \leftarrow (RA \neq 0) + (RB)$$

$$DCBTST(EA)$$

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 EA address is marked as cachable, the data block is loaded into the data cache.

If the EA is marked as non-cachable, or if the data block at the EA is in the data cache, no operation is performed.

This instruction is not allowed to cause data storage exceptions or data TLB miss exceptions. If execution of the instruction would cause such an exception, then no operation is performed, and no 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

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** brings data into the cache in “Exclusive” mode, which allows the program to alter the cached data. “Exclusive” mode is part of the MESI protocol for multi-processor systems, and is not implemented. 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 10-36.

This instruction is considered a “load” with respect to data address compare (DAC) debug exceptions. See “Debug Interrupt” on page 10-44.

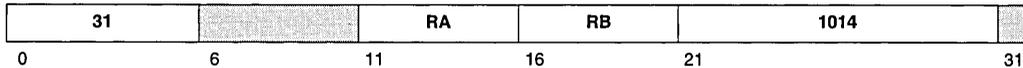
Architecture Note

This instruction is part of the IBM PowerPC Embedded Virtual Environment.

dcbz

Data Cache Block Set to Zero

dcbz RA, RB



$EA \leftarrow (RA \ll 10) + (RB)$
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 EA is marked as cachable and non-write-through, the data in the cache block is set to 0.

If the data block at the EA is not in the data cache and the EA is marked as cachable and non-write-through, a cache block is established and set to 0. Note that nothing is read from main storage, as described in the programming note.

If the data block at the EA is marked as either write-through or as non-cachable, 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 machine check exception could occur under these circumstances.

If **dcbz** is attempted to an EA which is marked as non-cachable, the software alignment exception handler should emulate the instruction by storing zeros to the block in main storage. If a data block corresponding to the EA exists in the cache, but the EA is non-cachable, stores (including **dcbz**) to that address are considered programming errors (the cache block should previously have been flushed).

If the EA is marked as write-through, the software alignment exception handler should emulate the instruction by storing zeros to the block in main storage. An EA that is marked as write-through required should also be marked as cachable; when **dcbz** is attempted to such an address, the alignment exception handler should maintain coherency of cache and memory.

Exceptions

An alignment exception occurs if the EA is marked as non-cachable or as write-through.

This instruction is considered a “store” with respect to data storage exceptions. See “Data Storage Interrupt” on page 10-36.

This instruction is considered a “store” with respect to data address compare (DAC) debug exceptions. See “Debug Interrupt” on page 10-44.

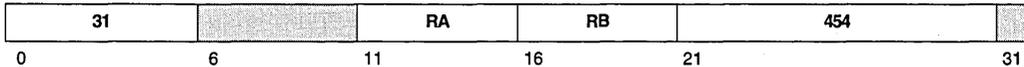
Architecture Note

This instruction is part of the IBM PowerPC Embedded Virtual Environment.

dccci

Data Cache Congruence Class Invalidate

dccci RA, RB



$EA \leftarrow (RA \ll 0) + (RB)$
DCCCI(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.

Both cache lines in the congruence class specified by $EA_{20:26}$ are invalidated, whether or not they match the EA. If modified data existed in the cache congruence class before the operation of this instruction, that data is lost.

The operation specified by this instruction is performed whether or not the EA is marked as cachable.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Note

Execution of this instruction is privileged.

This instruction is intended for use in the power-on reset routine to invalidate the entire data cache tag array before enabling the data cache. A series of **dccc** instruction should be executed, one for each congruence class. Cachability can then be enabled.

Exceptions

See “Access Protection for Cache Control Instructions” on page 6-15.

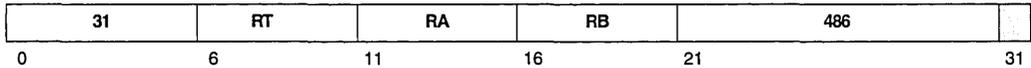
The execution of an **dccc** instruction can cause a data TLB miss exception, at the specified EA, regardless of the non-specific intent of that EA.

This instruction does not cause data address compare (DAC) debug exceptions. See “Debug Interrupt” on page 10-44.

Architecture Note

This instruction is implementation-specific and may not be portable to other implementations.

dcread RT, RA, RB



$EA \leftarrow (RA \ll 0) + (RB)$
 if $((CCR0[CIS] = 0) \wedge (CCR0[CWS] = 0))$ then $(RT) \leftarrow$ (d-cache data, way A)
 if $((CCR0[CIS] = 0) \wedge (CCR0[CWS] = 1))$ then $(RT) \leftarrow$ (d-cache data, way B)
 if $((CCR0[CIS] = 1) \wedge (CCR0[CWS] = 0))$ then $(RT) \leftarrow$ (d-cache tag, way A)
 if $((CCR0[CIS] = 1) \wedge (CCR0[CWS] = 1))$ then $(RT) \leftarrow$ (d-cache tag, way B)

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.

This instruction is a debugging tool for reading the data cache entries for the congruence class specified by $EA_{19:26}$, where m is the number of bits in the cache tag. The cache information is read into register RT.

If $CCR0[CIS] = 0$, the information is a word of data cache array data from the addressed congruence class. The word is specified by $EA_{27:29}$ ($EA_{0:19}$ and $EA_{30:31}$ are ignored). If $EA_{30:31}$ are not 00, an alignment exception occurs. If $CCR0[CWS] = 0$, the data is from the A-way; otherwise, the data is from the B-way.

If $CCR0[CIS] = 1$, the information is a cache tag from the addressed congruence class ($EA_{0:19}$ and $EA_{30:31}$ are ignored). If $CCR0[CWS] = 0$, the tag is from the A-way; otherwise the tag is from the B-way.

Data cache tag information is placed into register RT as shown:

0:19	TAG	Cache Tag
20:25		Reserved
26	D	Cache Line Dirty 0 Not dirty 1 Dirty
27	V	Cache Line Valid 0 Not valid 1 Valid
28:30		Reserved
31	LRU	Least Recently Used (LRU) 0 A-way LRU 1 B-way LRU

If instruction bit 31 contains 1, the contents of $CR[CR0]$ are undefined.

Registers Altered

- RT

dcread

Data Cache Read

Invalid Instruction Forms

- Reserved fields

Programming Note

Execution of this instruction is privileged.

Exceptions

If EA is not word-aligned, an alignment exception occurs.

This instruction is considered a “load” with respect to data storage exceptions, but cannot cause a data storage exception. See “Access Protection for Cache Control Instructions” on page 6-15.

The execution of an **dcread** instruction can cause a data TLB miss exception, at the specified EA, regardless of the non-specific intent of that effective address.

This instruction is considered a “load” with respect to data address compare (DAC) debug exceptions. See “Debug Interrupt” on page 10-44.

Architecture Note

This instruction is implementation-specific and may not be portable to other implementations.

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]_{LT, GT, EQ} are undefined. Either invalid division operation sets XER[OV, SO] to 1 if the OE field contains 1.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} 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.

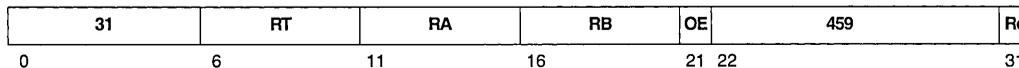
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

divwu

Divide Word Unsigned

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



$$(RT) \leftarrow (RA) \div (RB)$$

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]_{LT, GT, EQ} are also undefined. The invalid division operation also sets XER[OV, SO] to 1 if the OE field contains 1.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[OV, SO] if OE contains 1

Programming Note

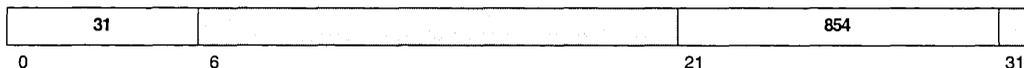
The 32-bit remainder can be calculated using the following sequence of instructions

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 zero.

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

eieio

The **eieio** instruction ensures that all loads and stores preceding **eieio** complete with respect to main storage before any loads and stores following **eieio** access main storage.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Note

Architecturally, **eieio** orders storage access, not instruction completion. Therefore, non-storage operations after **eieio** could complete before storage operations that were before **eieio**. The **sync** instruction guarantees ordering of both instruction completion and storage access. For the PPC405GP, the **eieio** instruction is implemented to behave as a **sync** instruction.

To write code that is portable between various PowerPC implementations, programmers should use the mnemonic that corresponds to the desired behavior.

Architecture Note

This instruction is part of the IBM PowerPC Embedded Virtual Environment.

icbi RA, RB



$$EA \leftarrow (RA)0 + (RB)$$

$$ICBI(EA)$$

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 cachable in the ICCR.

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 operations use MSR[DR], not MSR[IR], to determine translation of their operands.

When data translation is disabled, cachability for the EA of the operand of instruction cache operations is determined by the ICCR, not the DCCR.

Exceptions

Instruction storage exceptions and instruction-side TLB miss exceptions are associated with instruction *fetching*, not with instruction execution. Exceptions that occur during the *execution* of instruction cache operations cause data-side exceptions (data storage exceptions and data TLB miss exceptions).

This instruction is considered a “load” with respect to data storage exceptions. See “Data Storage Interrupt” on page 10-36.

This instruction is considered a “load” with respect to data address compare (DAC) debug exceptions.

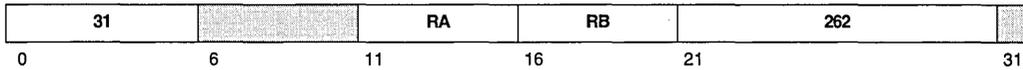
Architecture Note

This instruction is part of the IBM PowerPC Embedded Virtual Environment.

icbt

Instruction Cache Block Touch

icbt RA, RB



$$EA \leftarrow (RA \neq 0) + (RB)$$
$$ICBT(EA)$$

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 is marked as cachable, the instruction block is loaded into the instruction cache.

If the instruction block at the EA is in the instruction cache, or if the EA is marked as non-cachable, 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

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 operations use MSR[DR], not MSR[IR], to determine translation of their operands. When data translation is disabled, cachability for the effective address of the operand of instruction cache operations is determined by the ICCR, not the DCCR.

Exceptions

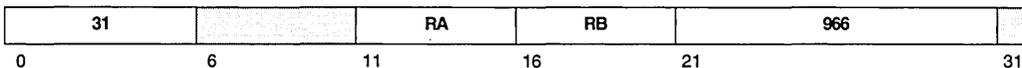
Instruction storage exceptions and instruction-side TLB miss exceptions are associated with instruction *fetching*, not with instruction execution. Exceptions occurring during *execution* of instruction cache operations cause data storage and data TLB miss exceptions.

If the execution of an **icbt** instruction would cause a data TLB miss exception, no operation is performed and no exception occurs.

This instruction is considered a "load" with respect to protection exceptions, but cannot cause data storage exceptions. This instruction is also considered a "load" with respect to data address compare (DAC) debug exceptions.

Architecture Note

This instruction is part of the IBM PowerPC Embedded Operating Environment.



EA \leftarrow (RAI0) + (RB)
 ICCCI(ICU cache array)

This instruction invalidates the entire ICU cache array. The EA is not used; previous implementations have used the EA for protection checks. The instruction form 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 cache tag array before enabling the cache. Cachability can then be enabled.

Architecture Note

This instruction is implementation-specific and may not be portable to other implementations.

icread

Instruction Cache Read

icread RA, RB



$EA \leftarrow (RA \ll 0) + (RB)$

if $((CCR0[CIS] = 0) \wedge (CCR0[CWS] = 0))$ then (ICDBDR) \leftarrow (i-cache data, way A)

if $((CCR0[CIS] = 0) \wedge (CCR0[CWS] = 1))$ then (ICDBDR) \leftarrow (i-cache data, way B)

if $((CCR0[CIS] = 1) \wedge (CCR0[CWS] = 0))$ then (ICDBDR) \leftarrow (i-cache tag, way A)

if $((CCR0[CIS] = 1) \wedge (CCR0[CWS] = 1))$ then (ICDBDR) \leftarrow (i-cache tag, way B)

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.

This instruction is a debugging tool for reading the instruction cache entries for the congruence class specified by $EA_{19:26}$. The cache information is read into the Instruction Cache Debug Data Register (ICDBDR), from where it can be read into a GPR using the extended mnemonic **mficdbdr**.

If $CCR0[CIS] = 0$, the information is a word of instruction cache data from the addressed line. The word is specified by $EA_{27:29}$ ($EA_{0:21}$ and $EA_{30:31}$ are ignored). If $CCR0[CWS] = 0$, the data is from the A-way, otherwise from the B-way.

If $(CCR0[CIS] = 1)$, the information is a cache tag from the addressed congruence class ($EA_{0:21}$ and $EA_{28:31}$ are ignored). If $(CCR0[CWS] = 0)$, the tag is from the A-way, otherwise from the B-way.

Instruction cache tag information is placed in the ICDBDR as shown:

0:21	TAG	Cache Tag
22:26		Reserved
27	V	Cache Line Valid 0 Not valid 1 Valid
28:30		Reserved
31	LRU	Least Recently Used (LRU) 0 A-way LRU 1 B-way LRU

If instruction bit 31 contains 1, the contents of $CR[CR0]$ are undefined.

Registers Altered

- ICDBDR

Invalid Instruction Forms

- Reserved fields

Programming Note

Execution of this instruction is privileged.

The instruction pipeline does not automatically wait for data from **icread** to arrive at the ICDBDR before attempting to use the contents of the ICDBDR. Therefore, insert an **isync** instruction between **icread** and **mficbdr**.

```
icread r5,r6 # read cache information
isync      # ensure completion of icread
mficbdr r7 # move information to GPR
```

Instruction cache operations use MSR[DR], not MSR[IR], to determine translation of their operands. When data translation is disabled, cachability for the EA of the operand of instruction cache operations is determined by the ICCR, not the DCCR.

Exceptions

Instruction storage exceptions and instruction-side TLB miss exceptions are associated with instruction *fetching*, not with instruction execution. Exceptions that occur during the *execution* of instruction cache operations cause data-side exceptions (data storage exceptions and data TLB miss exceptions).

The execution of **icread** can cause a data TLB miss exception, at the specified EA, regardless of the non-specific intent of that EA.

This instruction is considered a “load” and cannot cause a data storage exception.

This instruction is considered a “load” with respect to data address compare (DAC) debug exceptions, but will not cause DAC debug events.

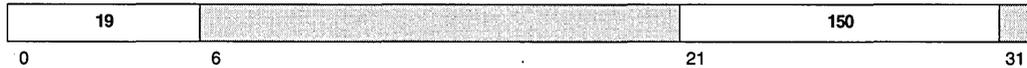
Architecture Note

This instruction is implementation-specific and may not be portable to other implementations.

isync

Instruction Synchronize

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.

No subsequent instructions are initiated by the processor until **isync** completes. Finally, execution of **isync** causes the processor to discard any prefetched instructions, with the effect that subsequent instructions are fetched and executed in the context established by the instructions preceding **isync**.

isync has no effect on caches.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Programming Note

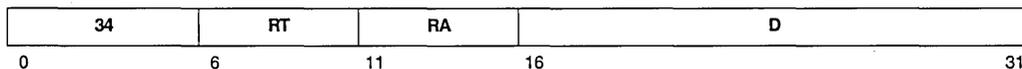
See the discussion of context synchronizing instructions in "Synchronization" on page 3-43.

The following code example illustrates the necessary steps for self-modifying code. This example assumes that `addr1` is both data and instruction cacheable.

```
    stw      regN, addr1      # data in regN is to become an instruction at addr1
    dcbst   addr1            # forces data from the data cache to memory
    sync    # wait until the data actually reaches the memory
    icbi    addr1            # the previous value at addr1 might already be in
                              # the instruction cache; invalidate in the cache
    isync   # the previous value at addr1 might already have been
                              # pre-fetched into the queue; invalidate the queue
                              # so that the instruction must be re-fetched
```

Architecture Note

This instruction is part of the IBM PowerPC Embedded Virtual Environment.

lbz RT, D(RA)
$$EA \leftarrow (RA \ll 0) + \text{EXTS}(D)$$
$$(RT) \leftarrow {}^{24}0 \parallel \text{MS}(EA, 1)$$

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

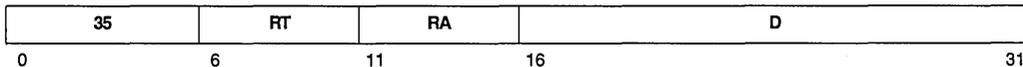
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lbzu

Load Byte and Zero with Update

lbzu RT, D(RA)



$EA \leftarrow (RA \ll 0) + \text{EXTS}(D)$
 $(RA) \leftarrow EA$
 $(RT) \leftarrow 2^0 \parallel \text{MS}(EA, 1)$

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

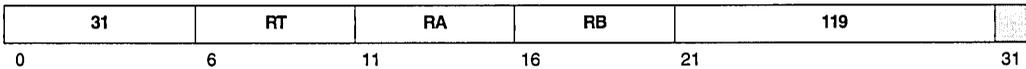
- RA
- RT

Invalid Instruction Forms

- RA=RT
- RA=0

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lbzux RT, RA, RB

$EA \leftarrow (RA \ll 0) + (RB)$
 $(RA) \leftarrow EA$
 $(RT) \leftarrow {}^{24}0 \parallel MS(EA, 1)$

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

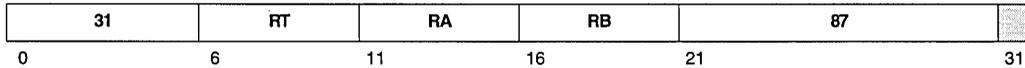
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lbzx

Load Byte and Zero Indexed

lbzx RT,RA, RB



$$EA \leftarrow (RA \neq 0) + (RB)$$
$$(RT) \leftarrow \text{24}0 \parallel \text{MS}(EA, 1)$$

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

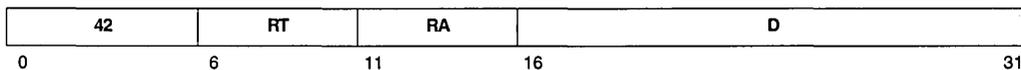
- RT

Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lha RT, D(RA)

$$EA \leftarrow (RA \ll 0) + \text{EXTS}(D)$$
$$(RT) \leftarrow \text{EXTS}(MS(EA, 2))$$

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 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

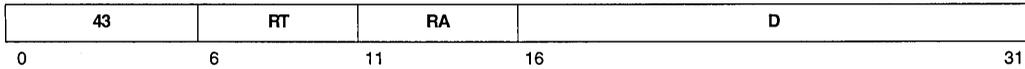
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lhau

Load Halfword Algebraic with Update

lhau RT, D(RA)



$EA \leftarrow (RA \ll 0) + \text{EXTS}(D)$
 $(RA) \leftarrow EA$
 $(RT) \leftarrow \text{EXTS}(\text{MS}(EA, 2))$

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 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

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lhaux RT, RA, RB

$EA \leftarrow (RA \ll 0) + (RB)$
 $(RA) \leftarrow EA$
 $(RT) \leftarrow EXTS(MS(EA, 2))$

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

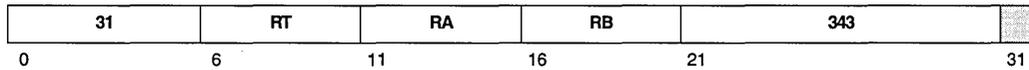
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lhax

Load Halfword Algebraic Indexed

lhax RT, RA, RB



$EA \leftarrow (RA \ll 0) + (RB)$
 $(RT) \leftarrow \text{EXTS}(\text{MS}(EA, 2))$

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 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

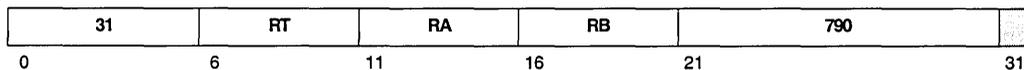
- RT

Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lhrbx RT, RA, RB

$$EA \leftarrow (RA \ll 0) + (RB)$$

$$(RT) \leftarrow {}^{16}0 \parallel MS(EA + 1, 1) \parallel MS(EA, 1)$$

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 halfword at the EA is byte-reversed. 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

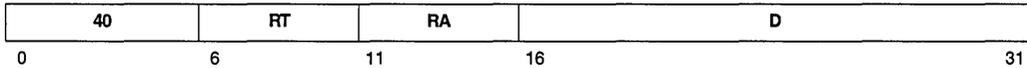
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lhz

Load Halfword and Zero

lhz RT, D(RA)



$$EA \leftarrow (RA \ll 2) + \text{EXTS}(D)$$
$$(RT) \leftarrow 160 \parallel \text{MS}(EA, 2)$$

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 extended to 32 bits by concatenating 16 0-bits to its left. The result is placed into register RT.

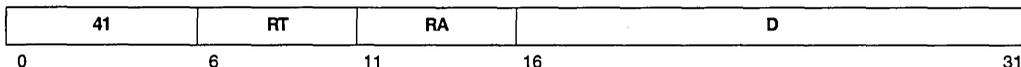
Registers Altered

- RT

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lhzu RT, D(RA)



$$EA \leftarrow (RA \ll 0) + \text{EXTS}(D)$$

$$(RA) \leftarrow EA$$

$$(RT) \leftarrow {}^{16}0 \parallel \text{MS}(EA, 2)$$

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 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

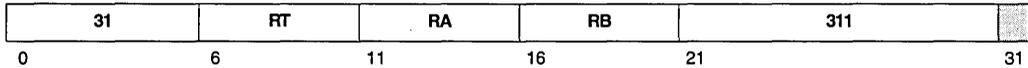
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lhzux

Load Halfword and Zero with Update Indexed

lhzux **RT, RA, RB**



$EA \leftarrow (RA \ll 0) + (RB)$
 $(RA) \leftarrow EA$
 $(RT) \leftarrow {}^{16}0 \parallel MS(EA,2)$

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

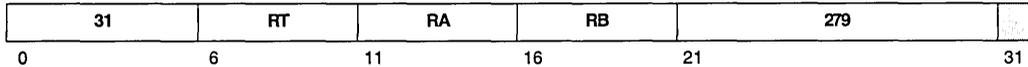
- RA
- RT

Invalid Instruction Forms

- Reserved fields
- RA = RT
- RA = 0

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lhzx RT, RA, RB

$$EA \leftarrow (RA \ll 0) + (RB)$$

$$(RT) \leftarrow {}^{16}0 \parallel MS(EA, 2)$$

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 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

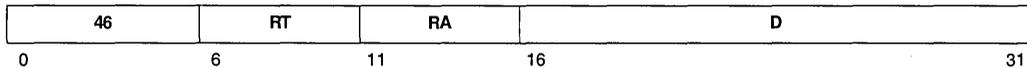
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lww

Load Multiple Word

lww RT, D(RA)



```
EA ← (RA|0) + EXTS(D)
r ← RT
do while r ≤ 31
  if ((r ≠ RA) ∨ (r = 31)) then
    (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). Register RA is not altered by this instruction (unless RA is GPR(31), which is an invalid form of this instruction). The word which would have been placed into register RA is discarded.

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.

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

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
        if ((r ≠ RA) ∨ (r = RFINAL)) then
            (GPR(r)) ← 0
        if ((r ≠ RA) ∨ (r = RFINAL)) then
            (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}. Register RA is not altered (unless RA = R_{FINAL}, an invalid form of this instruction). Bytes which would have been loaded into register RA are discarded.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- RT and subsequent GPRs as described above.

lswi

Load String Word Immediate

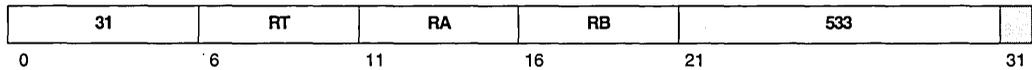
Invalid Instruction Forms

- Reserved fields
- RA is in the range of registers to be loaded
- RA = RT = 0

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lswx RT, RA, RB



```

EA ← (RA[0] + (RB))
CNT ← XER[TBC]
n ← CNT
R_FINAL ← ((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
    if (((r ≠ RA) ∧ (r ≠ RB)) ∨ (r = R_FINAL)) then
      (GPR(r)) ← 0
    if (((r ≠ RA) ∧ (r ≠ RB)) ∨ (r = R_FINAL)) then
      (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 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. Register RA is not altered (unless RA = R_FINAL, which is an invalid form of this instruction). Register RB is not altered (unless RB = R_FINAL, which is an invalid form of this instruction). Bytes which would have been loaded into registers RA or RB are discarded.

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.

lswx

Load String Word Indexed

Invalid Instruction Forms

- Reserved fields
- RA or RB is in the range of registers to be loaded.
- RA = RT = 0

Programming Note

If XER[TBC] = 0, the contents of register RT are unchanged and **lswx** is treated as a no-op.

The PowerPC Architecture states that, if XER[TBC] = 0 and if the EA is such that a precise data exception would normally occur (if not for the zero length), **lswx** is treated as a no-op and the precise exception will not occur. Data storage exceptions and alignment exceptions are examples of precise data exceptions.

However, the PowerPC Architecture makes no statement regarding imprecise exceptions related to **lswx** with XER[TBC] = 0. The PPC405GP generates an imprecise exception (machine check) on this instruction when all of the following conditions are true:

- The instruction passes all protection bounds checking
- The address is cachable
- The address is passed to the data cache
- The address misses in the data cache (resulting in a line fill request)
- The address encounters some form of bus error

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lwarx RT, RA, RB



$EA \leftarrow (RA \ll 0) + (RB)$
 $RESERVE \leftarrow 1$
 $(RT) \leftarrow MS(EA, 4)$

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

lwarx and the **stwcx.** instruction should be 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 asynchronous processes. Only **lwarx** can set the reservation bit to 1. **stwcx.** sets the reservation bit to 0 upon its completion, whether or not **stwcx.** sent (RS) to memory. CR[CR0]_{EQ} 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 register as required
      stwcx. # attempt to store semaphore; reset reservation
      bne loop # an asynchronous process has intervened; try again

```

If the asynchronous process in the code example had paired **lwarx** with a store other than **stwcx.**, the reservation bit would not have been cleared in the asynchronous process, and the code example would have overwritten the semaphore.

Exceptions

An alignment exception occurs if the EA is not word-aligned.

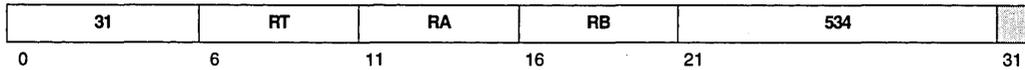
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lwbrx

Load Word Byte-Reverse Indexed

lwbrx RT, RA, RB



$$EA \leftarrow (RA10) + (RB)$$
$$(RT) \leftarrow MS(EA+3,1) \parallel MS(EA+2,1) \parallel MS(EA+1,1) \parallel MS(EA,1)$$

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 word at the EA is byte-reversed: the least significant byte becomes the most significant byte, the next least significant byte becomes the next most significant byte, and so on. The resulting word is placed into register RT.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

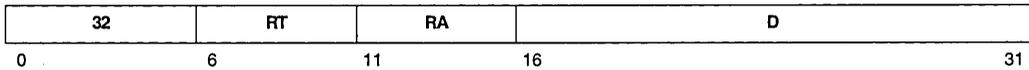
- RT

Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lwz RT, D(RA)
$$EA \leftarrow (RA \ll 0) + \text{EXTS}(D)$$
$$(RT) \leftarrow \text{MS}(EA, 4)$$

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

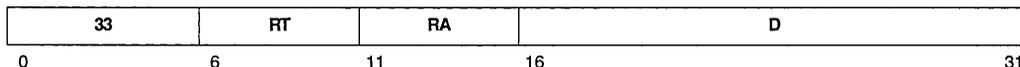
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lwzu

Load Word and Zero with Update

lwzu RT, D(RA)



$EA \leftarrow (RA \ll 0) + \text{EXTS}(D)$
 $(RA) \leftarrow EA$
 $(RT) \leftarrow \text{MS}(EA, 4)$

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

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lwzux RT, RA, RB



$$EA \leftarrow (RA \ll 0) + (RB)$$

$$(RA) \leftarrow EA$$

$$(RT) \leftarrow MS(EA, 4)$$

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

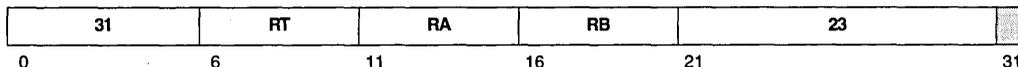
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

lwzx

Load Word and Zero Indexed

lwzx RT, RA, RB



$EA \leftarrow (RA \neq 0) + (RB)$
 $(RT) \leftarrow MS(EA, 4)$

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

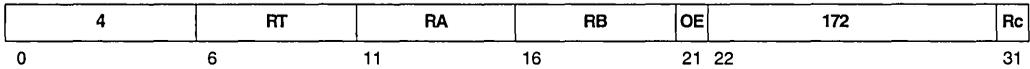
Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

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



$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ signed

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is summed with the contents of RT and the sum is stored in a 33-bit temporary register. The contents of RT are replaced by the low-order 32 bits of the temporary register.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

macchws

Multiply Accumulate Cross Halfword to Word Saturate Signed

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



```
prod0:31 ← (RA)16:31 × (RB)0:15 signed
temp0:32 ← prod0:31 + (RT)
if ((prod0 = RT0) ∧ (RT0 ≠ temp1)) then (RT) ← (RT0 || 31(-RT0))
else (RT) ← temp1:32
```

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 the sum is stored in a 33-bit temporary register.

If a result does not overflow, the low-order 32 bits of the temporary register are stored in RT.

If a result overflows, the returned result is the nearest representable value. Thus, if a result is less than -2^{31} , the value stored in RT is -2^{31} . Likewise, if a result is greater than $2^{31} - 1$, the value stored in RT is $2^{31} - 1$.

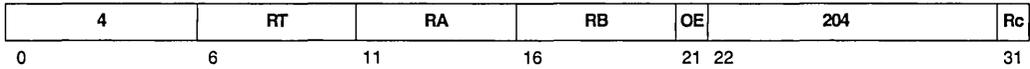
Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

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



$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ unsigned

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow (temp_{1:32} \vee^{32} temp_0)$

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 and the sum is stored in a 33-bit temporary register.

If a result does not overflow, the low-order 32 bits of the temporary register are stored in RT.

If a result overflows, the returned result is the nearest representable value. Thus, if a result is greater than $2^{32} - 1$, the value stored in RT is $2^{32} - 1$.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

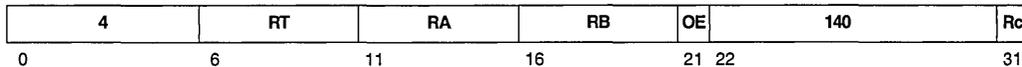
Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

macchwu

Multiply Accumulate Cross Halfword to Word Modulo Unsigned

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



$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ unsigned

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

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 the sum is stored in a 33-bit temporary register. The contents of RT are replaced by the low-order 32 bits of the temporary register.

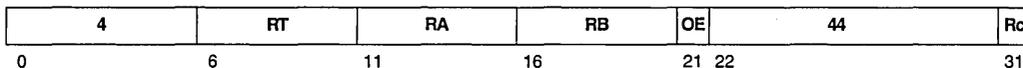
Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

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



$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ signed
 $temp_{0:32} \leftarrow prod_{0:31} + (RT)$
 $(RT) \leftarrow temp_{1:32}$

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 the sum is stored in a 33-bit temporary register. The contents of RT are replaced by the low-order 32 bits of the temporary register.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

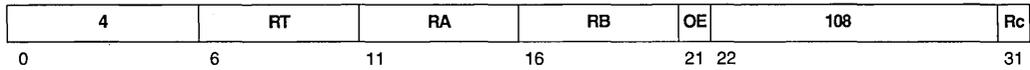
Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

machhws

Multiply Accumulate High Halfword to Word Saturate Signed

machhws	RT, RA, RB	OE=0, Rc=0
machhws.	RT, RA, RB	OE=0, Rc=1
machhws0	RT, RA, RB	OE=1, Rc=0
machhws0.	RT, RA, RB	OE=1, Rc=1



```
prod0:31 ← (RA)0:15 × (RB)0:15 signed
temp0:32 ← prod0:31 + (RT)
if ((prod0 = RT0) ∧ (RT0 ≠ temp1)) then (RT) ← (RT0 || 31(¬RT0))
else (RT) ← temp1:32
```

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 the sum is stored in a 33-bit temporary register.

If a result does not overflow, the low-order 32 bits of the temporary register are stored in RT.

If a result overflows, the returned result is the nearest representable value. Thus, if a result is less than -2^{31} , the value stored in RT is -2^{31} . Likewise, if a result is greater than $2^{31} - 1$, the value stored in RT is $2^{31} - 1$.

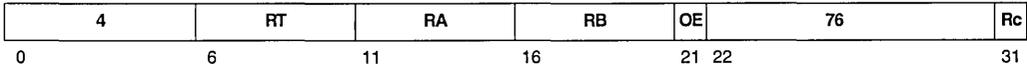
Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

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



$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ unsigned

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow (temp_{1:32} \vee^{32} temp_0)$

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 and the sum is stored in a 33-bit temporary register.

If a result does not overflow, the low-order 32 bits of the temporary register are stored in RT.

If a result overflows, the returned result is the nearest representable value. Thus, if a result is greater than $2^{32} - 1$, the value stored in RT is $2^{32} - 1$.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

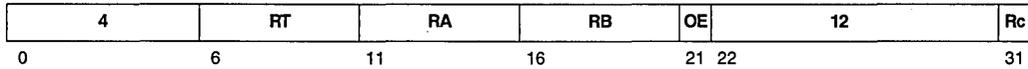
Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

machhwu

Multiply Accumulate High Halfword to Word Modulo Unsigned

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



$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ unsigned
 $temp_{0:32} \leftarrow prod_{0:31} + (RT)$
 $(RT) \leftarrow temp_{1:32}$

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 the sum is stored in a 33-bit temporary register. The contents of RT are replaced by the low-order 32 bits of the temporary register.

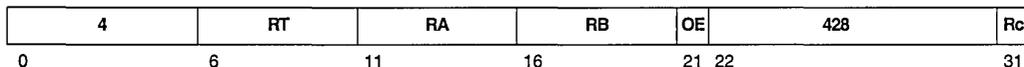
Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

maclhw	RT, RA, RB	OE=0, Rc=0
maclhw.	RT, RA, RB	OE=0, Rc=1
maclhwo	RT, RA, RB	OE=1, Rc=0
maclhwo.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ signed

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

The low-order halfword of RA is multiplied by the low-order halfword of RB. The signed product is summed with the contents of RT and the sum is stored in a 33-bit temporary register. The contents of RT are replaced by the low-order 32 bits of the temporary register.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

maclhws

Multiply Accumulate Low Halfword to Word Saturate Signed

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

$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ signed

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \parallel^{31}(-RT_0))$

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 summed with the contents of RT and the sum is stored in a 33-bit temporary register.

If a result does not overflow, the low-order 32 bits of the temporary register are stored in RT.

If a result overflows, the returned result is the nearest representable value. Thus, if a result is less than -2^{31} , the value stored in RT is -2^{31} . Likewise, if a result is greater than $2^{31} - 1$, the value stored in RT is $2^{31} - 1$.

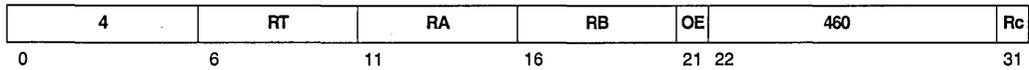
Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

machwsu	RT, RA, RB	OE=0, Rc=0
machwsu.	RT, RA, RB	OE=0, Rc=1
machwsuo	RT, RA, RB	OE=1, Rc=0
machwsuo.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ unsigned

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow (temp_{1:32} \vee^{32} temp_0)$

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 and the sum is stored in a 33-bit temporary register.

If a result does not overflow, the low-order 32 bits of the temporary register are stored in RT.

If a result overflows, the returned result is the nearest representable value. Thus, if a result is greater than $2^{32} - 1$, the value stored in RT is $2^{32} - 1$.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

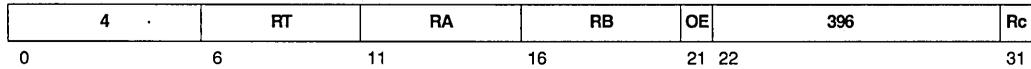
Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

maclhwu

Multiply Accumulate Low Halfword to Word Modulo Unsigned

maclhwu	RT, RA, RB	OE=0, Rc=0
maclhwu.	RT, RA, RB	OE=0, Rc=1
maclhwuo	RT, RA, RB	OE=1, Rc=0
maclhwuo.	RT, RA, RB	OE=1, Rc=1



$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ unsigned

$temp_{0:32} \leftarrow prod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

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 the sum is stored in a 33-bit temporary register. The contents of RT are replaced by the low-order 32 bits of the temporary register.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

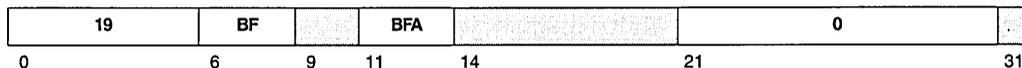
Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

mcrf

Move Condition Register Field

mcrf BF, BFA



$m \leftarrow \text{BFA}$
 $n \leftarrow \text{BF}$
 $(\text{CR}[\text{CR}_n]) \leftarrow (\text{CR}[\text{CR}_m])$

The contents of the CR field specified by the BFA field are placed into the CR field specified by the BF field.

Registers Altered

- $\text{CR}[\text{CR}_n]$ where n is specified by the BF field.

Invalid Instruction Forms

- Reserved fields

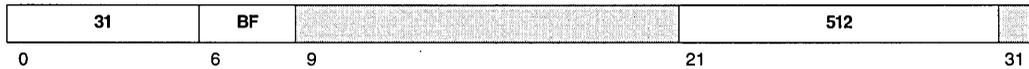
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

mcrxr

Move to Condition Register from XER

mcrxr **BF**



$n \leftarrow \text{BF}$
 $(\text{CR}[\text{CR}_n]) \leftarrow \text{XER}_{0:3}$
 $\text{XER}_{0:3} \leftarrow '0$

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.

This transfer is positional, by bit number, so the mnemonics associated with each bit are changed. See Table 24-18 for clarification.

Table 24-18. Transfer Bit Mnemonic Assignment

Bit	XER Usage	CR Usage
0	SO	LT
1	OV	GT
2	CA	EQ
3	Reserved	SO

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

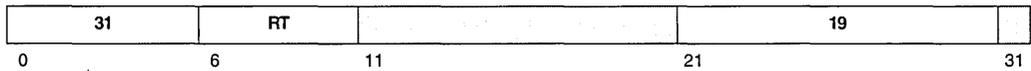
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

mfcrr

Move From Condition Register

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

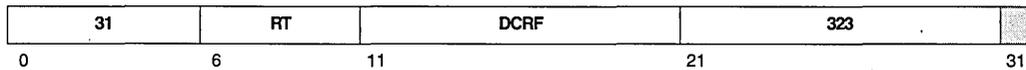
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

mfdcr

Move from Device Control Register

mfdcr RT, DCRN



$DCRN \leftarrow DCRF_{5:9} \parallel DCRF_{0:4}$
 $(RT) \leftarrow (DCR(DCRN))$

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
- Invalid DCRF values

Programming Note

Execution of this instruction is privileged.

The DCR number (DCRN) specified in the assembler language coding of **mfdcr** refers to a DCR number. The assembler handles the unusual register number encoding to generate the DCRF field.

Architecture Note

This instruction is implementation-specific and may not be portable to other implementations.

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.

Architecture Note

This instruction is part of the IBM PowerPC Embedded Operating Environment.

mfspr

Move From Special Purpose Register

mfspr RT, SPRN



$SPRN \leftarrow SPRF_{5:9} \parallel SPRF_{0:4}$
 $(RT) \leftarrow (SPR(SPRN))$

The contents of the SPR specified by the SPRF field are placed into register RT. See “Special Purpose Registers” on page 25-1 for a listing of SPR mnemonics and corresponding SPRN and SPRF 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 Mode Operation” on page 3-41.

The SPR number (SPRN) specified in the assembler language coding of **mfspir** refers to an SPR number (see “Special Purpose Registers” on page 25-1 for a list of SPRN values). The assembler handles the unusual register number encoding to generate the SPRF field. Also, see “Privileged SPRs” on page 3-42 for information about privileged SPRs.

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-19. Extended Mnemonics for mf spr

Mnemonic	Operands	Function	Other Registers Changed
mfccr0 mfctr mfdac1 mfdac2 mfdear mfdbcr0 mfdbcr1 mfdbsr mfdccr mfdcwr mfdvc1 mfdvc2 mfesr mfevpr mflac1 mflac2 mflac3 mflac4 mflccr mflcdbdr mflr mfpid mfpit mfpvr mfsgr mfsler mfsprg0 mfsprg1 mfsprg2 mfsprg3 mfsprg4 mfsprg5 mfsprg6 mfsprg7 mfsrr0 mfsrr1 mfsrr2 mfsrr3 mfsu0r mftcr mftsr mfxer mfzpr	RT	Move from special purpose register SPRN. <i>Extended mnemonic for</i> mf spr RT,SPRN See "Special Purpose Registers" on page 25-1 for a list of valid SPRN values.	

mftb

Move From Time Base

mftb RT, TBRN



$$\begin{aligned} \text{TBRN} &\leftarrow \text{TBRF}_{5:9} \parallel \text{TBRF}_{0:4} \\ (\text{RT}) &\leftarrow (\text{TBR}(\text{TBRN})) \end{aligned}$$

The contents of the time base register (TBR) specified by the TBRF field are placed into register RT. The following table lists the TBRN and TBRF values.

Table 24-20. Extended Mnemonics for mftb

Register Mnemonic	Register Name	TBRN		TBRF	Access
		Decimal	Hex		
TBL	Time Base Lower	268	0x10C	0x188	Read-only
TBU	Time Base Upper	269	0x10D	0x1A8	Read-only

If TBRN is a value other than those listed in the table, the results are boundedly undefined.

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields
- Invalid TBRF values

Programming Notes

The mnemonic **mftb** serves as both a hardware mnemonic and an extended mnemonic. The assembler recognizes an **mftb** mnemonic having two operands as the hardware form; an **mftb** mnemonic having one operand is recognized as the extended form.

The TBR number (TBRN) specified in the assembler language coding of the **mftb** instruction refers to a TBR number listed in the preceding table. The assembler handles the unusual register number encoding to generate the TBRF field.

Architecture Note

This instruction is part of the IBM PowerPC Embedded Virtual Environment.

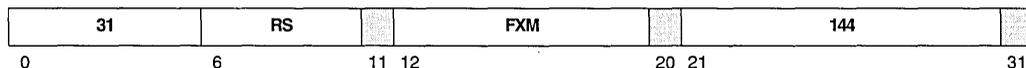
Table 24-21. Extended Mnemonics for mftb

Mnemonic	Operands	Function	Other Registers Altered
mftb	RT	Move the contents of TBL into RT. <i>Extended mnemonic for mftb RT,TBL</i>	
mftbu	RT	Move the contents of TBU into RT. <i>Extended mnemonic for mftb RT,TBU</i>	

mtcrf

Move to Condition Register Fields

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:

FXM Bit Number	Bits Controlled
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

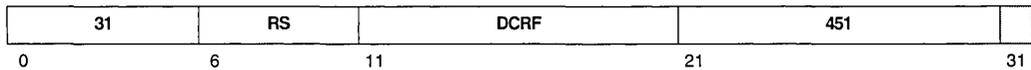
- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-22. Extended Mnemonics for mtcrf

Mnemonic	Operands	Function	Other Registers Altered
mtcr	RS	Move to CR. <i>Extended mnemonic for mtcrf 0xFF,RS</i>	

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
- Invalid DCRF values

Programming Note

Execution of this instruction is privileged.

The DCR number (DCRN) specified in the assembler language coding of **mtdcr** refers to a DCR number. The assembler handles the unusual register number encoding to generate the DCRF field.

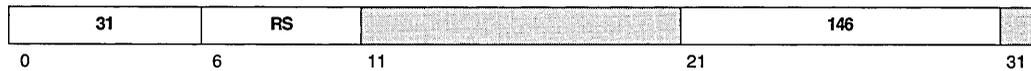
Architecture Note

This instruction is implementation-specific and may not be portable to other implementations.

mtmsr

Move To Machine State Register

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

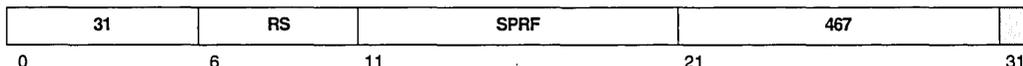
Programming Note

The **mtmsr** instruction is privileged and execution synchronizing.

Architecture Note

This instruction is part of the IBM PowerPC Embedded Operating Environment.

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” on page 25-1 for a listing of SPR mnemonics and corresponding SPRN and SPRF 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 is a 1. See “Privileged SPRs” on page 3-42 for more information.

The SPR number (SPRN) specified in the assembler language coding of the **mtspr** instruction refers to an SPR number (see “Special Purpose Registers” on page 25-1 for a list of SPRN values). The assembler handles the unusual register number encoding to generate the SPRF field.

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

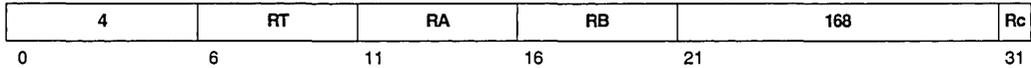
mtspr

Move To Special Purpose Register

Table 24-23. Extended Mnemonics for mtspr

Mnemonic	Operands	Function	Other Registers Altered
mtccr0 mtctr mtdac1 mtdac2 mtdbcr0 mtdbcr1 mtdbsr mtdccr mtdcwr mtdear mtdvc1 mtdvc2 mtesr mtevpr mtiac1 mtiac2 mtiac3 mtiac4 mticcr mticdbdr mtr mtpid mtpit mtpvr mtsgr mtsler mtsprg0 mtsprg1 mtsprg2 mtsprg3 mtsprg4 mtsprg5 mtsprg6 mtsprg7 mtsrr0 mtsrr1 mtsrr2 mtsrr3 mtsru0r mttbl mttbu mttcr mttsr mtxer mtzpr	RS	Move to special purpose register SPRN. <i>Extended mnemonic for mtspr SPRN,RS</i> See "Special Purpose Registers" on page 25-1 for a list of valid SPRN values.	

mulchw RT, RA, RB Rc=0
mulchw. RT, RA, RB Rc=1



$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ signed

The low-order halfword of RA is multiplied by the high-order halfword of RB. The resulting signed product replaces the contents of RT.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1

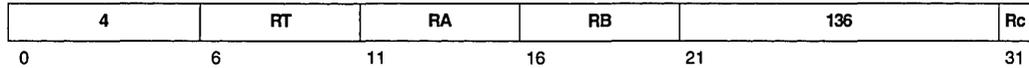
Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

mulchwu

Multiply Cross Halfword to Word Unsigned

mulchwu RT, RA, RB Rc=0
mulchwu. RT, RA, RB Rc=1



$$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15} \text{ unsigned}$$

The low-order halfword of RA is multiplied by the high-order halfword of RB. The resulting unsigned product replaces the contents of RT.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1

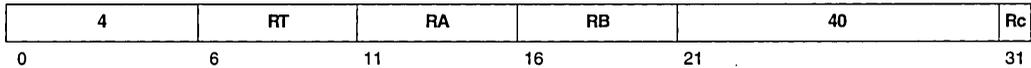
Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

mulhhw

Multiply High Halfword to Word Signed

mulhhw RT, RA, RB Rc=0
mulhhw. RT, RA, RB Rc=1



$(RT)_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ signed

The high-order halfword of RA is multiplied by the high-order halfword of RB. The resulting signed product replaces the contents of RT.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1

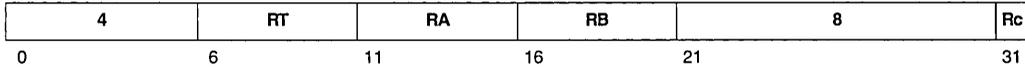
Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

mulhhu

Multiply High Halfword to Word Unsigned

mulhhu RT, RA, RB Rc=0
mulhhu. RT, RA, RB Rc=1



$(RT)_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ unsigned

The high-order halfword of RA is multiplied by the high-order halfword of RB. The resulting unsigned product replaces the contents of RT.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1

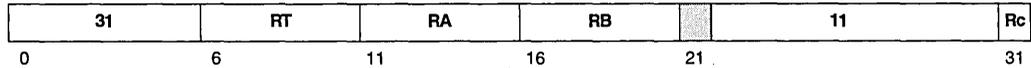
Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

mulhwu

Multiply High Word Unsigned

mulhwu RT, RA, RB Rc=0
mulhwu. RT, RA, RB Rc=1



$prod_{0:63} \leftarrow (RA) \times (RB)$ unsigned
 $(RT) \leftarrow prod_{0:31}$

The 64-bit unsigned product of registers RA and RB is formed. The most significant 32 bits of the result are placed into register RT.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} 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. The **mulhw** instruction generates the correct result when these operands are interpreted as signed quantities. The **mulhwu** instruction generates the correct result when these operands are interpreted as unsigned quantities.

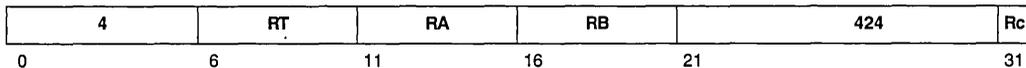
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

mullhw

Multiply Low Halfword to Word Signed

mullhw RT, RA, RB Rc=0
mullhw. RT, RA, RB Rc=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. The resulting signed product replaces the contents of RT.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1

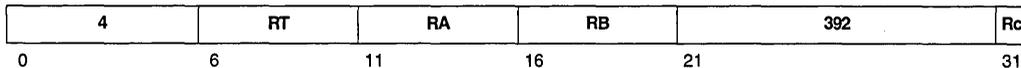
Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

mullhwu

Multiply Low Halfword to Word Unsigned

mullhwu RT, RA, RB OE=0, Rc=0
mullhwu. RT, RA, RB OE=0, Rc=1



$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ unsigned

The low-order halfword of RA is multiplied by the low-order halfword of RB. The resulting unsigned product replaces the contents of RT.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1

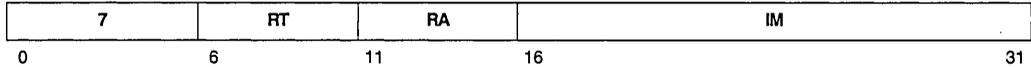
Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

multi

Multiply Low Immediate

multi RT, RA, IM



$prod_{0:47} \leftarrow (RA) \times EXTS(IM) \text{ signed}$
 $(RT) \leftarrow prod_{16:47}$

The 48-bit product of register RA and the sign-extended IM field is formed. Both register RA and the IM field are interpreted as signed quantities. 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.

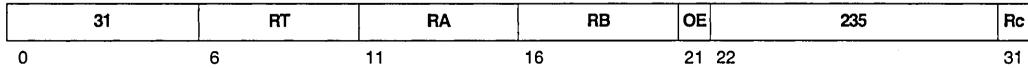
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

mullw

Multiply Low Word

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



$prod_{0:63} \leftarrow (RA) \times (RB)$ signed
 $(RT) \leftarrow prod_{32:63}$

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]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE=1

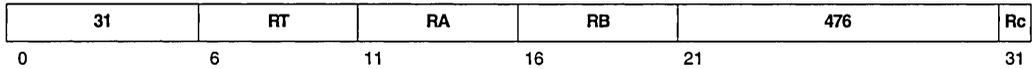
Programming Note

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 is correct only if the operands are regarded as signed numbers.

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

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]_{LT, GT, EQ, SO} if Rc contains 1

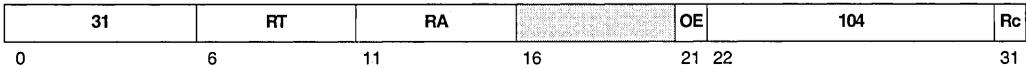
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

neg

Negate

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]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE=1

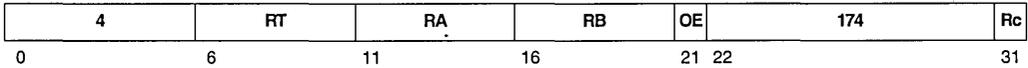
Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

nmacchw	RT, RA, RB	OE=0, Rc=0
nmacchw.	RT, RA, RB	OE=0, Rc=1
nmacchw0	RT, RA, RB	OE=1, Rc=0
nmacchw0.	RT, RA, RB	OE=1, Rc=1



$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{0:15})$ 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 negated signed product is summed with the contents of RT and the sum is stored in a 33-bit temporary register. The contents of RT are replaced by the low-order 32 bits of the temporary register.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1

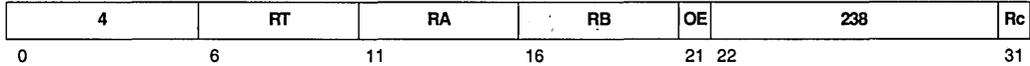
Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

nmacchws

Negative Multiply Accumulate Cross Halfword to Word Saturate Signed

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



```

nprod0:31 ← -((RA)16:31 × (RB)0:15 signed)
temp0:32 ← nprod0:31 + (RT)
if ((nprod0 = RT0) ∧ (RT0 ≠ temp1)) then (RT) ← (RT0 || 31(-RT0))
else (RT) ← temp1:32

```

The low-order halfword of RA is multiplied by the high-order halfword of RB. The negated signed product is summed with the contents of RT and the sum is stored in a 33-bit temporary register.

If a result does not overflow, the low-order 32 bits of the temporary register are stored in RT.

If a result overflows, the returned result is the nearest representable value. Thus, if a result is less than -2^{31} , the value stored in RT is -2^{31} . Likewise, if a result is greater than $2^{31} - 1$, the value stored in RT is $2^{31} - 1$.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

nmachhw

Negative Multiply Accumulate High Halfword to Word Modulo Signed

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



$nprod_{0:31} \leftarrow -((RA)_{0:15} \times (RB)_{0:15})$ signed

$temp_{0:32} \leftarrow nprod_{0:31} + (RT)$

$(RT) \leftarrow temp_{1:32}$

The high-order halfword of RA is multiplied by the high-order halfword of RB. The negated signed product is summed with the contents of RT and the sum is stored in a 33-bit temporary register. The contents of RT are replaced by the low-order 32 bits of the temporary register.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

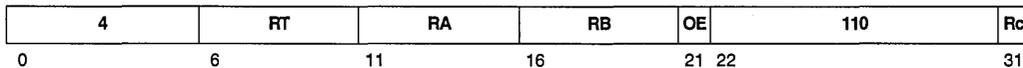
Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

nmachhws

Negative Multiply Accumulate High Halfword to Word Saturate Signed

nmachhws	RT, RA, RB	OE=0, Rc=0
nmachhws.	RT, RA, RB	OE=0, Rc=1
nmachhws0	RT, RA, RB	OE=1, Rc=0
nmachhws0.	RT, RA, RB	OE=1, Rc=1



```

nprod0:31 ← −((RA)0:15 × (RB)0:15) signed
temp0:32 ← nprod0:31 + (RT)
if ((nprod0 = RT0) ∧ (RT0 ≠ temp1)) then (RT) ← (RT0 || 31(−RT0))
else (RT) ← temp1:32

```

The high-order halfword of RA is multiplied by the high-order halfword of RB. The negated signed product is summed with the contents of RT and the sum is stored in a 33-bit temporary register.

If a result does not overflow (i.e., it is accurately representable in 32 bits), the low-order 32 bits of the temporary register are stored in RT.

If a result overflows, the returned result is the nearest representable value. Thus, if a result is less than -2^{31} , the value stored in RT is -2^{31} . Likewise, if a result is greater than $2^{31} - 1$, the value stored in RT is $2^{31} - 1$.

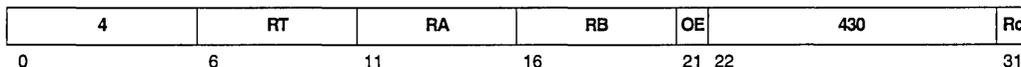
Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

nmac1hw	RT, RA, RB	OE=0, Rc=0
nmac1hw.	RT, RA, RB	OE=0, Rc=1
nmac1hwo	RT, RA, RB	OE=1, Rc=0
nmachlwo.	RT, RA, RB	OE=1, Rc=1



$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{16:31})$ 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 negated signed product is summed with the contents of RT and the sum is stored in a 33-bit temporary register. The contents of RT are replaced by the low-order 32 bits of the temporary register.

Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

nmaclhws

Negative Multiply Accumulate High Halfword to Word Saturate Signed

nmaclhws	RT, RA, RB	OE=0, Rc=0
nmaclhws.	RT, RA, RB	OE=0, Rc=1
nmaclhwsO	RT, RA, RB	OE=1, Rc=0
nmaclhwsO.	RT, RA, RB	OE=1, Rc=1



```

nprod0:31 ← -((RA)16:31 × (RB)16:31) signed
temp0:32 ← nprod0:31 + (RT)
if ((nprod0 = RT0) ∧ (RT0 ≠ temp1)) then (RT) ← (RT0 || 31(-RT0))
else (RT) ← temp1:32

```

The low-order halfword of RA is multiplied by the low-order halfword of RB. The negated signed product is summed with the contents of RT and the sum is stored in a 33-bit temporary register.

If a result does not overflow, the low-order 32 bits of the temporary register are stored in RT.

If a result overflows, the returned result is the nearest representable value. Thus, if a result is less than -2^{31} , the value stored in RT is -2^{31} . Likewise, if a result is greater than $2^{31} - 1$, the value stored in RT is $2^{31} - 1$.

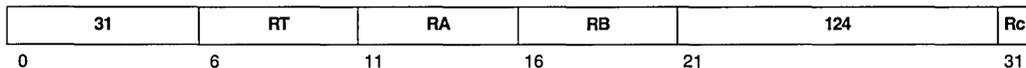
Registers Altered

- RT
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is part of the Multiply-Accumulate instruction set extensions and complies with the architectural requirements for APUs of the IBM PowerPC Embedded Environment. As such, it is not part of the PowerPC Architecture, nor is it part of the IBM PowerPC Embedded Environment. Programs that use this instruction may not be portable to other implementations.

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]_{LT, GT, EQ, SO} if Rc contains 1

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

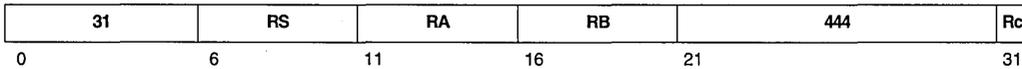
Table 24-24. Extended Mnemonics for nor, nor.

Mnemonic	Operands	Function	Other Registers Altered
not	RA, RS	Complement register. (RA) ← ¬(RS) <i>Extended mnemonic for nor RA,RS,RS</i>	
not.		<i>Extended mnemonic for nor. RA,RS,RS</i>	CR[CR0]

or

OR

or RA, RS, RB Rc=0
or. RA, RS, RB Rc=1



$$(RA) \leftarrow (RS) \vee (RB)$$

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]_{LT,GT,EQ,SO} if Rc contains 1

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-25. 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]_{LT, GT, EQ, SO} if Rc contains 1

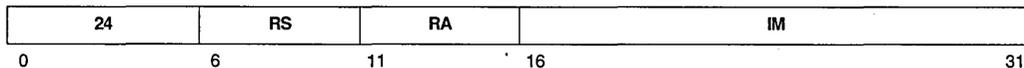
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

ori

OR Immediate

ori RA, RS, IM



$$(RA) \leftarrow (RS) \vee (160 \parallel IM)$$

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

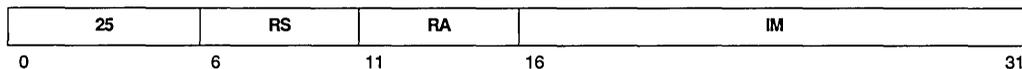
- RA

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-26. 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

$$(RA) \leftarrow (RS) \vee (IM \parallel 160)$$

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

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

rft

Return From Critical Interrupt

rft



(PC) ← (SRR2)
(MSR) ← (SRR3)

The program counter (PC) is restored with the contents of SRR2 and the MSR is restored with the contents of SRR3.

Instruction execution returns to the address contained in the PC.

Registers Altered

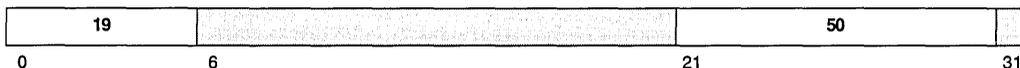
- MSR

Programming Note

Execution of this instruction is privileged and context-synchronizing.

Architecture Note

This instruction part of the IBM PowerPC Embedded Operating Environment.

rfi

(PC) ← (SRR0)
(MSR) ← (SRR1)

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.

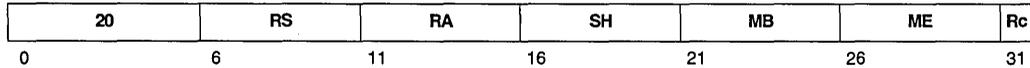
Architecture Note

This instruction is part of the IBM PowerPC Embedded Operating Environment.

rlwimi

Rotate Left Word Immediate then Mask Insert

rlwimi RA, RS, SH, MB, ME R_c=0
 rlwimi. RA, RS, SH, MB, ME R_c=1



$$r \leftarrow \text{ROTL}((RS), SH)$$

$$m \leftarrow \text{MASK}(MB, ME)$$

$$(RA) \leftarrow (r \wedge m) \vee ((RA) \wedge \neg m)$$

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]_{LT, GT, EQ, SO} if R_c contains 1

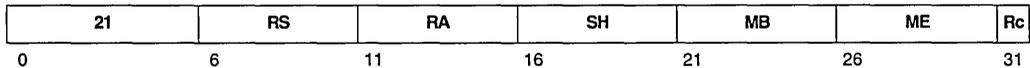
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-27. 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 RA, RS, SH, MB, ME Rc=0
rlwinm. RA, RS, SH, MB, ME Rc=1



$r \leftarrow \text{ROTL}((RS), SH)$
 $m \leftarrow \text{MASK}(MB, ME)$
 $(RA) \leftarrow r \wedge m$

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]_{LT, GT, EQ, SO} if Rc contains 1

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-28. Extended Mnemonics for rlwinm, rlwinm.

Mnemonic	Operands	Function	Other Registers Altered
clrlwi	RA, RS, n	Clear left immediate. ($n < 32$) $(RA)_{0:n-1} \leftarrow {}^n0$ <i>Extended mnemonic for rlwinm RA,RS,0,n,31</i>	
clrlwi.		<i>Extended mnemonic for rlwinm. RA,RS,0,n,31</i>	CR[CR0]
clrlslwi	RA, RS, b, n	Clear left and shift left immediate. ($n \leq b < 32$) $(RA)_{b:n:31-n} \leftarrow (RS)_{b:31}$ $(RA)_{32-n:31} \leftarrow {}^n0$ $(RA)_{0:b-n-1} \leftarrow b-n0$ <i>Extended mnemonic for rlwinm RA,RS,n,b-n,31-n</i>	
clrlslwi.		<i>Extended mnemonic for rlwinm. RA,RS,n,b-n,31-n</i>	CR[CR0]

rlwinm

Rotate Left Word Immediate then AND with Mask

Table 24-28. Extended Mnemonics for rlwinm, rlwinm. (continued)

Mnemonic	Operands	Function	Other Registers Altered
clrrwi	RA, RS, n	Clear right immediate. ($n < 32$) $(RA)_{32-n:31} \leftarrow ^n0$ <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]
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]
rotzwi	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	
rotzwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,32-n,0,31	CR[CR0]
slwi	RA, RS, n	Shift left immediate. ($n < 32$) $(RA)_{0:31-n} \leftarrow (RS)_{n:31}$ $(RA)_{32-n:31} \leftarrow ^n0$ <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]

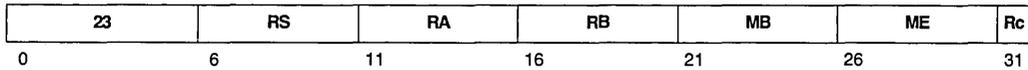
Table 24-28. Extended Mnemonics for rlwinm, rlwinm. (continued)

Mnemonic	Operands	Function	Other Registers Altered
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 rlwinm RA,RS,32-n,n,31</i>	
srwi.		<i>Extended mnemonic for rlwinm. RA,RS,32-n,n,31</i>	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]_{LT, GT, EQ, SO} if Rc contains 1

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-29. Extended Mnemonics for rlwnm, rlwnm.

Mnemonic	Operands	Function	Other Registers Altered
rotlw	RA, RS, RB	Rotate left. (RA) ← ROTL((RS), (RB) _{27:31}) <i>Extended mnemonic for rlwnm RA,RS,RB,0,31</i>	
rotlw.		<i>Extended mnemonic for rlwnm. RA,RS,RB,0,31</i>	CR[CR0]

sc



(SRR1) ← (MSR)
 (SRR0) ← (PC)
 PC ← EVPR_{0:15} || 0x0C00
 (MSR[WE, EE, PR, DR, IR]) ← 0

A system call exception is generated. 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 exception vector address. The exception vector address is calculated by concatenating the high halfword of the Exception Vector Prefix Register (EVPR) to the left of 0x0C00.

The MSR[WE, EE, PR, DR, IR] bits are set to 0.

Program execution continues at the new address in the PC.

The **sc** instruction is context synchronizing.

Registers Altered

- SRR0
- SRR1
- MSR[WE, EE, PR, DR, IR]

Invalid Instruction Forms

- Reserved fields

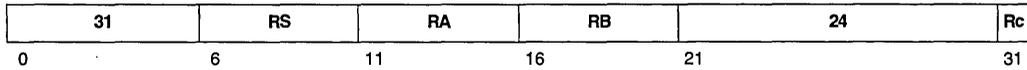
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

slw

Shift Left Word

slw RA, RS, RB Rc=0
slw. RA, RS, RB Rc=1



```
n ← (RB)27:31
r ← ROTL((RS), n)
if (RB)26 = 0 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_{27:31}. 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.

If RB₂₆ = 1, register RA is set to zero.

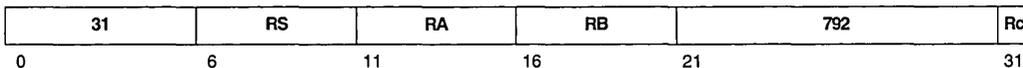
Registers Altered

- RA
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

sraw RA, RS, RB Rc=0
 sraw. RA, RS, RB Rc=1



```

n ← (RB)27:31
r ← ROTL((RS), 32 - n)
if (RB)26 = 0 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_{27:31}. Bits shifted out of the least significant bit are lost. Register 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.

If bit 26 of register RB contains 1, register RA and XER[CA] are set to bit 0 of register RS.

Registers Altered

- RA
- XER[CA]
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1

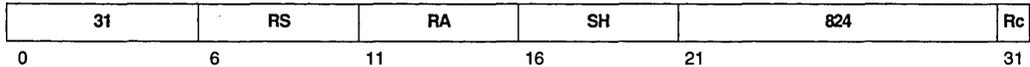
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

srawi

Shift Right Algebraic Word Immediate

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.

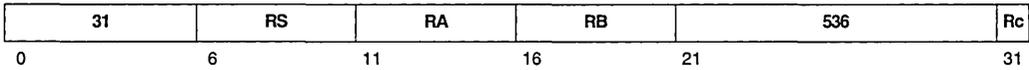
Registers Altered

- RA
- XER[CA]
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

srw RA, RS, RB Rc=0
srw. RA, RS, RB Rc=1



```

n ← (RB)27:31
r ← ROTL((RS), 32 - n)
if (RB)26 = 0 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_{27:31}. 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.

If bit 26 of register RB contains a one, register RA is set to 0.

Registers Altered

- RA
- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1

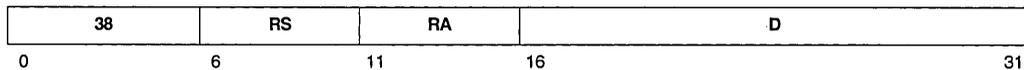
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

stb

Store Byte

stb RS, D(RA)



$EA \leftarrow (RAI0) + EXTS(D)$

$MS(EA, 1) \leftarrow (RS)_{24:31}$

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.

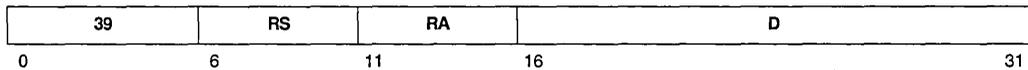
The least significant byte of register RS is stored into the byte at the EA.

Registers Altered

- None

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

stbu RS, D(RA) $EA \leftarrow (RA)_{10} + EXTS(D)$ $MS(EA, 1) \leftarrow (RS)_{24:31}$ $(RA) \leftarrow EA$

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

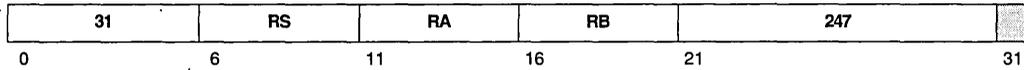
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

stbux

Store Byte with Update Indexed

stbux RS, RA, RB



$EA \leftarrow (RA \ll 0) + (RB)$
 $MS(EA, 1) \leftarrow (RS)_{24:31}$
 $(RA) \leftarrow EA$

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

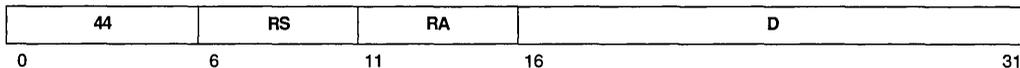
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

sth

Store Halfword

sth RS, D(RA)



$EA \leftarrow (RA \ll 0) + \text{EXTS}(D)$
 $MS(EA, 2) \leftarrow (RS)_{16:31}$

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.

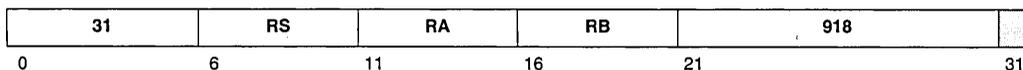
Registers Altered

- None

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

sthbrx RS, RA, RB



$$EA \leftarrow (RA \ll 0) + (RB)$$

$$MS(EA, 2) \leftarrow (RS)_{24:31} \parallel (RS)_{16:23}$$

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. The result 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

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

sth

Store Halfword with Update

sth RS, D(RA)



$EA \leftarrow (RA \ll 0) + \text{EXTS}(D)$
 $MS(EA, 2) \leftarrow (RS)_{16:31}$
 $(RA) \leftarrow EA$

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

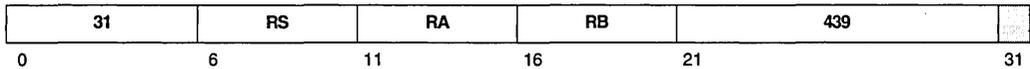
Invalid Instruction Forms

- RA = 0

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

sthux RS, RA, RB



$$EA \leftarrow (RA \ll 0) + (RB)$$

$$MS(EA, 2) \leftarrow (RS)_{16:31}$$

$$(RA) \leftarrow EA$$

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

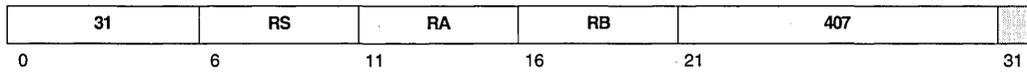
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

sthx

Store Halfword Indexed

sthx RS, RA, RB



$$EA \leftarrow (RA \ll 0) + (RB)$$
$$MS(EA, 2) \leftarrow (RS)_{16:31}$$

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

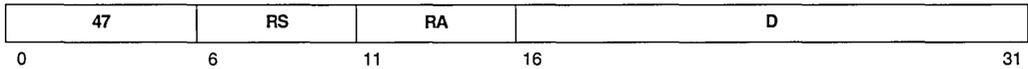
Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

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

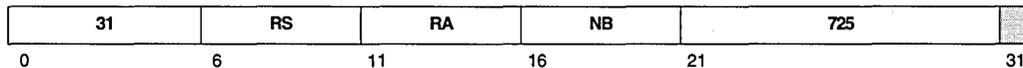
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

stswi

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), wrapping to GPR(0), 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.

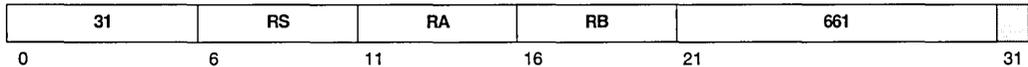
Registers Altered

- None

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

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), wrapping to GPR(0), 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

If XER[TBC] = 0, **stswx** is treated as a no-op.

The PowerPC Architecture states that if XER[TBC] = 0 and if the EA is such that a precise data exception would normally occur (if not for the zero length), **stswx** is treated as a no-op and the precise exception will not occur. Data storage exceptions and alignment exceptions are examples of precise data exceptions.

stswx

Store String Word Indexed

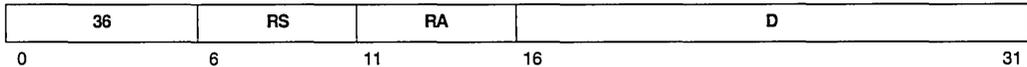
However, the architecture makes no statement regarding imprecise exceptions related to **stswx** when $XER[TBC] = 0$. IBM PowerPC Embedded controllers generate an imprecise exception (machine check) on this instruction when all of the following conditions are true:

- The instruction passes all protection bounds checking
- The address is cachable
- The address is passed to the data cache
- The address misses in the data cache (resulting in a line fill request)
- The address encounters some form of bus error (non-configured, for example)

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

stw RS, D(RA)



$EA \leftarrow (RA \ll 0) + \text{EXTS}(D)$
 $MS(EA, 4) \leftarrow (RS)$

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

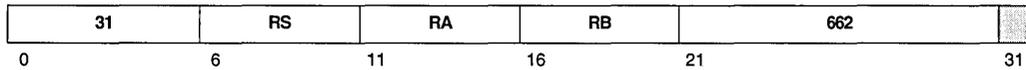
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

stwbrx

Store Word Byte-Reverse Indexed

stwbrx RS, RA, RB



$$EA \leftarrow (RA)0 + (RB)$$

$$MS(EA, 4) \leftarrow (RS)_{24:31} \parallel (RS)_{16:23} \parallel (RS)_{8:15} \parallel (RS)_{0:7}$$

An 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 byte-reversed: the least significant byte becomes the most significant byte, the next least significant byte becomes the next most significant byte, and so on. The result is 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

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

stwcx. RS, RA, RB



```

EA ← (RAI0) + (RB)
if RESERVE = 1 then
    MS(EA, 4) ← (RS)
    RESERVE ← 0
    (CR[CR0]) ← 20 || 1 || XERSO
else
    (CR[CR0]) ← 20 || 0 || XERSO
    
```

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]_{LT, GT} are cleared
- CR[CR0]_{EQ} is set to the state of the reservation bit at the start of the instruction
- CR[CR0]_{SO} is set to the contents of the XER[SO] bit

Registers Altered

- CR[CR0]_{LT, GT, EQ, SO}

Programming Note

lwarx and the **stwcx.** instruction should be 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 asynchronous processes. Only **lwarx** can set the reservation bit to 1. **stwcx.** sets the reservation bit to 0 upon its completion, whether or not **stwcx.** sent (RS) to memory. CR[CR0]_{EQ} 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 register as required
      stwcx. # attempt to store semaphore; reset reservation
      bne loop # an asynchronous process has intervened; try again
    
```

If the asynchronous process in the code example had paired **lwarx** with a store other than **stwcx.**, the reservation bit would not have been cleared in the asynchronous process, and the code example would have overwritten the semaphore.

Exceptions

An alignment exception occurs if the EA is not word-aligned.

stwcx.

Store Word Conditional Indexed

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

stwu RS, D(RA) $EA \leftarrow (RA \ll 0) + \text{EXTS}(D)$ $MS(EA, 4) \leftarrow (RS)$ $(RA) \leftarrow EA$

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

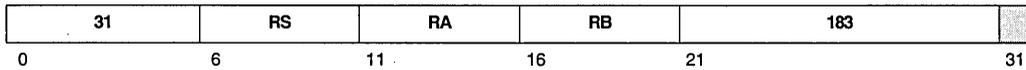
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

stwux

Store Word with Update Indexed

stwux RS, RA, RB



$EA \leftarrow (RA10) + (RB)$
 $MS(EA, 4) \leftarrow (RS)$
 $(RA) \leftarrow EA$

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

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

stwx RS, RA, RB



$$EA \leftarrow (RA \ll 0) + (RB)$$

$$MS(EA,4) \leftarrow (RS)$$

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

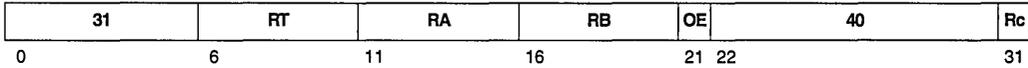
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

subf

Subtract From

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]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

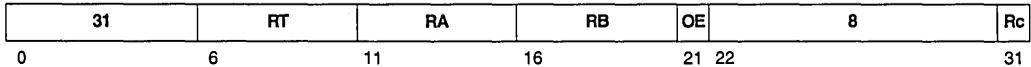
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-30. 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	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]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

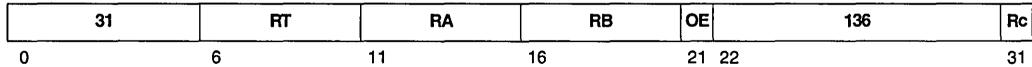
Table 24-31. Extended Mnemonics for subfc, subfc., subfco, subfco.

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]
subfco		<i>Extended mnemonic for subfco RT, RB, RA</i>	XER[SO, OV]
subfco.		<i>Extended mnemonic for subfco. RT, RB, RA</i>	CR[CR0] XER[SO, OV]

subfe

Subtract From Extended

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]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

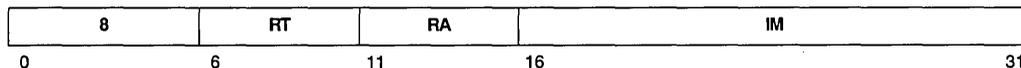
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

subfic

Subtract From Immediate Carrying

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]

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

subfme

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]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1
- XER[CA]

Invalid Instruction Forms

- Reserved fields

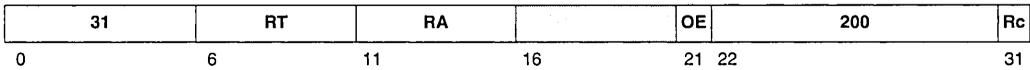
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

subfze

Subtract from Zero Extended

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]_{LT, GT, EQ, SO} if Rc contains 1
- XER[SO, OV] if OE contains 1

Invalid Instruction Forms

- Reserved fields

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

sync

Synchronize

sync



The **sync** instruction guarantees that all instructions initiated by the processor preceding **sync** will complete before **sync** completes, and that no subsequent instructions will be initiated by the processor until after **sync** completes. When **sync** completes, all storage accesses that were initiated by the processor before the **sync** instruction will have been completed with respect to all mechanisms that access storage.

If instruction bit 31 contains 1, the contents of CR[CR0] are undefined.

Registers Altered

- None.

Invalid Instruction Forms

- Reserved fields

Programming Note

Architecturally, the **eieio** instruction orders storage access, not instruction completion. Therefore, non-storage operations that follow **eieio** could complete before storage operations that precede **eieio**. The **sync** instruction guarantees ordering of instruction completion and storage access. For the PPC405GP, the **eieio** instruction is implemented to behave as a **sync** instruction.

To write code that is portable between various PowerPC implementations, programmers should use the mnemonic that corresponds to the desired behavior.

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

tlbia

All of the entries in the TLB are invalidated and become unavailable for translation by clearing the valid (V) bit in the TLBHI portion of each TLB entry. The rest of the fields in the TLB entries are unmodified.

Registers Altered

- None.

Invalid Instruction Forms

- None.

Programming Note

This instruction is privileged. Translation is not required to be active during the execution of this instruction. The effects of the invalidation are not guaranteed to be visible to the programming model until the completion of a context synchronizing operation.

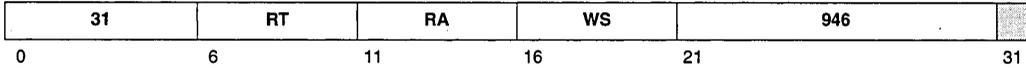
Architecture Note

This instruction is part of the IBM PowerPC Embedded Operating Environment.

tlbre

TLB Read Entry

tlbre RT, RA, WS



```
if WS4 = 1
  (RT) ← TLBLO[(RA26:31)]
else
  (RT) ← TLBHI[(RA26:31)]
  (PID) ← TID from TLB[(RA26:31)]
```

The contents of the selected TLB entry is placed into register RT (and possibly into PID).

Bits 26:31 of the contents of RA is used as an index into the TLB. If this index specifies a TLB entry that does not exist, the results are undefined.

The WS field specifies which portion (TLBHI or TLBLO) of the entry is loaded into RT. If TLBHI is being accessed, the PID SPR is set to the value of the TID field in the TLB entry.

If the WS field is not 0 or 1, 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
- PID (if WS = 0)

Invalid Instruction Forms

- Reserved fields
- Invalid WS value

Programming Notes

This instruction is privileged. Translation is not required to be active during the execution of this instruction.

The contents of RT after the execution of this instruction are interpreted as follows:

```
If WS = 0 (TLBHI):
  RT[0:21] ← EPN[0:21]
  RT[22:24] ← SIZE[0:2]
  RT[25] ← V
  RT[26] ← E
  RT[27] ← U0
  RT[28:31] ← 0
  PID[24:31] ← TID[0:7]; (note that the TID is copied to the PID, not to RT)
If WS = 1 (TLBLO):
  RT[0:21] ← RPN[0:21]
  RT[22:23] ← EX.WR
  RT[24:27] ← ZSEL[0:3]
  RT[28:31] ← WIMG
```

Architecture Note

This instruction part of the IBM PowerPC Embedded Operating Environment.

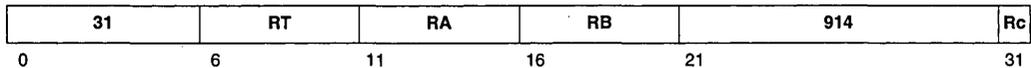
Table 24-32. Extended Mnemonics for tlbre

Mnemonic	Operands	Function	Other Registers Altered
tlbrehi	RT, RA	Load TLBHI portion of the selected TLB entry into RT. Load the PID register with the contents of the TID field of the selected TLB entry. $(RT) \leftarrow TLBHI[(RA)]$ $(PID) \leftarrow TLB[(RA)]_{TID}$ <i>Extended mnemonic for tlbre RT,RA,0</i>	
tlbrelo	RT, RA	Load TLBLO portion of the selected TLB entry into RT. $(RT) \leftarrow TLBLO[(RA)]$ <i>Extended mnemonic for tlbre RT,RA,1</i>	

tbsx

TLB Search Indexed

tbsx RT, RA, RB R_c=0
tbsx. RT, RA, RB R_c=1



```
EA ← (RAI0) + (RB)
if Rc = 1
    CR[CR0]LT ← 0
    CR[CR0]GT ← 0
    CR[CR0]SO ← XER[SO]
if Valid TLB entry matching EA and PID is in the TLB then
    (RT) ← Index of matching TLB Entry
    if Rc = 1
        CR[CR0]EQ ← 1
    else
        (RT) Undefined
    if Rc = 1
        CR[CR0]EQ ← 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 PID. See XREF for details. The record bit (R_c) specifies whether the results of the search will affect CR[CR0] as shown above. The intention is that CR[CR0]_{EQ} can be tested after a **tbsx.** instruction if there is a possibility that the search may fail.

Registers Altered

- CR[CR0]_{LT}, _{GT}, _{EQ}, _{SO} if R_c contains 1

Invalid Instruction Forms

- None.

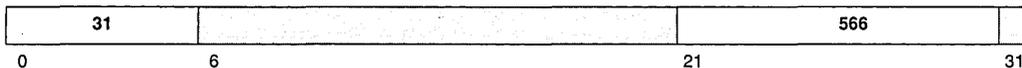
Programming Note

This instruction is privileged. Translation is not required to be active during the execution of this instruction.

Architecture Note

This instruction part of the IBM PowerPC Embedded Operating Environment.

tlbsync



The **tlbsync** instruction is provided in the PowerPC architecture to support synchronization of TLB operations among the processors of a multi-processor system. In the PPC405GP, this instruction performs no operation, and is provided to facilitate code portability.

Registers Altered

- None.

Invalid Instruction Forms

- None.

Programming Notes

This instruction is privileged. Translation is not required to be active during the execution of this instruction.

Since the PPC405GP does not support tightly-coupled multiprocessor systems, **tlbsync** performs no operation.

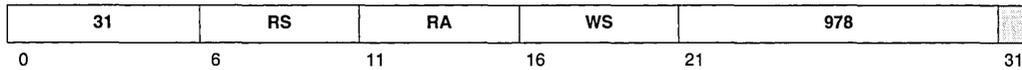
Architecture Note

This instruction is part of the IBM PowerPC Embedded Operating Environment.

tlbwe

TLB Write Entry

tlbwe RS, RA, WS



```
if WS4 = 1
    TLBLO[(RA26:31)] ← (RS)
else
    TLBHI[(RA26:31)] ← (RS)
    TID of TLB[(RA26:31)] ← (PID24:31)
```

The contents of the selected TLB entry is replaced with the contents of register RS (and possibly PID).

Bits 26:31 of the contents of RA are used as an index into the TLB. If this index specifies a TLB entry that does not exist, the results are undefined.

The WS field specifies which portion (TLBHI or TLBLO) of the entry is replaced from RS. For instructions that specify TLBHI, the TID field in the TLB entry is supplied from PID_{24:31}.

If the WS field is not 0 or 1, 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 Notes

This instruction is privileged. Translation is not required to be active during the execution of this instruction.

The effects of this update are not guaranteed to be visible to the programming model until the completion of a context synchronizing operation. For example, updating a zone selection field within the TLB while in supervisor code should be followed by an **isync** instruction (or other context synchronizing operation) to guarantee that the desired translation and protection domains are used.

tlbwe writes the TLB fields from RS and the PID as follows:

```
If WS = 0 (TLBHI):
    EPN[0:21] ← RS[0:21]
    SIZE[0:2] ← RS[22:24]
    V ← RS[25]
    E ← RS[26]
    U0 ← RS[27]
    TID[0:7] ← PID[24:31]; (note that the TID is written from the PID, not RS)
```

If WS = 1 (TLBLO):
 $RPN[0:21] \leftarrow RT[0:21]$
 $EX, WR \leftarrow RS[22:23]$
 $ZSEL[0:3] \leftarrow RS[24:27]$
 $WIMG \leftarrow RS[28:31]$

Architecture Note

This instruction part of the IBM PowerPC Embedded Operating Environment.

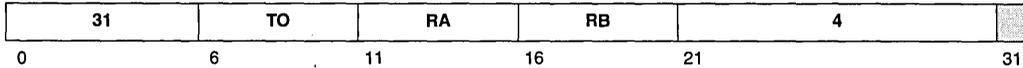
Table 24-33. Extended Mnemonics for tlbwe

Mnemonic	Operands	Function	Other Registers Altered
tlbwehi	RS, RA	Write TLBHI portion of the selected TLB entry from RS. Write the TID register of the selected TLB entry from the PID register. $TLBHI[(RA)] \leftarrow (RS)$ $TLB[(RA)]_{TID} \leftarrow (PID_{24:31})$ <i>Extended mnemonic for</i> tlbwe RS,RA,0	
tlbwelo	RS, RA	Write TLBLO portion of the selected TLB entry from RS. $TLBLO[(RA)] \leftarrow (RS)$ <i>Extended mnemonic for</i> tlbwe RS,RA,1	

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)) then TRAP (see details below)

Register RA is compared with register RB. If any comparison condition selected by the TO field is true, a TRAP occurs. The behavior of a TRAP depends upon the debug mode of the processor, as described below:

- If TRAP is not enabled as a debug event (DBCR[TDE] = 0 or DBCR[EDM, IDM] = 0,0):

TRAP causes a program interrupt. See “Program Interrupt” on page 10-40.

(SRR0) ← address of tw instruction
(SRR1) ← (MSR)
(ESR[PTR]) ← 1
(MSR[WE, EE, PR, DR, IR]) ← 0
PC ← EVPR_{0:15} || 0x0700

- If TRAP is enabled as an external debug event (DBCR[TDE] = 1 and DBCR[EDM] = 1):

TRAP goes to the debug stop state, to be handled by an external debugger with hardware control.

(DBSR[TIE]) ← 1

In addition, if TRAP is also enabled as an internal debug event (DBCR[IDM] = 1) and debug exceptions are disabled (MSR[DE] = 0), then report an imprecise event:

(DBSR[IDE]) ← 1
PC ← address of tw instruction

- If TRAP is enabled as an internal debug event and *not* an external debug event (DBCR[TDE] = 1 and DBCR[EDM, IDM] = 0,1) and debug exceptions are enabled (MSR[DE] = 1):

TRAP causes a debug interrupt. See “Debug Interrupt” on page 10-44.

(SRR2) ← address of tw instruction
(SRR3) ← (MSR)
(DBSR[TIE]) ← 1
(MSR[WE, EE, PR, CE, DE, DR, IR]) ← 0
PC ← EVPR_{0:15} || 0x2000

- If TRAP is enabled as an internal debug event and *not* an external debug event (DBCR[TDE] = 1 and DBCR[EDM, IDM] = 0,1) and Debug Exceptions are disabled (MSR[DE] = 0):

TRAP reports the debug event as an *imprecise* event and causes a program interrupt. See “Program Interrupt” on page 10-40.

(SRR0) ← address of **tw** instruction
 (SRR1) ← (MSR)
 (ESR(PTR)) ← 1
 (DBSR[TIE,IDE]) ← 1,1
 (MSR[WE, EE, PR, DR, IR]) ← 0
 PC ← EVPR_{0:15} || 0x0700

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 inserted into the execution stream by a debugger to implement breakpoints, and is not typically used by application code.

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-34. Extended Mnemonics for tw

Mnemonic	Operands	Function	Other Registers Altered
trap		Trap unconditionally. <i>Extended mnemonic for tw 31,0,0</i>	
tweq	RA, RB	Trap if (RA) equal to (RB). <i>Extended mnemonic for tw 4,RA,RB</i>	
twge	RA, RB	Trap if (RA) greater than or equal to (RB). <i>Extended mnemonic for tw 12,RA,RB</i>	
twgt	RA, RB	Trap if (RA) greater than (RB). <i>Extended mnemonic for tw 8,RA,RB</i>	
twle	RA, RB	Trap if (RA) less than or equal to (RB). <i>Extended mnemonic for tw 20,RA,RB</i>	
twlge	RA, RB	Trap if (RA) logically greater than or equal to (RB). <i>Extended mnemonic for tw 5,RA,RB</i>	
twlgt	RA, RB	Trap if (RA) logically greater than (RB). <i>Extended mnemonic for tw 1,RA,RB</i>	

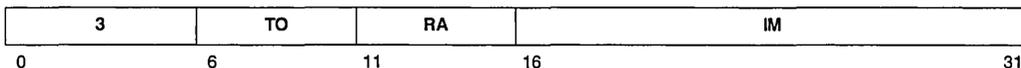
tw

Trap Word

Table 24-34. Extended Mnemonics for tw (continued)

Mnemonic	Operands	Function	Other Registers Altered
twle	RA, RB	Trap if (RA) logically less than or equal to (RB). <i>Extended mnemonic for tw 6,RA,RB</i>	
twlt	RA, RB	Trap if (RA) logically less than (RB). <i>Extended mnemonic for tw 2,RA,RB</i>	
twng	RA, RB	Trap if (RA) logically not greater than (RB). <i>Extended mnemonic for tw 6,RA,RB</i>	
twnl	RA, RB	Trap if (RA) logically not less than (RB). <i>Extended mnemonic for tw 5,RA,RB</i>	
twt	RA, RB	Trap if (RA) less than (RB). <i>Extended mnemonic for tw 16,RA,RB</i>	
twne	RA, RB	Trap if (RA) not equal to (RB). <i>Extended mnemonic for tw 24,RA,RB</i>	
twng	RA, RB	Trap if (RA) not greater than (RB). <i>Extended mnemonic for tw 20,RA,RB</i>	
twnl	RA, RB	Trap if (RA) not less than (RB). <i>Extended mnemonic for tw 12,RA,RB</i>	

twi TO, RA, IM



```

if ( ((RA) < EXTS(IM) ^ TO0 = 1) ∨
      ((RA) > EXTS(IM) ^ TO1 = 1) ∨
      ((RA) = EXTS(IM) ^ TO2 = 1) ∨
      ((RA) <sup>u</sup> EXTS(IM) ^ TO3 = 1) ∨
      ((RA) >sup>u</sup> EXTS(IM) ^ TO4 = 1) ) then TRAP (see details below)

```

Register RA is compared with the IM field, which has been sign-extended to 32 bits. If any comparison condition selected by the TO field is true, a TRAP occurs. The behavior of a TRAP depends upon the Debug Mode of the processor, as described below:

- If TRAP is not enabled as a debug event (DBCR[TDE] = 0 or DBCR[EDM,IDM] = 0,0):

TRAP causes a program interrupt. See “Program Interrupt” on page 10-40.

```

(SRR0) ← address of twi instruction
(SRR1) ← (MSR)
(ESR[PTR]) ← 1
(MSR[WE, EE, PR, DR, IR]) ← 0
PC ← EVPR0:15 || 0x0700

```

- If TRAP is enabled as an External debug event (DBCR[TDE] = 1 and DBCR[EDM] = 1):

TRAP goes to the Debug Stop state, to be handled by an external debugger with hardware control of the PPC405GP.

```

(DBSR[TIE]) ← 1
  In addition, if TRAP is also enabled as an Internal debug event (DBCR[IDM] = 1)
  and Debug Exceptions are disabled (MSR[DE] = 0), then report an imprecise event:
  (DBSR[IDE]) ← 1
PC ← address of twi instruction

```

- If TRAP is enabled as an Internal debug event and *not* an External debug event (DBCR[TDE] = 1 and DBCR[EDM,IDM] = 0,1) and Debug Exceptions are enabled (MSR[DE] = 1):

TRAP causes a Debug interrupt. See “Debug Interrupt” on page 10-44.

```

(SRR2) ← address of twi instruction
(SRR3) ← (MSR)
(DBSR[TIE]) ← 1
(MSR[WE, EE, PR, CE, DE, DR, IR]) ← 0
PC ← EVPR0:15 || 0x2000

```

- If TRAP is enabled as an Internal debug event and *not* an External debug event (DBCR[TDE] = 1 and DBCR[EDM,IDM] = 0,1) and Debug Exceptions are disabled (MSR[DE] = 0):

TRAP will report the debug event as an *imprecise* event and will cause a Program interrupt. See “Program Interrupt” on page 10-40.

twi

Trap Word Immediate

(SRR0) ← address of twi instruction
(SRR1) ← (MSR)
(ESR[PTR]) ← 1
(DBSR[TIE,IDE]) ← 1,1
(MSR[WE, EE, PR, DR, IR]) ← 0
PC ← EVPR_{0:15} || 0x0700

Registers Altered

- None

Programming Note

This instruction is inserted into the execution stream by a debugger to implement breakpoints, and is not typically used by application code.

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Table 24-35. Extended Mnemonics for twi

Mnemonic	Operands	Function	Other Registers Altered
tweqi	RA, IM	Trap if (RA) equal to EXTS(IM). <i>Extended mnemonic for twi 4,RA,IM</i>	
twgei	RA, IM	Trap if (RA) greater than or equal to EXTS(IM). <i>Extended mnemonic for twi 12,RA,IM</i>	
twgti	RA, IM	Trap if (RA) greater than EXTS(IM). <i>Extended mnemonic for twi 8,RA,IM</i>	
twlei	RA, IM	Trap if (RA) less than or equal to EXTS(IM). <i>Extended mnemonic for twi 20,RA,IM</i>	
twlgei	RA, IM	Trap if (RA) logically greater than or equal to EXTS(IM). <i>Extended mnemonic for twi 5,RA,IM</i>	
twlgti	RA, IM	Trap if (RA) logically greater than EXTS(IM). <i>Extended mnemonic for twi 1,RA,IM</i>	
twllei	RA, IM	Trap if (RA) logically less than or equal to EXTS(IM). <i>Extended mnemonic for twi 6,RA,IM</i>	
twllti	RA, IM	Trap if (RA) logically less than EXTS(IM). <i>Extended mnemonic for twi 2,RA,IM</i>	
twlngi	RA, IM	Trap if (RA) logically not greater than EXTS(IM). <i>Extended mnemonic for twi 6,RA,IM</i>	

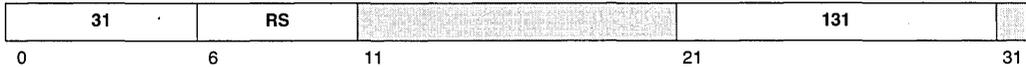
Table 24-35. Extended Mnemonics for twi (continued)

Mnemonic	Operands	Function	Other Registers Altered
twlnli	RA, IM	Trap if (RA) logically not less than EXTS(IM). <i>Extended mnemonic for twi 5,RA,IM</i>	
twlti	RA, IM	Trap if (RA) less than EXTS(IM). <i>Extended mnemonic for twi 16,RA,IM</i>	
twnei	RA, IM	Trap if (RA) not equal to EXTS(IM). <i>Extended mnemonic for twi 24,RA,IM</i>	
twngi	RA, IM	Trap if (RA) not greater than EXTS(IM). <i>Extended mnemonic for twi 20,RA,IM</i>	
twnli	RA, IM	Trap if (RA) not less than EXTS(IM). <i>Extended mnemonic for twi 12,RA,IM</i>	

wrtee

Write External Enable

wrtee RS



$MSR[EE] \leftarrow (RS)_{16}$

The 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 Note

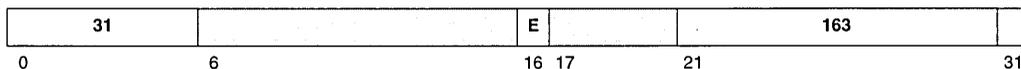
Execution of this instruction is privileged.

This instruction is used to provide atomic update of MSR[EE]. Typical usage is:

```
mfmsr Rn    #save EE in Rn[16]
wrteei 0    #Turn off EE
•          #Code with EE disabled
•
•
wrtee Rn    #restore EE without affecting any MSR changes that occurred in the disabled code
```

Architecture Note

This instruction part of the IBM PowerPC Embedded Operating Environment.

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 Note

Execution of this instruction is privileged.

This instruction is used to provide an atomic update of MSR[EE]. Typical usage is:

```

mfmsr Rn    #save EE in Rn[16]
wrteei 0    #Turn off EE
•          #Code with EE disabled
•
•
wrtee Rn    #restore EE without affecting any MSR changes that occurred in the disabled code

```

Architecture Note

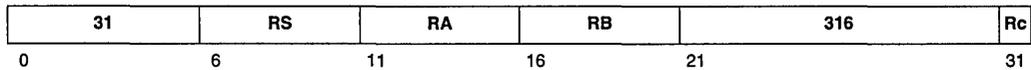
This instruction part of the IBM PowerPC Embedded Operating Environment.

xor

XOR

xor RA, RS, RB
xor. RA, RS, RB

Rc=0
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

- CR[CR0]_{LT, GT, EQ, SO} if Rc contains 1
- RA

Architecture Note

This instruction part of the IBM PowerPC Embedded Operating Environment.

xori RA, RS, IM



$$(RA) \leftarrow (RS) \oplus (160 \parallel IM)$$

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

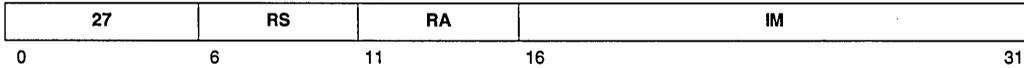
Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

xoris

XOR Immediate Shifted

xoris RA, RS, IM



$$(RA) \leftarrow (RS) \oplus (IM \ll 16)$$

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

Architecture Note

This instruction is part of the PowerPC User Instruction Set Architecture.

Chapter 25. Register Summary

The registers are grouped into categories, based on access mode: General Purpose Registers (GPRs), Special Purpose Registers (SPRs), Time Base Registers (TBRs), the Machine State Register (MSR), the Condition Register (CR), Device Control Registers (DCRs), and memory-mapped I/O (MMIO) registers.

25.1 Reserved Registers

Any register numbers not listed in the tables which follow are reserved, and should be neither read nor written. These reserved register numbers may be used for additional functions on future PowerPC embedded products.

25.2 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.

25.3 General Purpose Registers

The PPC405GP processor core provides 32 General Purpose Registers (GPRs). The contents of these registers can be loaded from memory using load instructions and stored to memory using store instructions. GPRs are also addressed by all integer instructions.

Table 25-1. PPC405GP General Purpose Registers

Mnemonic	Register Name	GPR Number		Access
		Decimal	Hex	
R0–R31	General Purpose Register 0–31	0–31	0x0–0x1F	Read/Write

25.4 Machine State Register and Condition Register

Because these registers are accessed using special instructions, they do not require addressing.

25.5 Special Purpose Registers

Special Purpose Registers (SPRs), which are part of the PowerPC Embedded Architecture, are accessed using the `mtspr` and `mfspr` instructions. SPRs control the use of the debug facilities, timers, interrupts, storage control attributes, and other architected processor resources.

Table 25-2 shows the mnemonics, names, and numbers of the SPRs. The columns under “SPRN” list the register numbers used as operands in assembler language coding of the **mf spr** and **mt spr** instructions. The column labeled “SPRF” lists the corresponding fields contained in the *machine code* of **mf spr** and **mt spr**. The SPRN field contains the five-bit subfields of the SPRF field, which are *reversed* in the machine code for the **mf spr** and **mt spr** instructions (SPRN ← SPRF_{5:9} || SPRF_{0:4}) for compatibility with the POWER Architecture. Note that the assembler handles the special coding transparently.

All SPRs are privileged, except the Count Register (CTR), the Link Register (LR), SPR General Purpose Registers (SPRG4–SPRG7, read-only), User SPR General Purpose Register (USPRG0), and the Fixed-point Exception Register (XER). Note that access to the Time Base Lower (TBL) and Time Base Upper (TBU) registers, when addressed as SPRs, is write-only and privileged. However, when addressed as Time Base Registers (TBRs), read access to these registers is not privileged. See “Time Base Registers” on page 25-3 for more information.

Table 25-2 lists the SPRs, their mnemonics and names, their numbers (SPRN) and the corresponding SPRF numbers, and access. All SPR numbers not listed are reserved, and should be neither read nor written.

Table 25-2. Special Purpose Registers

Mnemonic	Register Name	SPRN		SPRF	Access
		Decimal	Hex		
CCR0	Core Configuration Register 0	947	0x3B3	0x27D	Read/Write
CTR	Count Register	9	0x009	0x120	Read/Write
DAC1	Data Address Compare 1	1014	0x3F6	0x2DF	Read/Write
DAC2	Data Address Compare 2	1015	0x3F7	0x2FF	Read/Write
DBCR0	Debug Control Register 0	1010	0x3F2	0x25F	Read/Write
DBCR1	Debug Control Register 1	957	0x3BD	0x3BD	Read/Write
DBSR	Debug Status Register	1008	0x3F0	0x21F	Read/Clear
DCCR	Data Cache Cachability Register	1018	0x3FA	0x35F	Read/Write
DCWR	Data Cache Write-through Register	954	0x3BA	0x35D	Read/Write
DVC1	Data Value Compare 1	950	0x3B6	0x2DD	Read/Write
DVC2	Data Value Compare 2	951	0x3B7	0x2FD	Read/Write
DEAR	Data Error Address Register	981	0x3D5	0x2BE	Read/Write
ESR	Exception Syndrome Register	980	0x3D4	0x29E	Read/Write
EVPR	Exception Vector Prefix Register	982	0x3D6	0x2DE	Read/Write
IAC1	Instruction Address Compare 1	1012	0x3F4	0x29F	Read/Write
IAC2	Instruction Address Compare 2	1013	0x3F5	0x2B5	Read/Write
IAC3	Instruction Address Compare 3	948	0x3B4	0x29D	Read/Write
IAC4	Instruction Address Compare 4	949	0x3B5	0x2BD	Read/Write
ICCR	Instruction Cache Cachability Register	1019	0x3FB	0x37F	Read/Write
ICDBDR	Instruction Cache Debug Data Register	979	0x3D3	0x27E	Read-only
LR	Link Register	8	0x008	0x100	Read/Write
PID	Process ID	945	0x3B1	0x23D	Read/Write

Table 25-2. Special Purpose Registers (continued)

Mnemonic	Register Name	SPRN		SPRF	Access
		Decimal	Hex		
PIT	Programmable Interval Timer	987	0x3DB	0x37E	Read/Write
PVR	Processor Version Register	287	0x11F	0x3E8	Read-only
SGR	Storage Guarded Register	953	0x3B9	0x33D	Read/Write
SLER	Storage Little Endian Register	955	0x3BB	0x37D	Read/Write
SPRG0	SPR General 0	272	0x110	0x208	Read/Write
SPRG1	SPR General 1	273	0x111	0x228	Read/Write
SPRG2	SPR General 2	274	0x112	0x248	Read/Write
SPRG3	SPR General 3	275	0x113	0x268	Read/Write
SPRG4	SPR General 4	260	0x104	0x088	Read-only
SPRG4	SPR General 4	276	0x114	0x288	Read/Write
SPRG5	SPR General 5	261	0x105	0x0A8	Read-only
SPRG5	SPR General 5	277	0x115	0x2A8	Read/Write
SPRG6	SPR General 6	262	0x106	0x0C8	Read-only
SPRG6	SPR General 6	278	0x116	0x2C8	Read/Write
SPRG7	SPR General 7	263	0x107	0x0E8	Read-only
SPRG7	SPR General 7	279	0x117	0x2E8	Read/Write
SRR0	Save/Restore Register 0	26	0x01A	0x340	Read/Write
SRR1	Save/Restore Register 1	27	0x01B	0x360	Read/Write
SRR2	Save/Restore Register 2	990	0x3DE	0x3DE	Read/Write
SRR3	Save/Restore Register 3	991	0x3DF	0x3FE	Read/Write
SU0R	Storage User-defined 0 Register	956	0x3BC	0x39D	Read/Write
TBL	Time Base Lower	284	0x11C	0x388	Write-only
TBU	Time Base Upper	285	0x11D	0x3A8	Write-only
TCR	Timer Control Register	986	0x3DA	0x35E	Read/Write
TSR	Timer Status Register	984	0x3D8	0x31E	Read/Clear
USPRG0	User SPR General 0	256	0x100	0x008	Read/Write
XER	Fixed Point Exception Register	1	0x001	0x020	Read/Write
ZPR	Zone Protection Register	944	0x3B0	0x21D	Privileged

25.6 Time Base Registers

The PowerPC Architecture provides a 64-bit time base. Chapter 11, "Timer Facilities," describes the architected time base. In the PPC405GP, the time base is implemented as two 32-bit time base registers (TBRs). The low-order 32 bits of the time base are read from the TBL and the high-order 32 bits are read from the TBU.

User-mode access to the TBRs is read-only, and there is no explicitly privileged read access to the time base.

The **mftb** instruction reads from TBL and TBU. (Writing the time base is accomplished by moving the contents of a GPR to a pair of SPRs, which are also called TBL and TBU, using the **mtspr** instruction.)

Table 25-3 shows the mnemonics, names, and numbers of the TBRs. The columns under “TBRN” list the register numbers used as operands in assembler language coding of the **mftb** and **mtspr** instructions. The column labeled “TBRF” lists the corresponding fields contained in the *machine code* of **mftb** and **mtspr**. The TBRN field contains two five-bit subfields of the TBRF field; the subfields are *reversed* in the machine code for the **mftb** and **mtspr** instructions (TBRN ← TBRF_{5:9} || TBRF_{0:4}). Note that the assembler handles the special coding transparently.

Table 25-3. Time Base Registers

Mnemonic	Register Name	TBRN		TBRF	Access
		Decimal	Hex		
TBL	Time Base Lower (Read-only)	268	0x10C	0x188	Read-only
TBU	Time Base Upper (Read-only)	269	0x10D	0x1A8	Read-only

25.7 Device Control Registers

Device Control Registers (DCRs) are on-chip registers that are architecturally outside of the processor core. They are used to control, configure, and hold status for various functional units. DCRs are accessed using the **mfocr** and **mtocr** instructions.

The **mfocr** and **mtocr** 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.

25.7.1 Directly Accessed DCRs

The following DCRs are directly accessed, that is, they are accessed using their DCR numbers.

Table 25-4. Directly Accessed DCRs

Register	DCR Number	Access	Description
DCRs Used for Indirect Access			
SDRAM0_CFGADDR	0x010	R/W	Memory Controller Address Register
SDRAM0_CFGDATA	0x011	R/W	Memory Controller Data Register
EBC0_CFGADDR	0x012	R/W	Peripheral Controller Address Register
EBC0_CFGDATA	0x013	R/W	Peripheral Controller Data Register
DCP0_CFGADDR	0x014	R/W	Decompression Controller Address Register
DCP0_CFGDATA	0x015	R/W	Decompression Controller Data Register
On-Chip Memory			
OCM0_ISARC	0x018	R/W	OCM Instruction-Side Address Range Compare Register
OCM0_ISCNTL	0x019	R/W	OCM Instruction-Side Control Register
OCM0_DSARC	0x01A	R/W	OCM Data-Side Address Range Compare Register
OCM0_DSCNTL	0x01B	R/W	OCM Data-Side Control Register

Table 25-4. Directly Accessed DCRs (continued)

Register	DCR Number	Access	Description
On-Chip Buses			
PLB0_BESR	0x084	R/Clear	PLB Error Status Register
PLB0_BEAR	0x086	R/W	PLB Error Address Register
PLB0_ACR	0x087	R/W	PLB Arbiter Control Register
POB0_BESR0	0x0A0	R/Clear	PLB to OPB Error Status Register 0
POB0_BEAR	0x0A2	R	PLB to OPB Error Address Register
POB0_BESR1	0x0A4	R/Clear	PLB to OPB Error Status Register 1
Clocking, Power Management, and Chip Control			
CPC0_PLLMR	0x0B0	R/W	PLL Mode Register
CPC0_CR0	0x0B1	R/W	Chip Control Register 0
CPC0_CR1	0x0B2	R/W	Chip Control Register 1
CPC0_PSR	0x0B4	R	Chip Pin Strapping Register
CPC0_JTAGID	0xB5	R	JTAG ID Register
CPC0_SR	0x0B8	R	CPM Status Register
CPC0_ER	0x0B9	R/W	CPM Enable Register
CPC0_FR	0x0BA	R/W	CPM Force Register
Universal Interrupt Controller			
UIC0_SR	0x0C0	R/Clear	UIC Status Register
UIC0_ER	0x0C2	R/W	UIC Enable Register
UIC0_CR	0x0C3	R/W	UIC Critical Register
UIC0_PR	0x0C4	R/W	UIC Polarity Register
UIC0_TR	0x0C5	R/W	UIC Triggering Register
UIC0_MSR	0x0C6	R	UIC Masked Status Register
UIC0_VR	0x0C7	R	UIC Vector Register
UIC0_VCR	0x0C8	W	UIC 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
DMA0_DA0	0x102	R/W	DMA Destination Address Register 0
DMA0_SA0	0x103	R/W	DMA Source Address Register 0
DMA0_SG0	0x104	R/W	DMA Scatter/Gather Descriptor Address Register 0
DMA0_CR1	0x108	R/W	DMA Channel Control Register 1
DMA0_CT1	0x109	R/W	DMA Count Register 1
DMA0_DA1	0x10A	R/W	DMA Destination Address Register 1
DMA0_SA1	0x10B	R/W	DMA Source Address Register 1
DMA0_SG1	0x10C	R/W	DMA Scatter/Gather Descriptor Address Register 1
DMA0_CR2	0x110	R/W	DMA Channel Control Register 2
DMA0_CT2	0x111	R/W	DMA Count Register 2
DMA0_DA2	0x112	R/W	DMA Destination Address Register 2
DMA0_SA2	0x113	R/W	DMA Source Address Register 2
DMA0_SG2	0x114	R/W	DMA Scatter/Gather Descriptor Address Register 2

Table 25-4. Directly Accessed DCRs (continued)

Register	DCR Number	Access	Description
DMA0_CR3	0x118	R/W	DMA Channel Control Register 3
DMA0_CT3	0x119	R/W	DMA Count Register 3
DMA0_DA3	0x11A	R/W	DMA Destination Address Register 3
DMA0_SA3	0x11B	R/W	DMA Source Address Register 3
DMA0_SG3	0x11C	R/W	DMA Scatter/Gather Descriptor Address
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
Media Access Layer			
MAL0_CFG	0x180	R/W	MAL Configuration Register
MAL0_ESR	0x181	R/Clear	Error Status Register
MAL0_IER	0x182	R/W	Interrupt Enable Register
MAL0_TXCASR	0x184	R/W	Tx Channel Active Register (Set)
MAL0_TXCARR	0x185	R/W	Tx Channel Active Register (Reset)
MAL0_TXEOBISR	0x186	R/Clear	Tx End of Buffer Interrupt Status Register
MAL0_TXDEIR	0x187	R/Clear	Tx Descriptor Error Interrupt Register
MAL0_RXCASR	0x190	R/W	Rx Channel Active Register (Set)
MAL0_RXCARR	0x191	R/W	Rx Channel Active Register (Reset)
MAL0_RXEOBISR	0x192	R/Clear	Rx End of Buffer Interrupt Status Register
MAL0_RXDEIR	0x193	R/Clear	Rx Descriptor Error Interrupt Register
MAL0_TXCTP0R	0x1A0	R/W	Channel Tx 0 Channel Table Pointer Register
MAL0_TXCTP1R	0x1A1	R/W	Channel Tx 1 Channel Table Pointer Register
MAL0_RXCTP0R	0x1C0	R/W	Channel Rx 0 Channel Table Pointer Register
MAL0_RCBS0	0x1E0	R/W	Channel RX 0 Channel Buffer Size Register

25.7.2 Indirectly Accessed DCRs

The DCRs for the SDRAM controller, external bus controller (EBC), and decompression controller are indirectly accessed.

The following general procedure can be used to access the SDRAM controller, EBC, and decompression controller:

1. Write an offset to an address DCR.

Offsets for the SDRAM controller registers, listed in Table 25-6, are written to the Memory Controller Address Register (SDRAM0_CFGADDR); its address is in Table 25-5.

Offsets for the EBC registers, listed in Table 25-8, "Offsets for EBC Registers," on page 25-8, are written to the Peripheral Controller Address Register (EBC0_CFGADDR); its address is in Table 25-7, "EBC DCR Usage," on page 25-8.

Offsets for the decompression controller registers, listed in Table 25-10, “Offsets for Decompression Controller Registers,” on page 25-9, are written to the Decompression Controller Address Register (DCP0_CFGADDR); its address is in Table 25-9, “Decompression Controller DCR Usage,” on page 25-9.

2. Read data from or write data to a data DCR.

Data associated with the unit is written to or read from the target register.

25.7.2.1 Indirect Access of SDRAM Controller DCRs

The following procedure accesses the SDRAM controller registers listed in Table 25-5.

1. Write the offset from Table 25-6 to the Memory Controller Address Register (SDRAM0_CFGADDR).
2. Read data from or write data to the Memory Controller Data Register (SDRAM0_CFGDATA).

Table 25-5. SDRAM Controller DCR Usage

Register	DCR Number	Access	Description
SDRAM0_CFGADDR	0x010	R/W	Memory Controller Address Register
SDRAM0_CFGDATA	0x011	R/W	Memory Controller Data Register

Table 25-6. Offsets for SDRAM Controller Registers

Register	Offset	Access	Description
SDRAM0_BESR0	0x00	R/Clear	Bus Error Syndrome Register 0
SDRAM0_BESR1	0x08	R/Clear	Bus Error Syndrome Register 1
SDRAM0_BEAR	0x10	R/W	Bus Error Address Register
SDRAM0_CFG	0x20	R/W	Memory Controller Options
SDRAM0_RTR	0x30	R/W	Refresh Timer Register
SDRAM0_PMIT	0x34	R/W	Power Management Idle Timer
SDRAM0_B0CR	0x40	R/W	Memory Bank 0 Configuration
SDRAM0_B1CR	0x44	R/W	Memory Bank 1 Configuration
SDRAM0_B2CR	0x48	R/W	Memory Bank 2 Configuration
SDRAM0_B3CR	0x4C	R/W	Memory Bank 3 Configuration
SDRAM0_TR	0x80	R/W	SDRAM Timing Register
SDRAM0_ECCCFG	0x94	R/W	ECC Configuration
SDRAM0_ECCESR	0x98	R	ECC Error Status

25.7.2.2 Indirect Access of EBC DCRs

The following procedure accesses the EBC registers listed in Table 25-7.

1. Write the offset from Table 25-8 to the Peripheral Controller Address Register (EBC0_CFGADDR).
2. Read data from or write data to the Peripheral Controller Data Register (EBC0_CFGDATA).

Table 25-7. EBC DCR Usage

Register	DCR Number	Access	Description
EBC0_CFGADDR	0x012	R/W	Peripheral Controller Address Register
EBC0_CFGDATA	0x013	R/W	Peripheral Controller Data Register

Table 25-8. Offsets for EBC Registers

Register	Offset	Access	Description
EBC0_B0CR	0x00	R/W	Peripheral Bank 0 Configuration Register
EBC0_B1CR	0x01	R/W	Peripheral Bank 1 Configuration Register
EBC0_B2CR	0x02	R/W	Peripheral Bank 2 Configuration Register
EBC0_B3CR	0x03	R/W	Peripheral Bank 3 Configuration Register
EBC0_B4CR	0x04	R/W	Peripheral Bank 4 Configuration Register
EBC0_B5CR	0x05	R/W	Peripheral Bank 5 Configuration Register
EBC0_B6CR	0x06	R/W	Peripheral Bank 6 Configuration Register
EBC0_B7CR	0x07	R/W	Peripheral Bank 7 Configuration Register
EBC0_B0AP	0x10	R/W	Peripheral Bank 0 Access Parameters
EBC0_B1AP	0x11	R/W	Peripheral Bank 1 Access Parameters
EBC0_B2AP	0x12	R/W	Peripheral Bank 2 Access Parameters
EBC0_B3AP	0x13	R/W	Peripheral Bank 3 Access Parameters
EBC0_B4AP	0x14	R/W	Peripheral Bank 4 Access Parameters
EBC0_B5AP	0x15	R/W	Peripheral Bank 5 Access Parameters
EBC0_B6AP	0x16	R/W	Peripheral Bank 6 Access Parameters
EBC0_B7AP	0x17	R/W	Peripheral Bank 7 Access Parameters
EBC0_BEAR	0x20	R/W	Peripheral Bus Error Address Register
EBC0_BESR0	0x21	R/W	Peripheral Bus Error Status Register 0
EBC0_BESR1	0x22	R/W	Peripheral Bus Error Status Register 1
EBC0_CFG	0x23	R/W	External Peripheral Control Register

25.7.2.3 Indirect Access of Decompression Controller DCRs

The following procedure accesses the decompression controller registers listed in Table 25-9.

1. Write the offset from Table 25-10 to the Decompression Controller Address Register (DCP0_CFGADDR).
2. Read data from or write data to the Decompression Controller Data Register (DCP0_CFGDATA).

Table 25-9. Decompression Controller DCR Usage

Register	DCR Number	Access	Description
DCP0_CFGADDR	0x014	R/W	Decompression Controller Address Register
DCP0_CFGDATA	0x015	R/W	Decompression Controller Data Register

Table 25-10. Offsets for Decompression Controller Registers

Register	Offset	Access	Description
DCP0_ITOR0	0x00	R/W	Index Table Origin Register 0
DCP0_ITOR1	0x01	R/W	Index Table Origin Register 1
DCP0_ITOR2	0x02	R/W	Index Table Origin Register 2
DCP0_ITOR3	0x03	R/W	Index Table Origin Register 3
DCP0_ADDR0	0x04	R/W	Address Decode Definition Register 0
DCP0_ADDR1	0x05	R/W	Address Decode Definition Register 1
DCP0_CFG	0x40	R/W	Decompression Core Configuration Register
DCP0_ID	0x41	R	Decompression Core ID Register
DCP0_VER	0x42	R	Decompression Core Version Number Register
DCP0_PLBBEAR	0x50	R	Bus Error Address Register (PLB address)
DCP0_MEMBEAR	0x51	R	Bus Error Address Register (DCP to EBC address)
DCP0_ESR	0x52	R/Clear	Bus Error Status Register 0 (masters 0-3)
DCP0_RAM0– DCP0_RAM3FF	0x400–0x7FF	R/W	Decode Tables

25.8 MMIO Registers

Some registers associated with on-chip peripherals are MMIO registers. Such registers are accessed using load/store instructions.

25.8.1 Directly Accessed MMIO Registers

Directly-accessed MMIO registers are accessed using load/store instructions that contain the register addresses. Table 25-11 lists the directly-accessed MMIO registers.

Table 25-11. Directly Accessed MMIO Registers

Register	Address	Access	Description
MMIO Registers Used for Indirect Access			
PCIC0_CFGADDR	0xEEC00000	R/W	PCI Configuration Address Register
PCIC0_CFGDATA	0xEEC00004	R/W	PCI Configuration Data Register
PCI-to-PCI Bridge			
PCIL0_PMM0LA	0xEF400000	R/W	PMM 0 Local Address
PCIL0_PMM0MA	0xEF400004	R/W	PMM 0 Mask/Attribute
PCIL0_PMM0PCILA	0xEF400008	R/W	PMM 0 PCI Low Address
PCIL0_PMM0PCIHA	0xEF40000C	R/W	PMM 0 PCI High Address
PCIL0_PMM1LA	0xEF400010	R/W	PMM 1 Local Address
PCIL0_PMM1MA	0xEF400014	R/W	PMM 1 Mask/Attribute
PCIL0_PMM1PCILA	0xEF400018	R/W	PMM 1 PCI Low Address
PCIL0_PMM1PCIHA	0xEF40001C	R/W	PMM 1 PCI High Address
PCIL0_PMM2LA	0xEF400020	R/W	PMM 2 Local Address
PCIL0_PMM2MA	0xEF400024	R/W	PMM 2 Mask/Attribute
PCIL0_PMM2PCILA	0xEF400028	R/W	PMM 2 PCI Low Address
PCIL0_PMM2PCIHA	0xEF40002C	R/W	PMM 2 PCI High Address
PCIL0_PTM1MS	0xEF400030	R/W	PTM 1 Memory Size
PCIL0_PTM1LA	0xEF400034	R/W	PTM 1 Local Address
PCIL0_PTM2MS	0xEF400038	R/W	PTM 2 Memory Size
PCIL0_PTM2LA	0xEF40003C	R/W	PTM 2 Local Address
Serial Ports			
UART0_RBR	0xEF600300	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	0xEF600301	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	0xEF600302	R	UART 0 Interrupt Identification Register
UART0_FCR	0xEF600302	W	UART 0 FIFO Control Register
UART0_LCR	0xEF600303	R/W	UART 0 Line Control Register

Table 25-11. Directly Accessed MMIO Registers (continued)

Register	Address	Access	Description
UART0_MCR	0xEF600304	R/W	UART 0 Modem Control Register
UART0_LSR	0xEF600305	R/W	UART 0 Line Status Register
UART0_MSR	0xEF600306	R/W	UART 0 Modem Status Register
UART0_SCR	0xEF600307	R/W	UART 0 Scratch Register
UART1_RBR	0xEF600400	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	0xEF600401	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	0xEF600402	R	UART 1 Interrupt Identification Register
UART1_FCR	0xEF600402	W	UART 1 FIFO Control Register
UART1_LCR	0xEF600403	R/W	UART 1 Line Control Register
UART1_MCR	0xEF600404	R/W	UART 1 Modem Control Register
UART1_LSR	0xEF600405	R/W	UART 1 Line Status Register
UART1_MSR	0xEF600406	R/W	UART 1 Modem Status Register
UART1_SCR	0xEF600407	R/W	UART 1 Scratch Register
Inter-Integrated Circuit			
IIC0_MDBUF	0xEF600500	R/W	IIC0 Master Data Buffer
IIC0_SDBUF	0xEF600502	R/W	IIC0 Slave Data Buffer
IIC0_LMADR	0xEF600504	R/W	IIC0 Low Master Address
IIC0_HMADR	0xEF600505	R/W	IIC0 High Master Address
IIC0_CNTL	0xEF600506	R/W	IIC0 Control
IIC0_MDCNTL	0xEF600507	R/W	IIC0 Mode Control
IIC0_STS	0xEF600508	R/W	IIC0 Status
IIC0_EXTSTS	0xEF600509	R/W	IIC0 Extended Status
IIC0_LSADR	0xEF60050A	R/W	IIC0 Low Slave Address
IIC0_HSADR	0xEF60050B	R/W	IIC0 High Slave Address
IIC0_CLKDIV	0xEF60050C	R/W	IIC0 Clock Divide
IIC0_INTRMSK	0xEF60050D	R/W	IIC0 Interrupt Mask
IIC0_XFRCNT	0xEF60050E	R/W	IIC0 Transfer Count
IIC0_XTCNTLSS	0xEF60050F	R/W	IIC0 Extended Control and Slave Status
IIC0_DIRECTCNTL	0xEF600510	R/W	IIC0 Direct Control
OPB Arbiter			
OPBA0_PR	0xEF600600	R/W	OPB Arbiter Priority Register
OPBA0_CR	0xEF600601	R/W	OPB Arbiter Control Register

Table 25-11. Directly Accessed MMIO Registers (continued)

Register	Address	Access	Description
General-Purpose I/O			
GPIO0_OR	0xEF600700	R/W	GPIO0_IRO Output Register
GPIO0_TCR	0xEF600704	R/W	GPIO0_IRO Three-State Control Register
GPIO0_ODR	0xEF600718	R/W	GPIO0_IRO Open Drain Register
GPIO0_IR	0xEF60071C	R	GPIO0_IRO Input Register
Ethernet			
EMAC0_MR0	0xEF600800	R/W	Mode Register 0
EMAC0_MR1	0xEF600804	R/W	Mode Register 1
EMAC0_TMR0	0xEF600808	R/W	Transmit Mode Register 0
EMAC0_TMR1	0xEF60080C	R/W	Transmit Mode Register 1
EMAC0_RMR	0xEF600810	R/W	Receive Mode Register
EMAC0_ISR	0xEF600814	R/W	Interrupt Status Register
EMAC0_ISER	0xEF600818	R/W	Interrupt Status Enable Register
EMAC0_IAHR	0xEF60081C	R/W	Individual Address High
EMAC0_IALR	0xEF600820	R/W	Individual Address Low
EMAC0_VTPID	0xEF600824	R/W	VLAN TPID Register
EMAC0_VTCI	0xEF600828	R/W	VLAN TCI Register
EMAC0_PTR	0xEF60082C	R/W	Pause Timer Register
EMAC0_IAHT1	0xEF600830	R/W	Individual Address Hash Table 1
EMAC0_IAHT2	0xEF600834	R/W	Individual Address Hash Table 2
EMAC0_IAHT3	0xEF600838	R/W	Individual Address Hash Table 3
EMAC0_IAHT4	0xEF60083C	R/W	Individual Address Hash Table 4
EMAC0_GAHT1	0xEF600840	R/W	Group Address Hash Table 1
EMAC0_GAHT2	0xEF600844	R/W	Group Address Hash Table 2
EMAC0_GAHT3	0xEF600848	R/W	Group Address Hash Table 3
EMAC0_GAHT4	0xEF60084C	R/W	Group Address Hash Table 4
EMAC0_LSAH	0xEF600850	R	Last Source Address Low
EMAC0_LSAL	0xEF600854	R	Last Source Address High
EMAC0_IPGVR	0xEF600858	R/W	Inter-Packet Gap Value Register
EMAC0_STACR	0xEF60085C	R/W	STA Control Register
EM0TSDRAM0_TRTR	0xEF600860	R/W	Transmit Request Threshold Register
EMAC0_RWMR	0xEF600864	R/W	Receive Low/High Water Mark Register

25.8.2 Indirectly Accessed MMIO Registers

The PCI configuration registers, listed in Table 25-13, “PCI Configuration Registers,” on page 25-13, are indirectly accessed.

The following procedure accesses the PCI configuration registers, using the address and data registers listed in Table 25-12:

1. OR the Enable/Bus/Device/Function bits with the high-order 6 bits of the offset from Table 25-13 and write the result to register PCI Configuration Address Register (PCIC0_CFGADDR).
2. OR the low-order 2 bits of the offset from Table 25-13 with PCI Configuration Data Register (PCIC0_CFGDATA) to form an address.
3. Read data from or write data to the address.

Table 25-12. PCI Configuration Address and Data Registers

Register	Address	Access	Description
PCIC0_CFGADDR	0xEEC00000	R/W	PCI Configuration Address Register
PCIC0_CFGDATA	0xEEC00004	R/W	PCI Configuration Data Register

Table 25-13. PCI Configuration Registers

Register	Offset	Access		Description
		PLB	PCI	
PCIC0_VENDID	0x01–0x00	R/W	R	PCI Vendor ID
PCIC0_DEVID	0x03–0x02	R/W	R	PCI Device ID
PCIC0_CMD	0x05–0x04	R/W	R/W	PCI Command Register
PCIC0_STATUS	0x07–0x06	R/W	R/W	PCI Status Register
PCIC0_REVID	0x08	R/W	R/W	PCI Revision ID
PCIC0_CLS	0x0B–0x09	R/W	R	PCI Class Register
PCIC0_CACHELS	0x0C	R	R	PCI Cache Line Size
PCIC0_LATTIM	0x0D	R/W	R/W	PCI Latency Timer
PCIC0_HDTYPE	0x0E	R	R	PCI Header Type
PCIC0_BIST	0x0F	R	R	PCI Built In Self Test Control
PCIC0_BAR0	0x13–0x10	R	R	PCI Reserved BAR 0
PCIC0_PTM1BAR	0x17–0x14	R/W	R/W	PCI PTM 1 BAR
PCIC0_PTM2BAR	0x1B–0x18	R/W	R/W	PCI PTM 2 BAR
PCIC0_BAR3	0x1F–0x1C	—	—	PCI Reserved BAR 3
PCIC0_BAR4	0x23–0x20	—	—	PCI Reserved BAR 4
PCIC0_BAR5	0x24–0x27	—	—	PCI Reserved BAR 5
PCIC0_CISPTR	0x2B–0x28	—	—	Unused Cardbus CIS Pointer
PCIC0_SBSYSVID	0x2D–0x2C	R/W	R	PCI Subsystem Vendor ID
PCIC0_SBSYSID	0x2F–0x2E	R/W	R	PCI Subsystem ID
PCIC0_EROMBA	0x33–0x30	—	—	Unused Expansion ROM Base Address
PCIC0_CAP	0x34	R	R	PCI Capabilities Pointer
PCIC0_INTLN	0x3C	R/W	R/W	PCI Interrupt Line
PCIC0_INTPN	0x3D	R	R	PCI Interrupt Pin

Table 25-13. PCI Configuration Registers (continued)

Register	Offset	Access		Description
		PLB	PCI	
PCIC0_MINGNT	0x3E	R	R	PCI Minimum Grant
PCIC0_MAXLTNCY	0x3F	R	R	PCI Maximum Latency
PCIC0_ICS	0x44	R/W	R/W	PCI Interrupt Control/Status
PCIC0_ERREN	0x48	R/W	R/W	Error Enable
PCIC0_ERRSTS	0x49	R/W	R/W	Error Status
PCIC0_BRDGOPT1	0x4B–0x4A	R/W	R/W	PCI Bridge Options 1
PCIC0_PLBBESR0	0x4F–0x4C	R/W	R/W	PLB Slave Error Syndrome 0
PCIC0_PLBBESR1	0x53–0x50	R/W	R/W	PLB Slave Error Syndrome 1
PCIC0_PLBBEAR	0x57–0x54	R/W	R/W	PLB Slave Error Address Register
PCIC0_CAPID	0x58	R	R	Capability Identifier
PCIC0_NEXTIPTR	0x59	R	R	Next Item Pointer
PCIC0_PMC	0x5B–0x5A	R	R	Power Management Capabilities
PCIC0_PMCSR	0x5D–0x5C	R/W	R/W	Power Management Control Status
PCIC0_PMCSRBASE	0x5E	R	R	PMCSR PCI to PCI bridge Support Extensions
PCIC0_DATA	0x5F	—	—	Unused Data
PCIC0_BRDGOPT2	0x63–0x60	R/W	R/W	PCI Bridge Options 2
PCIC0_PMSCRR	0x64	R/W	R/W	Power Management State Change Request Register

25.9 Alphabetical Register Listing

The following pages list the registers available in the PPC405GP. For each register, the following information is supplied:

- Register mnemonic and name
- Cross
- Register type (SPR or TBR)
- Register number (address)
- 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 mnemonic, field bit location, field name, and the function associated with various field values

SPR 0x3B3

See “Cache Control and Debugging Features” on page 4-11.

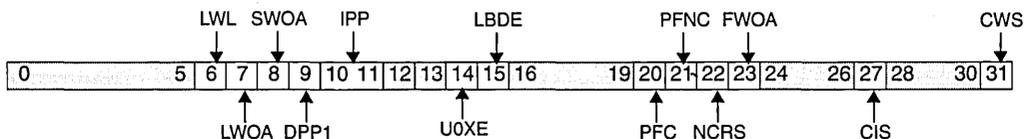


Figure 25-1. Core Configuration Register 0 (CCR0)

0:5		Reserved
6	LWL	Load Word as Line 0 The DCU performs load misses or non-cachable loads as words, halfwords, or bytes, as requested 1 For load misses or non-cachable loads, the DCU moves eight words (including the target word) into the line buffer
7	LWOA	Load Without Allocate 0 Load misses result in line fills 1 Load misses do not result in a line fill, but in non-cachable loads
8	SWOA	Store Without Allocate 0 Store misses result in line fills 1 Store misses do not result in line fills, but in non-cachable stores
9	DPP1	DCU PLB Priority Bit 1 0 DCU PLB priority 0 on bit 1 1 DCU PLB priority 1 on bit 1 Note: DCU logic dynamically controls DCU priority bit 0.
10:11	IPP	ICU PLB Priority Bits 0:1 00 Lowest ICU PLB priority 01 Next to lowest ICU PLB priority 10 Next to highest ICU PLB priority 11 Highest ICU PLB priority
12:13		Reserved
14	U0XE	Enable U0 Exception 0 Enables the U0 exception 1 Disables the U0 exception
15	LBDE	Load Debug Enable 0 Load data is invisible on data-side (on-chip memory (OCM)) 1 Load data is visible on data-side OCM
16:19		Reserved
20	PFC	ICU Prefetching for Cachable Regions 0 Disables prefetching for cachable regions 1 Enables prefetching for cachable regions

CCR0 (cont.)

Core Configuration Register 0

21	PFNC	ICU Prefetching for Non-Cachable Regions 0 Disables prefetching for non-cachable regions 1 Enables prefetching for non-cachable regions
22	NCRS	Non-cachable ICU request size 0 Requests are for four-word lines 1 Requests are for eight-word lines
23	FWOA	Fetch Without Allocate 0 An ICU miss results in a line fill. 1 An ICU miss does not cause a line fill, but results in a non-cachable fetch.
24:26		Reserved
27	CIS	Cache Information Select 0 Information is cache data. 1 Information is cache tag.
28:30		Reserved
31	CWS	Cache Way Select 0 Cache way is A. 1 Cache way is B.

DCR 0x0B1

See "Chip Control Register 0 (CPC0_CR0)" on page 7-12.

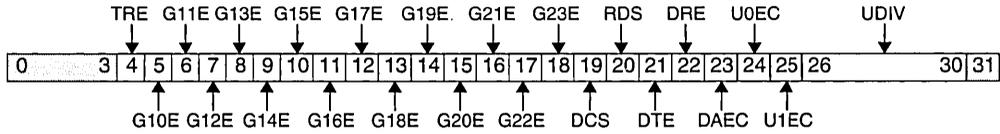


Figure 25-2. Chip Control Register 0 (CPC0_CR0)

0:3		Reserved	
4	TRE	CPU Trace Enable 0 GPIO1-9 are enabled 1 GPIO1-9 are disabled	Trace interface cannot be used when GPIO is enabled.
5	G10E	GPIO 10 Enable 0 Enable $\overline{\text{PerCS1}}$ as a chip select 1 Enable $\overline{\text{PerCS1}}$ as GPIO10	
6	G11E	GPIO 11 Enable 0 Enable $\overline{\text{PerCS2}}$ as a chip select 1 Enable $\overline{\text{PerCS2}}$ as GPIO11	
7	G12E	GPIO 12 Enable 0 Enable $\overline{\text{PerCS3}}$ as a chip select 1 Enable $\overline{\text{PerCS3}}$ as GPIO12	
8	G13E	GPIO 13 Enable 0 Enable $\overline{\text{PerCS4}}$ as a chip select 1 Enable $\overline{\text{PerCS4}}$ as GPIO13	
9	G14E	GPIO 14 Enable 0 Enable $\overline{\text{PerCS5}}$ as a chip select 1 Enable $\overline{\text{PerCS5}}$ as GPIO14	
10	G15E	GPIO 15 Enable 0 Enable $\overline{\text{PerCS6}}$ as a chip select 1 Enable $\overline{\text{PerCS6}}$ as GPIO15	
11	G16E	GPIO 16 Enable 0 Enable $\overline{\text{PerCS7}}$ as a chip select 1 Enable $\overline{\text{PerCS7}}$ as GPIO16	
12	G17E	GPIO 17 Enable 0 Enable interrupt IRQ0 as an interrupt 1 Enable interrupt IRQ0 as GPIO17	The purpose of GPIO_17_EN through GPIO_23_EN is to isolate the interrupt controller from activity on a shared pin when that pin is being used as a GPIO. For instance, when G17E is set to a 1, IRQ0 at the UIC will always be forced to a zero. Note: Setting G17E to a 0 will not prevent GPIO channel 17 (if configured as an output) from creating contention with the off-chip source of the IRQ input. Therefore, be sure to leave the shared GPIO channel disabled when using the pin as an interrupt input.

CPC0_CR0 (cont.)

Chip Control Register 0

13	G18E	GPIO 18 Enable 0 Enable interrupt IRQ1 as an interrupt 1 Enable interrupt IRQ1 as GPIO18
14	G19E	GPIO 19 Enable 0 Enable interrupt IRQ2 as an interrupt 1 Enable interrupt IRQ2 as GPIO19
15	G20E	GPIO 20 Enable 0 Enable interrupt IRQ3 as an interrupt 1 Enable interrupt IRQ3 as GPIO20
16	G21E	GPIO 21 Enable 0 Enable interrupt IRQ4 as an interrupt 1 Enable interrupt IRQ4 as GPIO21
17	G22E	GPIO 22 Enable 0 Enable interrupt IRQ5 as an interrupt 1 Enable interrupt IRQ5 as GPIO22
18	G23E	GPIO 23 Enable 0 Enable interrupt IRQ6 as an interrupt 1 Enable interrupt IRQ6 as GPIO23
19	DCS	DSR/CTS select 0 DSR is selected. 1 CTS is selected.
20	RDS	RTS/DTR select 0 RTS is selected. 1 DTR is selected.
21	DTE	DMA Transmit Enable for UART0 0 DMA transmit channel is disabled. 1 DMA transmit channel is enabled.
22	DRE	DMA Receive Enable for UART0 0 DMA receive channel is disabled. 1 DMA receive channel is enabled.
23	DAEC	DMA Allow Enable Clear for UART0 0 DTE and DRE for UART0 are not cleared when the UART receives a corresponding terminal count. 1 DTE and DRE for UART0 are cleared when the UART receives a corresponding terminal count.
24	U0EC	Select External Clock for UART0 0 UART0 uses the internally derived serial clock 1 UART0 uses the external serial clock input
25	U1EC	Select External Clock for UART1 0 UART1 uses the internally derived serial clock 1 UART1 uses the external serial clock input

CPC0_CR0 (cont.)

Chip Control Register 0

26:30	UDIV	UART Internal Clock Divisor 00000 Divide by 1 00001 Divide by 2 00010 Divide by 3 . . . 11110 Divide by 31 11111 Divide by 32	UDIV specifies the divisor of the CPU clock frequency used to derive a UART serial clock frequency. For example, if the CPU is running at 200MHz, a divider value of 20 sets the serial clock frequency to 10MHz. Note: The maximum serial clock frequency is less than $1/2 \times \text{OPB frequency}$
31		Reserved	

CPC0_CR1

Chip Control Register 1

DCR 0x0B2

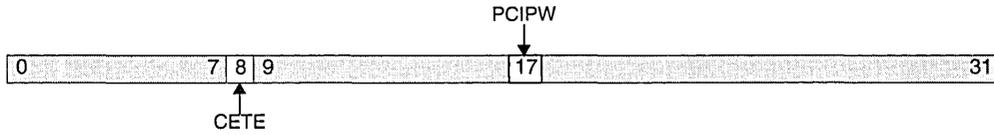


Figure 25-3. Chip Control Register 1 (CPC0_CR1)

0:7		Reserved
8	CETE	CPU External Timer Enable 0 CPU timers increment at CPU clock frequency. 1 CPU timers increment at frequency of TmrClk input.
9:16		Reserved
17	PCIPW	PCI Interrupt Enable/Peripheral Write Enable 0 PCI interrupt signal can be used. 1 Peripheral write enable signal can be enabled.
18:31		Reserved

DCR 0x0B9

See “CPM Enable Register (CPC0_ER)” on page 13-3.

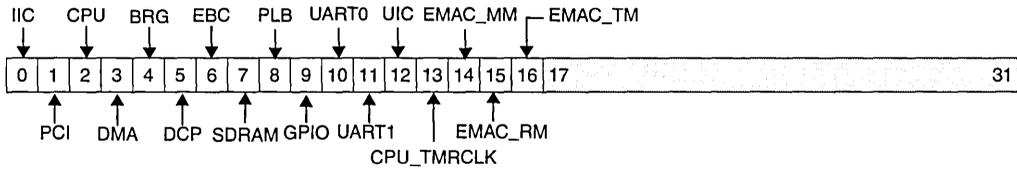


Figure 25-4. CPM Enable Register (CPC0_ER)

0	IIC	IIC Interface	Class 3
1	PCI	PCI Bridge	Class 3
2	CPU	PPC405GP Processor Core	Class 2
3	DMA	DMA Controller	Class 2
4	BRG	PLB to OPB Bridge	Class 2
5	DCP	CodePack	Class 2
6	EBC	ROM/SRAM Peripheral Controller	Class 2
7	SDRAM	SDRAM Memory Controller	Class 2
8	PLB	PLB Bus Arbiter	Class 2
9	GPIO	General Purpose Interrupt Controller	Class 1
10	UART0	Serial Port 0	Class 1
11	UART1	Serial Port 1	Class 1
12	UIC	Universal Interrupt Controller	Class 1
13	CPU_TMRCLK	CPU Timers	Class 1
14	EMAC_MM	Ethernet MM Unit	Class 1
15	EMAC_RM	Ethernet RM Unit	Class 1
16	EMAC_TM	Ethernet TM Unit	Class 1
17:31		Reserved	

CPC0_FR

CPM Force Register

DCR 0x0BA

See "CPM Force Register (CPC0_FR)" on page 13-3.

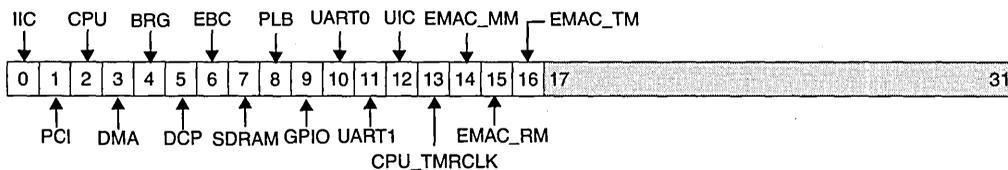


Figure 25-5. CPM Force Register (CPC0_FR)

0	IIC	IIC Interface	Class 3
1	PCI	PCI Bridge	Class 3
2	CPU	PPC405GP Processor Core	Class 2
3	DMA	DMA Controller	Class 2
4	BRG	PLB to OPB Bridge	Class 2
5	DCP	CodePack	Class 2
6	EBC	ROM/SRAM Peripheral Controller	Class 2
7	SDRAM	SDRAM Memory Controller	Class 2
8	PLB	PLB Bus Arbiter	Class 2
9	GPIO	General Purpose Interrupt Controller	Class 1
10	UART0	Serial Port 0	Class 1
11	UART1	Serial Port 1	Class 1
12	UIC	Universal Interrupt Controller	Class 1
13	CPU_TMRCLK	CPU Timers	Class 1
14	EMAC_MM	Ethernet MM Unit	Class 1
15	EMAC_RM	Ethernet RM Unit	Class 1
16	EMAC_TM	Ethernet TM Unit	Class 1
17:31		Reserved	

CPC0_JTAGID

JTAG ID Register

DCR 0x0B5 Read-Only

See "JTAG ID Register (CPC0_JTAGID)" on page 12-4.

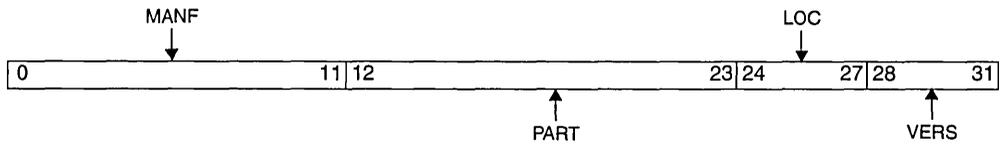


Figure 25-6. JTAG ID Register (CPC0_JTAGID)

0:11	MANF	Manufacturer Identifier
12:23	PART	Part Number
24:27	LOC	Developer Location
28:31	VERS	Version

CPC0_PLLMR

PLL Mode Register

DCR 0x0B

See “PLL Mode Register (CPC0_PLLMR)” on page 7-10.

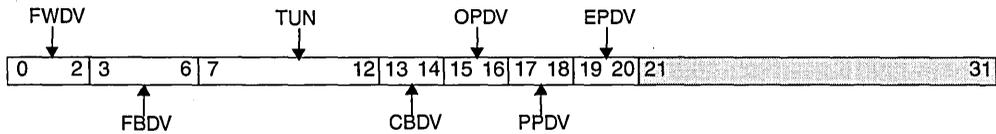


Figure 25-7. PLL Mode Register (CPC0_PLLMR)

0:2	FWDV	<p>Forward Divisor</p> <p>000 Forward divisor is 8. 001 Forward divisor is 7. 010 Forward divisor is 6. 011 Forward divisor is 5. 100 Forward divisor is 4. 101 Forward divisor is 3. 110 Forward divisor is 2. 111 Forward divisor is 1.</p>	<p>PLLOUTA Value:</p> <p>50 MHz–100 MHz 58 MHz–114 MHz 66 MHz–134 MHz 80 MHz–160 MHz 100 MHz–200 MHz 133 MHz–267 MHz 200 MHz–400 MHz 400 MHz–800 MHz</p> <p>Note: PLLOUTA is the CPU clock in PPC405GP.</p>
3:6	FBDV	<p>Feedback Divisor</p> <p>0000 Feedback divisor is 16. 0001 Feedback divisor is 1. 0010 Feedback divisor is 2. 0011 Feedback divisor is 3. 0100 Feedback divisor is 4. 0101 Feedback divisor is 5. 0110 Feedback divisor is 6. 0111 Feedback divisor is 7. 1000 Feedback divisor is 8. 1001 Feedback divisor is 9. 1010 Feedback divisor is 10. 1011 Feedback divisor is 11. 1100 Feedback divisor is 12. 1101 Feedback divisor is 13. 1110 Feedback divisor is 14. 1111 Feedback divisor is 15.</p>	
7:12	TUN	<p>TUNE[5:0] Field</p>	<p>Note: The tune bits adjust parameters that control PLL jitter. The recommended values minimize jitter for the PLL implemented in the PPC405GP. These bits are shown for information only, and do not require modification except in special clocking circumstances, such as spread spectrum clocking. For details on the use of spread spectrum clock generators (SSCGs) with the PPC405GP, visit the technical documents area of the IBM PowerPC web site.</p>

CPC0_PLLMR (cont.)

PLL Mode Register

13:14	CBDV	CPU:PLB Frequency Divisor 00 CPU:PLB divisor is 1 01 CPU:PLB divisor is 2 10 CPU:PLB divisor is 3 11 CPU:PLB divisor is 4	
15:16	OPDV	OPB-PLB Frequency Divisor 00 OPB-PLB divisor is 1 01 OPB-PLB divisor is 2 10 OPB-PLB divisor is 3 11 OPB-PLB divisor is 4	
17:18	PPDV	PCI-PLB Frequency Divisor 00 PCI-PLB divisor is 1 01 PCI-PLB divisor is 2 10 PCI-PLB divisor is 3 11 PCI-PLB divisor is 4	See "PCI Clocks" on page 7-8 for details regarding asynchronous PCI clocking and how it relates to synchronous clocking.
19:20	EPDV	External Bus-PLB Frequency Divisor 00 External bus-PLB divisor ratio is 2:1 01 External bus-PLB divisor ratio is 3:1 10 External bus-PLB divisor ratio is 4:1 11 External bus-PLB divisor ratio is 5:1	
21:31		Reserved	

CPC0_PSR

Chip Pin Strapping Register

DCR 0x0B3 Read-Only

See "Chip Pin Strapping Register (CPC0_PSR)" on page 9-1.

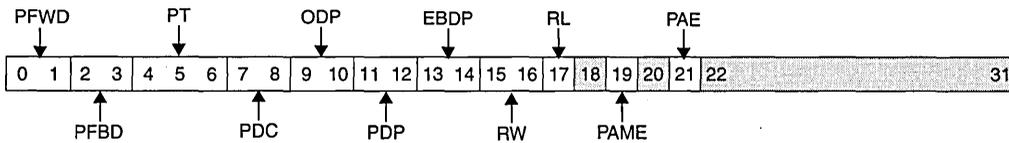


Figure 25-8. Chip Pin Strapping Register (CPC0_PSR)

0:1	PFWD	PLL Forward Divider 00 Bypass Mode 01 Divide by 3 10 Divide by 4 11 Divide by 6	
2:3	PFBD	PLL Feedback Divider 00 Divide By 1 01 Divide By 2 10 Divide By 3 11 Divide By 4	
4:6	PT	PLL Tuning 000 Choice 1; TUNE[5:0] = 010001 001 Choice 2; TUNE[5:0] = 010010 010 Choice 3; TUNE[5:0] = 010011 011 Choice 4; TUNE[5:0] = 010100 100 Choice 5; TUNE[5:0] = 010101 101 Choice 6; TUNE[5:0] = 010110 110 Choice 7; TUNE[5:0] = 010111 111 Choice 8; TUNE[5:0] = 100100	Note: The tune bits adjust parameters that control PLL jitter. The recommended values minimize jitter for the PLL implemented in the PPC405GP. These bits are shown for information only, and do not require modification except in special clocking circumstances, such as spread spectrum clocking. For details on the use of spread spectrum clock generators (SSCGs) with the PPC405GP, visit the technical documents area of the IBM PowerPC web site.
7:8	PDC	PLB Divider from CPU 00 Divide By 1 01 Divide By 2 10 Divide By 3 11 Divide By 4	
9:10	ODP	OPB Divider from PLB 00 Divide By 1 01 Divide By 2 10 Divide By 3 11 Divide By 4	
11:12	PDP	PCI Divider from PLB 00 Divide By 1 01 Divide By 2 10 Divide By 3 11 Divide By 4	

CPC0_PSR (cont.)

Chip Pin Strapping Register

13:14	EBDP	External Bus Divider from PLB 00 Divide By 2 01 Divide By 3 10 Divide By 4 11 Divide By 5
15:16	RW	ROM Width 00 8-bit ROM 01 16-bit ROM 10 32-bit ROM 11 Reserved
17	RL	ROM Location 0 405GP Peripheral Attach 1 405GP PCI Attach
18		Reserved
19	PAME	PCI Asynchronous Mode Enable 0 Synchronous PCI Mode 1 Asynchronous Mode
20		Reserved
21	PAE	PCI Arbiter Enable 0 Internal Arbiter Disabled 1 Internal Arbiter Enabled
22:31		Reserved

CPC0_SR

CPM Status Register

DCR 0x0B8 Read-Only

See “CPM Status Register (CPC0_SR)” on page 13-3.

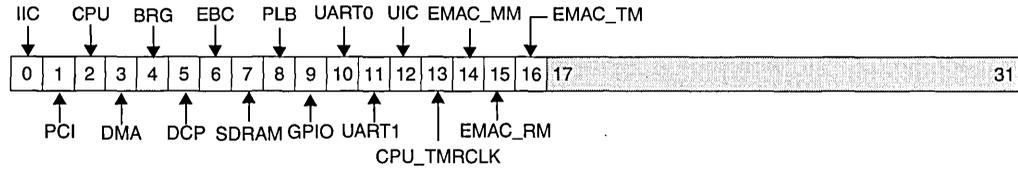


Figure 25-9. CPM Status Register (CPC0_SR)

0	IIC	IIC Interface	Class 3
1	PCI	PCI Bridge	Class 3
2	CPU	PPC405GP Processor Core	Class 2
3	DMA	DMA Controller	Class 2
4	BRG	PLB to OPB Bridge	Class 2
5	DCP	CodePack	Class 2
6	EBC	ROM/SRAM Peripheral Controller	Class 2
7	SDRAM	SDRAM Memory Controller	Class 2
8	PLB	PLB Bus Arbiter	Class 2
9	GPIO	General Purpose Interrupt Controller	Class 1
10	UART0	Serial Port 0	Class 1
11	UART1	Serial Port 1	Class 1
12	UIC	Universal Interrupt Controller	Class 1
13	CPU_TMRCLK	CPU Timers	Class 1
14	EMAC_MM	Ethernet MM Unit	Class 1
15	EMAC_RM	Ethernet RM Unit	Class 1
16	EMAC_TM	Ethernet TM Unit	Class 1
17:31		Reserved	

See "Condition Register (CR)" on page 3-12.

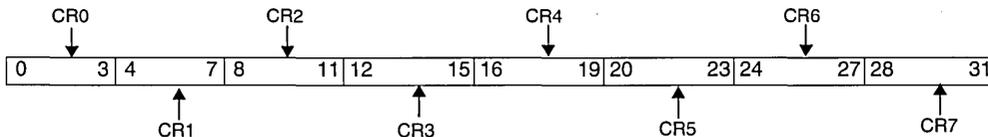


Figure 25-10. Condition Register (CR)

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

CTR

Count Register

SPR 0x009

See "Count Register (CTR)" on page 3-7.



Figure 25-11. Count Register (CTR)

0:31	Count	Used as count for branch conditional with decrement instructions, or as address for branch-to-counter instructions.
------	-------	---

DAC1–DAC2

Data Address Compare Registers

SPR 0x3F6–0x3F7

See “Data Address Compare Registers (DAC1–DAC2)” on page 12-14.



Figure 25-12. Data Address Compare Registers (DAC1–DAC2)

0:31	Data Address Compare (DAC) byte address	DBCR0[D1S] determines which address bits are examined.
------	---	--

DBCRO

Debug Control Register 0

SPR 0x3F2

See “Debug Control Registers” on page 12-9.

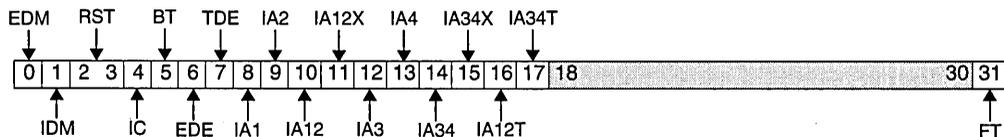


Figure 25-13. Debug Control Register 0 (DBCRO)

0	EDM	External Debug Mode 0 Disabled 1 Enabled	
1	IDM	Internal Debug Mode 0 Disabled 1 Enabled	
2:3	RST	Reset 00 No action 01 Core reset 10 Chip reset 11 System reset	Causes a processor reset request when set by software.
Attention: Writing 01, 10, or 11 to this field causes a processor reset request.			
4	IC	Instruction Completion Debug Event 0 Disabled 1 Enabled	
5	BT	Branch Taken Debug Event 0 Disabled 1 Enabled	
6	EDE	Exception Debug Event 0 Disabled 1 Enabled	
7	TDE	Trap Debug Event 0 Disabled 1 Enabled	
8	IA1	IAC 1 Debug Event 0 Disabled 1 Enabled	
9	IA2	IAC 2 Debug Event 0 Disabled 1 Enabled	
10	IA12	Instruction Address Range Compare 1–2 0 Disabled 1 Enabled	Registers IAC1 and IAC2 define an address range used for IAC address comparisons.
11	IA12X	Enable Instruction Address Exclusive Range Compare 1–2 0 Inclusive 1 Exclusive	Selects the range defined by IAC1 and IAC2 to be inclusive or exclusive.

DBCR0 (cont.)

Debug Control Register 0

12	IA3	IAC 3 Debug Event 0 Disabled 1 Enabled	
13	IA4	IAC 4 Debug Event 0 Disabled 1 Enabled	
14	IA34	Instruction Address Range Compare 3–4 0 Disabled 1 Enabled	Registers IAC3 and IAC4 define an address range used for IAC address comparisons.
15	IA34X	Instruction Address Exclusive Range Compare 3–4 0 Inclusive 1 Exclusive	Selects range defined by IAC3 and IAC4 to be inclusive or exclusive.
16	IA12T	Instruction Address Range Compare 1-2 Toggle 0 Disabled 1 Enable	Toggles range 12 inclusive, exclusive DBCR[IA12X] on debug event.
17	IA34T	Instruction Address Range Compare 3–4 Toggle 0 Disabled 1 Enable	Toggles range 34 inclusive, exclusive DBCR[IA34X] on debug event.
18:30		Reserved	
31	FT	Freeze timers on debug event 0 Timers not frozen 1 Timers frozen	

DBCR1

Debug Control Register 1

SPR 0x3BD

See “Debug Control Registers” on page 12-9.

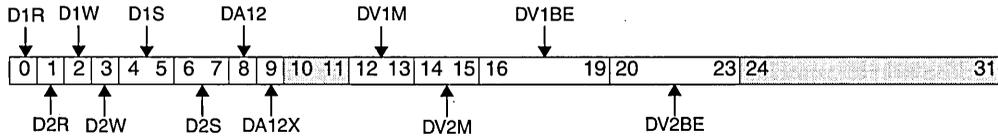


Figure 25-14. Debug Control Register 1 (DBCR1)

0	D1R	DAC1 Read Debug Event 0 Disabled 1 Enabled	
1	D2R	DAC 2 Read Debug Event 0 Disabled 1 Enabled	
2	D1W	DAC 1 Write Debug Event 0 Disabled 1 Enabled	
3	D2W	DAC 2 Write Debug Event 0 Disabled 1 Enabled	
4:5	D1S	DAC 1 Size 00 Compare all bits 01 Ignore lsb (least significant bit) 10 Ignore two lsbs 11 Ignore five lsbs	Address bits used in the compare: Byte address Halfword address Word address Cache line (8-word) address
6:7	D2S	DAC 2 Size 00 Compare all bits 01 Ignore lsb (least significant bit) 10 Ignore two lsbs 11 Ignore five lsbs	Address bits used in the compare: Byte address Halfword address Word address Cache line (8-word) address
8	DA12	Enable Data Address Range Compare 1:2 0 Disabled 1 Enabled	Registers DAC1 and DAC2 define an address range used for DAC address comparisons
9	DA12X	Data Address Exclusive Range Compare 1:2 0 Inclusive 1 Exclusive	Selects range defined by DAC1 and DAC2 to be inclusive or exclusive
10:11		Reserved	

DBCR1 (cont.)

Debug Control Register 1

12:13	DV1M	Data Value Compare 1 Mode 00 Undefined 01 AND 10 OR 11 AND-OR	Type of data comparison used: All bytes selected by DBCR1[DV1BE] must compare to the appropriate bytes of DVC1. One of the bytes selected by DBCR1[DV1BE] must compare to the appropriate bytes of DVC1. The upper halfword or lower halfword must compare to the appropriate halfword in DVC1. When performing halfword compares set DBCR1[DV1BE] = 0011, 1100, or 1111.
14:15	DV2M	Data Value Compare 2 Mode 00 Undefined 01 AND 10 OR 11 AND-OR	Type of data comparison used All bytes selected by DBCR1[DV2BE] must compare to the appropriate bytes of DVC2. One of the bytes selected by DBCR1[DV2BE] must compare to the appropriate bytes of DVC2. The upper halfword or lower halfword must compare to the appropriate halfword in DVC2. When performing halfword compares set DBCR1[DV2BE] = 0011, 1100, or 1111.
16:19	DV1BE	Data Value Compare 1 Byte 0 Disabled 1 Enabled	Selects which data bytes to use in data value comparison
20:23	DV2BE	Data Value Compare 2 Byte 0 Disabled 1 Enabled	Selects which data bytes to use in data value comparison
24:31		Reserved	

DBSR

Debug Status Register

SPR 0x3F0 Read/Clear

See "Debug Status Register (DBSR)" on page 12-12.



Figure 25-15. Debug Status Register (DBSR)

0	IC	Instruction Completion Debug Event 0 Event did not occur 1 Event occurred
1	BT	Branch Taken Debug Event 0 Event did not occur 1 Event occurred
2	EDE	Exception Debug Event 0 Event did not occur 1 Event occurred
3	TIE	Trap Instruction Debug Event 0 Event did not occur 1 Event occurred
4	UDE	Unconditional Debug Event 0 Event did not occur 1 Event occurred
5	IA1	IAC1 Debug Event 0 Event did not occur 1 Event occurred
6	IA2	IAC2 Debug Event 0 Event did not occur 1 Event occurred
7	DR1	DAC1 Read Debug Event 0 Event did not occur 1 Event occurred
8	DW1	DAC1 Write Debug Event 0 Event did not occur 1 Event occurred
9	DR2	DAC2 Read Debug Event 0 Event did not occur 1 Event occurred
10	DW2	DAC2 Write Debug Event 0 Event did not occur 1 Event occurred

DBSR (cont.)

Debug Status Register

11	IDE	Imprecise Debug Event 0 No circumstance that would cause a debug event (if MSR[DE] = 1) occurred 1 A debug event would have occurred, but debug exceptions were disabled (MSR[DE] = 1)	
12	IA3	IAC3 Debug Event 0 Event did not occur 1 Event occurred	
13	IA4	IAC4 Debug Event 0 Event did not occur 1 Event occurred	
14:21		Reserved	
22:23	MRR	Most Recent Reset 00 No reset has occurred since last cleared by software. 01 Core reset 10 Chip reset 11 System reset	This field is set to a value, indicating the type of reset, when a reset occurs.
24:31		Reserved	

DCCR

Data Cache Cacheability Register

SPR 0x3FA

See "Real-mode Storage Attribute Control" on page 6-17.

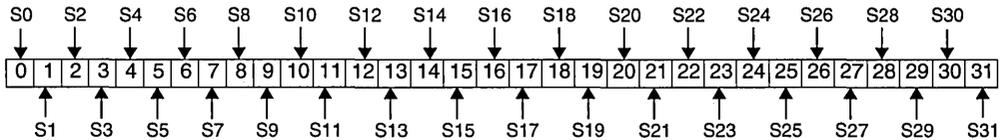


Figure 25-16. Data Cache Cachability Register (DCCR)

0	S0	0 Noncachable 1 Cachable	0x0000 0000–0x07FF FFFF
1	S1	0 Noncachable 1 Cachable	0x0800 0000–0x0FFF FFFF
2	S2	0 Noncachable 1 Cachable	0x1000 0000–0x17FF FFFF
3	S3	0 Noncachable 1 Cachable	0x1800 0000–0x1FFF FFFF
4	S4	0 Noncachable 1 Cachable	0x2000 0000–0x27FF FFFF
5	S5	0 Noncachable 1 Cachable	0x2800 0000–0x2FFF FFFF
6	S6	0 Noncachable 1 Cachable	0x3000 0000–0x37FF FFFF
7	S7	0 Noncachable 1 Cachable	0x3800 0000–0x3FFF FFFF
8	S8	0 Noncachable 1 Cachable	0x4000 0000–0x47FF FFFF
9	S9	0 Noncachable 1 Cachable	0x4800 0000–0x4FFF FFFF
10	S10	0 Noncachable 1 Cachable	0x5000 0000–0x57FF FFFF
11	S11	0 Noncachable 1 Cachable	0x5800 0000–0x5FFF FFFF
12	S12	0 Noncachable 1 Cachable	0x6000 0000–0x67FF FFFF
13	S13	0 Noncachable 1 Cachable	0x6800 0000–0x6FFF FFFF
14	S14	0 Noncachable 1 Cachable	0x7000 0000–0x77FF FFFF
15	S15	0 Noncachable 1 Cachable	0x7800 0000–0x7FFF FFFF

DCCR (cont.)

Data Cache Cacheability Register

16	S16	0 Noncachable 1 Cachable	0x8000 0000–0x87FF FFFF
17	S17	0 Noncachable 1 Cachable	0x8800 0000–0x8FFF FFFF
18	S18	0 Noncachable 1 Cachable	0x9000 0000–0x97FF FFFF
19	S19	0 Noncachable 1 Cachable	0x9800 0000–0x9FFF FFFF
20	S20	0 Noncachable 1 Cachable	0xA000 0000–0xA7FF FFFF
21	S21	0 Noncachable 1 Cachable	0xA800 0000–0xAFFF FFFF
22	S22	0 Noncachable 1 Cachable	0xB000 0000–0xB7FF FFFF
23	S23	0 Noncachable 1 Cachable	0xB800 0000–0xBFFF FFFF
24	S24	0 Noncachable 1 Cachable	0xC000 0000–0xC7FF FFFF
25	S25	0 Noncachable 1 Cachable	0xC800 0000–0xCFFF FFFF
26	S26	0 Noncachable 1 Cachable	0xD000 0000–0xD7FF FFFF
27	S27	0 Noncachable 1 Cachable	0xD800 0000–0xDFFF FFFF
28	S28	0 Noncachable 1 Cachable	0xE000 0000–0xE7FF FFFF
29	S29	0 Noncachable 1 Cachable	0xE800 0000–0xEFFF FFFF
30	S30	0 Noncachable 1 Cachable	0xF000 0000–0xF7FF FFFF
31	S31	0 Noncachable 1 Cachable	0xF800 0000–0xFFFF FFFF

DCP0_ADDR0–DCP0_ADDR1

Address Decode Definition Register 0–1

DCR Accessed using DCP0_CFGADDR, DCP0_CFGDATA; Offset 0x04–0x05

See “Decompression Address Decode Definition Registers (DCP0_ADDR0–DCP0_ADDR1)” on page 14-5.

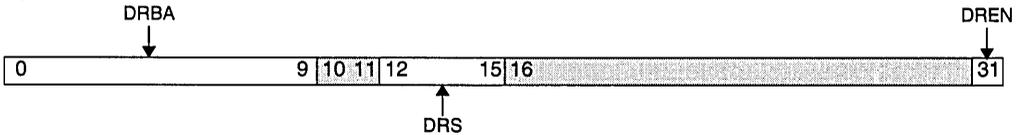


Figure 25-17. Decompression Address Decode Definition Registers (DCP0_ADDR0–DCP0_ADDR1)

0:9	DRBA	Decode Region Base Address	High-order decode region address bits
10:11		Reserved	
12:15	DRS	Decode Region Size 0000–0100 Reserved 0101 4MB 0110 8MB 0111 16MB 1000 32MB 1001 64MB 1010 128MB 1011 256MB 1100 512MB 1101 1GB 1110 2GB 1111 4GB	
16:30		Reserved	
31	DREN	Enable Decode Region 0 Decoding is disabled for region 1 Decoding is enabled for region.	

DCP0_CFG

Decompression Core Configuration Register

DCR Accessed using DCP0_CFGADDR, DCP0_CFGDATA; Offset 0x40

See "Decompression Configuration Register (DCP0_CFG)" on page 14-5.

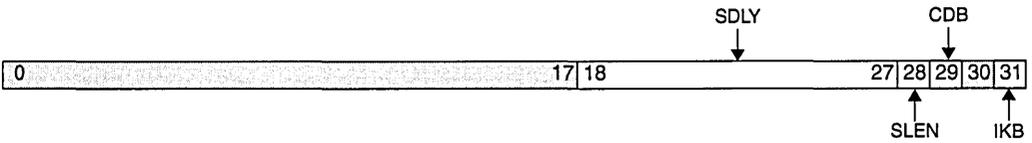


Figure 25-18. Decompression Controller Configuration Register (DCP0_CFG)

0:17		Reserved
18:27	SLDY	Sleep Delay Sleep delay
28	SLEN	Sleep Enable 0 Disable sleep 1 Enable sleep
29	CDB	Clear Decompression Buffer Self-clearing; always reads 0 0 Normal operation 1 Clear decompression buffer
30		Reserved
31	IKB	Enable Decompression 0 Decompression is enabled; U0 storage attribute is recognized 1 Decompression is disabled; U0 storage attribute is ignored

DCP0_CFGADDR

Decompression Controller Address Register

DCR 0x014

See “Decompression Controller Registers” on page 14-3.

This register is used to determine offsets for decompression controller DCRs.

DCP0_CFGDATA

Decompression Controller Data Register

DCR 0x015

See “Decompression Controller Registers” on page 14-3.

This register is used to indirectly access decompression controller DCRs.

DCP0_ESR

Decompression Controller Bus Error Status Register 0

DCR Accessed using DCP0_CFGADDR, DCP0_CFGDATA; Offset 0x52 Read/Clear

See “Decompression Controller Error Status Register 0 (DCP0_ESR)” on page 14-7.

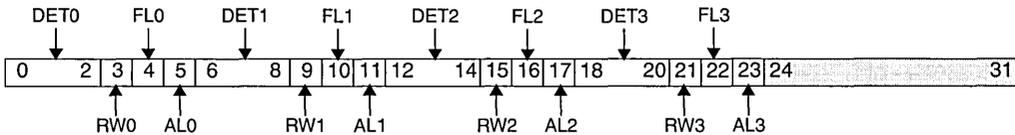


Figure 25-19. Decompression Controller Error Status Register 0 (DCP0_ESR)

0:2	DET0	Decompression Error Type for Master 0 000 Time-out during ITE fetch 001 Time-out during block fetch 010 Memory controller error during ITE fetch 011 Memory controller error during block fetch 100–111 Reserved	Master 0 is the processor core instruction cache unit (ICU).
3	RW0	Read/Write Status for Master 0 0 Error operation was a write 1 Error operation was a read	This implementation only reports errors for compressed reads.
4	FLO	DCP0_ESR Field Lock for Master 0 0 DCP0_ESR fields are unlocked 1 DCP0_ESR fields are locked	DCP0_ESR register fields are locked when the PLB_lockErr signal was active in the cycle in which the error occurred.
5	AL0	DCP0_MEMBEAR/DCP0_PLBBEAR Address Lock for Master 0 0 Master has not locked DCP0_MEMBEAR and DCP0_PLBBEAR 1 Master has locked DCP0_MEMBEAR and DCP0_PLBBEAR	DCP0_MEMBEAR and DCP0_PLBBEAR are locked to this master address when the PLB_lockErr signal was active in the cycle in which the error occurred.
6:8	DET1	Decompression Error Type for Master 1	See description for DCP0_ESR[DET0]. Master1 is the processor core data cache unit (DCU).
9	RW1	Read/Write Status for Master 1 0 Error operation was a write 1 Error operation was a read	This implementation only reports errors for compressed reads.
10	FL1	DCP0_ESR Field Lock for Master 1 0 DCP0_ESR fields are unlocked 1 DCP0_ESR fields are locked	DCP0_ESR register fields are locked when the PLB_lockErr signal was active in the cycle in which the error occurred.
11	AL1	DCP0_MEMBEAR/DCP0_PLBBEAR Address Lock for Master 1 0 Master has not locked DCP0_MEMBEAR and DCP0_PLBBEAR 1 Master has locked DCP0_MEMBEAR and DCP0_PLBBEAR	DCP0_MEMBEAR and DCP0_PLBBEAR are locked to this master address when the PLB_lockErr signal was active in the cycle in which the error occurred.

DCP0_ESR (cont.)

Decompression Controller Bus Error Status Register 0

12:14	DET2	Decompression Error Type for Master 2	See description for DCP0_ESR[DET0]. Master 2 is the external master.
15	RW2	Read/Write Status for Master 2 0 Error operation was a write 1 Error operation was a read	This implementation only reports errors for compressed reads.
16	FL2	DCP0_ESR Field Lock for Master 2 0 DCP0_ESR fields are unlocked 1 DCP0_ESR fields are locked	DCP0_ESR register fields are locked when the PLB_lockErr signal was active in the cycle in which the error occurred.
17	AL2	DCP0_MEMBEAR/DCP0_PLBBEAR Address Lock for Master 2 0 Master has not locked DCP0_MEMBEAR and DCP0_PLBBEAR 1 Master has locked DCP0_MEMBEAR and DCP0_PLBBEAR	DCP0_MEMBEAR and DCP0_PLBBEAR are locked to this master address when the PLB_lockErr signal was active in the cycle in which the error occurred.
18:20	DET3	Decompression Error Type for Master 3	See description for DCP0_ESR[DET0]. Master 3 is PCI.
21	RW3	Read/Write Status for Master 3 0 Error operation was a write 1 Error operation was a read	This implementation only report errors for compressed reads.
22	FL3	DCP0_ESR Field Lock for Master 3 0 DCP0_ESR fields are unlocked 1 DCP0_ESR fields are locked	DCP0_ESR register fields are locked when the PLB_lockErr signal was active in the cycle in which the error occurred.
23	AL3	DCP0_MEMBEAR/DCP0_PLBBEAR Address Lock for Master 3 0 Master has not locked DCP0_MEMBEAR and DCP0_PLBBEAR 1 Master has locked DCP0_MEMBEAR and DCP0_PLBBEAR	DCP0_MEMBEAR and DCP0_PLBBEAR are locked to this master address when the PLB_lockErr signal was active in the cycle in which the error occurred.
24:31		Reserved	

DCP0_ID

Decompression Controller ID Register

DCR Accessed using DCP0_CFGADDR, DCP0_CFGDATA; Offset 0x41 Read-Only

See "Decompression Controller ID Register (DCP0_ID)" on page 14-6.

0	31
---	----

Figure 25-20. Decompression Controller ID Register (DCP0_ID)

0:31	Decompression Controller ID	Read-only, value is 0000504D
------	-----------------------------	------------------------------

DCP0_ITOR0–DCP0_ITOR3

Index Table Origin Register 0–3

DCR Accessed using DCP0_CFGADDR, DCP0_CFGDATA; Offset 0x00–0x03

See “Index Table Origin Registers (DCP0_ITOR0–DCP0_ITOR3)” on page 14-4.



Figure 25-21. Decompression Index Table Origin Registers (DCP0_ITOR0–DCP0_ITOR3)

0:20		Reserved
21:31	ITO	Index Table Origin High-order index table address bits

DCP0_MEMBEAR

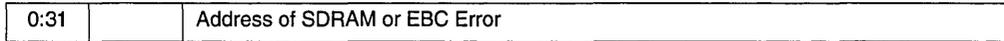
Decompression Controller Bus Error Address Register

DCR Accessed using DCP0_CFGADDR, DCP0_CFGDATA; Offset 0x51 Read-Only

See “Decompression Controller Bus Error Address Register (DCP0_MEMBEAR)” on page 14-7.



Figure 25-22. Decompression Controller Bus Error Address Register (DCP0_MEMBEAR)



DCP0_PLBBEAR

Decompression Controller PLB Error Address Register

DCR Accessed using DCP0_CFGADDR, DCP0_CFGDATA; Offset 0x50 Read-Only

See "Decompression Controller PLB Error Address Register (DCP0_PLBBEAR)" on page 14-7.

0	31
---	----

Figure 25-23. Decompression Controller PLB Error Address Register (DCP0_PLBBEAR)

0:31	Address of PLB Error
------	----------------------

DCP0_RAM0–DCP0_RAM3FF

Decompression Controller SRAM/ROM

DCR 0x400–0x7FF

See “Decompression Controller Registers” on page 14-3.

These registers store decode tables for the decompression controller.

DCP0_VER

Decompression Controller Version Register

DCR Accessed using DCP0_CFGADDR, DCP0_CFGDATA; Offset 0x42 Read-Only

See "Decompression Controller Version Register (DCP0_VER)" on page 14-6.



Figure 25-24. Decompression Controller Version Register (DCP0_VER)

0:31	Decompression Controller Version	Read-only, value is 00000200
------	----------------------------------	------------------------------

DCWR

Data Cache Write-through Register

SPR 0x3BA

See “Real-mode Storage Attribute Control” on page 6-17.

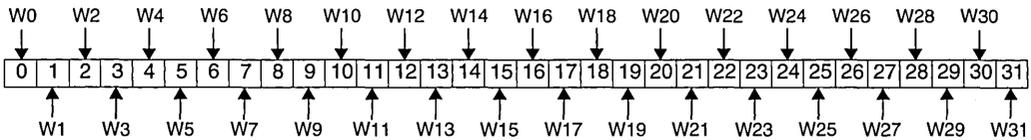


Figure 25-25. Data Cache Write-through Register (DCWR)

0	W0	0 Write-back 1 Write-through	0x0000 0000–0x07FF FFFF
1	W1	0 Write-back 1 Write-through	0x0800 0000–0x0FFF FFFF
2	W2	0 Write-back 1 Write-through	0x1000 0000–0x17FF FFFF
3	W3	0 Write-back 1 Write-through	0x1800 0000–0x1FFF FFFF
4	W4	0 Write-back 1 Write-through	0x2000 0000–0x27FF FFFF
5	W5	0 Write-back 1 Write-through	0x2800 0000–0x2FFF FFFF
6	W6	0 Write-back 1 Write-through	0x3000 0000–0x37FF FFFF
7	W7	0 Write-back 1 Write-through	0x3800 0000–0x3FFF FFFF
8	W8	0 Write-back 1 Write-through	0x4000 0000–0x47FF FFFF
9	W9	0 Write-back 1 Write-through	0x4800 0000–0x4FFF FFFF
10	W10	0 Write-back 1 Write-through	0x5000 0000–0x57FF FFFF
11	W11	0 Write-back 1 Write-through	0x5800 0000–0x5FFF FFFF
12	W12	0 Write-back 1 Write-through	0x6000 0000–0x67FF FFFF
13	W13	0 Write-back 1 Write-through	0x6800 0000–0x6FFF FFFF
14	W14	0 Write-back 1 Write-through	0x7000 0000–0x77FF FFFF
15	W15	0 Write-back 1 Write-through	0x7800 0000–0x7FFF FFFF

DCWR (cont.)

Data Cache Write-through Register

16	W16	0 Write-back 1 Write-through	0x8000 0000–0x87FF FFFF
17	W17	0 Write-back 1 Write-through	0x8800 0000–0x8FFF FFFF
18	W18	0 Write-back 1 Write-through	0x9000 0000–0x97FF FFFF
19	W19	0 Write-back 1 Write-through	0x9800 0000–0x9FFF FFFF
20	W20	0 Write-back 1 Write-through	0xA000 0000–0xA7FF FFFF
21	W21	0 Write-back 1 Write-through	0xA800 0000–0xAFFF FFFF
22	W22	0 Write-back 1 Write-through	0xB000 0000–0xB7FF FFFF
23	W23	0 Write-back 1 Write-through	0xB800 0000–0xBFFF FFFF
24	W24	0 Write-back 1 Write-through	0xC000 0000–0xC7FF FFFF
25	W25	0 Write-back 1 Write-through	0xC800 0000–0xCFFF FFFF
26	W26	0 Write-back 1 Write-through	0xD000 0000–0xD7FF FFFF
27	W27	0 Write-back 1 Write-through	0xD800 0000–0xDFFF FFFF
28	W28	0 Write-back 1 Write-through	0xE000 0000–0xE7FF FFFF
29	W29	0 Write-back 1 Write-through	0xE800 0000–0xEFFF FFFF
30	W30	0 Write-back 1 Write-through	0xF000 0000–0xF7FF FFFF
31	W31	0 Write-back 1 Write-through	0xF800 0000–0xFFFF FFFF

DEAR

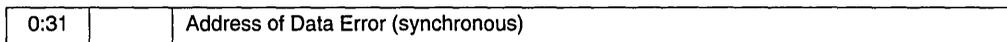
Data Exception Address Register

SPR 0x3D5

See "Data Exception Address Register (DEAR)" on page 10-34.



Figure 25-26. Data Exception Address Register (DEAR)



DMA0_CR0–DMA0_CR3

DMA Channel Control Registers 0–3

DCR 0x100, 0x108, 0x110, 0x118

See “DMA Channel Control Registers (DMA0_CR0–DMA0_CR3)” on page 18-8.

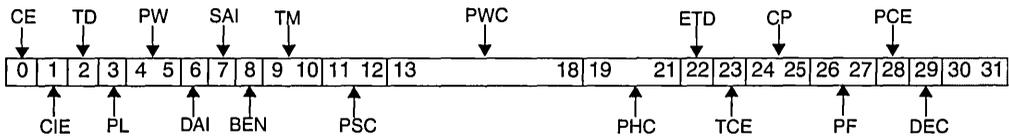


Figure 25-27. DMA Channel Control Registers (DMA0_CR0–DMA0_CR3)

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 are generated for terminal count, end of transfer, and errors conditions. See “DMA Interrupts” on page 18-15..
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 0 External peripheral (EBC) bus 1 OPB (UART0)	
4:5	PW	Peripheral Width/Memory alignment 00 Byte (8 bits) 01 Halfword (16 bits) 10 Word (32 bits) 11 Doubleword (64 bits) memory-to-memory transfers only	Transfer width equals peripheral width for peripherals.
6	DAI	Destination Address Increment 0 Do not increment destination address 1 After each data transfer increment the destination address by: 1, if the transfer width is a byte (8 bits) 2, if the transfer width is a halfword (16 bits) 4, if the transfer width is a word (32 bits) 8, if the transfer width is a doubleword (64 bit)	
7	SAI	Source Address Increment 0 Do not increment source address 1 After each data transfer increment the source address by: 1, if the transfer width is a byte (8 bits) 2, if the transfer width is a halfword (16 bits) 4, if the transfer width is a word (32 bits) 8, if the transfer width is a doubleword (64 bit)	

DMA0_CR0–DMA0_CR3 (cont.)

DMA Channel Control Registers 0–3

8	BEN	Buffer Enable 0 Disable DMA 32-byte buffer 1 Enable DMA 32-byte buffer	If BEN=0 discrete read and write operations occur for each data transfer.
9:10	TM	Transfer mode 00 Peripheral 01 Reserved 10 Software-initiated memory-to-memory 11 Device-paced memory-to-memory	
11:12	PSC	Peripheral Setup Cycles 0-3	Number of PerClk cycles that the EBC peripheral bus is idle from the last peripheral bus transaction to DMAAckn becoming active. Used only for the peripheral side of peripheral mode transfers.
13:18	PWC	Peripheral Wait Cycles 0-63	DMAAckn remains active for PWC+1 PerClk cycles. Used only for the peripheral side of peripheral mode transfers.
19:21	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.
22	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.
23	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.
24:25	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. See "Channel Priorities" on page 18-13 for more information.
26:27	PF	Memory Read Prefetch Transfer 00 Prefetch 1 doubleword 01 Prefetch 2 doublewords 10 Prefetch 4 doublewords 11 Reserved	Used only during memory-to-peripheral and device-paced memory-to-memory transfers. To enable prefetching it is required that BEN=1.
28	PCE	Parity Check Enable 0 Disable parity checking 1 Enable parity checking	Enables parity checking for peripheral mode transfers. See "Data Parity During DMA Peripheral Transfers" on page 18-14.
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).

DMA0_CR0–DMA0_CR3 (cont.)

DMA Channel Control Registers 0–3

30:31		Reserved
-------	--	----------

DMA0_CT0–DMA0_CT3

DMA Count Register 0–3

DCR 0x101, 0x109, 0x111, 0x119

See “DMA Count Registers (DMA0_CT0–DMA0_CT3)” on page 18-11.

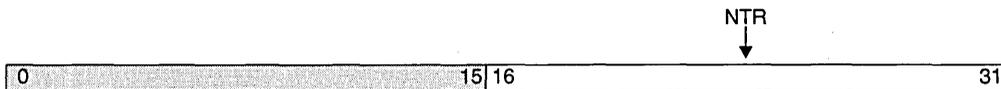


Figure 25-28. DMA Count Registers (DMA0_CT0–DMA0_CT3)

0:15		Reserved
16:31	NTR	Number of transfers remaining

DMA0_DA0–DMA0_DA3

DMA Destination Address Register 0–3

DCR 0x102, 0x10A, 0x112, 0x11A

See “DMA Destination Address Registers (DMA0_DA0–DMA0_DA3)” on page 18-11.

0	31
---	----

Figure 25-29. DMA Destination Address Registers (DMA0_DA0–DMA0_DA3)

0:31	Destination address for memory-to-memory and peripheral-to-memory transfers.
------	--

DMA0_POL

DMA Polarity Configuration Register

DCR 0x126

See “DMA Polarity Configuration Register (DMA0_POL)” on page 18-5.

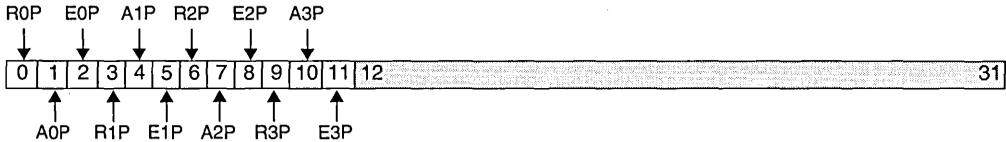


Figure 25-30. DMA Polarity Configuration Register (DMA0_POL)

0	R0P	DMAReq0 Polarity 0 DMAReq0 is active high 1 DMAReq0 is active low
1	A0P	DMAAck0 Polarity 0 DMAAck0 is active high 1 DMAAck0 is active low
2	E0P	EOT0[TC0] Polarity 0 EOT0[TC0] is active high 1 EOT0[TC0] is active low
3	R1P	DMAReq1 Polarity 0 DMAReq1 is active high 1 DMAReq1 is active low
4	A1P	DMAAck1 Polarity 0 DMAAck1 is active high 1 DMAAck1 is active low
5	E1P	EOT1[TC1] Polarity 0 EOT1[TC1] is active high 1 EOT1[TC1] is active low
6	R2P	DMAReq2 Polarity 0 DMAReq2 is active high 1 DMAReq2 is active low
7	A2P	DMAAck2 Polarity 0 DMAAck2 is active high 1 DMAAck2 is active low
8	E2P	EOT2[TC2] Polarity 0 EOT2[TC2] is active high 1 EOT2[TC2] is active low
9	R3P	DMAReq3 Polarity 0 DMAReq3 is active high 1 DMAReq3 is active low
10	A3P	DMAAck3 Polarity 0 DMAAck3 is active high 1 DMAAck3 is active low

DMA0_POL

DMA Polarity Configuration Register

11	E3P	EOT3[TC3] Polarity 0 EOT3[TC3] is active high 1 EOT3[TC3] is active low
12:31		Reserved

DMA0_SA0–DMA0_SA3

DMA Source Address Registers 0–3

DCR 0x103, 0x10B, 0x113, 0x11B

See “DMA Source Address Registers (DMA0_SA0–DMA0_SA3)” on page 18-10.

0	31
---	----

Figure 25-31. DMA Source Address Registers (DMA0_SA0–DMA0_SA3)

0:31	Source address for memory-to-memory and memory-to-peripheral transfers.
------	---

DMA0_SG0–DMA0_SG3

DMA Scatter/Gather Descriptor Address Registers 0–3

DCR 0x104, 0x10C, 0x114, 0x11C

See “DMA Scatter/Gather Descriptor Address Registers (DMA0_SG0–DMA0_SG3)” on page 18-12.



Figure 25-32. DMA Scatter/Gather Descriptor Address Registers (DMA0_SG0–DMA0_SG3)

0:31	Address of next scatter/gather descriptor table.
------	--

DMA0_SGC

DMA Scatter/Gather Command Register

DCR 0x123

See "DMA Scatter/Gather Command Register (DMA0_SGC)" on page 18-13.

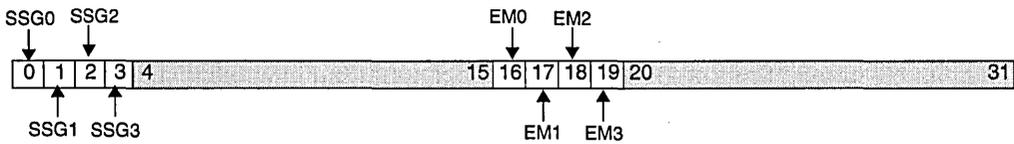


Figure 25-33. DMA Scatter/Gather Command Register (DMA0_SGC)

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.
4:15		Reserved	
16:19	EM[0:3]	Enable Mask for channels 0-3. 0 Writes to SSG[n] are ignored 1 Allow writing to SSG[n]	To write SSG[n], EM[n] must be set. Otherwise, writing SSG[n] has no effect.
20:31		Reserved	

DCR 0x125

See "DMA Sleep Mode Register (DMA0_SLP)" on page 18-6.

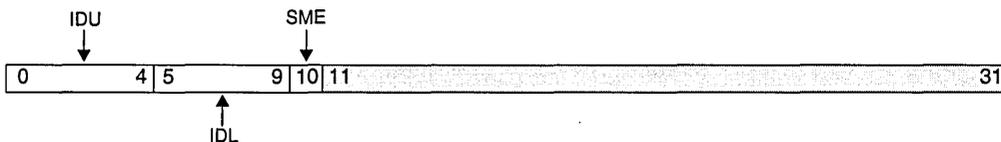


Figure 25-34. DMA Sleep Mode Register (DMA0_SLP)

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

DCR 0x120

See "DMA Scatter/Gather Command Register (DMA0_SGC)" on page 18-13.

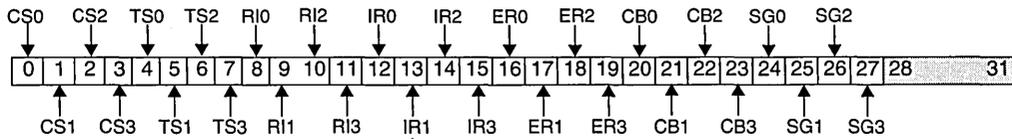


Figure 25-35. DMA Status Register (DMA0_SR)

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.
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.
8:11	RI[0:3]	Channel 0-3 Error Status 0 No error 1 Error occurred	See "Errors" on page 18-14 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	
24:27	SG[0:3]	Scatter/Gather Status 0 No scatter/gather operation in progress 1 Scatter/gather operation in progress	
28:31		Reserved	

DVC1–DVC2

Data Value Compare Registers

SPR 0x3B6–0x3B7

See “Data Value Compare Registers (DVC1–DVC2)” on page 12-15.

0	31
---	----

Figure 25-36. Data Value Compare Registers (DVC1–DVC2)

0:31	Data Value to Compare
------	-----------------------

EBC0_BEAR

Peripheral Bus Error Address Register

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x20

See "Peripheral Bus Error Address Register (EBC0_BEAR)" on page 16-29.



Figure 25-37. Peripheral Bus Error Address Register (EBC0_BEAR)

0:31	Address of Bus Error (asynchronous)
------	-------------------------------------

EBC0_BESR0

Peripheral Bus Error Status Register 0

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x21

See "Peripheral Bus Error Status Register 0 (EBC0_BESR0)" on page 16-30.

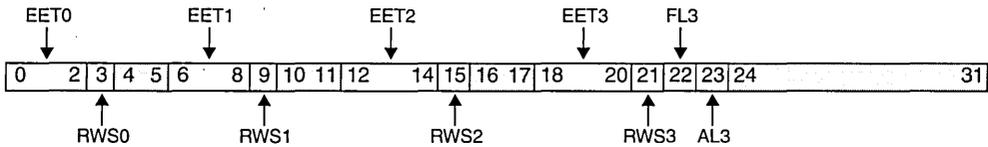


Figure 25-38. Peripheral Bus Error Status Register 0 (EBC0_BESR0)

0:2	EET0	Error type for master 0 Master 0 is the processor instruction fetcher. 000 No error 001 Parity error 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error
3	RWS0	Read/write status for master 0 0 Error operation was a write operation 1 Error operation was a read operation
4:5		Reserved
6:8	EET1	Error type for master 1 Master 1 is the processor data side. 000 No error 001 Parity error 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error
9	RWS1	Read/write status for master 1 0 Error operation was a write operation 1 Error operation was a read operation
10:11		Reserved
12:14	EET2	Error type for master 2 Master 2 is the external bus master. 000 No error 001 Parity error 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error
15	RWS2	Read/write status for master 2 0 Error operation was a write operation 1 Error operation was a read operation

EBC0_BESR0 (cont.)

Peripheral Bus Error Status Register 0

16:17		Reserved
18:20	EET3	Error type for master 3 000 No error 001 Parity error 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error
21	RWS3	Read/write status for master 3 0 Error operation was a write operation 1 Error operation was a read operation
22	FL3	Field lock for master 3 0 EET3 and RWS3 fields are unlocked 1 EET3 and RWS3 fields are locked
23	AL3	EBC0_BEAR address lock for master 3 0 EBC0_BEAR address unlocked 1 EBC0_BEAR address locked
24:31		Reserved

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x22

See "Peripheral Bus Error Status Register 1 (EBC0_BESR1)" on page 16-32.

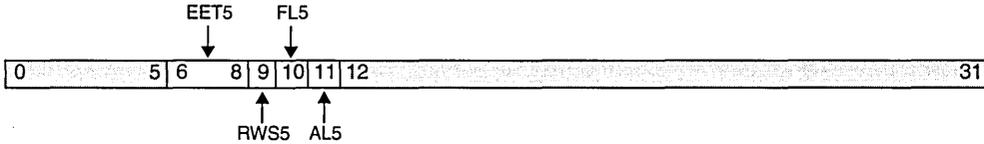


Figure 25-39. Peripheral Bus Error Status Register 1 (EBC0_BESR1)

0:2	EET4	Error type for master 4 000 No error 001 Parity error 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error	Master 4 is the MAL.
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 EET4 and RWS4 fields are unlocked 1 EET4 and RWS4 fields are locked	
5	AL4	EBC0_BEAR address lock for master 4 0 EBC0_BEAR address unlocked 1 EBC0_BEAR address locked	
6:8	EET5	Error type for master 5 000 No error 001 Parity error 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error	Master 5 is the DMA controller.
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 EET5 and RWS5 fields are unlocked 1 EET5 and RWS5 fields are locked	
11	AL5	EBC0_BEAR address lock for master 5 0 EBC0_BEAR address unlocked 1 EBC0_BEAR address locked	

EBC0_BESR1 (cont.)

Peripheral Bus Error Status Register 1

12:31		Reserved
-------	--	----------

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x10–0x17

See “Peripheral Bank Access Parameters (EBC0_BnAP)” on page 16-26.

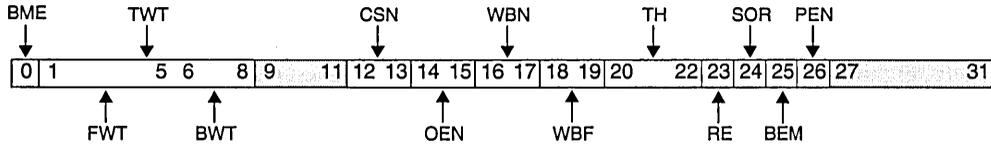


Figure 25-40. Peripheral Bank Access Parameters (EBC0_BnAP)

0	BME	Burst Mode Enable 0 Bursting is disabled 1 Bursting is enabled	
1:8	TWT	Transfer Wait 0-255 PerClk cycles	Wait states on all transfers when BME=0.
1:5	FWT	First Wait 0-31 PerClk cycles	If BME=1, number of wait states on the first transfer of a burst.
6:8	BWT	Burst Wait 0-7 PerClk cycles	If BME=1, number of wait states on non-first transfers of a burst.
9:11		Reserved	
12:13	CSN	Chip Select On Timing 0-3 PerClk cycles	Number of cycles from peripheral address driven to PerCSn low.
14:15	OEN	Output Enable On Timing 0-3 PerClk cycles	Number of cycles from PerCSn low to PerOE low.
16:17	WBN	Write Byte Enable On Timing 0-3 PerClk cycles	If BEM=0, number of cycles from PerCSn low to PerWBEO:3 active.
18:19	WBF	Write Byte Enable Off Timing 0-3 PerClk cycles	If BEM=0 and RE=0, number of cycles PerWBEn becomes inactive prior to PerCSn inactive.
20:22	TH	Transfer Hold 0-7 PerClk cycles	Contains the number of hold cycles inserted at the end of a transfer.
23	RE	Ready Enable 0 PerReady is disabled 1 PerReady is enabled	
24	SOR	Sample on Ready 0 Data transfer occurs one PerClk cycle after PerReady is sampled active 1 Data transfer occurs in the same PerClk cycle that PerReady becomes active	
25	BEM	Byte Enable Mode 0 PerWBEO:3 are only active for write cycles 1 PerWBEO:3 are active for read and write cycles	If BEM=0, PerWBEO:3 timing is controlled by WBN and WBF. If BEM=1, PerWBEO:3 has the same timing as PerAddr0:31.
26	PEN	Parity Enable 0 Disable parity checking 1 Enable parity checking	The EBC implements odd parity.

EBC0_BnAP (cont.)

Peripheral Bank 0-7 Access Parameters

27:31		Reserved
-------	--	----------

EBC0_CFG

External Peripheral Control Register

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x23

See "EBC Configuration Register (EBC0_CFG)" on page 16-23.

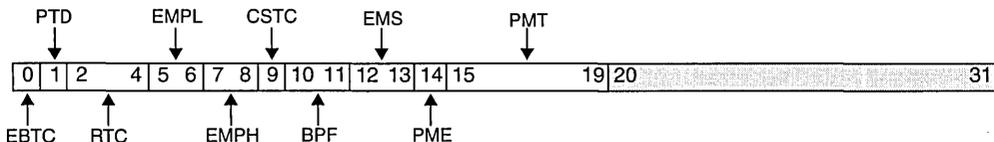


Figure 25-42. EBC Configuration Register (EBC0_CFG)

0	EBTC	External Bus Three-State Control 0 Address, data and control signals are high-Z between EBC transfers. 1 Between EBC transfers the peripheral data bus, address bus and control signals are driven.	Default after reset is EBTC=1. See "Effect of Driver Enable Programming on EBC Signal States" on page 16-5.
1	PTD	Device-Paced Time-out Disable 0 Enabled time-outs 1 Disable time-outs	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 cycles from PerAddr0:31 changing until a time-out error occurs.
5:6	EMPL	External Master Priority Low 00 Low 01 Medium low 10 Medium high 11 High	The PLB priority for external master initiated transfers when the external master hold priority input is low (HoldPri=0).
7:8	EMPH	External Master Priority High 00 Low 01 Medium low 10 Medium high 11 High	The PLB priority for external master initiated transfers when the external master hold priority input is high (HoldPri=1).
9	CSTC	Chip Select Three-state Control 0 PerCS0:7 are high-Z between EBC transfers and when an external master is active (HoldAck=1) 1 PerCS0:7 are always driven.	Default after reset is CSTC=1. See "Effect of Driver Enable Programming on EBC Signal States" on page 16-5.
10:11	BPF	Burst Prefetch 00 Prefetch 1 doubleword 01 Prefetch 2 doublewords 10 Prefetch 4 doublewords 11 Reserved	Controls the amount of data prefetching when the EBC is servicing a PLB burst read. For most applications set this field to 0b00.

EBC0_CFG (cont.)

External Peripheral Control Register

12:13	EMS	External Master Size 00 8-bit 01 16-bit 10 32-bit 11 No external master attached	Width of the attached external master.
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.
20:31		Reserved	

EBC0_CFGADDR

Peripheral Controller Address Register

DCR 0x012

See “EBC Registers” on page 16-23.

This register is used to determine offsets for the indirectly-accessed EBC DCRs.

DCR 0x013

See “EBC Registers” on page 16-23.

This register is used to access the indirectly-accessed EBC DCRs.

EMAC0_GAHT1–EMAC0_GAHT4

Group Address Hash Tables 1–4

MMIO 0xEF600840–0xEF60084C

See “Group Address Hash Tables 1–4 (EMAC0_GAHT1–EMAC0_GAHT4)” on page 19-37.

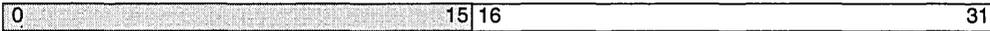


Figure 25-43. Group Address Hash Tables 1–4 (EMAC0_GAHT1–EMAC0_GAHT4)

0:15		Reserved
16:31		Group Address Hash Number

EMAC0_IAHR

Individual Address High Register

MMIO 0xEF60081C

See "Individual Address High (EMAC0_IAHR)" on page 19-35.



Figure 25-44. Individual Address High Register (EMAC0_IAHR)

0:15		Reserved	
16:31		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).

EMAC0_IAHT1-EMAC0_IAHT4

Individual Address Hash Tables 1-4

MMIO 0xEF600830-0xEF60083C

See "Individual Address Hash Tables 1-4 (EMAC0_IAHT1-EMAC0_IAHT4)" on page 19-37.

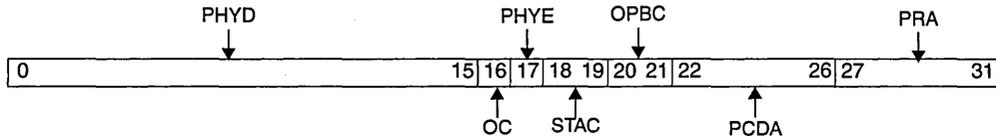


Figure 25-45. STA Control Register (EMAC0_STACR)

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 EMAC0_STACR is addressed 1 PHY data transfer complete	
17	PHYE	PHY Error 0 Successful read transaction 1 Read transaction was not successful	EMAC0_STACR[PHYE] = 0 when a read is successful.
18:19	STAC	STA Command 00 Reserved 01 Read 10 Write 11 Reserved	EMAC sets EMAC0_STACR[STAC] = 0 when the command is completed.
20:21	OPBC	OPB Bus Clock Frequency 00 50 MHz 01 66 MHz 10 83 MHz 11 100 MHz	EMAC0_STACR[OPBC] is used to generate the Management Data Clock (EMCMDClk). When the operational frequency differs from those in the list, then the next greater frequency should be chosen.
22:26	PCDA	PHY Command Destination Address	
27:31	PRA	PHY Register Address	

EMAC0_TMR0

Transmit Mode Register 0

MMIO 0xEF600808

See "Transmit Mode Register 0 (EMAC0_TMR0)" on page 19-27.



Figure 25-46. Transmit Mode Register 0 (EMAC0_TMR0)

0	GNP0	Get New Packet 0 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel 0	EMAC0_TMR0[GNP0] = 0 if EMAC is programmed in dependent mode.
1	GNP1	Get New Packet 1 0 Writing 0 has no effect. 1 Packet ready for transmission on TX Channel 1	EMAC0_TMR0[GNP1] = 0 if EMAC is programmed in dependent mode.
2	GNPD	Get New Packet for Dependent Mode 0 Writing 0 to this bit has no effect 1 Packet ready for transmission in dependent mode	EMAC0_TMR0[GNPD] = 0 if EMAC is not programmed in dependent mode. EMAC0_TMR0[GNPD] = 1 activates the EMAC transmit path in dependent mode.
3	FC	First Channel 0 Activate TX Channel 0 first when GNPD is 1 1 Activate TX Channel 1 first when GNPD is 1	EMAC0_TMR0[FC] is only meaningful in dependent mode, after resetting EMAC0_ISR[DBDM]. EMAC0_TMR0[FC] = 0 if EMAC is not programmed in dependent mode.
4:31		Reserved	

EMAC0_TMR1

Transmit Mode Register 1

MMIO 0xEF60080C

See "Transmit Mode Register 1 (EMAC0_TMR1)" on page 19-28.

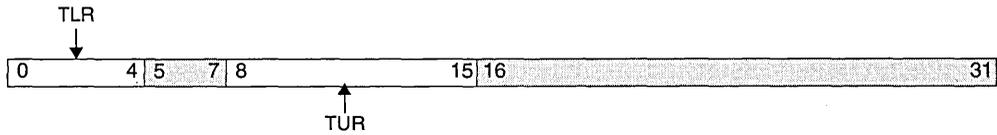


Figure 25-47. Transmit Mode Register 1 (EMAC0_TMR1)

0:4	TLR	Transmit Low Request
5:7		Reserved
8:15	TUR	Transmit Urgent Request
16:31		Reserved

EMAC0_TRTR

Transmit Request Threshold Register

MMIO 0xEF600860

See "Transmit Request Threshold Register (EMAC0_TRTR)" on page 19-40.



Figure 25-48. Transmit Request Threshold Register (EMAC0_TRTR)

0:4	TRT	Transmit Request Threshold The following number of bytes must be placed in the Transmit FIFO before initiating a transmit request. 00000 64 bytes 00001 128 bytes 00010 192 bytes 00011 256 bytes . . . 11111 2048 bytes
5:31		Reserved

EMAC0_VTCI

VLAN TCI Register

MMIO 0xEF600828

See "VLAN TCI Register (EMAC0_VTCI)" on page 19-36.

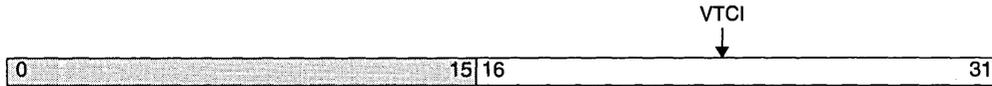


Figure 25-49. VLAN TCI Register (EMAC0_VTCI)

0:15		Reserved
16:31	VTCI	VLAN TCI tag

EMAC0_VTPID

VLAN TPID Register

MMIO 0xEF600824

See "VLAN TPID Register (EMAC0_VTPID)" on page 19-36.

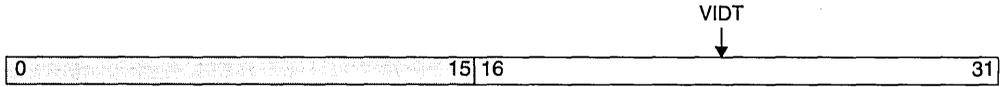


Figure 25-50. VLAN TPID Register (EMAC0_VTPID)

0:15		Reserved
16:31	VIDT	VLAN ID tag

ESR

Exception Syndrome Register

SPR 0x3D4

See "Exception Syndrome Register (ESR)" on page 10-31.

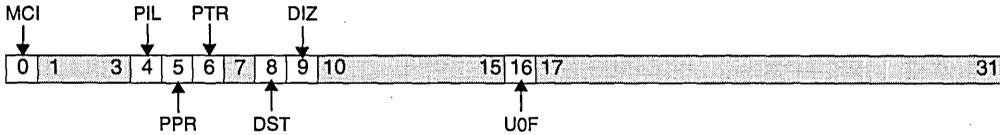


Figure 25-51. Exception Syndrome Register (ESR)

0	MCI	Machine check—instruction 0 Instruction machine check did not occur. 1 Instruction machine check occurred.
1:3		Reserved
4	PIL	Program interrupt—illegal 0 Illegal Instruction error did not occur. 1 Illegal Instruction error occurred.
5	PPR	Program interrupt—privileged 0 Privileged instruction error did not occur. 1 Privileged instruction error occurred.
6	PTR	Program interrupt—trap 0 Trap with successful compare did not occur. 1 Trap with successful compare occurred.
7		Reserved
8	DST	Data storage interrupt—store fault 0 Excepting instruction was not a store. 1 Excepting instruction was a store (includes dcbl , dcbz , and dcccl).
9	DIZ	Data/instruction storage interrupt—zone fault 0 Excepting condition was not a zone fault. 1 Excepting condition was a zone fault.
10:15		Reserved
16	UOF	Data storage interrupt—U0 fault 0 Excepting instruction did not cause a U0 fault. 1 Excepting instruction did cause a U0 fault.
17:31		Reserved

EVPR

Exception Vector Prefix Register

SPR 0x3D6

See "Exception Vector Prefix Register (EVPR)" on page 10-31.

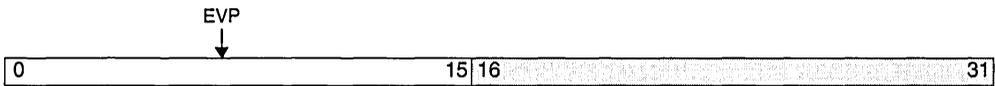


Figure 25-52. Exception Vector Prefix Register (EVPR)

0:15	EVP	Exception Vector Prefix
16:31		Reserved

GPIO0_IR

GPIO Input Register

MMIO 0xEF60071C Read-Only

See "GPIO Input Register (GPIO0_IR)" on page 23-6.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	31
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Figure 25-53. GPIO Input Register (GPIO0_IR)

0		Reserved
1:23		GPIO register bits
24:31		Reserved

GPIO0_ODR

GPIO Open Drain Register

MMIO 0xEF600718

See "GPIO Open Drain Register (GPIO0_ODR)" on page 23-6.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	31
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Figure 25-54. GPIO Open Drain Register (GPIO0_ODR)

0		Reserved
1:23		GPIO register bits
24:31		Reserved

GPIO_OR

GPIO Output Register

MMIO 0xEF600700

See "GPIO Output Register (GPIO_OR)" on page 23-5.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	31
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Figure 25-55. GPIO Output Register (GPIO_OR)

0		Reserved
1:23		GPIO register bits
24:31		Reserved

GPIO0_TCR

GPIO Three-State Control Register

MMIO 0xEF600704

See “GPIO Three-State Control Register (GPIO0_TCR)” on page 23-5.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	31
---	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Figure 25-56. GPIO Three-State Register (GPIO0_TCR)

0	Reserved
1:23	GPIO register bits
24:31	Reserved

GPR0–GPR31

General Purpose Registers

See “General Purpose Registers (R0-R31)” on page 3-6.

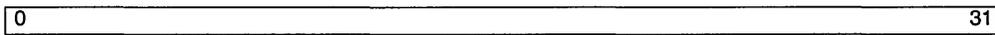
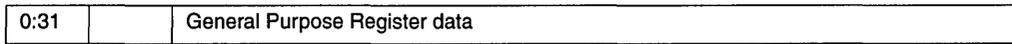


Figure 25-57. General Purpose Registers (R0-R31)



IAC1-IAC4

Instruction Address Compare Registers

SPR 0x3F4-0x3F5, 0x3B4-0x3B5

See "Instruction Address Compare Registers (IAC1-IAC4)" on page 12-14.

0	29	30	31
---	----	----	----

Figure 25-58. Instruction Address Compare Registers (IAC1-IAC4)

0:29	Instruction Address Compare word address	Omit two low-order bits of complete address.
30:31	Reserved	

ICCR

Instruction Cache Cacheability Register

SPR 0x3FB

See "Real-mode Storage Attribute Control" on page 6-17.

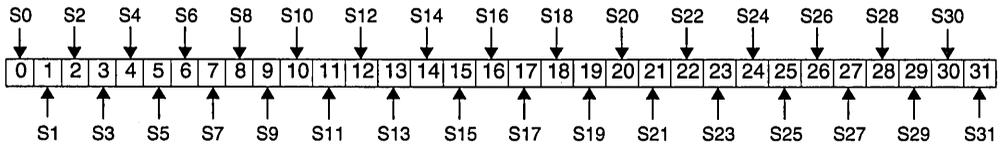


Figure 25-59. Instruction Cache Cachability Register (ICCR)

0	S0	0 Noncachable 1 Cachable	0x0000 0000–0x07FF FFFF
1	S1	0 Noncachable 1 Cachable	0x0800 0000–0x0FFF FFFF
2	S2	0 Noncachable 1 Cachable	0x1000 0000–0x17FF FFFF
3	S3	0 Noncachable 1 Cachable	0x1800 0000–0x1FFF FFFF
4	S4	0 Noncachable 1 Cachable	0x2000 0000–0x27FF FFFF
5	S5	0 Noncachable 1 Cachable	0x2800 0000–0x2FFF FFFF
6	S6	0 Noncachable 1 Cachable	0x3000 0000–0x37FF FFFF
7	S7	0 Noncachable 1 Cachable	0x3800 0000–0x3FFF FFFF
8	S8	0 Noncachable 1 Cachable	0x4000 0000–0x47FF FFFF
9	S9	0 Noncachable 1 Cachable	0x4800 0000–0x4FFF FFFF
10	S10	0 Noncachable 1 Cachable	0x5000 0000–0x57FF FFFF
11	S11	0 Noncachable 1 Cachable	0x5800 0000–0x5FFF FFFF
12	S12	0 Noncachable 1 Cachable	0x6000 0000–0x67FF FFFF
13	S13	0 Noncachable 1 Cachable	0x6800 0000–0x6FFF FFFF
14	S14	0 Noncachable 1 Cachable	0x7000 0000–0x77FF FFFF
15	S15	0 Noncachable 1 Cachable	0x7800 0000–0x7FFF FFFF

ICCR (cont.)

Instruction Cache Cacheability Register

16	S16	0 Noncachable 1 Cachable	0x8000 0000–0x87FF FFFF
17	S17	0 Noncachable 1 Cachable	0x8800 0000–0x8FFF FFFF
18	S18	0 Noncachable 1 Cachable	0x9000 0000–0x97FF FFFF
19	S19	0 Noncachable 1 Cachable	0x9800 0000–0x9FFF FFFF
20	S20	0 Noncachable 1 Cachable	0xA000 0000–0xA7FF FFFF
21	S21	0 Noncachable 1 Cachable	0xA800 0000–0xAFFF FFFF
22	S22	0 Noncachable 1 Cachable	0xB000 0000–0xB7FF FFFF
23	S23	0 Noncachable 1 Cachable	0xB800 0000–0xBFFF FFFF
24	S24	0 Noncachable 1 Cachable	0xC000 0000–0xC7FF FFFF
25	S25	0 Noncachable 1 Cachable	0xC800 0000–0xCFFF FFFF
26	S26	0 Noncachable 1 Cachable	0xD000 0000–0xD7FF FFFF
27	S27	0 Noncachable 1 Cachable	0xD800 0000–0xDFFF FFFF
28	S28	0 Noncachable 1 Cachable	0xE000 0000–0xE7FF FFFF
29	S29	0 Noncachable 1 Cachable	0xE800 0000–0xEFFF FFFF
30	S30	0 Noncachable 1 Cachable	0xF000 0000–0xF7FF FFFF
31	S31	0 Noncachable 1 Cachable	0xF800 0000–0xFFFF FFFF

ICDBDR

Instruction Cache Debug Data Register

SPR 0x3D3 Read-Only

See "ICU Debugging" on page 4-14.

0	31
---	----

Figure 25-60. Instruction Cache Debug Data Register (ICDBDR)

0:31		Instruction cache information	See icread on page 24-68.
------	--	-------------------------------	----------------------------------

ICU tag information is placed into the ICDBDR as shown:

0:21	TAG	Cache Tag
22:26		Reserved
27	V	Cache Line Valid 0 Not valid 1 Valid
28:30		Reserved
31	LRU	Least Recently Used (LRU) 0 A-way LRU 1 B-way LRU

IIC0_CLKDIV

IIC0 Clock Divide

MMIO 0xEF60050C

See "IIC0 Clock Divide Register" on page 22-15.

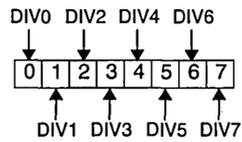


Figure 25-61. IIC0 Clock Divide Register (IIC0_CLKDIV)

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

IIC0_CNTL

IIC0 Control

MMIO 0xEF600506

See "IIC0 Control Register" on page 22-6.

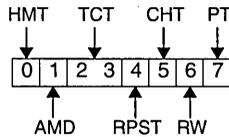


Figure 25-62. IIC0 Control Register (IIC0_CNTL)

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. IIC0_CNTL[PT] needs not be set. If IIC0_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 signal to be issued 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.	

IIC0_DIRECTCNTL

IIC0 Direct Control

MMIO 0xEF600510

See "IIC0 Direct Control Register" on page 22-20.

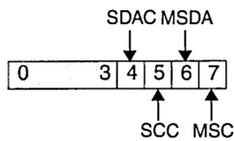


Figure 25-63. IIC0 Direct Control Register (IIC0_DIRECTCNTL)

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	SCC	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	MSC	Monitor IICSCSCL. Used to monitor the IICSCSCL input. 0 IICSCSCL is a logic 0 1 IICSCSCL is a logic 1	Read-only

IIC0_EXTSTS

IIC0 Extended Status

MMIO 0xEF600509

See “IIC0 Extended Control and Slave Status Register” on page 22-18.

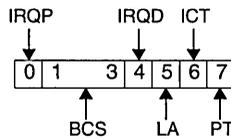


Figure 25-64. IIC0 Extended Status Register (IIC0_EXTSTS)

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"> • IIC0_EXTSTS[IRQP] might be set momentarily while an IRQ moves from the Pending to the On-deck state. • An interrupt remains pending, IIC0_EXTSTS[IRQP]=1, until the current on-deck interrupt becomes active, IIC0_STS[IRQD]=0 and IIC0_STS[IRQA]=1. • Writing 1 to IIC0_EXTSTS[IRQP] clears the field. • When the IIC interrupt is disabled, IIC0_MDCNTL[IRQP] = 0, IIC0_EXTSTS[IRQP] should be ignored.
1:3	BCS	<p>Bus Control State</p> <p>000 Unused; if this value is read, a major IIC hardware problem 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, a major IIC hardware problem occurred.</p>	Read-only.

IIC0_EXTSTS (cont.)

IIC0 Extended Status

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"> • IIC0_EXTSTS[IRQD] might be set momentarily while an IRQ moves from the On-deck to the Active state. • An interrupt remains on-deck, IIC0_EXTSTS[IRQD] = 1, until the current active interrupt is no longer active, IIC0_STS[IRQA] = 0. • If IIC0_EXTSTS[IRQP] = 1, IIC0_EXTSTS[IRQD] is set on the next OPB clock. • Writing 1 to IIC0_EXTSTS[IRQD] clears the field. • When the IIC interrupt is disabled, IIC0_MDCNTL[IRQP]=0, IIC0_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, IIC0_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>

IIC0_HMADR

IIC0 High Master Address

MMIO 0xEF600505

See “IIC0 High Master Address Register” on page 22-6.

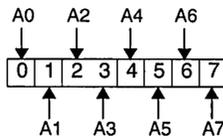


Figure 25-65. IIC0 High Master Address Register (IIC0_HMADR)

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

MMIO 0xEF60050B

See “IIC0 High Slave Address Register” on page 22-14.

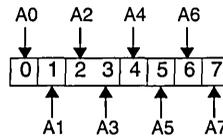


Figure 25-66. IIC0 High Slave Address Register (IIC0_HSADR)

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

IIC0_INTRMSK

IIC0 Interrupt Mask

MMIO 0xEF60050D

See “IIC0 Interrupt Mask Register” on page 22-16.

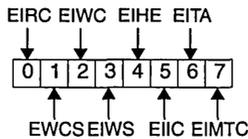


Figure 25-67. IIC0 Interrupt Mask Register (IIC0_INTRMSK)

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. IIC0_XTCNTLSS[SRC] = 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: IIC0_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: IIC0_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: IIC0_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	

MMIO 0xEF600504

See “IIC0 Low Master Address Register” on page 22-5.

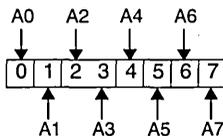


Figure 25-68. IIC0 Low Master Address Register (IIC0_LMADR)

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

IIC0_LSADR

IIC0 Low Slave Address

MMIO 0xEF60050A

See “IIC0 Low Slave Address Register” on page 22-14.

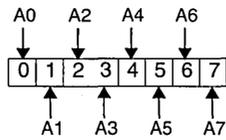


Figure 25-69. IIC0 Low Slave Address Register (IIC0_LSADR)

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

IIC0_MDBUF

IIC0 Master Data Buffer

MMIO 0xEF600500

See "IIC0 Master Data Buffer" on page 22-3.



Figure 25-70. IIC0 Master Data Buffer (IIC0_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

IIC0_MDCNTL

IIC0 Mode Control

MMIO 0xEF600507

See “IIC0 Mode Control Register” on page 22-8.

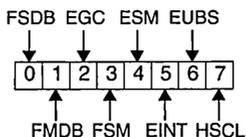


Figure 25-71. IIC0 Mode Control Register (IIC0_MDCNTL)

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.	IIC0_MDCNTL[ESM] overrides this field; if IIC0_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 IIC0_LSADR and IIC0_HSADR before setting this field.
5	EINT	Enable Interrupt 0 Interrupts are disabled. 1 Enables interrupts for interrupts enabled in IIC0_INTRMSK.	
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 IIC0_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.

MMIO 0xEF600502

See "IIC0 Slave Data Buffer" on page 22-4.



Figure 25-72. IIC0 Slave Data Buffer (IIC0_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

IIC0_STS

IIC0 Status

MMIO 0xEF600508

See “IIC0 Status Register” on page 22-10.

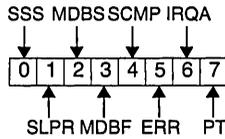


Figure 25-73. IIC0 Status Register (IIC0_STS)

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: IIC0_XTCNTLSS[SRCS, SRRS, SWC, SWRS].
1	SLPR	Sleep Request 0 Normal operation. 1 Sleep mode (CPC0_ER[IIC] = 1).	Read-only. The IIC interface is awakened when a start signal is detected on the IIC bus or when the CPC0_ER[IIC] 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 IIC0_STS[SCMP], set IIC0_STS[SCMP] = 1.
5	ERR	Error 0 No error has occurred. 1 One of the following fields is set: IIC0_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 IIC0_STS[IRQA], set IIC0_STS[IRQA] = 1. If IIC0_MDCNTL[EINT] = 0, then IIC0_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 0xEF60050E

See "IIC0 Transfer Count Register" on page 22-17.

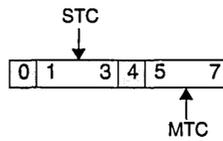


Figure 25-74. IIC0 Transfer Count Register (IIC0_XFRCNT)

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

IIC0_XTCNTLSS

IIC0 Extended Control and Slave Status

MMIO 0xEF60050F

See “IIC0 Extended Control and Slave Status Register” on page 22-18.

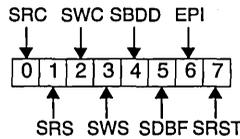


Figure 25-75. IIC0 Extended Control and Slave Status Register (IIC0_XTCNTLSS)

0	SRC	<p>Slave Read Complete</p> <p>0 Normal operation, or IIC0_MDCNTL[HSCL] = 0, IIC0_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>	<p>Check whether the read operation emptied IIC0_SDBUF.</p>
1	SRS	<p>Slave Read Needs Service</p> <p>0 Normal operation or slave read does not need service.</p> <p>1 IIC0_SDBUF is empty, and a read operation was requested on the IIC bus.</p> <p>The set condition may also indicate that IIC0_SDBUF is empty due to a slave read and additional data is requested by the master.</p>	<p>1. If IIC0_MDCNTL[HSCL]=0 and IIC0_SDBUF contains no data, the slave will issue a NACK and IIC0_XTCNTLSS[SRS] is set.</p> <p>2. If IIC0_MDCNTL[HSCL]=0, and IIC0_SDBUF contains data, the slave will send the data. IIC0_XTCNTLSS[SRS] is not set unless the master request additional data.</p> <p>3. If IIC0_MDCNTL[HSCL]=0, and IIC0_SDBUF contains no data, the slave will hold IIC_SCL low to indicate the slave is busy. IIC0_XTCNTLSS[SRS] is set until the IIC0_SDBUF is filled. Once filled, IIC_SCL is released, IIC0_XTCNTLSS[SRS] is cleared, and the slave sends the data.</p> <p>4. If IIC0_MDCNTL[HSCL]=1, and IIC0_SDBUF contains data, the slave will send the data. IIC0_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>	

IIC0_XTCNTLSS (cont.)

IIC0 Extended Control and Slave Status

3	SWS	<p>Slave Write Needs Service</p> <p>0 Normal operation or slave write does not need service.</p> <p>1 IIC0_SDBUF is full during a slave write.</p>	<p>1. If IIC0_MDCNTL[HSCL] = 1 and IIC0_SDBUF is full, the slave will hold IIC0_SCL low to indicate the slave is busy. IIC0_XTCNTLSS[SWS] is set until IIC0_SDBUF is empty. Once empty, IIC0_SCL is released, IIC0_XTCNTLSS[SWS] is cleared, and the slave receives the data.</p> <p>2. If IIC0_MDCNTL[HSCL] = 0 and IIC0_SDBUF is full, the slave will issue a NACK and IIC0_XTCNTLSS[SWS] is set.</p>
4	SBDD	<p>Slave Data Buffer Has Data</p> <p>0 IIC0_SDBUF is empty</p> <p>1 IIC0_SDBUF contains data</p>	Read-only
5	SDBF	<p>Slave Data Buffer Full</p> <p>0 IIC0_SDBUF is not full</p> <p>1 IIC0_SDBUF is full</p>	Read-only
6	EPI	<p>Enable Pulsed IRQ on Transfer Aborted</p> <p>0 The internal IIC interrupt signal to the UIC remains active until the status is cleared, IIC0STS[IRQA] = 0.</p> <p>1 The internal IIC interrupt signal to the PPC405GP UIC is active for one OPB clock cycle.</p>	
7	SRST	<p>Soft Reset</p> <p>0 Normal operation</p> <p>1 Soft reset</p>	

LR

Link Register

SPR 0x008

See "Link Register (LR)" on page 3-8.

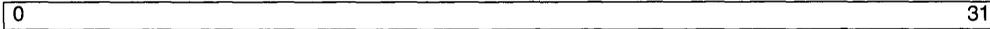


Figure 25-76. Link Register (LR)

0:31	Link Register contents	If (LR) represents an instruction address, LR _{30:31} should be 0.
------	------------------------	---

DCR 0x180

See “MAL Configuration Register (MAL0_CFG)” on page 20-25.

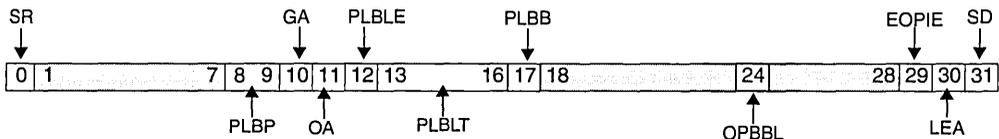


Figure 25-77. MAL Configuration Register (MAL0_CFG)

0	SR	MAL Software Reset 0 MAL reset is complete 1 Reset the MAL	This bit is used to generate 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	PLBP	PLB Priority 00 Lowest 01 10 11 Highest	Determines the priority of MAL requests on the PLB.
10	GA	Guarded Active 0 GUARDED signal not applied to the PLB slave 1 GUARDED signal applied to the PLB slave	When this bit is set, MAL applies the GUARDED signal to the PLB slave when it is the initiator on the bus. When set, the slave can access all the memory in the current page as well as the subsequent page.
11	OA	Ordered Active 0 ORDERED signal not applied to the PLB slave 1 ORDERED signal applied to the PLB slave	When this bit is set, MAL applies the ORDERED signal to the PLB slave when it is initiator on the bus during data write transactions. Note that the ORDERED signal is always driven active during status write transactions.
12	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.
13:16	PLBLT	PLB Latency Timer	Determines the number of cycles allowed for burst transactions on the PLB.
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.

MAL0_CFG (cont.)

MAL Configuration Register

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 COMMACs.
18:28		Reserved	
29	EOPIE	End of Packet Interrupt Enable 0 Generate interrupt on every end-of-packet only if the buffers I bit is set 1 Generate interrupt is on every end-of-packet	When this bit is set, an interrupt is generated on every end of packet (both transmit and receive). When clear, end of packet/buffer interrupt is generated only if the buffers I bit is set (1). Note: An interrupt is generated for every descriptor on which the I bit is set, regardless of the state of the EOPIE bit.
30	LEA	Locked Error Active 0 Handle errors in a non-locked mode 1 Handle errors in locked mode	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.
31	SD	MAL Scroll Descriptor 0 Do not scroll to the first descriptor of the next packet 1 Scroll to the first descriptor of the next packet	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.

DCR 0x181 Read/Clear

See "MAL Error Status Register (MALO_ESR)" on page 20-29.

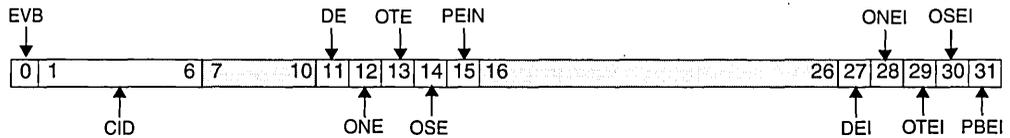


Figure 25-78. MAL Error Status Register (MALO_ESR)

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>
1:6	CID	Channel ID	<p>This field contains the number of the channel which caused the locked error. Bit 1 indicates whether the channel ID represents an RX channel (1) or a TX channel (0).</p> <p>Bits 2:6 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>
7:10		Reserved	
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	ONE	<p>OPB Non-fullword Error</p> <p>0 No error</p> <p>1 Non-fullword asserted</p>	Indicates that the error is a non-fullword acknowledge asserted by an OPB slave.
13	OTE	<p>OPB Timeout Error</p> <p>0 No error</p> <p>1 OPB timeout</p>	Indicates the error is an OPB timeout.

MALO_ESR (cont.)

MAL Error Status Register

14	OSE	OPB Slave Error 0 No error 1 OPB slave error	Indicates the error is an error indication asserted by an OPB slave.
15	PEIN	PLB Bus Error Indication 0 No error 1 PLB bus error	When this bit is set, the detected error is a PLB error. There is no meaning to the Channel ID field in this case.
16:26		Reserved	
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	ONEI	OPB Non-fullword Error Interrupt 0 No error 1 Non-fullword acknowledgment from a slave	This bit is set following a non-fullword acknowledgment coming from a slave. Set condition for this bit generates a maskable interrupt.
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.

MAL0_IER

MAL Interrupt Enable Register

DCR 0x182 Read/Only

See "MAL Interrupt Enable Register (MAL0_IER)" on page 20-31.

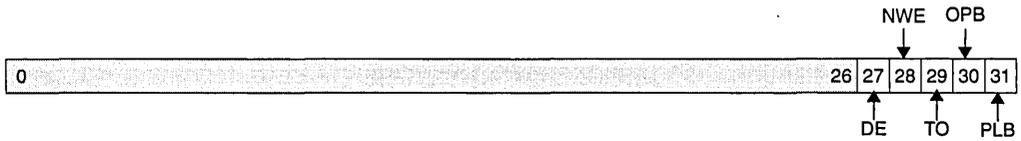


Figure 25-79. MAL Interrupt Enable Register (MAL0_IER)

Bit	Field	Field Name	Description
0:26		Reserved	
27	DE	Descriptor Error	When set, this bit enables the descriptor error (descriptor not valid) interrupt.
28	NWE	Non_W_Err_Int_Enable	When set, this bit enables OPB non-word transfer error interrupt.
29	TO	Time_Out_Int_Enable	When set, this bit enables OPB time-out error interrupt.
30	OPB	OPB_Err_Int_Enable	When set, this bit enables the OPB Slave error interrupt.
31	PLB	PLB_Err_Int_Enable	When set, this bit enables the PLB error interrupt.

MAL0_RCBS0

MAL Receive Channel Buffer Size Register

DCR 0x1E0

See "Buffer Length for Receive" on page 20-13.



Figure 25-80. RX Channel Buffer Size Register 0 (MAL0_RCBS0)

0:23		Reserved
24:31		Receive Channel Buffer Size

MAL0_RXCARR

MAL RX Channel_Active Reset Register

DCR 0x191

See "Channel Active Set and Reset Registers" on page 20-26.

0	1	31
---	---	----

Figure 25-81. RX Channel_Active Reset Register (MAL0_RXCARR)

0		Receive Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, etc.). When 0 is written to the bit, channel operation is disabled. There is only one RX channel in the PPC405GP.
1:31		Reserved	

MAL0_RXCASR

MAL RX Channel Active Set Register

DCR 0x190

See "Channel Active Set and Reset Registers" on page 20-26.



Figure 25-82. RX Channel_Active Set Register (MAL0_RXCASR)

0		Receive Channel Active Set	Each bit represents its related channel (bit 0 for channel 0 etc.). When 1 is written to the bit, channel operation is enabled. There is only one RX channel in the PPC405GP.
1:31		Reserved	

MAL0_RXCTP0R

MAL Receive Channel 0 Pointer Register

DCR 0x1C0

See "Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTP0R)" on page 20-33.

0	31
---	----

Figure 25-83. RX Channel Table Pointer x Register (MAL0_RXCTPxR)

0:31	Channel Table Pointer	Pointer to the base address in memory of the buffer descriptor table used by the channel. The value entered should be a pointer to a location in memory accommodating an aligned double fullword (this requires the three least significant bits of this pointer must be 000) There is one receive channel (x = 0).
------	-----------------------	--

MAL0_RXDEIR

MAL Receive Descriptor Interrupt Register

DCR 0x193 Read/Clear

See "Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)" on page 20-32.

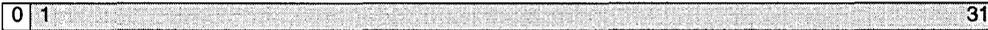


Figure 25-84. RX Descriptor Error Interrupt Register (MAL0_RXDEIR)

0		Receive Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0 etc.). When one or more bits are set, MAL_DESC_ERR_INT is set. Writing 1 to a bit resets it. There is only one RX channel in the PPC405GP.
1:31		Reserved	

MAL0_RXEOBISR

MAL Receive End-of-Buffer Interrupt Status Register

DCR 0x192 Read/Clear

See "End of Buffer Interrupt Status Registers" on page 20-28.

0	1	31
---	---	----

Figure 25-85. RX End of Buffer Interrupt Status Register (MAL0_RXEOBISR)

0	Receive Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, etc.). Writing 1 to a bit resets it. There is only one RX channel in the PPC405GP.
1:31	Reserved	

MAL0_TXCARR

MAL TX Channel Active Reset Register

DCR 0x185

See "Channel Active Set and Reset Registers" on page 20-26.

0	1	2	31
---	---	---	----

Figure 25-86. TX Channel_Active Reset Register (MAL0_TXCARR)

0:1		Transmit Channel Active Reset	Each bit represents its related channel (bit 0 for channel 0, etc.). When 1 is written to the bit, channel operation is disabled. There are only two TX channels in the PPC405GP.
2:31		Reserved	

MAL0_TXCASR

MAL TX Channel Active Set Register

DCR 0x184

See "Channel Active Set and Reset Registers" on page 20-26.

0	1	2	31
---	---	---	----

Figure 25-87. TX Channel_Active Set Register (MAL0_TXCASR)

0:1		Transmit Channel Active Set	Each bit represents its related channel (bit 0 for channel 0, etc.). When 1 is written to the bit, channel operation is enabled. There are only two TX channels in the PPC405GP.
2:31		Reserved	

MAL0_TXCTPxR

MAL Transmit Channel Pointer Registers

DCR 0x1A0–0x1A1 Read/Write

See “Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTP0R)” on page 20-33.

0	31
---	----

Figure 25-88. TX Channel Table Pointer x Register (MAL0_TXCTPxR)

0:31	Channel Table Pointer	Pointer to the base address in memory of the buffer descriptor table used by the channel. The value entered should be a pointer to a location in memory accommodating an aligned double fullword (this requires the three least significant bits of this pointer must be 000). There are two transmit channels (x = 0 and 1).
------	-----------------------	---

MAL0_TXDEIR

MAL Transmit Descriptor Interrupt Register

DCR 0x187 Read/Clear

See "Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)" on page 20-32.

0	1	2	31
---	---	---	----

Figure 25-89. TX Descriptor Error Interrupt Register (MAL0_TXDEIR)

0:1		Transmit Descriptor Error Interrupt	Each bit represents its related channel (bit 0 for channel 0, etc.). When one or more bits are set, MAL_DESC_ERR_INT is set. Writing 1 to a bit resets it. There are only two TX channels in the PPC405GP.
2:31		Reserved	

MAL0_TXEOBISR

MAL Transmit End-of-Buffer Interrupt Status Register

DCR 0x186 Read/Clear

See "End of Buffer Interrupt Status Registers" on page 20-28.

0	1	2	31
---	---	---	----

Figure 25-90. TX End of Buffer Interrupt Status Register (MAL0_TXEOBISR)

0:1		Transmit Channel End-of-Buffer Interrupt	Each bit represents its related channel (bit 0 for channel 0, etc.). Writing 1 to a bit resets it. There are only two TX channels in the PPC405GP.
2:31		Reserved	

See "Machine State Register (MSR)" on page 10-28.

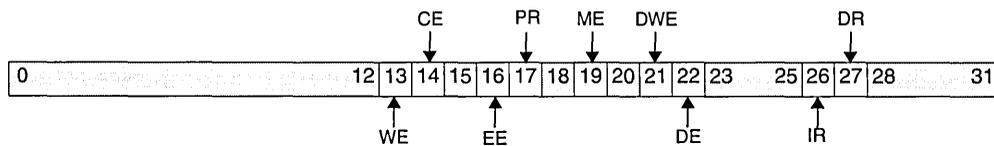


Figure 25-91. Machine State Register (MSR)

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 interrupts are disabled. 1 Critical interrupts are enabled.	Controls the critical interrupt input and watchdog timer first time-out interrupts.
15		Reserved	
16	EE	External Interrupt Enable 0 Asynchronous interrupts (external to the processor core) are disabled. 1 Asynchronous interrupts are enabled.	Controls the non-critical external interrupt input, PIT, and FIT interrupts.
17	PR	Problem State 0 Supervisor state (all instructions allowed). 1 Problem state (some instructions not allowed).	
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 Debug wait mode is disabled. 1 Debug wait mode is enabled.	
22	DE	Debug Interrupts Enable 0 Debug interrupts are disabled. 1 Debug interrupts are enabled.	
23:25		Reserved	
26	IR	Instruction Relocate 0 Instruction address translation is disabled. 1 Instruction address translation is enabled.	

MSR (cont.)

Machine State Register

27	DR	Data Relocate 0 Data address translation is disabled. 1 Data address translation is enabled.
28:31		Reserved

OCM0_DSARC

OCM Data-Side Address Range Compare Register

DCR 0x01A

See "OCM Data-Side Address Range Compare Register (OCM0_DSARC)" on page 5-6.



Figure 25-92. OCM Data-Side Address Range Compare Register (OCM0_DSARC)

0:5	DSAR	Data-side OCM address range
6:31		Reserved

OCM0_DSCNTL

OCM Data-Side Address Control Register

DCR 0x01B

See "OCM Data-Side Control Register (OCM0_DSCNTL)" on page 5-7.

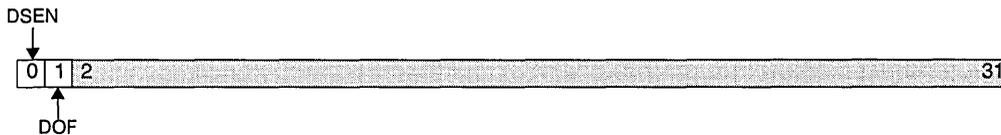


Figure 25-93. OCM Data-Side Control Register (OCM0_DSCNTL)

0	DSEN	Data-Side OCM Enable 0 Data-side OCM accesses are disabled. 1 Data-side OCM accesses are enabled.
1	DOF	This field should remain set to 1.
2:31		Reserved

OCM0_ISARC

OCM Instruction-Side Address Range Compare Register

DCR 0x018

See "OCM Instruction-Side Address Range Compare Register (OCM0_ISARC)" on page 5-5.



Figure 25-94. OCM Instruction-Side Address Range Compare Register (OCM0_ISARC)

0:5	ISAR	Instruction-side OCM address range
6:31		Reserved

OCM0_ISCNTL

OCM Instruction-Side Control Register

DCR 0x019

See “OCM Instruction-Side Control Register (OCM0_ISCNTL)” on page 5-6.

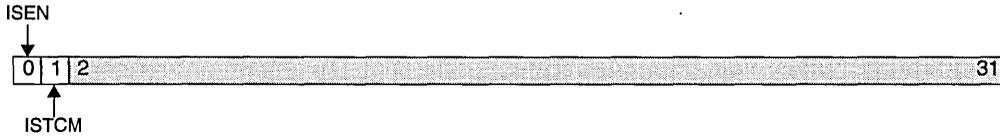


Figure 25-95. OCM Instruction-Side Control Register (OCM0_ISCNTL)

0	ISEN	<p>Instruction-Side OCM Enable</p> <p>0 Instruction-side OCM accesses are disabled.</p> <p>1 Instruction-side OCM accesses are enabled.</p>	
1	ISTCM	<p>Instruction-Side Two Cycle Mode</p> <p>0 Instruction-side OCM accesses are returned in one cycle.</p> <p>1 Instruction-side OCM accesses are returned in two cycles.</p>	<p>OCM0_ISCNTL[ISTCM], which has a reset value of 1, should be set to OCM0_ISCNTL[ISTCM] = 0 during chip initialization.</p>
2:31		Reserved	

OPBA0_CR

OPB Arbiter Control Register

MMIO 0xEF600600

See "OPB Arbiter Control Register (OPBA0_CR)" on page 2-12.



Figure 25-96. 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 01 Reserved 10 Master ID 2 11 Reserved Master 0 is DMA; master 2 is the OPB to PLB bridge.
5:31		Reserved

OPBA0_PR

OPB Arbiter Priority Register

MMIO 0xEF600601

See “OPB Arbiter Priority Register (OPBA0_PR)” on page 2-13.

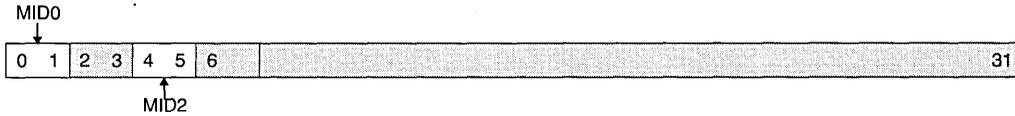


Figure 25-97. OPB Arbiter Priority Register (OPBA0_PR)

0:1	MID0	High Priority Master ID 00 Master ID 0 01 Reserved 10 Reserved 11 Reserved	At reset, this priority is assigned to DMA.
2:3		Reserved	
4:5	MID2	Low Priority master ID 00 Reserved 01 Reserved 10 Master ID 2 11 Reserved	At reset, this priority is assigned to the OPB to PLB bridge.
6:31		Reserved	

PCIC0_BAR0

PCI Base Address Register 0

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x10 Read-Only

See "Unused PCI Base Address Register Space" on page 17-37.

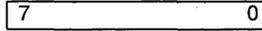


Figure 25-98. PCI Base Address Register (PCIC0_BAR0)

7:0	PCI Base Address	Unimplemented; always returns 0.
-----	------------------	----------------------------------

PCIC0_BIST

PCI Built In Self Test Control

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x0F Read-Only

See "PCI Built-In Self Test (BIST) Control Register (PCIC0_BIST)" on page 17-37.

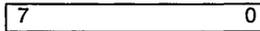


Figure 25-99. PCI Built-in Self Test Control Register (PCIC0_BIST)

7:0	PCI BIST Control
-----	------------------

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x4B-0x4A

See "Bridge Options 1 Register (PCIC0_BRDGOPT1)" on page 17-44.

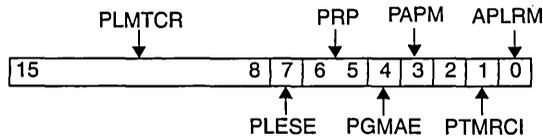


Figure 25-100. Bridge Options 1 Register (PCIC0_BRDGOPT1)

15:8	PMLTCR	PLB Master Latency Timer Count Register	PMLTCR contains the value used by the PLB master to load its latency timer. The granularity of this timer is 16 PLB cycles; therefore, the low-order bits of this register are read-only and are hardwired to 1.
7	PLESE	PLB Lock Error Status Enable 0 Slave error locking is disabled. 1 Slave error locking is enabled.	PLESE controls the handling of slave error locking.
6:5	PRP	PLB Request Priority 11 Highest 10 Next highest 01 Next highest 00 Lowest	PRP controls the request priority for PLB accesses.
4	PGMAE	PLB Guarded Memory Access Enable 0 Bridge PLB master memory accesses are unguarded. 1 Bridge PLB master memory accesses are guarded.	PGMAE controls whether PLB accesses are guarded or unguarded.
3	PAMP	PCI Arbiter Park Mode 0 The arbiter parks on requester 0 (the bridge PCI master). 1 The arbiter parks on the last master granted the bus.	PAPB defines how the internal PCI arbiter handles bus parking.
2:1	PTMRCI	PCI Target Memory Read Command Interpretation 00 Memory Read 01 Memory Read Line 10 Memory Read Multiples 11 Reserved	PTMRCI enables the PCI bridge to be forced to treat a PCI memory read as a memory read multiple, or as a memory read line, with respect to the burst size implied by the read commands. This is for masters that use memory read for multiple beat bursts.
0	APLRM	Atomic PLB Line Read Mode 0 1 PLB slave asserts Addrack and begins its data tenure immediately after the PCI master receives the first read data word.	APLRM controls the behavior of the bridge PLB slave with respect to PLB line reads. APLRM must not be set to 1 unless all PCI target devices can guarantee no disconnects for PLB line reads.

PCIC0_BRDGOPT2

PCI Bridge Options 2

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x61-0x60

See "Bridge Options 2 Register (PCIC0_BRDGOPT2)" on page 17-52.

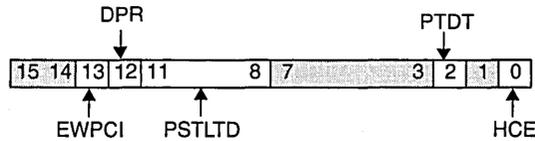


Figure 25-101. Bridge Options 2 Register (PCIC0_BRDGOPT2)

15:14		Reserved	
13	EWPCI	External Write to PCI Command Interrupt 0 No write to PCIC0_CMD has occurred. 1 External PCI master has written to PCIC0_CMD.	Software can also set or clear this bit.
12	DPR	Drive PCI Reset 0 Normal operation 1 Causes PCIReset pin to be asserted.	Software that asserts this bit must leave it asserted long enough to guarantee the PCI pulse width requirements. DPR does not reset PLB bus interface registers or PCI bridge registers. PCIReset is also asserted when the PCI bridge is reset.
11:8	PSSTLD	Subsequent Target Latency Timer Duration Specifies the number of PCI clocks that a PCI master burst can be held in a wait state before a target disconnect is initiated.	Only set on reads. In synchronous mode, PSSTLD equals the maximum number of PCI clocks to disconnect. In asynchronous mode, PSSTLD plus 3 equals the maximum number of PCI clocks to disconnect. The asynchronous value must be 2 or less.
7:3		Reserved	
2	PTDT	PCI Discard Timer Disable 0 Disabled 1 Enabled	When enabled, the PCI bridge never discards delayed read data.
1		Reserved	
0	HCE	Host Configuration Enable 0 Disabled 1 Enabled	HCE controls host PCI access to the PCI bridge configuration registers. All host attempts to access the PCI bridge PCI configuration registers are retried. This gives the local CPU (PLB master) time to initialize them before the host sees them.

PCIC0_CACHELS

PCI Cache Line Size

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x0C Read-Only

See "PCI Cache Line Size Register (PCIC0_CACHELS)" on page 17-35.



Figure 25-102. PCI Cache Line Size Register (PCIC0_CACHELS)

7:0	PCI Cache Line Size
-----	---------------------

PCIC0_CAP

PCI Capabilities Pointer

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x34 Read-Only

See "PCI Capabilities Pointer (PCIC0_CAP)" on page 17-40.



Figure 25-103. PCI Capabilities Pointer (PCIC0_CAP)



PCIC0_CAPID

PCI Capability Identifier

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x58 Read-Only

See "Capability Identifier (PCIC0_CAPID)" on page 17-49.

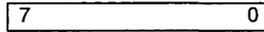


Figure 25-104. Capability Identifier (PCIC0_CAPID)

7:0	PCI Capability Identifier
-----	---------------------------

PCIC0_CFGADDR

PCI Configuration Address Register

0xEEC00000

See "PCI Configuration Address Register (PCIC0_CFGADDR)" on page 17-29.

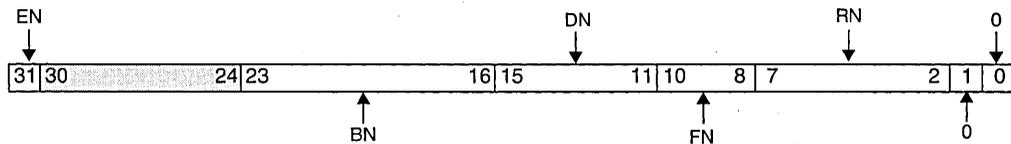


Figure 25-105. PCI Configuration Address Register (PCIC0_CFGADDR)

31	EN	Enable 0 Disabled 1 Enabled
30:24		Reserved
23:16	BN	Bus Number
15:11	DN	Device Number
10:8	FN	Function Number
7:2	RN	Register Number
1	0	
0	0	

PCIC0_CFGDATA

PCI Configuration Data Register

0xEEC00004

See "PCI Configuration Data Register (PCIC0_CFGDATA)" on page 17-30.

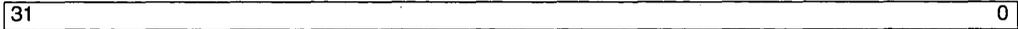


Figure 25-106. PCI Configuration Data Register (PCIC0_CFGDATA)



PCIC0_CLS

PCI Class Register

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x09 Read-Only (PCI), R/W (PLB)

See "PCI Class Register (PCIC0_CLS)" on page 17-35.

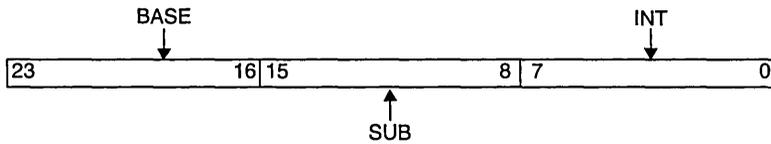


Figure 25-107. PCI Class Register (PCIC0_PCICLS)

23:16	BASE	Base Class	Reset to 0x06, which indicates bridge device. Users of the RISCWatch debugger must use the PCIC0_BASECC register to access this field.
15:8	SUB	Subclass	Reset to 00, which indicates host bridge. Users of the RISCWatch debugger must use the PCIC0_SUBCLS register to access this field.
7:0	INT	Interface Class	Reset to 00. Users of the RISCWatch debugger must use the PCIC0_INTCLS register to access this field.

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x05-0x04

See “PCI Command Register (PCIC0_CMD)” on page 17-31.

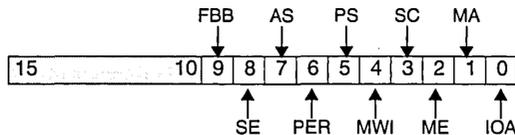


Figure 25-108. PCI Command Register (PCIC0_CMD)

15:10		Reserved	
9	FBB	Fast Back-to-Back Write Enable	Enables PCI masters to perform fast back-to-back transactions. Because the PCI bridge does not perform fast back-to-back transactions; FBB is read-only and returns 0 when read.
8	SE	PCISERR Enable 0 Disabled 1 Enabled	Enables driving PCISERR when a PCI bus parity error is detected when the PCI bridge is the PCI target. PCIC0_CMD[PER] must be enabled for address parity errors. PCIC0_CMD[PER] and PCIC0_ERREN[WDPE] must be enabled for write data parity errors.
7	AS	Address stepping wait states.	The PCI bridge does not address step (except for address stepping when generating a Config Type 0 cycle); AS is read-only and returns 0 when read.
6	PER	Parity error response 0 Disabled 1 Enabled	This bit is enabled for all types of PCI bus parity errors, including the following: <ul style="list-style-type: none"> • 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, detection of these errors is masked and PCIPERR (PERR#) is not asserted, although parity is still generated.
5	PS	Palette Snooping	Enable special palette snooping. The PCI bridge is not a VGA device; PS is read-only and returns 0 when read.
4	MWI	Memory Write and Invalidate Enable	The PCI bridge does not generate this command; MWI is read-only and returns 0 when read.
3	SC	Special Cycle Operations Enable	The PCI bridge never monitors special cycles; SC is read-only and returns 0 when read.

PCIC0_CMD (cont.)

PCI Command Register

2	ME	Master Enable 0 Disabled 1 Enabled	Enables PCI bridge-to-master cycles on the PCI bus. When ME is 0, the PCI bridge only responds as a PLB slave to PCIC0_CFGADDR, PCIC0_CFGDATA, and PCI bridge local configuration register access. Except for configuration cycles, the PCI bridge cannot master cycles to the PCI bus. If the pin strapping setting reflected in CPC0_PSR[RL] = 1, ME resets to 1.
1	MA	Memory Access 0 Disabled 1 Enabled	Controls PCI bridge response as a PCI memory target. MA is disabled at reset.
0	IOA	I/O Access	Controls the PCI bridge response as a PCI I/O target. The PCI bridge does not respond to I/O space accesses; IOA is read-only and returns 0 when read.

PCIC0_DATA

PCI Data

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x5F Read-Only

See "PCI Data Register (PCIC0_DATA)" on page 17-52.

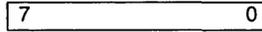
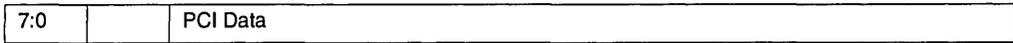


Figure 25-109. PCI Data (PCIC0_DATA)



PCIC0_DEVID

PCI Device ID

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x03–0x02 Read-Only (PCI), R/W (PLB)

See "PCI Device ID Register (PCIC0_DEVID)" on page 17-31.



Figure 25-110. PCI Device ID Register (PCIC0_DEVID)



Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x48

See “Error Enable Register (PCIC0_ERREN)” on page 17-42.

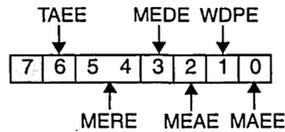


Figure 25-111. Error Enable Register (PCIC0_ERREN)

7		Reserved	
6	TAE	Target Abort Error Enable 0 Disabled 1 Enabled	While the PCI bridge is the PCI master, this bit enables the detection of a target abort as an error condition. If TAE is enabled, the PCI bridge reports PLB bus errors.
5:4	MERE	PLB Bus Error Response Enable 00 No action is taken. 01 The PCI target should drive $\overline{PCISERR}$ on the PCI bus. 10 Target should target abort the offending read. 11 Indicates the PCI target should drive $\overline{PCISERR}$ and target abort.	MERE controls the response taken by the PCI bridge on the PCI bus (as the PCI target) when PLB bus errors are asserted to the PCI bridge PLB master. Note: Only reads can be target aborted. Note: Modes 10 and 11 cannot be used in asynchronous mode.
3	MEDE	PLB Master Error Detection Enable 0 Disables detection of PLB master errors. 1 Enables detection of PLB master errors.	MEDE enables the detection of PLB bus errors when the PCI bridge is a PLB master.
2	MEAE	PLB Bus Error Assertion Enable 0 Disabled 1 Enabled	MEAE enables the reporting of a PLB bus error when the PCI bridge is a PLB slave.
1	WDPE	Write Data Parity $\overline{PCISERR}$ Enable 0 Disabled 1 Enabled.	The PCI bridge drives $\overline{PCISERR}$ when a data parity error is detected on a write cycle when the PCI bridge is the PCI target. PCIC0_CMD[SE] must also be 1.
0	MAEE	Master Abort Error Enable 0 Disabled 1 Enabled	MAEE enables the detection of a master abort as an error condition when the PCI bridge is the master. The PCI bridge drives SI_MErr on the PLB bus in response to a master abort. If this bit is disabled, driving of SI_MErr in response to master abort is masked.

PCIC0_ERRSTS

PCI Error Status

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x49

See “Error Status Register (PCIC0_ERRSTS)” on page 17-43.

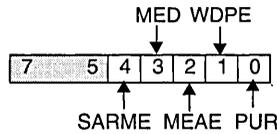


Figure 25-112. Error Status Register (PCIC0_ERRSTS)

7:5		Reserved	
4	SARME	PCISerr Asserted on Received PLB Bus Error	Set when PCI bridge asserts PCISerr on the PCI bus in response to PCI bridge receiving a PLB bus error while PLB master.
3	MED	PLB Bus Error Detected 1 Error detected	Set when a PLB bus error signal is asserted when PCI bridge is the PLB master. MED is set regardless of whether the PCI bridge is enabled to treat this as an error condition (the setting of MED is not maskable).
2	MEAE	PLB Bus Error Assertion Event 1 An PCI bridge error, which can cause a PLB bus error, occurred.	Set when an error occurs that would cause PCI bridge (as PLB slave) to assert a PLB bus error signal. MEAE is set regardless of whether the the PLB bus error assertion is enabled (the setting of MEAE is not maskable).
1	WDPE	PCISerr on Write Data Parity Error	Set when the PCI bridge drives PCISerr in response to a data parity error detected on a PCI write to PLB memory. PCIPerr is also driven.
0	PUR	PLB Unsupported Request	Set when the PCI bridge is a PLB slave and detects an unsupported request from a PLB master to an address range that PCI bridge decodes. The PCI bridge allows such requests to time out.

PCIC0_HDTYPE

PCI Header Type

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x0E Read-Only

See "PCI Header Type Register (PCIC0_HDTYPE)" on page 17-36.

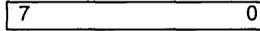


Figure 25-113. PCI Header Type Register (PCIC0_HDTYPE)

7:0	PCI Header Type
-----	-----------------

PCIC0_ICS

PCI Interrupt Control/Status

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x44

See "PCI Interrupt Control/Status Register (PCIC0_ICS)" on page 17-42.



Figure 25-114. PCI Interrupt Control/Status Register

7:1		Reserved	These bits return 0 when read.
0	API	Asset PCI interrupt	When software sets this bit, the PCI bridge asserts its Interrupt pin.

PCIC0_INTLN

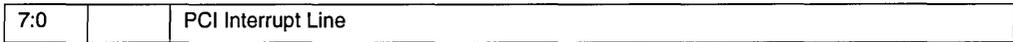
PCI Interrupt Line

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x3C

See "PCI Interrupt Line Register (PCIC0_INTLN)" on page 17-40.



Figure 25-115. PCI Interrupt Line Register (PCIC0_INTLN)



PCIC0_INTPN

PCI Interrupt Pin

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x3D Read-Only

See "PCI Interrupt Pin Register (PCIC0_INTPN)" on page 17-41.



Figure 25-116. PCI Interrupt Pin Register (PCIC0_INTPN)



PCIC0_LATTIM

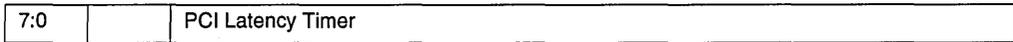
PCI Latency Timer

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x0D

See "PCI Latency Timer Register (PCIC0_LATTIM)" on page 17-36.



Figure 25-117. PCI Latency Timer Register (PCIC0_LATTIM)



PCIC0_MAXLTNCY

PCI Maximum Latency

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x3F Read-Only

See "PCI Maximum Latency Register (PCIC0_MAXLTNCY)" on page 17-41.



Figure 25-118. PCI Maximum Latency Register (PCIC0_MAXLTNCY)

7:0	PCI Maximum Latency
-----	---------------------

PCIC0_MINGNT

PCI Minimum Grant

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x3E Read-Only

See "PCI Minimum Grant Register (PCIC0_MINGNT)" on page 17-41.

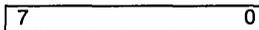


Figure 25-119. PCI Minimum Grant Register (PCIC0_MINGNT)

7:0	PCI Minimum Grant
-----	-------------------

PCIC0_NEXTIPTR

PCI Next Item Pointer

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x59 Read-Only

See "Next Item Pointer (PCIC0_NEXTIPTR)" on page 17-49.



Figure 25-120. Next Item Pointer (PCIC0_NEXTIPTR)



PCIC0_PLBBEAR

PLB Slave Error Address Register

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x57-0x54

See "PLB Slave Error Address Register (PCIC0_PLBBEAR)" on page 17-48.



Figure 25-121. PLB Slave Error Address Register (PCIC0_PLBBEAR)



PCIC0_PLBBESR0

PLB Slave Error Syndrome Register 0

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x4F-0x4C

See "PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0)" on page 17-45.

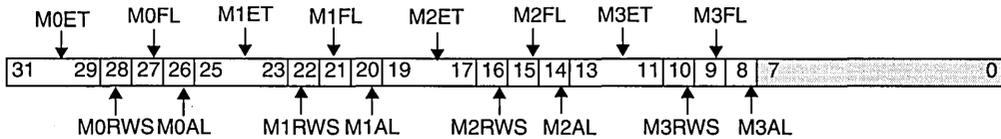


Figure 25-122. PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0)

31:29	M0ET	Master 0 Error Type 000 No Error 001 Parity Error 010 Reserved 011 Reserved 100 Reserved 101 Non-configured Bank Error 110 Reserved 111 Reserved
28	M0RWS	Master 0 Read/Write Status 0 Error operation was a write 1 Error operation was a read
27	M0FL	Master 0 PCIC0_PLBBESR0 Field Lock 0 PCIC0_PLBB ESR0 unlocked 1 PCIC0_PLBB ESR0 locked
26	M0AL	Master 0 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR unlocked by Master 0 1 PCIC0_PLBBEAR locked by Master 0
25:23	M1ET	Master 1 Error Type See PCIC0_PLBBESR0[M0ET]
22	M1RWS	Master 1 Read/Write Status 0 Error operation was a write 1 Error operation was a read
21	M1FL	Master 1 PCIC0_PLBBESR0 Field Lock 0 PCIC0_PLBB ESR0 unlocked 1 PCIC0_PLBB ESR0 locked
20	M1AL	Master 1 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR unlocked by Master 1 1 PCIC0_PLBBEAR locked by Master 1
19:17	M2ET	Master 2 Error Type See PCIC0_PLBBESR0[M0ET]
16	M2RWS	Master 2 Read/Write Status 0 Error operation was a write 1 Error operation was a read
15	M2FL	Master 2 PCIC0_PLBBESR0 Field Lock 0 PCIC0_PLBB ESR0 unlocked 1 PCIC0_PLBB ESR0 locked

PCIC0_PLBBESR0 (cont.)

PLB Slave Error Syndrome Register 0

14	M2AL	Master 2 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR unlocked by Master 2 1 PCIC0_PLBBEAR locked by Master 2
13:11	M3ET	Master 3 Error Type See PCIC0_PLBBESR0[M0ET]
10	M3RWS	Master 3 Read/Write Status 0 Error operation was a write 1 Error operation was a read
9	M3FL	Master 3 PCIC0_PLBBESR0 Field Lock 0 PCIC0_PLBBESR0 unlocked 1 PCIC0_PLBBESR0 locked
8	M3AL	Master 3 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR Unlocked by Master 2 1 PCIC0_PLBBEAR Locked by Master 2
7:0		Reserved

PCIC0_PLBBESR1

PLB Slave Error Syndrome Register 1

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x53–0x50

See “PLB Slave Error Syndrome Register 1 (PCIC0_PLBBESR1)” on page 17-47.

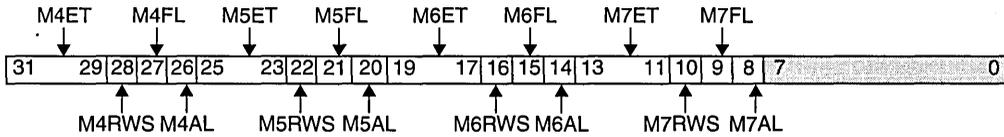


Figure 25-123. PLB Slave Error Syndrome 1 (PCIC0_PLBBESR1)

31:29	M4ET	Master 4 Error Type 000 No Error 001 Parity Error 010 Reserved 011 Reserved 100 Reserved 101 Non-configured Bank Error 110 Reserved 111 Reserved
28	M4RWS	Master 4 Read/Write Status 0 Write error operation 1 Read error operation
27	M4FL	Master 4 PCIC0_PLBBESR1 Field Lock 0 PCIC0_PLBBESR1 unlocked 1 PCIC0_PLBBESR1 locked
26	M4AL	Master 4 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR unlocked by Master 4 1 PCIC0_PLBBEAR locked by Master 4
25:23	M5ET	Master 5 Error Type See PCIC0_PLBBESR1[M4ET]
22	M5RWS	Master 5 Read/Write Status 0 Write error operation 1 Read error operation
21	M5FL	Master 5 PCIC0_PLBBESR1 Field Lock 0 PCIC0_PLBBESR1 Unlocked 1 PCIC0_PLBBESR1 Locked
20	M5AL	Master 5 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR unlocked by Master 5 1 PCIC0_PLBBEAR locked by Master 5
19:17	M6ET	Master 6 Error Type See PCIC0_PLBBESR1[M4ET]
16	M6RWS	Master 6 Read/Write Status 0 Write error operation 1 Read error operation
15	M6FL	Master 6 PCIC0_PLBBESR1 Field Lock 0 PCIC0_PLBBESR1 unlocked 1 PCIC0_PLBBESR1 locked

PCIC0_PLBBESR1 (cont.)

PLB Slave Error Syndrome Register 1

14	M6AL	Master 6 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR unlocked by Master 6 1 PCIC0_PLBBEAR locked by Master 6
13:11	M7ET	Master 7 Error Type See PCIC0_PLBBESR1[M4ET]
10	M7RWS	Master 7 Read/Write Status 0 Write error operation 1 Read error operation
9	M7FL	Master 7 PCIC0_PLBB ESR1 Field Lock 0 PCIC0_PLBBESR1 unlocked 1 PCIC0_PLBBESR1 locked
8	M7AL	Master 7 PCIC0_PLBBEAR Address Lock 0 PCIC0_PLBBEAR unlocked by Master 7 1 PCIC0_PLBBEAR locked by Master 7
7:0		Reserved

PCIC0_PMC

PCI Power Management Capabilities

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x5A (Read-Only)

See "Power Management Capabilities (PCIC0_PMC)" on page 17-50.

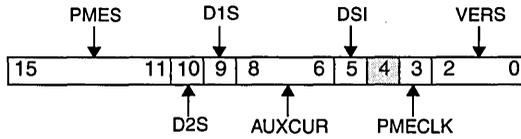


Figure 25-124. Power Management Capabilities Register (PCIC0_PMC)

15:11	PMES	PME Support	The PCI bridge does not support PME#; therefore, PMES is hardwired to 0b00000.
10	D2S	D2 Support Determines if the D2 power management state is supported.	The PCI bridge does not support the D2 power management state; therefore, D2S is hardwired to 0.
9	D1S	D1 Support Determines if the D1 power management state is supported.	The PCI bridge supports the D1 power management state; therefore, D1S is hardwired to 1.
8:6	AUXCUR	Auxiliary Current Support	The PCI bridge does not support Aux_Current; therefore, AUXCUR is hardwired to 0b000.
5	DSI	Device Specific Initialization 0 after reset	This 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.
4		Reserved	Always read as 0.
3	PMECLK		This bit is hardwired to 0 indicating that the function does not support PME# generation in any state.
2:0	VERS		Returns 0b010 on reads, indicating that PMC complies with Revision 1.1 of <i>PCI Power Management Interface Specification</i> .

PCIC0_PMCSR

PCI Power Management Control Status

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x5D–0x5C

See “Power Management Control/Status Register (PCIC0_PMCSR)” on page 17-50.

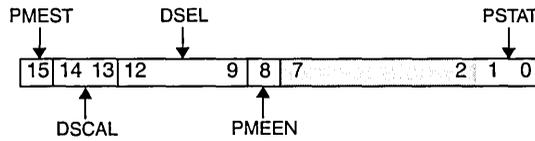


Figure 25-125. Power Management Control/Status Register (PCIC0_PMCSR)

15	PMEST		The PCI bridge does not support PME#, therefore, PMEST is hardwired to 0.
14:13	DSCAL		The PCI bridge does not support data register; therefore, DSCAL is hardwired to 0b00.
12:9	DSEL		The PCI bridge does not support a data register; therefore, DSEL is hardwired to 0b0000.
8	PMEEN		The PCI bridge does not support PME generation; therefore, PMEEN is hardwired to 0.
7:2		Reserved	Returns 0 when read.
1:0	PSTAT	Determine the current power state of a function and sets the function into a new power state. 00 D0 01 D1 10 D2 11 D3 Hot	If software attempts to write a value for an unsupported power state to PSTAT, its value does not change. Writing this field may change PCIC0_PMSCR.

PCIC0_PMCSRBSE

PMCSR PCI to PCI Bridge Support Extensions

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x5E Read-Only

See "PMCSR PCI-to-PCI Bridge Support Extensions (PCIC0_PMCSRBSE)" on page 17-51.



Figure 25-126. PMCSR PCI to PCI Bridge Support Extensions (PCIC0_PMCSRBSE)

7:0	PCI to PCI Bridge Support Extensions
-----	--------------------------------------

PCIC0_PMSCRR

PCI Power Management State Change Request Register

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x64

See “Power Management State Change Request Register (PCIC0_PMSCRR)” on page 17-54.

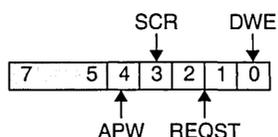


Figure 25-127. Power Management State Change Request Register (PCIC0_PMSCRR)

7:5		Reserved	Always read as 0.
4	APW	Accept PMCI0_PCMSR Writes Always 1 if DWE is 0.	The local processor sets APW when the local processor is ready to change the power management state. APW is cleared when the host configuration writes to the PCIC0_PMCSR register is accepted. The local processor can write 0 to APW.
3	SCR	State Change Request	The PCI bridge sets SCR when a host writes PCIC0_PMCSR to request a power management state change. This drives an interrupt to the local processor informing it of a state change request. The local processor must simultaneously clear SCR and set APW = 1 when the local processor is ready to change the state. After SCR is cleared, new requests are not detected until the outstanding delayed write is accepted. The local processor can set SCR = 1. Note that any host side write to any byte (0x5C–0x5F) is considered a power state change request.
2:1	REQST	Request State	Indicates the new power management state requested by a delayed host write to PCIC0_PMCSR. This field is read-only from the PLB side.
0	DWE	Delayed Write Enable 0 Immediate write 1 Delayed write	When DWE is set to 1, any configuration write to the PCIC0_PMCSR is completed as a delayed write. All writes to PCIC0_PMCSR are retried until the local processor sets the “Accept PCIC0_PMCSR Write bit” (bit 4). When 0, any configuration write to the PCIC0_PMCSR is completed immediately. DWE is a don’t care if a host write to PCIC0_PMCSR requests a state change from D3hot to D0.

PCIC0_PTM1BAR

PCI PTM 1 Base Address Range

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x17-0x14

See "PCI PTM 1 BAR (PCIC0_PTM1BAR)" on page 17-37.

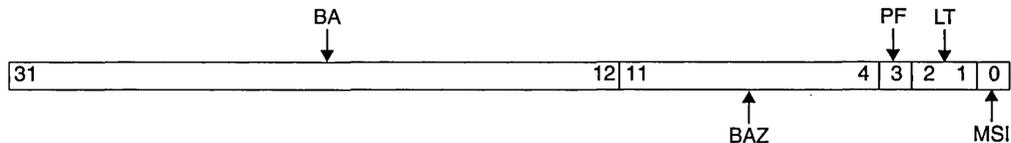


Figure 25-128. PCI PTM 1 BAR Register (PCIC0_PTM1BAR)

31:12	BA	Base Address These bits determine where in PCI memory address space this region is located.	Only corresponding bits in PCIL0_PTM1MS that are set to 1 are writable. Bits in PCIL0_PTM1MS that are set to 0 cause the corresponding Base Address register bits to be always 0. PCIL0_PTM1MS must be initialized by a PLB master before any PCI device is allowed to configure this register.
11:4	BAZ	Base Address Always Zero	BAZ = 0x00 because the minimum size of this range is 4KB.
3	PF	Prefetchable	PR = 1 to indicate that prefetching is allowed.
2:1	LT	Location Type	LT = 0b00 to indicate that the memory space can be located anywhere in the 32-bit address space.
0	MSI	Memory Space Indicator	MSI = 0 to indicate memory space, rather than I/O space.

PCIC0_PTM2BAR

PCI PTM 2 Base Address Range

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x1B-0x18

See "PCI PTM 2 BAR (PCIC0_PTM2BAR)" on page 17-38.

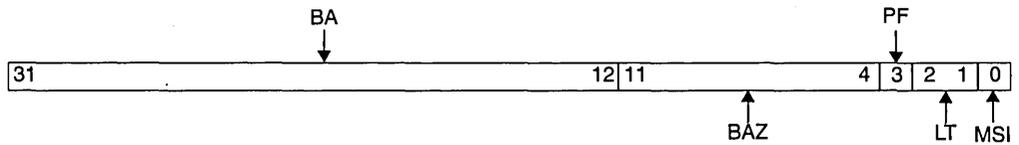


Figure 25-129. PCI PTM 2 BAR Register (PCIC0_PTM2BAR)

31:12	BA	Base Address These bits determine where in PCI Memory address space this region is located.	Only corresponding bits in PCIL0_PTM2MS that are set to 1 are writable. Bits in PCIL0_PTM2MS that are set to 0 cause the corresponding Base Address register bits to be always 0. PCIL0_PTM2MS must be initialized by a PLB master before any PCI device can configure this register.
11:4	BAZ	Base Address Always Zero	BAZ = 0x00 because the minimum size of this range is 4KB.
3	PF	Prefetchable	PF = 1 to indicate that prefetching is allowed.
2:1	LT	Location Type	LT = 0b00 to indicate that the memory space can be located anywhere in the 32-bit address space.
0	MSI	Memory Space Indicator	MSI = 0 to indicate memory space, rather than I/O space.

PCIC0_REVID

PCI Revision ID

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x08 Read-Only

See "PCI Revision ID Register (PCIC0_REVID)" on page 17-34.

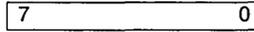


Figure 25-130. PCI Revision ID Register (PCIC0_REVID)

7:0	Revision ID	Revision level of device.
-----	-------------	---------------------------

PCIC0_SBSYSID

PCI Subsystem ID

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; Offset 0x2D–0x2C

See “PCI Subsystem ID Register (PCIC0_SBSYSID)” on page 17-39.

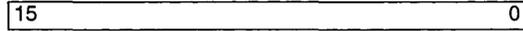


Figure 25-131. PCI Subsystem ID Register (PCIC0_SBSYSID)



PCIC0_SBSYSVID

PCI Subsystem Vendor ID

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; Offset 0x2F-0x2E

See "PCI Subsystem Vendor ID Register (PCIC0_SBSYSVID)" on page 17-39.



Figure 25-132. PCI Subsystem Vendor ID Register (PCIC0_SBSYSVID)

15:0	PCI Subsystem Vendor ID
------	-------------------------

PCIC0_STATUS

PCI Status Register

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x07–0x06

See “PCI Status Register (PCIC0_STATUS)” on page 17-33.

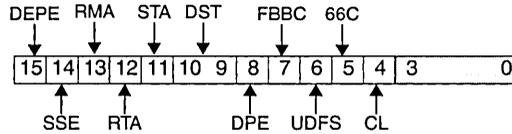


Figure 25-133. PCI Status Register (PCIC0_STATUS)

15	DEPE	Detected Parity Error Write 1 to clear.	The PCI bridge sets DEPE when the PCI bridge detects a PCI bus parity error, regardless of the setting of any enable bits (DEPE is non-maskable). The following events set DEPE: <ul style="list-style-type: none"> • PCI address bus parity error detected when PCI bridge is a target. • PCI data bus parity error detected when a PCI master writes to PLB memory (PCI bridge is the target). • PCI data bus parity error detected when PCI bridge masters a PCI read cycle.
14	SSE	Signaled System Error Write 1 to clear.	The PCI bridge sets SSE if the PCI bridge asserts PCISerr (see “Error Handling” on page 17-55 for causes of PCISerr assertion).
13	RMA	Received Master Abort Write 1 to clear.	The PCI bridge sets RTA when a PCI cycle for which the PCI bridge is the master is terminated with master abort.
12	RTA	Received Target Abort Write 1 to clear.	The PCI bridge sets RTA when a PCI cycle for which it is the master is terminated with target abort.
11	STA	Signaled Target Abort Write 1 to clear.	The PCI bridge sets STA when a PCI cycle for which it is the target is terminated with target abort.
10:9	DST	PCIDevSel Response Timing Read-only.	The PCI bridge asserts PCIDevSel on the second clock after PCIFframe is asserted (called medium response time). Read-only; always returns 0b01 when read.
8	DPE	Data Parity Error Detected Write 1 to clear.	DPE is set when the following conditions are met: <ul style="list-style-type: none"> • The PCI bridge detects a data parity error (PCIPerr is asserted) when the PCI bridge is the master on a PCI read cycle, or is the master when it samples PCIPerr asserted on a PCI write cycle. • PCIC0_CMD[PER] = 1.

PCIC0_STATUS (cont.)

PCI Status Register

7	FBBC	Fast Back-to-Back Capable Read-only; returns 0 when read.	Indicates that the PCI target can accept fast back-to-back transactions when the transactions are not to the same agent. The PCI bridge target does not accept this type of fast back-to-back transaction.
6	UDFS	UDF Supported Read-only; returns 0 when read.	Indicates device support of user-definable features. The PCI bridge does not support user-definable features.
5	66C	66 MHz Capable 0 At reset 1 PCI bridge is configured for 66MHz operation.	Indicates that the device can run at 66 MHz. The PCI bridge can be configured to run at 33 MHz max or 66 MHz. The local CPU (PLB master) sets 66C to 1 if PCI bridge is configured for 66 MHz operation.
4	CL	Capabilities List This bit is read only and returns 1 when read.	Indicates that the value at offset 0x34 is a pointer in configuration space to a linked list of new capabilities.
3:0		Reserved	These bits return 0s when read.

PCIC0_VENDID

PCI Vendor ID

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; Offset 0x01–0x00 Read-Only (PCI), R/W (PLB)

See “PCI Vendor ID Register (PCIC0_VENDID)” on page 17-31.

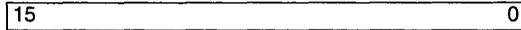


Figure 25-134. PCI Vendor ID Register (PCIC0_VENDID)



PCIL0_PMM0LA

PMM 0 Local Address

MMIO 0xEF400000

See "PMM 0 Local Address Register (PCIL0_PMM0LA)" on page 17-21.

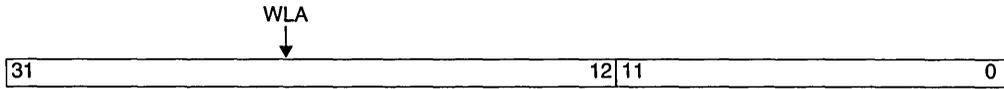


Figure 25-135. PMM 0 Local Address Register (PCIL0_PMM0LA)

31:12	WLA	Writable PLB Local Address
11:0		PLB Local Address Always 0

PCILO_PMM0MA

PMM 0 Mask/Attribute

MMIO 0xEF400004

See "PMM 0 Mask/Attribute Register (PCILO_PMM0MA)" on page 17-22.

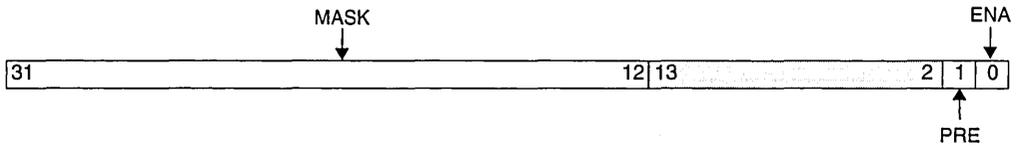


Figure 25-136. PMM 0 Mask/Attribute Register (PCILO_PMM0MA)

31:12	MASK	The mask bits determine the size of the address map range.	The mask must be of the form 111....0000. Bits set to 1 cause the corresponding PCILO_PMM0LA bits to be compared with incoming PLB addresses. Note that the minimum range size is 4KB, and valid ranges are powers of 2. For example, a 128MB range would be encoded as 0xF8000 and a 4KB range would be encoded as all ones.
11:2		Reserved	Returns 0 when read.
1	PRE	Read Prefetching Enable 1 Read prefetching is enabled.	If read prefetch is enabled, the PCI bridge prefetches 64 bytes from PCI memory in response to a PLB single-beat, byte-burst, or half word burst read from PMM 0.
0	ENA	PLB to PCI Memory Mapping Enable 1 Memory mapping is enabled.	Note that PCILO_PMM0LA, PCILO_PMM0PCIHA, and PCILO_PMM0PCILA must be initialized before enabling.

PCILO_PMM0PCIHA

PMM 0 PCI High Address

MMIO 0xEF40000C

See "PMM 0 PCI High Address Register (PCILO_PMM0PCIHA)" on page 17-23.



Figure 25-137. PMM 1 PCI High Address Register (PCILO_PMM1PCIHA)



PCILO_PMM0PCILA

PMM 0 PCI Low Address

MMIO 0xEF400008

See "PMM 0 PCI Low Address Register (PCILO_PMM0PCILA)" on page 17-22.

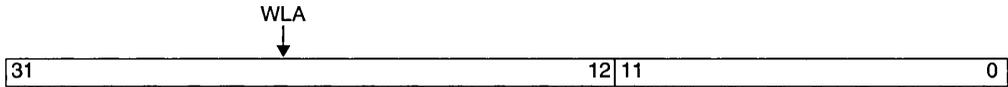


Figure 25-138. PMM 0 PCI Low Address Register (PCILO_PMM0PCILA)

31:12	WLA	Writable PCI Low Address
11:0		PCI Low Address Always 0

PCI0_PMM1LA

PMM 1 Local Address

MMIO 0xEF400010

See "PMM 1 Local Address Register (PCI0_PMM1LA)" on page 17-23.



Figure 25-139. PMM 1 Local Address Register (PCI0_PMM1LA)



PCIL0_PMM1MA

PMM 1 Mask/Attribute

MMIO 0xEF400014

See "PMM 1 Mask/Attribute Register (PCIL0_PMM1MA)" on page 17-24.

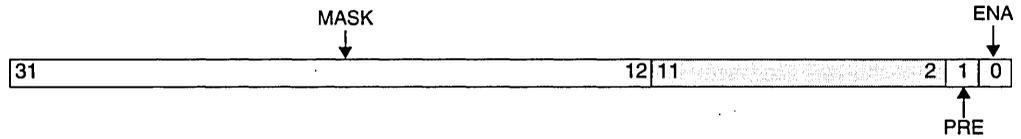


Figure 25-140. PMM 1 Mask/Attribute Register (PCIL0_PMM1MA)

31:12	MASK	The mask bits determine the size of the address map range.	The mask must be of the form 111....0000. Bits set to 1 cause the corresponding PCIL0_PMM1LA bits to be compared with incoming PLB addresses. Note that the minimum range size is 4KB, and valid ranges are powers of 2. For example, a 128MB range would be encoded as 0xF8000 and a 4KB range would be encoded as 0x11111.
11:2		Reserved	Returns 0 when read.
1	PRE	Read Prefetching Enable 1 Read prefetching is enabled.	If read prefetch is enabled, the PCI bridge prefetches 64 bytes from PCI memory in response to a PLB single-beat, byte-burst, or half word burst read from PMM 0.
0	ENA	PLB to PCI Memory Mapping Enable 1 Memory mapping is enabled.	Note that PCIL0_PMM1LA, PCIL0_PMM1PCIHA, and PCIL0_PMM1PCILA must be initialized before enabling.

PCIL0_PMM1PCIHA

PMM 1 PCI High Address

MMIO 0xEF40001C

See "PMM 1 PCI High Address Register (PCIL0_PMM1PCIHA)" on page 17-25.

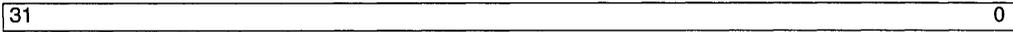


Figure 25-141. PMM 1 PCI High Address Register (PCIL0_PMM1PCIHA)



PCIL0_PMM1PCILA

PMM 1 PCI Low Address

MMIO 0xEF400018

See "PMM 1 PCI Low Address Register (PCIL0_PMM1PCILA)" on page 17-24.

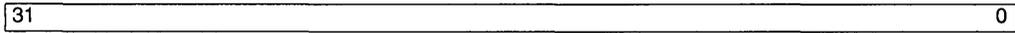


Figure 25-142. PMM 1 PCI Low Address Register (PCIL0_PMM1PCILA)



PCIL0_PMM2LA

PMM 2 Local Address

MMIO 0xEF400020

See "PMM 2 Local Address Register (PCIL0_PMM2LA)" on page 17-25.



Figure 25-143. PMM 2 Local Address Register (PCIL0_PMM2LA)



PCI0_PMM2MA

PMM 2 Mask/Attribute

MMIO 0xEF400024

See "PMM 2 Mask/Attribute Register (PCI0_PMM2MA)" on page 17-25.

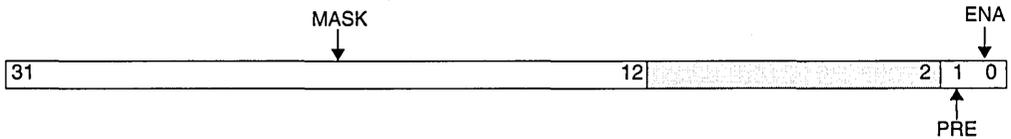


Figure 25-144. PMM 2 Mask/Attribute Register (PCI0_PMM2MA)

31:12	MASK	The mask bits determine the size of the address map range.	The mask must be of the form 111....0000. Bits set to 1 cause the corresponding PCI0_PMM2LA bits to be compared with incoming PLB addresses. Note that the minimum range size is 4KB, and valid ranges are powers of 2. For example, a 128MB range would be encoded as 0xF8000 and a 4KB range would be encoded as 0x11111.
11:2		Reserved	Returns 0 when read.
1	PRE	Read Prefetching Enable 1 Read prefetching is enabled.	If read prefetch is enabled, the PCI bridge prefetches 64 bytes from PCI memory in response to a PLB single-beat, byte-burst, or half word burst read from PMM 0.
0	ENA	PLB to PCI Memory Mapping Enable 1 Memory mapping is enabled.	Note that PCI0_PMM2LA, PCI0_PMM2PCIHA, and PCI0_PMM2PCILA must be initialized before enabling.

PCIL0_PMM2PCIHA

PMM 2 PCI High Address

MMIO 0xEF40002C

See "PMM 2 PCI High Address Register (PCIL0_PMM2PCIHA)" on page 17-27.



Figure 25-145. PMM 2 PCI High Address Register (PCIL0_PMM2PCIHA)



PCIL0_PMM2PCILA

PMM 2 PCI Low Address

MMIO 0xEF400028

See "PMM 2 PCI Low Address Register (PCIL0_PMM2PCILA)" on page 17-26.



Figure 25-146. PMM 2 Low Address Register (PCIL0_PMM2PCILA)



PCI0_PTM1LA

PTM 1 Local Address

MMIO 0xEF400034

See "PTM 1 Local Address Register (PCI0_PTM1LA)" on page 17-27.



Figure 25-147. PTM 2 Local Address Register (PCI0_PTM1LA)

31:12	WLA	Writable PTM 1 Local Address	Writable
11:0		PTM 1 Local Address	Always 0

PCI0_PTM1MS

PTM 1 Memory Size

MMIO 0xEF400030

See "PTM 1 Memory Size/Attribute Register (PCI0_PTM1MS)" on page 17-27.

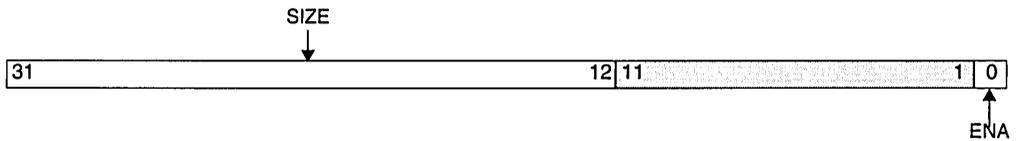


Figure 25-148. PTM 1 Memory Size/Attribute Register (PCI0_PTM1MS)

31:12	MASK	Defines the size of the region of PCI memory space that is mapped to local (PLB) space using PTM 1.	The minimum range size is 4KB. Valid ranges are always a power of 2. For example, a value of 0xFF000000 indicates that the region contains 16MB.
11:1		Reserved	Returns 0 when read.
0	EMM	Determines if range 1 is enabled to map PCI memory space to PLB space. Always 1 (enabled).	

PCIL0_PTM2LA

PTM 2 Local Address

MMIO 0xEF40003C

See "PTM 2 Local Address Register (PCIL0_PTM2LA)" on page 17-28.

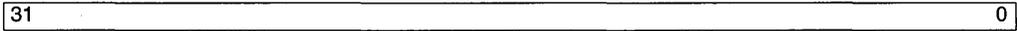
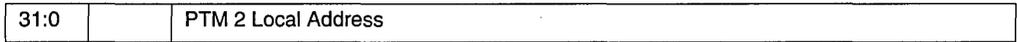


Figure 25-149. PTM 2 Local Address Register (PCIL0_PTM2LA)



PCIL0_PTM2MS

PTM 2 Memory Size

MMIO 0xEF400038

See "PTM 2 Memory Size/Attribute Register (PCIL0_PTM2MS)" on page 17-28.

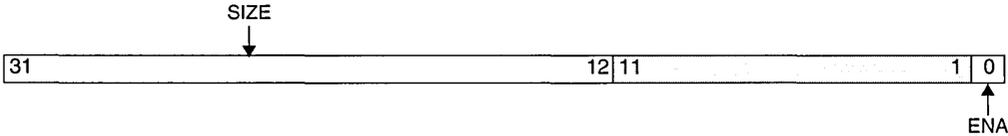


Figure 25-150. PTM 2 Memory Size/Attribute Register (PCIL0_PTM2MS)

31:12	MASK	Defines the size of the region of PCI memory space mapped to local (PLB) space using PTM 2.	The minimum range size is 4KB. Valid ranges are always a power of 2. For example, a value of 0xFF000000 indicates that the region contains 16MB.
11:1		Defines the size of the region of PCI memory space mapped to local (PLB) space using PTM 2.	The minimum range size is 4KB. Valid ranges are always a power of 2. For example, a value of 0xFF000000 indicates that the region contains 16MB.
0	EMM	Determines if range 2 is enabled to map PCI memory space to PLB space.	When EMM is disabled, PCIC0_PTM2BAR cannot be written. Set PCIC0_PTM2BAR to 0 before disabling EMM.

PID

Process ID

SPR 0x3B1

See "Address Translation" on page 6-1.



Figure 25-151. Process ID (PID)

0:23	Reserved
24:31	Process ID

SPR 0x3DB

See "Programmable Interval Timer (PIT)" on page 11-4.



Figure 25-152. Programmable Interval Timer (PIT)

0:31		Programmed interval remaining	Number of clocks remaining until the PIT event
------	--	-------------------------------	--

PLB0_ACR

PLB Arbiter Control Register

DCR 0x087

See "PLB Arbiter Control Register (PLB0_ACR)" on page 2-5.



Figure 25-153. PLB Arbiter Control Register (PLB0_ACR)

0	PPM	PLB Priority Mode 0 Fixed 1 Fair
1:3	PPO	PLB Priority Order 000 Masters 0, 1, 2, 3, 4, 5 001 Masters 1, 2, 3, 4, 5, 0 010 Masters 2, 3, 4, 5, 0, 1 011 Masters 3, 4, 5, 0, 1, 2 100 Masters 4, 5, 0, 1, 2, 3 101 Masters 5, 0, 1, 2, 3, 4 110 Reserved 111 Reserved
4	HBU	High Bus Utilization 0 Disabled 1 Enabled
5:31		Reserved

PLB0_BEAR

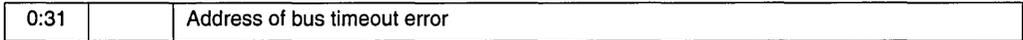
PLB Error Address Register

DCR 0x086

See "PLB Error Address Register (PLB0_BEAR)" on page 2-5.



Figure 25-154. PLB Error Address Register (PLB0_BEAR)



PLB0_BESR

PLB Error Status Register

DCR 0x084 Read/Clear

See "PLB Error Status Register (PLB0_BESR)" on page 2-6.

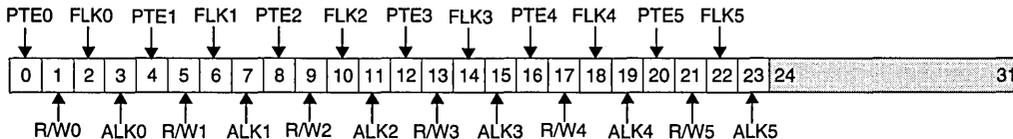


Figure 25-155. PLB Error Status Register (PLB0_BESR)

0	PTE0	Master 0 PLB Timeout Error Status 0 No master 0 timeout error 1 Master 0 timeout error	Master 0 is the processor core 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 processor core 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 external master.
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	AL2	Master 2 PLB0_BEAR Address Lock 0 Master 2 PLB0_BEAR is unlocked 1 Master 2 PLB0_BEAR is locked	

PLB0_BESR (cont.)

PLB Error Status Register

12	PTE3	Master 3 PLB Timeout Error Status 0 No Master 3 timeout error 1 Master 3 timeout error	Master 3 is PCI.
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 MAL.
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 DMA.
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	
24:31		Reserved	

POB0_BEAR

Bridge Error Address Register

DCR 0x0A2 Read-Only

See "Bridge Error Address Register (POB0_BEAR)" on page 2-8.



Figure 25-156. Bridge Error Address Register (POB0_BEAR)

0:31	0x0B2	Address of bus error
------	-------	----------------------

POB0_BESR0

Bridge Error Status Register 0

DCR 0x0A0 Read/Clear

See "Bridge Error Status Registers (POB0_BESR0–POB0_BESR1)" on page 2-8.

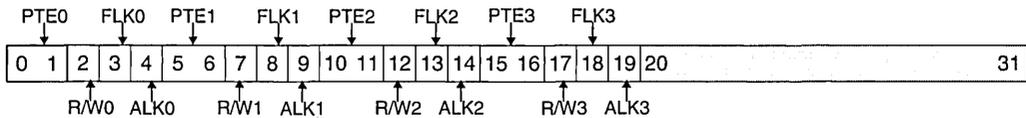


Figure 25-157. Bridge Error Status Register 0 (POB0_BESR0)

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 processor core ICU.
2	R/W0	Read Write Status Master 0 0 Master 0 error operation is a read 1 Master 0 error operation is a write	
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	
5:6	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 processor core DCU.
7	R/W1	Read/Write Status Master 1 0 Master 1 error operation is a read 1 Master 1 error operation is a write	
8	FLK1	POB0_BESR0 Field Lock Master 1 0 Master 1 POB0_BESR0 field is unlocked 1 Master 1 POB0_BESR0 field is locked	
9	ALK1	POB0_BEAR Address Lock Master 1 0 Master 1 POB0_BEAR address is unlocked 1 Master 1 POB0_BEAR address is locked	
10:11	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 external master.
12	R/W2	Read/Write Status Master 2 0 Master 2 error operation is a read 1 Master 2 error operation is a write	

POB0_BESR0 (cont.)

Bridge Error Status Register 0

13	FLK2	POB0_BESR0 Field Lock Master 2 0 Master 2 POB0_BESR0 field is unlocked 1 Master 2 POB0_BESR0 field is locked
14	ALK2	POB0_BEAR Address Lock Master 2 0 Master 2 POB0_BEAR address is unlocked 1 Master 2 POB0_BEAR address is locked
15:16	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
17	R/W3	Read/Write Status Master 3 0 Master 3 error operation is a read 1 Master 3 error operation is a write
18	FLK3	POB0_BESR0 Field Lock Master 3 0 Master 3 POB0_BESR0 field is unlocked 1 Master 3 POB0_BESR0 field is locked
19	ALK3	POB0_BEAR Address Lock Master 3 0 Master 3 POB0_BEAR address is unlocked 1 Master 3 POB0_BEAR address is locked
20:31		Reserved

DCR 0x0A4 Read/Clear

See “Bridge Error Status Registers (POB0_BESR0–POB0_BESR1)” on page 2-8.

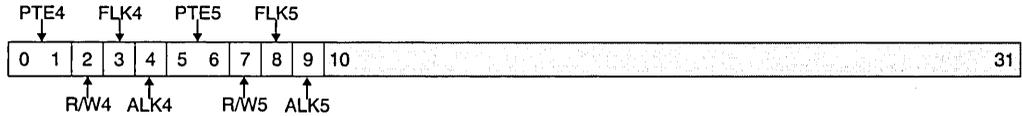


Figure 25-158. Bridge Error Status Register 1 (POB0_BESR1)

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 MAL.
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_BESR1 Field Lock Master 4 0 Master 4 POB0_BESR1 field is unlocked 1 Master 4 POB0_BESR1 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	
5:6	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 DMA.
7	R/W5	Read/Write Status Master 5 0 Master 5 error operation is a read 1 Master 5 error operation is a write	
8	FLK5	POB0_BESR1 Field Lock Master 5 0 Master 5 POB0_BESR1 field is unlocked 1 Master 5 POB0_BESR1 field is locked	
9	ALK5	POB0_BEAR Address Lock Master 5 0 Master 5 POB0_BEAR address is unlocked 1 Master 5 POB0_BEAR address is locked	
10:31		Reserved	

PVR

Processor Version Register

SPR 0x11F Read-Only

See "Processor Version Register (PVR)" on page 3-12.

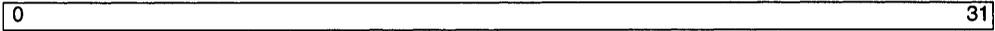


Figure 25-159. Processor Version Register (PVR)



SDRAM0_B0CR–SDRAM0_B3CR

Memory Bank 0–3 Configuration Registers

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x40–0x4C

See “Memory Bank 0–3 Configuration (SDRAM0_B0CR–SDRAM0_B3CR)” on page 15-6.

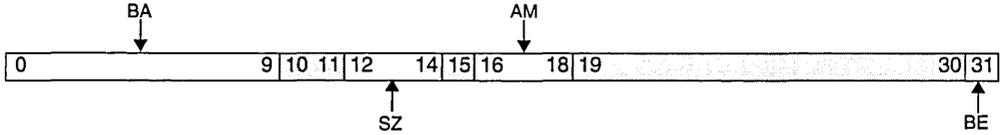


Figure 25-160. Memory Bank 0–3 Configuration Registers (SDRAM0_B0CR–SDRAM0_B3CR)

0:9	BA	Base Address	The base address must be aligned on a boundary that matches the size of the region as defined by the SZ field. For example, a 4MB region must begin on an address that is divisible by 4MB.
10:11		Reserved	
12:14	SZ	Size 000 4M byte 001 8M byte 010 16M byte 011 32M byte 100 64M byte 101 128M byte 110 256M byte 111 Reserved	
15		Reserved	
16:18	AM	Addressing Mode 000 Mode 1 001 Mode 2 010 Mode 3 011 Mode 4 100 Mode 5 101 Mode 6 110 Mode 7 111 Reserved	See “SDRAM Addressing Modes” on page 15-7.
19:30		Reserved	
31	BE	Memory Bank Enable 0 Bank is disabled 1 Bank is enabled	

SDRAM0_BEAR

PLB Master Bus Error Address Register

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x10

See "Bus Error Address Register (SDRAM0_BEAR)" on page 15-17.

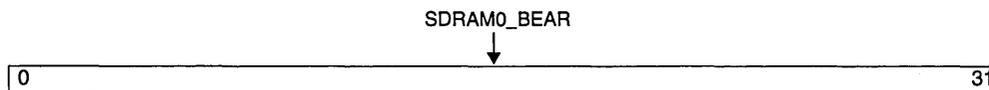


Figure 25-161. Bus Error Address Register (SDRAM0_BEAR)

0:31	SDRAM0_BEAR	Address of ECC Error.
------	-------------	-----------------------

SDRAM0_BESR0

Bus Error Syndrome Register 0

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x00 Read/Clear

See "Bus Error Syndrome Register 0 (SDRAM0_BESR0)" on page 15-17.

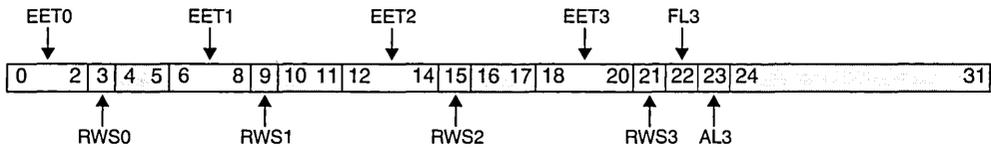


Figure 25-162. Bus Error Syndrome Register 0 (SDRAM0_BESR0)

0:2	EET0	Error type for master 0 000 No error 001 Reserved 01X ECC uncorrectable error 1XX Reserved	Master 0 is the processor instruction fetcher.
3	RWS0	Read/write status for master 0 0 Error operation was a write operation 1 Error operation was a read operation	
4:5		Reserved	
6:8	EET1	Error type for master 1 000 No error 001 Reserved 01X ECC uncorrectable error 1XX Reserved	Master 1 is the processor data side.
9	RWS1	Read/write status for master 1 0 Error operation was a write operation 1 Error operation was a read operation	
10:11		Reserved	
12:14	EET2	Error type for master 2 000 No error 001 Reserved 01X ECC uncorrectable error 1XX Reserved	Master 2 is the external bus master.
15	RWS2	Read/write status for master 2 0 Error operation was a write operation 1 Error operation was a read operation	
16:17		Reserved	
18:20	EET3	Error type for master 3 000 No error 001 Reserved 01X ECC uncorrectable error 1XX Reserved	Master 3 is the PCI bridge.
21	RWS3	Read/write status for master 3 0 Error operation was a write operation 1 Error operation was a read operation	

SDRAM0_BESR0 (cont.)

Bus Error Syndrome Register 0

22	FL3	Field lock for master 3 0 EET3 and RWS3 fields are unlocked 1 EET3 and RWS3 fields are locked
23	AL3	SDRAM0_BEAR address lock for master 3 0 SDRAM0_BEAR address unlocked 1 SDRAM0_BEAR address locked
24:31		Reserved

SDRAM0_BESR1

Bus Error Syndrome Register B

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x08 Read/Clear

See “Bus Error Syndrome Register 1 (SDRAM0_BESR1)” on page 15-18.

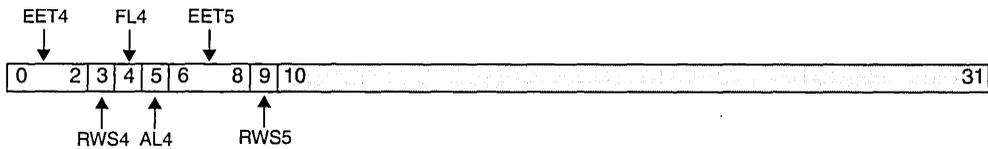


Figure 25-163. Bus Error Status Register 1 (SDRAM0_BESR1)

0:2	EET4	Error type for master 4 000 No error 001 Reserved 01X ECC uncorrectable error 1XX Reserved	Master 4 is the MAL.
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 EET4 and RWS4 fields are unlocked 1 EET4 and RWS4 fields are locked	
5	AL4	SDRAM0_BEAR address lock for master 4 0 SDRAM0_BEAR address unlocked 1 SDRAM0_BEAR address locked	
6:8	EET5	Error type for master 5 000 No error 001 Reserved 01X ECC uncorrectable error 1XX Reserved	Master 5 is the DMA controller.
9	RWS5	Read/write status for master 5 0 Error operation was a write operation 1 Error operation was a read operation	
10:31		Reserved	

SDRAM0_CFG

Memory Controller Configuration Register

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x20

See “Memory Controller Configuration Register (SDRAM0_CFG)” on page 15-3.

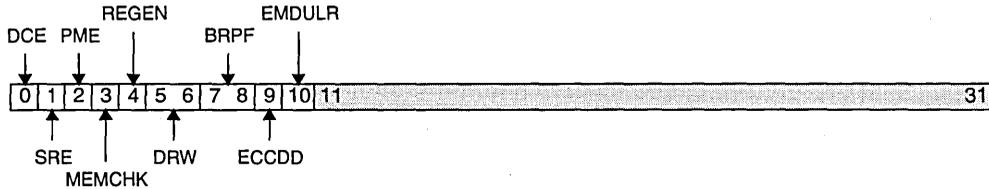


Figure 25-164. Memory Controller Configuration (SDRAM0_CFG)

0	DCE	SDRAM Controller Enable 0 Disable 1 Enable	All SDRAM controller configuration registers must be initialized and valid prior to setting DCE.
1	SRE	Self-Refresh Enable 0 Disable 1 Enable	See “Self-Refresh” on page 15-19.
2	PME	Power Management Enable 0 Disable 1 Enabled	See “Power Management” on page 15-20.
3	MEMCHK	Memory Data Error Checking 0 None 1 ECC	See “Error Checking and Correction (ECC)” on page 15-14.
4	REGEN	Registered Memory Enable 0 Disabled 1 Enabled	
5:6	DRW	SDRAM Width 00 32-bit 01 Reserved 10 Reserved 11 Reserved	Must be set to 2'b00.
7:8	BRPF	Burst Read Prefetch Granularity 00 Reserved 01 16 bytes 10 32 bytes 11 Reserved	Most applications should set with field to 2'b01.
9	ECCDD	ECC Driver Disable 0 Check bit data output on ECC7:0. 1 ECC7:0 are placed in high-Z state.	Regardless of whether ECC checking is enabled, SDRAM writes cause the PPC405GP to output check bit data on ECC7:0. If ECC is not used, setting ECCDD=1 disables these drivers.
10	EMDULR	Enable Memory Data Unless Read 0 MemData0:31 are high-Z unless a memory write is being performed. 1 MemData0:31 are driven unless a memory read is being performed.	

SDRAM0_CFG

Memory Controller Configuration Register

11:31		Reserved
-------	--	----------

SDRAM0_CFGADDR

Memory Controller Address Register

DCR 0x010

See "Accessing SDRAM Registers" on page 15-2.

This register is used to determine offsets for SDRAM controller DCRs.

SDRAM0_CFGDATA

Memory Controller Data Register

DCR 0x011

See “Accessing SDRAM Registers” on page 15-2.

This register is used to indirectly access SDRAM controller DCRs.

SDRAM_ECCCFG

ECC Configuration Register

Offset 0x94 DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x94

See "ECC Configuration Register (SDRAM0_ECCCFG)" on page 15-14.



Figure 25-165. ECC Configuration Register (SDRAM0_ECCCFG)

0:7		Reserved	
8:11	CEn	ECC Correction Enable for bank n. 0 Disabled 1 Enabled	When CEn is set, ECC correction is enabled for bank n (<u>BankSel</u> n). When cleared, the ECC logic ignores the check bits and passes read data along unmodified.
12:31		Reserved	

SDRAM0_ECCESR

ECC Error Status Register

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x98 Read-Only

See "ECC Error Status Register (SDRAM0_ECCESR)" on page 15-16.

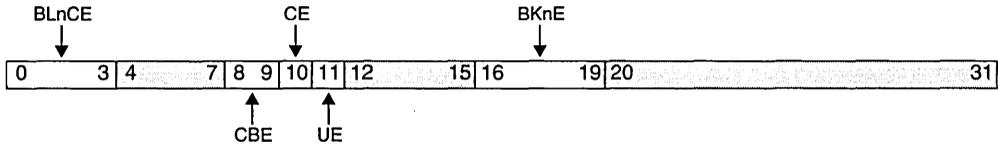


Figure 25-166. ECC Error Status Register (SDRAM0_ECCESR)

0:3	BLnCE	Byte Lane n Corrected Error 0 No error 1 Error occurred in byte lane n
4:7		Reserved
8:9	CBE	Error Detected in Check bits 00 No error 01 Error in lower check bits 10 Error in upper check bits 11 Error in both sets of check bits
10	CE	Correctable Error
11	UE	Uncorrectable Error
12:15		Reserved
16:19	BKnE	Bank n Error 0 No error 1 Error occurred in bank n
20:31		Reserved

SDRAM0_PMIT

Power Management Idle Timer

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x34

See "Power Management Idle Timer (SDRAM0_PMIT)" on page 15-20.

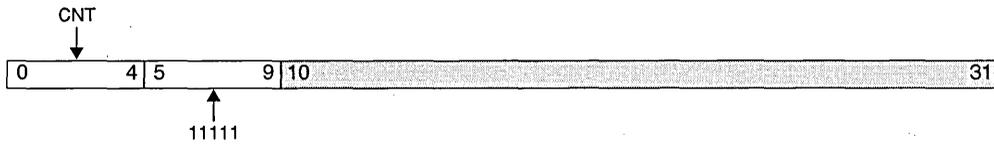


Figure 25-167. Power Management Idle Timer (SDRAM0_PMIT)

0:4	CNT	Count 0-31
5:9		Always 0b11111
10:31		Reserved

SDRAM0_RTR

Refresh Timer Register

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x30

See "Refresh Timer Register (SDRAM0_RTR)" on page 15-13.

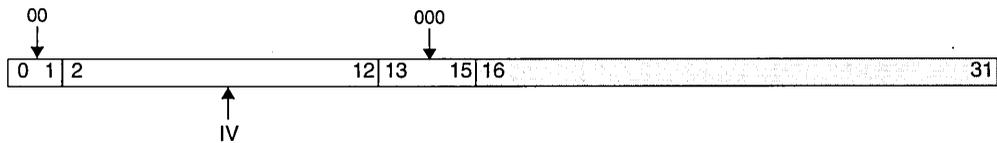


Figure 25-168. Refresh Timing Register (SDRAM0_RTR)

0:1		Always 0b00
2:12	IV	Interval Programmable between 0b000000000000 and 0b111111111111 (that is, 0x0000 to 0x3FF8 for the complete 16-bit field). Reset value is 0b00010111110 (that is, 0x05F0 for the complete 16-bit field)
13:15		Always 0b000
16:31		Reserved

SDRAM0_TR

SDRAM Timing Register

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x80

See “SDRAM Timing Register (SDRAM0_TR)” on page 15-9.

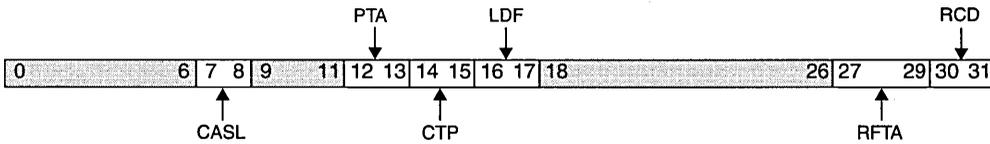


Figure 25-169. SDRAM Timing Register (SDRAM0_TR)

0:6		Reserved
7:8	CASL	SDRAM $\overline{\text{CAS}}$ latency. 00 Reserved 01 2 MemClkOut1:0 cycles 10 3 MemClkOut1:0 cycles 11 4 MemClkOut1:0 cycles
9:11		Reserved
12:13	PTA	SDRAM Precharge Command to next Activate Command minimum. 00 Reserved 01 2 MemClkOut1:0 cycles 10 3 MemClkOut1:0 cycles 11 4 MemClkOut1:0 cycles
14:15	CTP	SDRAM Read / Write Command to Precharge Command minimum. 00 Reserved 01 2 MemClkOut1:0 cycles 10 3 MemClkOut1:0 cycles 11 4 MemClkOut1:0 cycles
16:17	LDF	SDRAM Command Leadoff. 00 Reserved 01 2 MemClkOut1:0 cycles 10 3 MemClkOut1:0 cycles 11 4 MemClkOut1:0 cycles
18:26		Reserved
27:29	RFTA	SDRAM $\overline{\text{CAS}}$ before $\overline{\text{RAS}}$ Refresh Command to next Activate Command minimum. 000 4 MemClkOut1:0 cycles 001 5 MemClkOut1:0 cycles 010 6 MemClkOut1:0 cycles 011 7 MemClkOut1:0 cycles 100 8 MemClkOut1:0 cycles 101 9 MemClkOut1:0 cycles 110 10 MemClkOut1:0 cycles 111 Reserved

SDRAM0_TR

SDRAM Timing Register

30:31	RCD	SDRAM RAS to CAS Delay 00 Reserved 01 2 MemClkOut1:0 cycles 10 3 MemClkOut1:0 cycles 11 4 MemClkOut1:0 cycles
-------	-----	---

SGR

Storage Guarded Register

SPR 0x3B9

See “Real-mode Storage Attribute Control” on page 6-17.

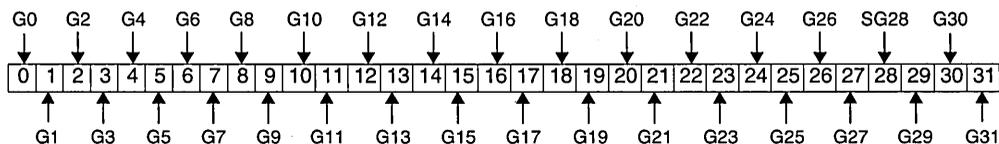


Figure 25-170. Storage Guarded Register (SGR)

0	G0	0 Normal 1 Guarded	0x0000 0000–0x07FF FFFF
1	G1	0 Normal 1 Guarded	0x0800 0000–0x0FFF FFFF
2	G2	0 Normal 1 Guarded	0x1000 0000–0x17FF FFFF
3	G3	0 Normal 1 Guarded	0x1800 0000–0x1FFF FFFF
4	G4	0 Normal 1 Guarded	0x2000 0000–0x27FF FFFF
5	G5	0 Normal 1 Guarded	0x2800 0000–0x2FFF FFFF
6	G6	0 Normal 1 Guarded	0x3000 0000–0x37FF FFFF
7	G7	0 Normal 1 Guarded	0x3800 0000–0x3FFF FFFF
8	G8	0 Normal 1 Guarded	0x4000 0000–0x47FF FFFF
9	G9	0 Normal 1 Guarded	0x4800 0000–0x4FFF FFFF
10	G10	0 Normal 1 Guarded	0x5000 0000–0x57FF FFFF
11	G11	0 Normal 1 Guarded	0x5800 0000–0x5FFF FFFF
12	G12	0 Normal 1 Guarded	0x6000 0000–0x67FF FFFF
13	G13	0 Normal 1 Guarded	0x6800 0000–0x6FFF FFFF
14	G14	0 Normal 1 Guarded	0x7000 0000–0x77FF FFFF
15	G15	0 Normal 1 Guarded	0x7800 0000–0x7FFF FFFF

SGR (cont.)

Storage Guarded Register

16	G16	0 Normal 1 Guarded	0x8000 0000–0x87FF FFFF
17	G17	0 Normal 1 Guarded	0x8800 0000–0x8FFF FFFF
18	G18	0 Normal 1 Guarded	0x9000 0000–0x97FF FFFF
19	G19	0 Normal 1 Guarded	0x9800 0000–0x9FFF FFFF
20	G20	0 Normal 1 Guarded	0xA000 0000–0xA7FF FFFF
21	G21	0 Normal 1 Guarded	0xA800 0000–0xAFFF FFFF
22	G22	0 Normal 1 Guarded	0xB000 0000–0xB7FF FFFF
23	G23	0 Normal 1 Guarded	0xB800 0000–0xBFFF FFFF
24	G24	0 Normal 1 Guarded	0xC000 0000–0xC7FF FFFF
25	G25	0 Normal 1 Guarded	0xC800 0000–0xCFFF FFFF
26	G26	0 Normal 1 Guarded	0xD000 0000–0xD7FF FFFF
27	G27	0 Normal 1 Guarded	0xD800 0000–0xDFFF FFFF
28	G28	0 Normal 1 Guarded	0xE000 0000–0xE7FF FFFF
29	G29	0 Normal 1 Guarded	0xE800 0000–0xEFFF FFFF
30	G30	0 Normal 1 Guarded	0xF000 0000–0xF7FF FFFF
31	G31	0 Normal 1 Guarded	0xF800 0000–0xFFFF FFFF

SLER

Storage Little-Endian Register

SPR 0x3BB

See “Real-mode Storage Attribute Control” on page 6-17.

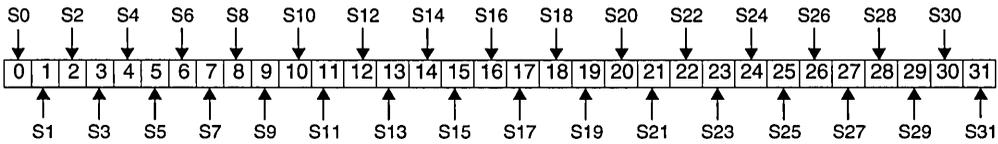


Figure 25-171. Storage Little-Endian Register (SLER)

0	S0	0 Big endian 1 Little endian	0x0000 0000–0x07FF FFFF
1	S1	0 Big endian 1 Little endian	0x0800 0000–0x0FFF FFFF
2	S2	0 Big endian 1 Little endian	0x1000 0000–0x17FF FFFF
3	S3	0 Big endian 1 Little endian	0x1800 0000–0x1FFF FFFF
4	S4	0 Big endian 1 Little endian	0x2000 0000–0x27FF FFFF
5	S5	0 Big endian 1 Little endian	0x2800 0000–0x2FFF FFFF
6	S6	0 Big endian 1 Little endian	0x3000 0000–0x37FF FFFF
7	S7	0 Big endian 1 Little endian	0x3800 0000–0x3FFF FFFF
8	S8	0 Big endian 1 Little endian	0x4000 0000–0x47FF FFFF
9	S9	0 Big endian 1 Little endian	0x4800 0000–0x4FFF FFFF
10	S10	0 Big endian 1 Little endian	0x5000 0000–0x57FF FFFF
11	S11	0 Big endian 1 Little endian	0x5800 0000–0x5FFF FFFF
12	S12	0 Big endian 1 Little endian	0x6000 0000–0x67FF FFFF
13	S13	0 Big endian 1 Little endian	0x6800 0000–0x6FFF FFFF
14	S14	0 Big endian 1 Little endian	0x7000 0000–0x77FF FFFF
15	S15	0 Big endian 1 Little endian	0x7800 0000–0x7FFF FFFF

SLER (cont.)

Storage Little-Endian Register

16	S16	0 Big endian 1 Little endian	0x8000 0000–0x87FF FFFF
17	S17	0 Big endian 1 Little endian	0x8800 0000–0x8FFF FFFF
18	S18	0 Big endian 1 Little endian	0x9000 0000–0x97FF FFFF
19	S19	0 Big endian 1 Little endian	0x9800 0000–0x9FFF FFFF
20	S20	0 Big endian 1 Little endian	0xA000 0000–0xA7FF FFFF
21	S21	0 Big endian 1 Little endian	0xA800 0000–0xAFFF FFFF
22	S22	0 Big endian 1 Little endian	0xB000 0000–0xB7FF FFFF
23	S23	0 Big endian 1 Little endian	0xB800 0000–0xBFFF FFFF
24	S24	0 Big endian 1 Little endian	0xC000 0000–0xC7FF FFFF
25	S25	0 Big endian 1 Little endian	0xC800 0000–0xCFFF FFFF
26	S26	0 Big endian 1 Little endian	0xD000 0000–0xD7FF FFFF
27	S27	0 Big endian 1 Little endian	0xD800 0000–0xDFFF FFFF
28	S28	0 Big endian 1 Little endian	0xE000 0000–0xE7FF FFFF
29	S29	0 Big endian 1 Little endian	0xE800 0000–0xEFFF FFFF
30	S30	0 Big endian 1 Little endian	0xF000 0000–0xF7FF FFFF
31	S31	0 Big endian 1 Little endian	0xF800 0000–0xFFFF FFFF

SPRG0–SPRG7

Storage Little-Endian Register

SPR 0x104–0x107 (Read-only); 0x110–0x113 (Read/Write); 0x114–0x117 (Read/Write)

See “Special Purpose Register General (SPRG0–SPRG7)” on page 3-11.

0	31
---	----

Figure 25-172. Special Purpose Registers General (SPRG0–SPRG7)

0:31	General data	Software value; no hardware usage.
------	--------------	------------------------------------

SRR0

Save/Restore Register 0

SPR 0x01A

See "Save/Restore Registers 0 and 1 (SRR0–SRR1)" on page 10-29.

0	29	30	31
---	----	----	----

Figure 25-173. Save/Restore Register 0 (SRR0)

0:29		SRR0 receives an instruction address when a non-critical interrupt is taken; the Program Counter is restored from SRR0 when rfi executes.
30:31		Reserved

SRR1

Save/Restore Register 1

SPR 0x01B

See "Save/Restore Registers 0 and 1 (SRR0–SRR1)" on page 10-29.

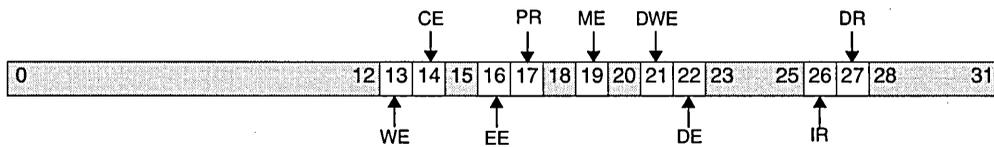


Figure 25-174. Save/Restore Register 1 (SRR1)

0:31	SRR1 receives a copy of the MSR when an interrupt is taken; the MSR is restored from SRR1 when <i>rfi</i> executes.
------	---

SRR2

Save/Restore Register 2

SPR 0x3DE

See "Save/Restore Registers 2 and 3 (SRR2–SRR3)" on page 10-30.

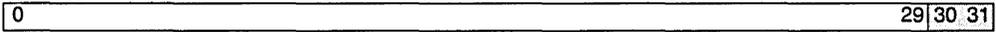


Figure 25-175. Save/Restore Register 2 (SRR2)

0:29		SRR2 receives an instruction address when a critical interrupt is taken; the Program Counter is restored from SRR2 when <i>rfci</i> executes.
30:31		Reserved

SRR3

Save/Restore Register 3

SPR 0x3DF

See "Save/Restore Registers 2 and 3 (SRR2–SRR3)" on page 10-30.

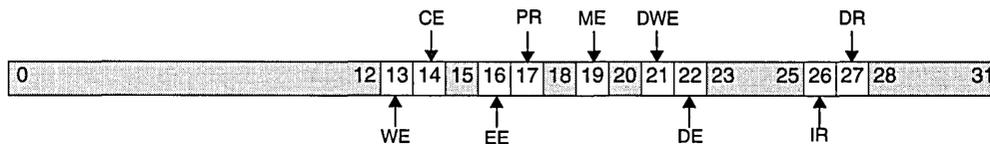


Figure 25-176. Save/Restore Register 3 (SRR3)

0:31	SRR3 receives a copy of the MSR when a critical interrupt is taken; the MSR is restored from SRR3 when <i>rfci</i> executes.
------	--

SPR 0x3BC

See "Real-mode Storage Attribute Control" on page 6-17.

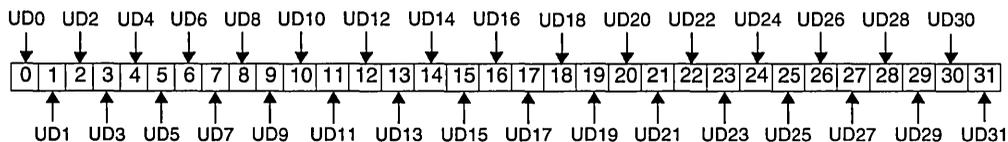


Figure 25-177. Storage User-defined 0 Register (SU0R)

0	UD0	0 Storage compression is off 1 Storage compression is on	0x0000 0000–0x07FF FFFF
1	UD1	0 Storage compression is off 1 Storage compression is on	0x0800 0000–0x0FFF FFFF
2	UD2	0 Storage compression is off 1 Storage compression is on	0x1000 0000–0x17FF FFFF
3	UD3	0 Storage compression is off 1 Storage compression is on	0x1800 0000–0x1FFF FFFF
4	UD4	0 Storage compression is off 1 Storage compression is on	0x2000 0000–0x27FF FFFF
5	UD5	0 Storage compression is off 1 Storage compression is on	0x2800 0000–0x2FFF FFFF
6	UD6	0 Storage compression is off 1 Storage compression is on	0x3000 0000–0x37FF FFFF
7	UD7	0 Storage compression is off 1 Storage compression is on	0x3800 0000–0x3FFF FFFF
8	UD8	0 Storage compression is off 1 Storage compression is on	0x4000 0000–0x47FF FFFF
9	UD9	0 Storage compression is off 1 Storage compression is on	0x4800 0000–0x4FFF FFFF
10	UD10	0 Storage compression is off 1 Storage compression is on	0x5000 0000–0x57FF FFFF
11	UD11	0 Storage compression is off 1 Storage compression is on	0x5800 0000–0x5FFF FFFF
12	UD12	0 Storage compression is off 1 Storage compression is on	0x6000 0000–0x67FF FFFF
13	UD13	0 Storage compression is off 1 Storage compression is on	0x6800 0000–0x6FFF FFFF
14	UD14	0 Storage compression is off 1 Storage compression is on	0x7000 0000–0x77FF FFFF
15	UD15	0 Storage compression is off 1 Storage compression is on	0x7800 0000–0x7FFF FFFF

SUOR (cont.)

Storage User-Defined 0 Register

16	UD16	0 Storage compression is off 1 Storage compression is on	0x8000 0000–0x87FF FFFF
17	UD17	0 Storage compression is off 1 Storage compression is on	0x8800 0000–0x8FFF FFFF
18	UD18	0 Storage compression is off 1 Storage compression is on	0x9000 0000–0x97FF FFFF
19	UD19	0 Storage compression is off 1 Storage compression is on	0x9800 0000–0x9FFF FFFF
20	UD20	0 Storage compression is off 1 Storage compression is on	0xA000 0000–0xA7FF FFFF
21	UD21	0 Storage compression is off 1 Storage compression is on	0xA800 0000–0xAFFF FFFF
22	UD22	0 Storage compression is off 1 Storage compression is on	0xB000 0000–0xB7FF FFFF
23	UD23	0 Storage compression is off 1 Storage compression is on	0xB800 0000–0xBFFF FFFF
24	UD24	0 Storage compression is off 1 Storage compression is on	0xC000 0000–0xC7FF FFFF
25	UD25	0 Storage compression is off 1 Storage compression is on	0xC800 0000–0xCFFF FFFF
26	UD26	0 Storage compression is off 1 Storage compression is on	0xD000 0000–0xD7FF FFFF
27	UD27	0 Storage compression is off 1 Storage compression is on	0xD800 0000–0xDFFF FFFF
28	UD28	0 Storage compression is off 1 Storage compression is on	0xE000 0000–0xE7FF FFFF
29	UD29	0 Storage compression is off 1 Storage compression is on	0xE800 0000–0xEFFF FFFF
30	UD30	0 Storage compression is off 1 Storage compression is on	0xF000 0000–0xF7FF FFFF
31	UD31	0 Storage compression is off 1 Storage compression is on	0xF800 0000–0xFFFF FFFF

TBL

Time Base Lower

TBR 0x10C (Read-only); SPR 0x11C (Privileged write-only)

See "Time Base" on page 11-2.



Figure 25-178. Time Base Lower (TBL)

0:31	Time Base Lower	Current count; low-order 32 bits of time base.
------	-----------------	--

TBU

Time Base Upper

TBR 0x10D (Read-only); SPR 0x11D (Privileged write-only)

See "Time Base" on page 11-2.



Figure 25-179. Time Base Upper (TBU)

0:31		Time Base Upper	Current count, high-order 32 bits of time base.
------	--	-----------------	---

SPR 0x3DA

See "Timer Control Register (TCR)" on page 11-9.

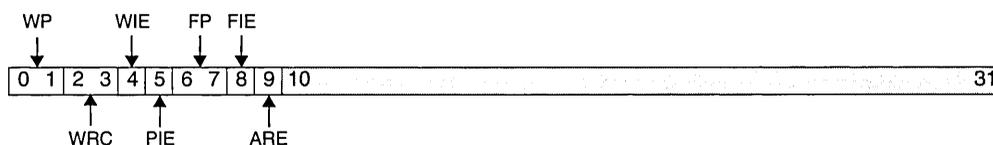


Figure 25-180. Timer Control Register (TCR)

0:1	WP	Watchdog Period 00 2^{17} clocks 01 2^{21} clocks 10 2^{25} clocks 11 2^{29} clocks	
2:3	WRC	Watchdog Reset Control 00 No Watchdog reset will occur. 01 Core reset will be forced by the Watchdog. 10 Chip reset will be forced by the Watchdog. 11 System reset will be forced by the Watchdog.	TCR[WRC] resets to 00. This field can be set by software, but cannot be cleared by software, except by a software-induced reset.
4	WIE	Watchdog Interrupt Enable 0 Disable watchdog interrupt. 1 Enable watchdog interrupt.	
5	PIE	PIT Interrupt Enable 0 Disable PIT interrupt. 1 Enable PIT interrupt.	
6:7	FP	FIT Period 00 2^9 clocks 01 2^{13} clocks 10 2^{17} clocks 11 2^{21} clocks	
8	FIE	FIT Interrupt Enable 0 Disable FIT interrupt. 1 Enable FIT interrupt.	
9	ARE	Auto Reload Enable 0 Disable auto reload. 1 Enable auto reload.	Disables on reset.
10:31		Reserved	

TSR

Timer Status Register

SPR 0x3D8 Read/Clear

See “Timer Status Register (TSR)” on page 11-8.

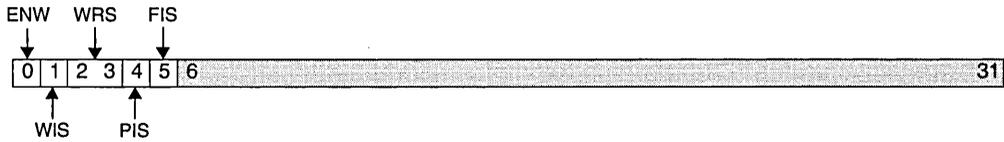


Figure 25-181. Timer Status Register (TSR)

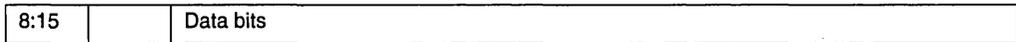
0	ENW	Enable Next Watchdog 0 Action on next watchdog event is to set TSR[ENW] = 1. 1 Action on next watchdog event is governed by TSR[WIS].	Software must reset TSR[ENW] = 0 after each watchdog timer event.
1	WIS	Watchdog Interrupt Status 0 No Watchdog interrupt is pending. 1 Watchdog interrupt is pending.	
2:3	WRS	Watchdog Reset Status 00 No Watchdog reset has occurred. 01 Core reset was forced by the watchdog. 10 Chip reset was forced by the watchdog. 11 System reset was forced by the watchdog.	
4	PIS	PIT Interrupt Status 0 No PIT interrupt is pending. 1 PIT interrupt is pending.	
5	FIS	FIT Interrupt Status 0 No FIT interrupt is pending. 1 FIT interrupt is pending.	
6:31		Reserved	

MMIO 0xEF600300, 0xEF600400

See "UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)" on page 21-14.



Figure 25-182. UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)



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

UARTx_DLM

UART Baud-Rate Divisor Latch MSB Registers

MMIO 0xEF600301, 0xEF600401

See "UART Baud-Rate Divisor Latch (MSB) Registers (UARTx_DLM)" on page 21-14.



Figure 25-183. 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 0xEF600302, 0xEF600402 Write-Only

See “FIFO Control Registers (UARTx_FCR)” on page 21-7.

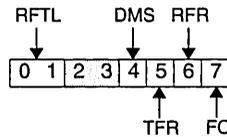


Figure 25-184. UART FIFO Control Registers (UARTx_FCR)

0:1	RFTL	Receiver FIFO Trigger Level 00 1 byte 01 4 bytes 10 8 bytes 11 14 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.

UARTx_IER

UART Interrupt Enable Registers

MMIO 0xEF600301, xEF600401

See "Interrupt Enable Registers (UARTx_IER)" on page 21-5.

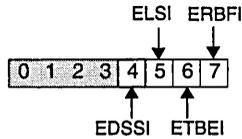


Figure 25-185. UART Interrupt Enable Registers (UARTx_IER)

0:3		Reserved	Always 0.
4	EDSSI	Enable Modem Status Interrupt	
5	ELSI	Receiver Line Status Interrupt enable 0 Enable receiver line status interrupt 1 Disable receiver line status interrupt	
6	ETBEI	Transmitter Holding Register Empty Interrupt enable 0 Enable transmitter holding register empty interrupt 1 Disable 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.

UARTx_IIR

UART Interrupt Identification Registers

MMIO 0xEF600302, 0xEF600402 Read-Only

See "Interrupt Identification Registers (UARTx_IIR)" on page 21-6.

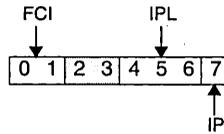


Figure 25-186. UART Interrupt Identification Registers (UARTx_IIR)

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	See Table 21-3. 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.

UARTx_LCR

UART Line Control Registers

MMIO 0xEF600303, 0xEF600403

See "Line Control Registers (UARTx_LCR)" on page 21-8.

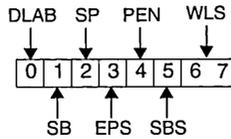


Figure 25-187. UART Line Control Registers (UARTx_LCR)

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[PE] = 1, the parity bit is transmitted and checked as 0. If UARTx_LCR[EPS] = 0 and UARTx_LCR[PE] = 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[PE] = 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[CL] = 00, characters have 1.5 stop bits. For any other value of UARTx_LCR[CL], characters have 2 stop bits. The receiver checks the first stop bit only, regardless of how many stop bits are selected.
6:7	WLS0, WLS1	Word Length Select Bits 0,1 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.

MMIO 0xEF600305, 0xEF600405

See “Line Status Registers (UARTx_LSR)” on page 21-11.

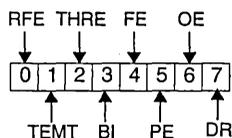


Figure 25-188. UART Line Status Registers (UARTx_LSR)

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 PPC405GP 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>	<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.</p>

UARTx_LSR (cont.)

UART Line Status Registers

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>	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	<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.

MMIO 0xEF600304, 0xEF600404

See "Modem Control Registers (UARTx_MCR)" on page 21-10.

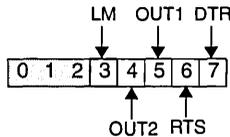


Figure 25-189. UART Modem Control Registers (UARTx_MCR)

0:2		Reserved	Always 0.
3	LM	<p>Loopback Mode</p> <p>0 Disabled</p> <p>1 Enabled</p>	<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	<p>User Output 2</p> <p>0 $\overline{OUT2}$ inactive (1)</p> <p>1 $\overline{OUT2}$ active (0)</p>	Auxiliary user designated output.
5	OUT1	<p>User Output 1</p> <p>0 $\overline{OUT1}$ inactive (1)</p> <p>1 $\overline{OUT1}$ active (0)</p>	Auxiliary user designated output.
6	RTS	<p>Request To Send</p> <p>0 \overline{RTS} inactive (1)</p> <p>1 \overline{RTS} active (0)</p>	
7	DTR	<p>Data Terminal Ready</p> <p>0 \overline{DTR} inactive (1)</p> <p>1 \overline{DTR} active (0)</p>	

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

UARTx_MSR

UART Modem Status Registers

MMIO 0xEF600306, 0xEF600406

See "Modem Status Registers (UARTx_MSR)" on page 21-13.

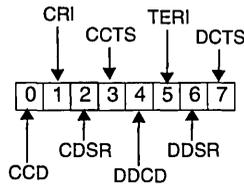


Figure 25-190. UART Modem Status Registers (UARTx_MSR)

0	DCD	Data Carrier Detect	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT2].
1	CRI	Complement of Ring Indicator	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[OUT1].
2	CDSR	Complement of Data Set Ready	In loopback mode (UARTx_MCR[LB] is 1), it is equivalent to UARTx_MCR[DTR].
3	CCTS	Complement of 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 $\overline{\text{DSR}}$ 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.

UARTx_RBR

UART Receiver Buffer Registers

MMIO 0xEF600300, 0xEF600400 Read-Only

See "Receiver Buffer Registers (UARTx_RBR)" on page 21-5.



Figure 25-191. 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.

UARTx_SCR

UART Scratchpad Registers

MMIO 0xEF600307, 0xEF600407

See "Scratchpad Registers (UARTx_SCR)" on page 21-13.



Figure 25-192. Scratchpad Registers (UARTx_SCR)



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

UARTx_THR

UART Transmitter Holding Registers

MMIO 0xEF600300, 0xEF600400 Write-Only

See "Transmitter Holding Registers (UARTx_THR)" on page 21-5.



Figure 25-193. UART Transmitter Holding Registers (UARTx_THR)



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

UIC0_CR

UIC Critical Register

DCR 0x0C3

See “UIC Critical Register (UIC0_CR)” on page 10-8.

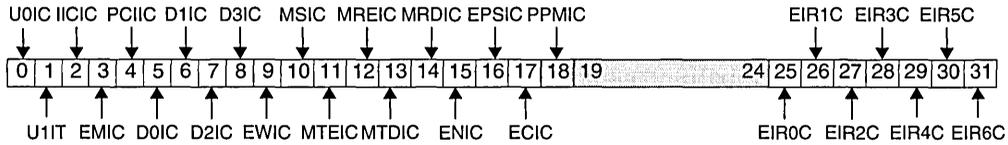


Figure 25-194. UIC Critical Register (UIC0_CR)

0	U0IC	UART0 Interrupt Class 0 UART0 interrupt is non-critical. 1 UART0 interrupt is critical.
1	U1IC	UART1 Interrupt Class 0 UART1 interrupt is non-critical. 1 UART1 interrupt is critical.
2	IIC	IIC Interrupt Class 0 IIC interrupt is non-critical. 1 IIC interrupt is critical.
3	EMIC	External Master Interrupt Class 0 External master interrupt is non-critical. 1 External master interrupt is critical.
4	PCIIC	PCI Interrupt Class 0 PCI interrupt is non-critical. 1 PCI interrupt is critical.
5	D0IC	DMA Channel 0 Interrupt Class 0 DMA channel 0 interrupt is non-critical. 1 DMA channel 0 interrupt is critical.
6	D1IC	DMA Channel 1 Interrupt Class 0 DMA channel 1 interrupt is non-critical. 1 DMA channel 1 interrupt is critical.
7	D2IC	DMA Channel 2 Interrupt Class 0 DMA channel 2 interrupt is non-critical. 1 DMA channel 2 interrupt is critical.
8	D3IC	DMA Channel 3 Interrupt Class 0 DMA channel 3 interrupt is non-critical. 1 DMA channel 3 interrupt is critical.
9	EWIC	Ethernet Wake-up Interrupt Class 0 Ethernet wake-up interrupt is non-critical. 1 Ethernet wake-up interrupt is critical.
10	MSIC	MAL SERR Interrupt Class 0 MAL SERR interrupt is non-critical. 1 MAL SERR interrupt is critical.
11	MTEIC	MAL TX EOB Interrupt Class 0 MAL TX EOB interrupt is non-critical. 1 MAL TX EOB interrupt is critical.

UIC0_CR (cont.)

UIC Critical Register

12	MREIC	MAL RX EOB Interrupt Class 0 MAL RX EOB interrupt is non-critical. 1 MAL RX EOB interrupt is critical.
13	MTDIC	MAL TX DE Interrupt Class 0 MAL TX DE interrupt is non-critical. 1 MAL TX DE interrupt is critical.
14	MRDIC	MAL RX DE Interrupt Class 0 MAL RX DE interrupt is non-critical. 1 MAL RX DE interrupt is critical.
15	EIC	Ethernet Interrupt Class 0 An Ethernet interrupt is non-critical. 1 An Ethernet interrupt is critical.
16	EPSIC	External PCI SERR Interrupt Class 0 External PCI SERR interrupt is non-critical. 1 External PCI SERR interrupt is critical.
17	ECIC	ECC Correctable Error Interrupt Class 0 ECC correctable error interrupt is non-critical. 1 ECC correctable error interrupt is critical.
18	PPMIC	PCI Power management Interrupt Class 0 PCI power management interrupt is non-critical. 1 PCI power management interrupt is critical.
19:24		Reserved
25	EIR0C	External IRQ 0 Class 0 An external IRQ 0 interrupt is non-critical. 1 An external IRQ 0 interrupt is critical.
26	EIR1C	External IRQ 1 Class 0 An external IRQ 1 interrupt is non-critical. 1 An external IRQ 1 interrupt is critical.
27	EIR2C	External IRQ 2 Class 0 An external IRQ 2 interrupt is non-critical. 1 An external IRQ 2 interrupt is critical.
28	EIR3C	External IRQ 3 Class 0 An external IRQ 3 interrupt is non-critical. 1 An external IRQ 3 interrupt is critical.
29	EIR4C	External IRQ 4 Class 0 An external IRQ 4 interrupt is non-critical. 1 An external IRQ 4 interrupt is critical.

UIC0_CR (cont.)

UIC Critical Register

30	EIR5C	External IRQ 5 Class 0 An external IRQ 5 interrupt is non-critical. 1 An external IRQ 5 interrupt is critical.
31	EIR6C	External IRQ 6 Class 0 An external IRQ 6 interrupt is non-critical. 1 An external IRQ 6 interrupt is critical.

UIC0_ER

UIC Interrupt Enable Register

DCR 0x0C2

See “UIC Enable Register (UIC0_ER)” on page 10-6.

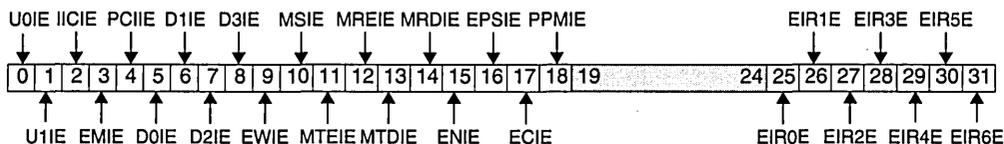


Figure 25-195. UIC Enable Register (UIC0_ER)

0	U0IE	UART0 Interrupt Enable 0 UART0 interrupt is disabled. 1 UART0 interrupt is enabled.
1	U1IE	UART1 Interrupt Enable 0 UART1 interrupt is disabled. 1 UART1 interrupt is enabled.
2	IICIE	IIC Interrupt Enable 0 IIC interrupt is disabled. 1 IIC interrupt is enabled.
3	EMIE	External Master Interrupt Enable 0 External master interrupt is disabled. 1 0xxExternal master interrupt is enabled.
4	PCIIE	PCI Interrupt Enable 0 PCI interrupt is disabled. 1 PCI interrupt is enabled.
5	D0IE	DMA Channel 0 Interrupt Enable 0 DMA channel 0 interrupt is disabled. 1 DMA channel 0 interrupt is enabled.
6	D1IE	DMA Channel 1 Interrupt Enable 0 DMA channel 1 interrupt is disabled. 1 DMA channel 1 interrupt is enabled.
7	D2IE	DMA Channel 2 Interrupt Enable 0 DMA channel 2 interrupt is disabled. 1 DMA channel 2 interrupt is enabled.
8	D3IE	DMA Channel 3 Interrupt Enable 0 DMA channel 3 interrupt is disabled. 1 DMA channel 3 interrupt is enabled.
9	EWIE	Ethernet Wake-up Interrupt Enable 0 Ethernet wake-up interrupt is disabled. 1 Ethernet wake-up interrupt is enabled.
10	MSIE	MAL SERR Interrupt Enable 0 MAL SERR interrupt is disabled. 1 MAL SERR interrupt is enabled.

UIC0_ER (cont.)

UIC Interrupt Enable Register

11	MTEIE	MAL TX EOB Interrupt Enable 0 MAL TX EOB interrupt is disabled. 1 MAL TX EOB interrupt is enabled.
12	MREIE	MAL RX EOB Interrupt Enable 0 MAL RX EOB interrupt is disabled. 1 MAL RX EOB interrupt is enabled.
13	MTDIE	MAL TX DE Interrupt Enable 0 MAL TX DE interrupt is disabled. 1 MAL TX DE interrupt is enabled.
14	MRDIE	MAL RX DE Interrupt Enable 0 MAL RX DE interrupt is disabled. 1 MAL RX DE interrupt is enabled.
15	EIE	Ethernet Interrupt Enable 0 An Ethernet interrupt is disabled. 1 An Ethernet interrupt is enabled.
16	EPSIE	External PCI SERR Interrupt Enable 0 External PCI SERR interrupt is disabled. 1 External PCI SERR interrupt is enabled.
17	ECIE	ECC Correctable Error Interrupt Enable 0 ECC correctable error interrupt is disabled. 1 ECC correctable error interrupt is enabled.
18	PPMI	PCI Power management Interrupt Enable 0 PCI power management interrupt is disabled. 1 PCI power management interrupt is enabled.
19:24		Reserved
25	EIR0E	External IRQ 0 Enable 0 An external IRQ 0 interrupt is disabled. 1 An external IRQ 0 interrupt is enabled.
26	EIR1E	External IRQ 1 Enable 0 An external IRQ 1 interrupt is disabled. 1 An external IRQ 1 interrupt is enabled.
27	EIR2E	External IRQ 2 Enable 0 An external IRQ 2 interrupt is disabled. 1 An external IRQ 2 interrupt is enabled.
28	EIR3E	External IRQ 3 Enable 0 An external IRQ 3 interrupt is disabled. 1 An external IRQ 3 interrupt is enabled.
29	EIR4E	External IRQ 4 Enable 0 An external IRQ 4 interrupt is disabled. 1 An external IRQ 4 interrupt is enabled.
30	EIR5E	External IRQ 5 Enable 0 An external IRQ 5 interrupt is disabled. 1 An external IRQ 5 interrupt is enabled.

UIC0_ER (cont.)

UIC Interrupt Enable Register

31	EIR6E	External IRQ 6 Enable 0 An external IRQ 6 interrupt is disabled. 1 An external IRQ 6 interrupt is enabled.
----	-------	--

UIC0_MSR

UIC Masked Status Register

DCR 0x0C6 Read-Only

See "UIC Masked Status Register (UIC0_MSR)" on page 10-16.

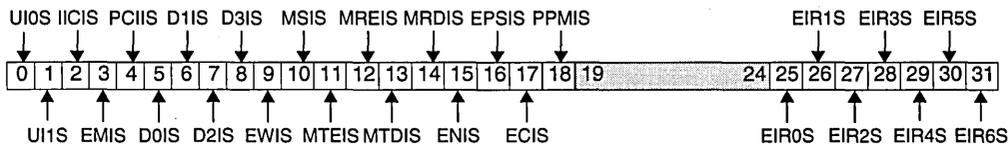


Figure 25-196. UIC Masked Status Register (UIC0_MSR)

0	U0IS	UART0 Masked Interrupt Status 0 A UART0 interrupt has not occurred. 1 A UART0 interrupt occurred.
1	U1IS	UART1 Masked Interrupt Status 0 A UART1 interrupt has not occurred. 1 A UART1 interrupt occurred.
2	IICIS	IIC Masked Interrupt Status 0 An IIC interrupt has not occurred. 1 An IIC interrupt occurred.
3	EMIS	External Master Masked Interrupt Status 0 An external master interrupt has not occurred. 1 An external master interrupt occurred.
4	PCIIS	PCI Masked Interrupt Status 0 A PCI interrupt has not occurred. 1 A PCI interrupt occurred.
5	D0IS	DMA Channel 0 Masked Interrupt Status 0 A DMA channel 0 interrupt has not occurred. 1 A DMA channel 0 interrupt occurred.
6	D1IS	DMA Channel 1 Masked Interrupt Status 0 A DMA channel 1 interrupt has not occurred. 1 A DMA channel 1 interrupt occurred.
7	D2IS	DMA Channel 2 Masked Interrupt Status 0 A DMA channel 2 interrupt has not occurred. 1 A DMA channel 2 interrupt occurred.
8	D3IS	DMA Channel 3 Masked Interrupt Status 0 A DMA channel 3 interrupt has not occurred. 1 A DMA channel 3 interrupt occurred.

UIC0_MSR (cont.)

UIC Masked Status Register

9	EWIS	Ethernet Wake-up Masked Interrupt Status 0 An Ethernet wake-up interrupt has not occurred. 1 An Ethernet wake-up interrupt occurred.
10	MSIS	MAL SERR Masked Interrupt Status 0 A MAL SERR interrupt has not occurred. 1 A MAL SERR interrupt occurred.
11	MTEIS	MAL TX EOB Masked Interrupt Status 0 A MAL TX EOB interrupt has not occurred. 1 A MAL TX EOB interrupt occurred.
12	MREIS	MAL RX EOB Masked Interrupt Status 0 A MAL RX EOB interrupt has not occurred. 1 A MAL RX EOB interrupt occurred.
13	MTDIS	MAL TX DE Masked Interrupt Status 0 A MAL TX DE interrupt has not occurred. 1 A MAL TX DE interrupt occurred.
14	MRDIS	MAL RX DE Masked Interrupt Status 0 A MAL RX DE interrupt has not occurred. 1 A MAL RX DE interrupt occurred.
15	EIS	Ethernet Masked Interrupt Status 0 An Ethernet interrupt has not occurred. 1 An Ethernet interrupt occurred.
16	EPSIE	External PCI SERR Masked Interrupt Status 0 An external PCI SERR interrupt has not occurred. 1 An external PCI SERR interrupt occurred.
17	ECIS	ECC Correctable Error Masked Interrupt Status 0 An ECC correctable error interrupt did not occur. 1 An ECC correctable error interrupt occurred.
18	PPMIS	PCI Power Management Masked Interrupt Status 0 A PCI power management interrupt did not occur. 1 A PCI power management interrupt occurred.
19:24		Reserved
25	EIR0E	External IRQ 0 Masked Status 0 An external IRQ 0 interrupt has not occurred. 1 An external IRQ 0 interrupt occurred.

UIC0_MSR (cont.)

UIC Masked Status Register

26	EIR1S	External IRQ 1 Masked Status 0 An external IRQ 1 interrupt has not occurred. 1 An external IRQ 1 interrupt occurred.
27	EIR2S	External IRQ 2 Masked Status 0 An external IRQ 2 interrupt has not occurred. 1 An external IRQ 2 interrupt occurred.
28	EIR3S	External IRQ 3 Masked Status 0 An external IRQ 3 interrupt has not occurred. 1 An external IRQ 3 interrupt occurred.
29	EIR4S	External IRQ 4 Masked Status 0 An external IRQ 4 interrupt has not occurred. 1 An external IRQ 4 interrupt occurred.
30	EIR5S	External IRQ 5 Masked Status 0 An external IRQ 5 interrupt has not occurred. 1 An external IRQ 5 interrupt occurred.
31	EIR6S	External IRQ 6 Masked Status 0 An external IRQ 6 interrupt has not occurred. 1 An external IRQ 6 interrupt occurred.

DCR 0x0C4

See “UIC Polarity Register (UIC0_PR)” on page 10-10.

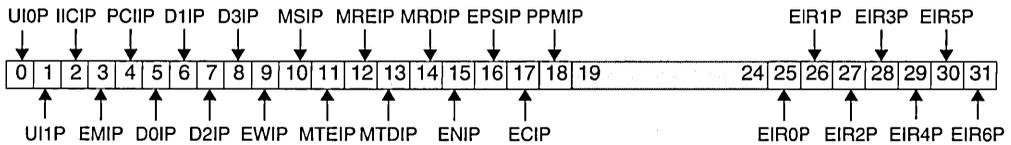


Figure 25-197. UIC Polarity Register (UIC0_PR)

0	U0IP	UART0 Interrupt Polarity 0 UART0 interrupt has negative polarity. 1 UART0 interrupt has positive polarity.	Must be set to 1.
1	U1IP	UART1 Interrupt Polarity 0 UART1 interrupt has negative polarity. 1 UART1 interrupt has positive polarity.	Must be set to 1.
2	IICIP	IIC Interrupt Polarity 0 IIC interrupt has negative polarity. 1 IIC interrupt has positive polarity.	Must be set to 1.
3	EMIP	External Master Interrupt Polarity 0 External master interrupt has negative polarity. 1 External master interrupt has positive polarity.	Must be set to 1.
4	PCIIP	PCI Interrupt Polarity 0 PCI interrupt has negative polarity. 1 PCI interrupt has positive polarity.	Must be set to 1.
5	D0IP	DMA Channel 0 Interrupt Polarity 0 DMA channel 0 interrupt has negative polarity. 1 DMA channel 0 interrupt has positive polarity.	Must be set to 1.
6	D1IP	DMA Channel 1 Interrupt Polarity 0 DMA channel 1 interrupt has negative polarity. 1 DMA channel 1 interrupt has positive polarity.	Must be set to 1.
7	D2IP	DMA Channel 2 Interrupt Polarity 0 DMA channel 2 interrupt has negative polarity. 1 DMA channel 2 interrupt has positive polarity.	Must be set to 1.
8	D3IP	DMA Channel 3 Interrupt Polarity 0 DMA channel 3 interrupt has negative polarity. 1 DMA channel 3 interrupt has positive polarity.	Must be set to 1.
9	EWIP	Ethernet Wake-up Interrupt Polarity 0 Ethernet wake-up interrupt has negative polarity. 1 Ethernet wake-up interrupt has positive polarity.	Must be set to 1.
10	MSIP	MAL SERR Interrupt Polarity 0 MAL SERR interrupt has negative polarity. 1 MAL SERR interrupt has positive polarity.	Must be set to 1.
11	MTEIP	MAL TX EOB Interrupt Polarity 0 MAL TX EOB interrupt has negative polarity. 1 MAL TX EOB interrupt has positive polarity.	Must be set to 1.

UIC0_PR (cont.)

UIC Polarity Register

12	MREIP	MAL RX EOB Interrupt Polarity 0 MAL RX EOB interrupt has negative polarity. 1 MAL RX EOB interrupt has positive polarity.	Must be set to 1.
13	MTDIP	MAL TX DE Interrupt Polarity 0 MAL TX DE interrupt has negative polarity. 1 MAL TX DE interrupt has positive polarity.	Must be set to 1.
14	MRDIP	MAL RX DE Interrupt Polarity 0 MAL RX DE interrupt has negative polarity. 1 MAL RX DE interrupt has positive polarity.	Must be set to 1.
15	EIP	Ethernet Interrupt Polarity 0 An Ethernet interrupt has negative polarity. 1 An Ethernet interrupt has positive polarity.	Must be set to 1.
16	EPSIP	External PCI SERR Interrupt Polarity 0 External PCI SERR interrupt has negative polarity. 1 External PCI SERR interrupt has positive polarity.	Must be set to 1.
17	ECIP	ECC Correctable Error Interrupt Polarity 0 ECC correctable error interrupt has negative polarity. 1 ECC correctable error interrupt has positive polarity.	Must be set to 1.
18	PPMIC	PCI Power management Interrupt Class 0 PCI power management interrupt has negative polarity. 1 PCI power management interrupt has positive polarity.	Must be set to 1.
19:24		Reserved	
25	EIR0P	External IRQ 0 Polarity 0 An external IRQ 0 interrupt has negative polarity. 1 An external IRQ 0 interrupt has positive polarity.	
26	EIR1P	External IRQ 1 Polarity 0 An external IRQ 1 interrupt has negative polarity. 1 An external IRQ 1 interrupt has positive polarity.	
27	EIR2P	External IRQ 2 Polarity 0 An external IRQ 2 interrupt has negative polarity. 1 An external IRQ 2 interrupt has positive polarity.	
28	EIR3P	External IRQ 3 Polarity 0 An external IRQ 3 interrupt has negative polarity. 1 An external IRQ 3 interrupt has positive polarity.	
29	EIR4P	External IRQ 4 Polarity 0 An external IRQ 4 interrupt has negative polarity. 1 An external IRQ 4 interrupt has positive polarity.	
30	EIR5P	External IRQ 5 Polarity 0 An external IRQ 5 interrupt has negative polarity. 1 An external IRQ 5 interrupt has positive polarity.	

UIC0_PR (cont.)

UIC Polarity Register

31	EIR6P	External IRQ 6 Polarity 0 An external IRQ 6 interrupt has negative polarity. 1 An external IRQ 6 interrupt has positive polarity.
----	-------	---

UIC0_SR

UIC Status Register

DCR 0x0C0 Read/Clear

See "UIC Status Register (UIC0_SR)" on page 10-3.

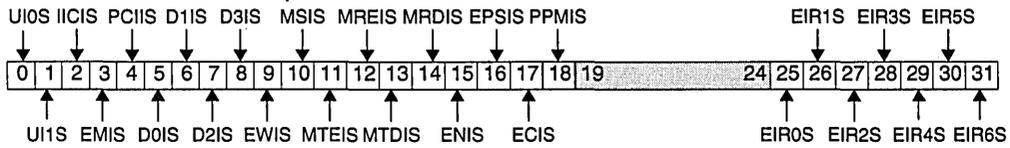


Figure 25-198. UIC Status Register (UIC0_SR)

0	U0IS	UART0 Interrupt Status 0 A UART0 interrupt has not occurred. 1 A UART0 interrupt occurred.
1	U1IS	UART1 Interrupt Status 0 A UART1 interrupt has not occurred. 1 A UART1 interrupt occurred.
2	IICIS	IIC Interrupt Status 0 An IIC interrupt has not occurred. 1 An IIC interrupt occurred.
3	EMIS	External Master Interrupt Status 0 An external master interrupt has not occurred. 1 An external master interrupt occurred.
4	PCIIS	PCI Interrupt Status 0 A PCI interrupt has not occurred. 1 A PCI interrupt occurred.
5	D0IS	DMA Channel 0 Interrupt Status 0 A DMA channel 0 interrupt has not occurred. 1 A DMA channel 0 interrupt occurred.
6	D1IS	DMA Channel 1 Interrupt Status 0 A DMA channel 1 interrupt has not occurred. 1 A DMA channel 1 interrupt occurred.
7	D2IS	DMA Channel 2 Interrupt Status 0 A DMA channel 2 interrupt has not occurred. 1 A DMA channel 2 interrupt occurred.
8	D3IS	DMA Channel 3 Interrupt Status 0 A DMA channel 3 interrupt has not occurred. 1 A DMA channel 3 interrupt occurred.
9	EWIS	Ethernet Wake-up Interrupt Status 0 An Ethernet wake-up interrupt has not occurred. 1 An Ethernet wake-up interrupt occurred.

UIC0_SR (cont.)

UIC Status Register

10	MSIS	MAL SERR Interrupt Status 0 A MAL SERR interrupt has not occurred. 1 A MAL SERR interrupt occurred.
11	MTEIS	MAL TX EOB Interrupt Status 0 A MAL TX EOB interrupt has not occurred. 1 A MAL TX EOB interrupt occurred.
12	MREIS	MAL RX EOB Interrupt Status 0 A MAL RX EOB interrupt has not occurred. 1 A MAL RX EOB interrupt occurred.
13	MTDIS	MAL TX DE Interrupt Status 0 A MAL TX DE interrupt has not occurred. 1 A MAL TX DE interrupt occurred.
14	MRDIS	MAL RX DE Interrupt Status 0 A MAL RX DE interrupt has not occurred. 1 A MAL RX DE interrupt occurred.
15	EIS	Ethernet Interrupt Status 0 An Ethernet interrupt has not occurred. 1 An Ethernet interrupt occurred.
16	EPSIS	External PCI SERR Interrupt Status 0 An external PCI SERR interrupt has not occurred. 1 An external PCI SERR interrupt occurred.
17	ECIS	ECC Correctable Error Interrupt Status 0 An ECC correctable error interrupt did not occur. 1 An ECC correctable error interrupt occurred.
18	PPMIS	PCI Power Management Interrupt Status 0 A PCI power management interrupt did not occur. 1 A PCI power management interrupt occurred.
19:24		Reserved
25	EIR0S	External IRQ 0 Status 0 An external IRQ 0 interrupt has not occurred. 1 An external IRQ 0 interrupt occurred.
26	EIR1S	External IRQ 1 Status 0 An external IRQ 1 interrupt has not occurred. 1 An external IRQ 1 interrupt occurred.
27	EIR2S	External IRQ 2 Status 0 An external IRQ 2 interrupt has not occurred. 1 An external IRQ 2 interrupt occurred.

UIC0_SR (cont.)

UIC Status Register

28	EIR3S	External IRQ 3 Status 0 An external IRQ 3 interrupt has not occurred. 1 An external IRQ 3 interrupt occurred.
29	EIR4S	External IRQ 4 Status 0 An external IRQ 4 interrupt has not occurred. 1 An external IRQ 4 interrupt occurred.
30	EIR5S	External IRQ 5 Status 0 An external IRQ 5 interrupt has not occurred. 1 An external IRQ 5 interrupt occurred.
31	EIR6S	External IRQ 6 Status 0 An external IRQ 6 interrupt has not occurred. 1 An external IRQ 6 interrupt occurred.

DCR 0x0C5

See “UIC Trigger Register (UIC0_TR)” on page 10-13.

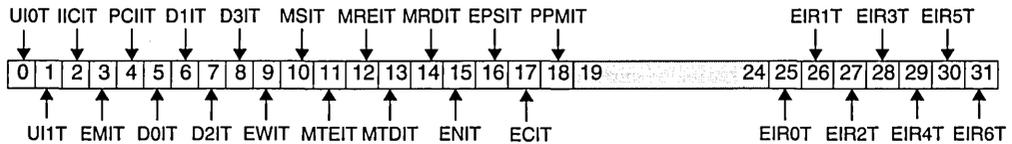


Figure 25-199. UIC Trigger Register (UIC0_TR)

0	U0IT	UART0 Interrupt Trigger 0 UART0 interrupt is level-sensitive. 1 UART0 interrupt is edge-sensitive.	Must be set to 0.
1	U1IT	UART1 Interrupt Trigger 0 UART1 interrupt is level-sensitive. 1 UART1 interrupt is edge-sensitive.	Must be set to 0.
2	IICIT	IIC Interrupt Trigger 0 IIC interrupt is level-sensitive. 1 IIC interrupt is edge-sensitive.	Must be set to 0.
3	EMIT	External Master Interrupt Trigger 0 External master interrupt is level-sensitive. 1 External master interrupt is edge-sensitive.	Must be set to 1.
4	PCIIT	PCI Interrupt Trigger 0 PCI interrupt is level-sensitive. 1 PCI interrupt is edge-sensitive.	Must be set to 0.
5	D0IT	DMA Channel 0 Interrupt Trigger 0 DMA channel 0 interrupt is level-sensitive. 1 DMA channel 0 interrupt is edge-sensitive.	Must be set to 0.
6	D1IT	DMA Channel 1 Interrupt Trigger 0 DMA channel 1 interrupt is level-sensitive. 1 DMA channel 1 interrupt is edge-sensitive.	Must be set to 0.
7	D2IT	DMA Channel 2 Interrupt Trigger 0 DMA channel 2 interrupt is level-sensitive. 1 DMA channel 2 interrupt is edge-sensitive.	Must be set to 0.
8	D3IT	DMA Channel 3 Interrupt Trigger 0 DMA channel 3 interrupt is level-sensitive. 1 DMA channel 3 interrupt is edge-sensitive.	Must be set to 0.

UIC0_TR (cont.)

UIC Triggering Register

9	EWIT	Ethernet Wake-up Interrupt Trigger 0 Ethernet wake-up interrupt is level-sensitive. 1 Ethernet wake-up interrupt is edge-sensitive.	Must be set to 0.
10	MSIT	MAL SERR Interrupt Trigger 0 MAL SERR interrupt is level-sensitive. 1 MAL SERR interrupt is edge-sensitive.	Must be set to 0.
11	MTEIT	MAL TX EOB Interrupt Trigger 0 MAL TX EOB interrupt is level-sensitive. 1 MAL TX EOB interrupt is edge-sensitive.	Must be set to 0.
12	MREIT	MAL RX EOB Interrupt Trigger 0 MAL RX EOB interrupt is level-sensitive. 1 MAL RX EOB interrupt is edge-sensitive.	Must be set to 0.
13	MTDIT	MAL TX DE Interrupt Trigger 0 MAL TX DE interrupt is level-sensitive. 1 MAL TX DE interrupt is edge-sensitive.	Must be set to 0.
14	MRDIT	MAL RX DE Interrupt Trigger 0 MAL RX DE interrupt is level-sensitive. 1 MAL RX DE interrupt is edge-sensitive.	Must be set to 0.
15	EIT	Ethernet Interrupt Trigger 0 An Ethernet interrupt is level-sensitive. 1 An Ethernet interrupt is edge-sensitive.	Must be set to 0.
16	EPSIT	External PCI SERR Interrupt Trigger 0 External PCI SERR interrupt is level-sensitive. 1 External PCI SERR interrupt is edge-sensitive.	Must be set to 0.
17	ECIT	ECC Correctable Error Interrupt Trigger 0 ECC correctable error interrupt is level-sensitive. 1 ECC correctable error interrupt is edge-sensitive.	Must be set to 0.
18	PPMIT	PCI Power management Interrupt Trigger 0 PCI power management interrupt is level-sensitive. 1 PCI power management interrupt is edge-sensitive.	Must be set to 0.
19:24		Reserved	
25	EIROT	External IRQ 0 Trigger 0 An external IRQ 0 interrupt is level-sensitive. 1 An external IRQ 0 interrupt is edge-sensitive.	

UIC0_TR (cont.)

UIC Triggering Register

26	EIR1T	External IRQ 1 Trigger 0 An external IRQ 1 interrupt is level-sensitive. 1 An external IRQ 1 interrupt is edge-sensitive.
27	EIR2T	External IRQ 2 Trigger 0 An external IRQ 2 interrupt is level-sensitive. 1 An external IRQ 2 interrupt is edge-sensitive.
28	EIR3T	External IRQ 3 Trigger 0 An external IRQ 3 interrupt is level-sensitive. 1 An external IRQ 3 interrupt is edge-sensitive.
29	EIR4T	External IRQ 4 Trigger 0 An external IRQ 4 interrupt is level-sensitive. 1 An external IRQ 4 interrupt is edge-sensitive.
30	EIR5T	External IRQ 5 Trigger 0 An external IRQ 5 interrupt is level-sensitive. 1 An external IRQ 5 interrupt is edge-sensitive.
31	EIR6T	External IRQ 6 Trigger 0 An external IRQ 6 interrupt is level-sensitive. 1 An external IRQ 6 interrupt is edge-sensitive.

UIC0_VCR

UIC Vector Configuration Register

DCR 0x0C8 Write-Only

See "UIC Vector Configuration Register (UIC0_VCR)" on page 10-18.

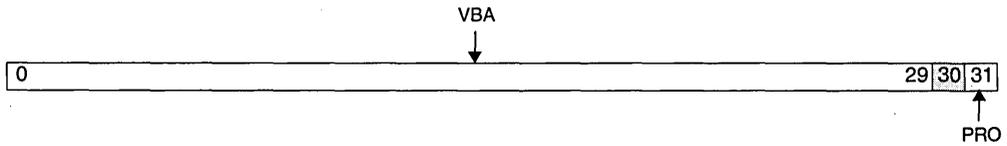


Figure 25-200. UIC Vector Configuration Register (UIC0_VCR)

0:29	VBA	Vector Base Address
30		Reserved
31	PRO	Priority Ordering 0 UIC0_SR[0] is the highest priority interrupt. 1 UIC0_SR[31] is the highest priority interrupt. Note: Vector generation is not performed for non-critical interrupts.

DCR 0x0C7 Read-Only

See "UIC Vector Register (UIC0_VR)" on page 10-19.

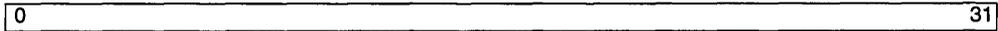
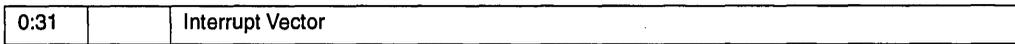


Figure 25-201. UIC Vector Register (UIC0_VR)



USPRG0

User Special Purpose Register General 0

SPR 0x100 (User R/W)

See "Special Purpose Register General (SPRG0–SPRG7)" on page 3-11.

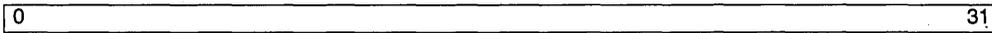


Figure 25-202. User SPR General 0 (USPRG0)

0:31	General data	Software value; no hardware usage.
------	--------------	------------------------------------

SPR 0x001

See "Fixed Point Exception Register (XER)" on page 3-8.

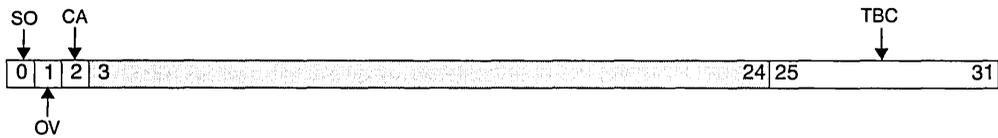


Figure 25-203. Fixed Point Exception Register (XER)

0	SO	Summary Overflow 0 No overflow has occurred. 1 Overflow has occurred.	Can be <i>set</i> by mtspr or by using "o" form instructions; 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 using "o" form instructions; can be <i>reset</i> by mtspr , by mcrxr , or "o" form instructions.
2	CA	Carry 0 Carry has not occurred. 1 Carry has occurred.	Can be <i>set</i> by mtspr or arithmetic instructions that update the CA field; can be <i>reset</i> by mtspr , by mcrxr , or by arithmetic instructions that update the CA field.
3:24		Reserved	
25:31	TBC	Transfer Byte Count	Used by lswx and stswx ; written by mtspr .

ZPR

Zone Protection Register

SPR 0x3B0

See "Zone Protection" on page 6-13.

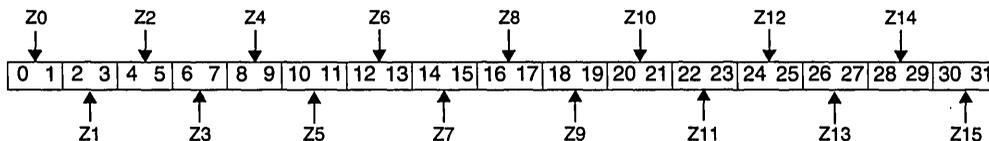


Figure 25-204. Zone Protection Register (ZPR)

0:1	Z0	TLB page access control for all pages in this zone.	
		In the problem state (MSR[PR] = 1): 00 No access 01 Access controlled by applicable TLB_entry[EX, WR] 10 Access controlled by applicable TLB_entry[EX, WR] 11 Accessed as if execute and write permissions (TLB_entry[EX, WR]) are granted.	In the supervisor state (MSR[PR] = 0): 00 Access controlled by applicable TLB_entry[EX, WR] 01 Access controlled by applicable TLB_entry[EX, WR] 10 Access controlled by applicable TLB_entry[EX, WR] 11 Accessed as if execute and write permissions (TLB_entry[EX, WR]) are granted.
2:3	Z1	See the description of Z0.	
4:5	Z2	See the description of Z0.	
6:7	Z3	See the description of Z0.	
8:9	Z4	See the description of Z0.	
10:11	Z5	See the description of Z0.	
12:13	Z6	See the description of Z0.	
14:15	Z7	See the description of Z0.	
16:17	Z8	See the description of Z0.	
18:19	Z9	See the description of Z0.	
20:21	Z10	See the description of Z0.	
22:23	Z11	See the description of Z0.	
24:25	Z12	See the description of Z0.	
26:27	Z13	See the description of Z0.	
28:29	Z14	See the description of Z0.	
30:31	Z15	See the description of Z0.	

Chapter 26. Signal Summary

This chapter provides detailed information on the PPC405GP I/O signals.

26.1 Signals Listed Alphabetically

Table 26-1 lists the PPC405GP signals in alphabetical order. For each signal there is an indication of the interface group to which it belongs and a page reference to the description of the signal in Table 26-2, "Signal Descriptions," on page 26-5.

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, $\overline{\text{ExtAck}}$).

Table 26-1. Alphabetical Signal List

Signal Name	Interface	Page
BA1:0	SDRAM	26-7
$\overline{\text{BankSel}}0:3$	SDRAM	26-7
BusReq	External Master Peripheral	26-8
$\overline{\text{CAS}}$	SDRAM	26-7
ClkEn0:1	SDRAM	26-7
DMAAck0:3	External Slave Peripheral	26-8
DMAReq0:3	External Slave Peripheral	26-8
DQM0:3	SDRAM	26-7
DQM $\overline{\text{CB}}$	SDRAM	26-7
DrvrInh1:2	System	26-10
ECC0:7	SDRAM	26-7
EMCMDClk	Ethernet	26-6
EOT0:3[TC0:3]	External Slave Peripheral	26-8
EMCMDIO[PHYMDIO]	Ethernet	26-7
EMCTxD0:3	Ethernet	26-6
EMCTxEn	Ethernet	26-6
EMCTxErr	Ethernet	26-6
$\overline{\text{ExtAck}}$	External Master Peripheral	26-8
$\overline{\text{ExtReq}}$	External Master Peripheral	26-8
$\overline{\text{ExtReset}}$	External Master Peripheral	26-8

Table 26-1. Alphabetical Signal List

Signal Name	Interface	Page
GPIO1:2[TS1E:TS2E]	System	26-2
GPIO3:4[TS1O:TS2O]	System	26-2
GPIO5:8[TS3:6]	System	26-2
GPIO9[TrcClk]	System	26-2
Halt	System	26-9
HoldAck	External Master Peripheral	26-8
HoldPri	External Master Peripheral	26-8
HoldReq	External Master Peripheral	26-8
IIC_SCL	Internal Peripheral	26-9
IIC_SDA	Internal Peripheral	26-9
IRQ0:6[GPIO17:23]	Interrupts	26-2
MemAddr0:12	SDRAM	26-7
MemClkOut0:1	SDRAM	26-7
MemData0:31	SDRAM	26-7
PCIA_D0:31	PCI	26-5
PCIO_0:3[BE0:3]	PCI	26-5
PCIClk	PCI	26-5
PCIDevSel	PCI	26-5
PCIIDSel	PCI	26-5
PCIFrame	PCI	26-5
PCIGnt0[Req]	PCI	26-6
PCIGnt1:5	PCI	26-6
PCIINT[PerWE]	PCI	26-6
PCIIRDY	PCI	26-5
PCIParity	PCI	26-5
PCIPerr	PCI	26-5
PCIREQ0[Gnt]	PCI	26-6
PCIREQ1:5	PCI	26-6
PCIReset	PCI	26-5
PCISerr	PCI	26-5
PCIStop	PCI	26-5

Table 26-1. Alphabetical Signal List

Signal Name	Interface	Page
PCITRDY	PCI	26-5
PerAddr0:31	External Slave Peripheral	26-7
PerBLast	External Slave Peripheral	26-8
PerClk	External Master Peripheral	26-8
PerCS0	External Slave Peripheral	26-7
PerCS1:7[GPIO10:16]	External Slave Peripheral	26-3
PerData0:31	External Slave Peripheral	26-7
PerErr	External Master Peripheral	26-8
PerOE	External Slave Peripheral	26-7
PerPar0:3	External Slave Peripheral	26-7
PerReady	External Slave Peripheral	26-8
PerR/W	External Slave Peripheral	26-8
PerWBE0:3	External Slave Peripheral	26-7
PHYCol	Ethernet	26-6
PHYRxClk	Ethernet	26-6
PHYCrS	Ethernet	26-6
PHYRxD0:3	Ethernet	26-6
PHYRxDV	Ethernet	26-6
PHYRxErr	Ethernet	26-6
PHYTxClk	Ethernet	26-6
RAS	SDRAM	26-7
RcvrInh	System	26-10
SysClk	System	26-9
SysErr	System	26-9
SysReset	System	26-9
TCK	JTAG	26-9
TDI	JTAG	26-9
TDO	JTAG	26-9
TestEn	System	26-10
TmrClk	System	26-10
TMS	JTAG	26-9

Table 26-1. Alphabetical Signal List

Signal Name	Interface	Page
$\overline{\text{TRST}}$	JTAG	26-9
UART0_CTS	Internal Peripheral	26-9
$\overline{\text{UART0_DCD}}$	Internal Peripheral	26-9
UART0_DSR	Internal Peripheral	26-9
UART0_DTR	Internal Peripheral	26-9
$\overline{\text{UART0_RI}}$	Internal Peripheral	26-9
UART0_RTS	Internal Peripheral	26-9
UART0_Rx	Internal Peripheral	26-8
UART0_Tx	Internal Peripheral	26-8
UART1_DSR[UART1_CTS]	Internal Peripheral	26-9
UART1_RTS[UART1_DTR]	Internal Peripheral	26-9
UART1_Rx	Internal Peripheral	26-9
UART1_Tx	Internal Peripheral	26-9
UARTSerClk	Internal Peripheral	26-8
$\overline{\text{WE}}$	SDRAM	26-7

26.2 Signal Descriptions

Each I/O signal is listed with the other signals in the same interface group.

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, $\overline{\text{ExtAck}}$).

Table 26-2. Signal Descriptions

Signal Name	I/O	Function
PCI Interface		
PCIAD0:31	I/O	PCI Address/Data Bus. Multiplexed address and data bus
PCIC0:3[$\overline{\text{BE0:3}}$]	I/O	PCI C (bus command) or Byte enable
PCIParity	I/O	PCI parity. Parity is even across PCIAD0:31 and PCIC0:3[$\overline{\text{BE0:3}}$]. PCIParity is valid one cycle after either an address or data phase. The PCI device that drives PCIAD0:31 is responsible for driving PCIParity on the next PCI bus clock.
$\overline{\text{PCIFrame}}$	I/O	$\overline{\text{PCIFrame}}$ is driven by the current PCI bus master to indicate beginning and duration of a PCI access.
$\overline{\text{PCIIRDY}}$	I/O	$\overline{\text{PCIIRDY}}$ is driven by the current PCI bus master. Assertion of $\overline{\text{PCIIRDY}}$ indicates that the PCI initiator is ready to transfer data.
$\overline{\text{PCITRDY}}$	I/O	The target of the current PCI transaction drives $\overline{\text{PCITRDY}}$. Assertion of $\overline{\text{PCITRDY}}$ indicates that the PCI target is ready to transfer data.
$\overline{\text{PCIStop}}$	I/O	The target of the current PCI transaction may assert $\overline{\text{PCIStop}}$ to indicate to the requesting PCI master that it wants to end the current transaction.
$\overline{\text{PCIDevSel}}$	I/O	$\overline{\text{PCIDevSel}}$ is driven by the target of the current PCI transaction. A PCI target asserts $\overline{\text{PCIDevSel}}$ when it has decoded an address and command encoding and claims the transaction.
PCIIDSel	I	PCIIDSel is used during configuration cycles to select the PCI slave interface for configuration
$\overline{\text{PCISErr}}$	I/O	$\overline{\text{PCISErr}}$ is used for reporting address parity errors or catastrophic failures detected by a PCI target.
$\overline{\text{PCIPErr}}$	I/O	$\overline{\text{PCIPErr}}$ is used for reporting data parity errors on PCI transactions. $\overline{\text{PCIPErr}}$ is driven active by the device receiving PCIAD0:31, PCIC0:3[$\overline{\text{BE0:3}}$], and PCIParity, two PCI clocks following the data in which bad parity is detected.
PCIClk	I	PCIClk is used as the asynchronous PCI clock when in Async mode. It is unused when the PCI interface is operated synchronously with the PLB bus.
$\overline{\text{PCIReset}}$	O	PCI specific reset

Table 26-2. Signal Descriptions (continued)

Signal Name	I/O	Function
PCIINT[PerWE]	O	PCI interrupt or Peripheral write enable. Logical AND of the four PerWBE0:3 write byte enables
PCIReq0[Gnt]	I	PCIReq0 when internal arbiter is used or Gnt when external arbiter is used.
PCIReq1:5	I	PCIReq1:5 input when internal arbiter is used
PCIGnt0[Req]	O	PCIGnt0 when internal arbiter is used or Req when external arbiter is used.
PCIGnt1:5	O	PCIGnt1:5 output when internal arbiter is used.
Ethernet Interface		
PHYRxD3:0	I	Received data. A nibble-wide bus from the physical layer device (PHY). The data is synchronous with the PHYRxClk.
EMCTxD3:0	O	Transmit data. A nibble-wide bus towards the network. The data is synchronous to the PHYTxClk.
PHYRxErr	I	Receive Error. This signal comes from the PHY and is synchronous with the PHYRxClk.
PHYRxClk	I	Receiver Medium clock. This signal is generated by the PHY.
PHYRxDV	I	Receive Data Valid. Data on the Data Bus is valid when this signal is activated. Deassertion of this signal indicates end of the frame reception.
PHYCrS	I	Carrier Sense signal from the PHY. This is an asynchronous signal.
EMCTxErr	O	Transmit Error. This signal is driven by EMAC and is connected to the PHY. This signal informs the PHY that an error was detected. This signal is synchronous to the PHYTxClk.
EMCTxEn	O	Transmit Data Enabled. EMCTxEnI is driven by EMAC to the PHY. Data is valid during the active state. Deassertion of EMCTxEn indicates end of frame transmission. This signal is synchronous to the PHYTxClk.
PHYTxClk	I	This clock comes from the PHY and is the Medium Transmit clock.
PHYCoI	I	Collision signal from the PHY. This is an asynchronous signal.
EMCMDClk	O	Management Data Clock. The EMCMDClk is sourced to the PHY. This clock has a period of 400 ns. EMCMDClk is derived from the OPB frequency. EMAC0_STACR[OPBC] must be set appropriately. Management information is transferred synchronously with respect to this clock.

Table 26-2. Signal Descriptions (continued)

Signal Name	I/O	Function
EMCMDIO[PHYMDIO]	I/O	Management Data Input/Output is a bidirectional signal between EMAC and the PHY. It is used to transfer control and status information.
SDRAM Interface		
MemData0:31	I/O	Memory data bus
MemAddr0:12	O	Memory address bus
BA1:0	O	Bank Address supporting up to 4 internal banks
RAS	O	Row Address Strobe
CAS	O	Column Address Strobe
DQM0:3	O	DQM for byte lanes 0, 1, 2, and 3.
DQMCB	O	DQM for ECC check bits
ECC0:7	I/O	ECC check bits 0:7
BankSel0:3	O	Select up to four external SDRAM banks
WE	O	Write Enable
ClkEn0:1	O	ClkEn0:1 provide duplicate clock enables for MemClkOut0:1.
MemClkOut0:1	O	Duplicate SDRAM clock outputs. In limited cases, this allows glueless SDRAM attach without requiring the signal to be repowered by a PLL or zero-delay buffer.
External Slave Peripheral Interface		
PerData0:31	I/O	Peripheral data bus used by PPC405GP when not in external master mode, otherwise used by external master
PerAddr0:31	I/O	Peripheral address bus used by PPC405GP when not in external master mode, otherwise used by external master.
PerPar0:3	I/O	Peripheral byte parity signals
PerWBE0:3	I/O	As outputs, these signals can act as byte-enables which are valid for an entire cycle or as write-byte-enables which are valid for each byte on each data transfer, allowing partial word transactions. As outputs, the signals are used by the peripheral controller or DMA controller, depending upon the type of transfer involved. These signals are used as inputs when an external bus master owns the external interface
PerCS0	O	Peripheral chip select 0
PerCS1:7[GPIO10:16]	O [I/O]	Seven additional peripheral chip selects or General Purpose I/O - To access this function, software must toggle a DCR bit.
PerOE	O	Used by either peripheral controller or DMA controller depending upon the type of transfer involved.

Table 26-2. Signal Descriptions (continued)

Signal Name	I/O	Function
PerR \overline{W}	I/O	Used by PPC405GP when not in external master mode, otherwise used by external master. As output, the signal is used by either peripheral controller or DMA controller depending upon the type of transfer involved.
PerReady	I	Used by a peripheral slave to indicate it is ready to transfer data
PerBLast	I/O	Used by PPC405GP when not in external master mode, otherwise used by external master.
DMAReq0:3	I	Used by slave peripherals to indicate that they are prepared to transfer data.
DMAAck0:3	O	Used by PPC405GP to indicate that data transfers have occurred.
EOT0:3[TC0:3]	I/O	End Of Transfer or Terminal Count
External Master Peripheral Interface		
PerClk	O	Peripheral clock to be used by an external master and by synchronous peripheral slaves
ExtReset	O	Peripheral reset to be used by an external master and by synchronous peripheral slaves
HoldReq	I	Hold Request, used by an external master to request ownership of the peripheral bus
HoldAck	O	Hold Acknowledge, used by PPC405GP to transfer ownership of peripheral bus to an external master
ExtReq	I	Used by an external master to indicate it is prepared to transfer data
ExtAck	O	Used by PPC405GP to indicate that a data transfer occurred.
HoldPri	I	Used by an external master to indicate the priority of a given transfer (0 = high, 1 = low)
BusReq	O	Used when PPC405GP needs to regain control of peripheral interface from an external Master
PerErr	I	Used to record external Master errors and external slave peripheral errors
Internal Peripheral Interface		
UARTSerClk	I	Serial Clock provides an alternative clock to the internally generated serial clock. Used when internally generated baud rates are not satisfactory. This input can be connected to UART0, using CPC0_CR0[U0EC] = 1, or UART1, using CPC0_CR0[U1EC] = 1, or both (CPC0_CR0[U0EC, U1EC] = 1.
UART0_Rx	I	UART0 Serial in data
UART0_Tx	O	UART0 Serial out data

Table 26-2. Signal Descriptions (continued)

Signal Name	I/O	Function
UART0_DCD	I	UART0 Data Carrier Detect
UART0_DSR	I	UART0 Data Set Ready
UART0_CTS	I	UART0 Clear To Send
UART0_DTR	O	UART0 Data Terminal Ready
UART0_RTS	O	UART0 Request To Send
UART0_RI	I	UART0 Ring Indicator
UART1_Rx	I	UART1 Serial In data.
UART1_Tx	O	UART1 Serial Out data
UART1_DSR[UART1_CTS]	I	UART1 Data Set Ready or UART1 Clear To Send. To access this function, software must toggle a DCR bit.
UART1_RTS[UART1_DTR]	O	UART1 Request To Send or UART1 Data Terminal Ready. To access this function, software must toggle a DCR bit.
IIC_SCL	I/O	IIC Serial Clock
IIC_SDA	I/O	IIC Serial Data
Interrupts Interface		
IRQ0:6[GPIO17:23]	I [I/O]	Interrupt requests 0–6 or General Purpose I/O. To access this function, software must toggle a DCR bit.
JTAG Interface		
TDI	I	Test data in
TMS	I	JTAG test mode select
TDO	O	Test data out
TCK	I	JTAG test clock
TRST	I	JTAG reset
System Interface		
SysClk	I	Main system clock input
SysReset	I/O	Main system reset. This signal may be redriven by the PPC405GP to allow a system reset to occur.
SysErr	O	Asserted when a machine check exception is generated.
Halt	I	Halt from external debugger.

Table 26-2. Signal Descriptions (continued)

Signal Name	I/O	Function
GPIO1:2[TS1E:TS2E]	I/O [O]	General Purpose I/O or Even Trace execution status. To access this function, software must toggle a DCR bit.
GPIO3:4[TS1O:TS2O]	I/O [O]	General Purpose I/O or Odd Trace execution status. To access this function, software must toggle a DCR bit.
GPIO5:8[TS3:6]	I/O [O]	General Purpose I/O or Trace status. To access this function, software must toggle a DCR bit.
GPIO9[TrcClk]	I/O [O]	General Purpose I/O or Trace interface clock.
TestEn	I	Test Enable. Reserved for manufacturing test.
RcvrInh	I	Receiver Inhibit. Reserved for manufacturing test.
DrvrInh1 DrvrInh2	I	Driver Inhibit 1 and 2. Reserved for manufacturing test.
TmrClk	I	Timer clock. TmrClk is an alternative clock source for the timer facilities. Used when the allowable timer clock intervals provided by the CPU clock are not satisfactory. Enabled when CPC0_CR1[CETE] = 1.

Appendix A. Instruction Summary

This appendix contains PPC405GP instructions summarized alphabetically and by opcode.

“Instruction Set and Extended Mnemonics – Alphabetical” lists all PPC405GP mnemonics, including extended mnemonics, alphabetically. A short functional description is included for each mnemonic.

“Instructions Sorted by Opcode,” on page A-33, lists all PPC405GP instructions, sorted by primary and secondary opcodes. Extended mnemonics are not included in the opcode list.

“Instruction Formats,” on page A-41, illustrates the PPC405GP instruction forms (allowed arrangements of fields within instructions).

A.1 Instruction Set and Extended Mnemonics – Alphabetical

Table A-1 summarizes the PPC405GP 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 24, “Instruction Set,” which is also alphabetized under the root form. Chapter 24 also describes the instruction operands and notation.

Note the following for the branch conditional mnemonic:

Bit 4 of the BO field provides a hint about the most likely outcome of a conditional branch. (See “Branch Prediction” on page 3-36 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 the table below, $BO_4 = 0$ has always been assumed. The assembler must allow the programmer to specify branch prediction. To do this, the assembler supports a suffixes for the conditional branch mnemonics:

- + Predict branch to be taken.
- Predict branch not to be taken.

As specific examples, **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 3-36 for more information.

Table A-1. PPC405GP Instruction Syntax Summary

Mnemonic	Operands	Function	Other Registers Changed	Page
add	RT, RA, RB	Add (RA) to (RB). Place result in RT.		24-6
add.			CR[CR0]	
addo			XER[SO, OV]	
addo.			CR[CR0] XER[SO, OV]	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
addc	RT, RA, RB	Add (RA) to (RB). Place result in RT. Place carry-out in XER[CA].		24-6
addc.			CR[CR0]	
addcco			XER[SO, OV]	
addcco.			CR[CR0] XER[SO, OV]	
adde	RT, RA, RB	Add XER[CA], (RA), (RB). Place result in RT. Place carry-out in XER[CA].		24-8
adde.			CR[CR0]	
addeo			XER[SO, OV]	
addeo.			CR[CR0] XER[SO, OV]	
addi	RT, RA, IM	Add EXTS(IM) to (RAI0). Place result in RT.		24-9
addic	RT, RA, IM	Add EXTS(IM) to (RAI0). Place result in RT. Place carry-out in XER[CA].		24-6
addic.	RT, RA, IM	Add EXTS(IM) to (RAI0). Place result in RT. Place carry-out in XER[CA].	CR[CR0]	24-11
addis	RT, RA, IM	Add (IM ¹⁶ 0) to (RAI0). Place result in RT.		24-12
addme	RT, RA	Add XER[CA], (RA), (-1). Place result in RT. Place carry-out in XER[CA].		24-13
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].		24-14
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.		24-15
and.			CR[CR0]	
andc	RA, RS, RB	AND (RS) with \neg (RB). Place result in RA.		24-16
andc.			CR[CR0]	
andi.	RA, RS, IM	AND (RS) with (¹⁶ 0 IM). Place result in RA.	CR[CR0]	24-17
andis.	RA, RS, IM	AND (RS) with (IM ¹⁶ 0). Place result in RA.	CR[CR0]	24-18

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
b	target	Branch unconditional relative. LI ← (target – CIA) _{6:29} NIA ← CIA + EXTS(LI ² 0)		24-19
ba		Branch unconditional absolute. LI ← target _{6:29} NIA ← EXTS(LI ² 0)		
bl		Branch unconditional relative. LI ← (target – CIA) _{6:29} NIA ← CIA + EXTS(LI ² 0)	(LR) ← CIA + 4.	
bla		Branch unconditional absolute. LI ← target _{6:29} NIA ← EXTS(LI ² 0)	(LR) ← CIA + 4.	
bc	BO, BI, target	Branch conditional relative. BD ← (target – CIA) _{16:29} NIA ← CIA + EXTS(BD ² 0)	CTR if BO ₂ = 0.	24-20
bca		Branch conditional absolute. BD ← target _{16:29} NIA ← EXTS(BD ² 0)	CTR if BO ₂ = 0.	
bcl		Branch conditional relative. BD ← (target – CIA) _{16:29} NIA ← CIA + EXTS(BD ² 0)	CTR if BO ₂ = 0. (LR) ← CIA + 4.	
bcla		Branch conditional absolute. BD ← target _{16:29} NIA ← EXTS(BD ² 0)	CTR if BO ₂ = 0. (LR) ← CIA + 4.	
bcctr	BO, BI	Branch conditional to address in CTR. Using (CTR) at exit from instruction, NIA ← CTR _{0:29} ² 0.	CTR if BO ₂ = 0.	24-26
bcctrl			CTR if BO ₂ = 0. (LR) ← CIA + 4.	
bclr	BO, BI	Branch conditional to address in LR. Using (LR) at entry to instruction, NIA ← LR _{0:29} ² 0.	CTR if BO ₂ = 0.	24-30
bctrl			CTR if BO ₂ = 0. (LR) ← CIA + 4.	
bctr		Branch unconditionally to address in CTR. <i>Extended mnemonic for</i> bcctr 20,0		24-26
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		24-20
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.	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bdnzlr		Decrement CTR. Branch if CTR \neq 0 to address in LR. <i>Extended mnemonic for</i> bclr 16,0		24-30
bdnzlrl		<i>Extended mnemonic for</i> bclrl 16,0	(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		24-20
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.	
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		24-30
bdnzflrl		<i>Extended mnemonic for</i> bclrl 0,cr_bit	(LR) \leftarrow CIA + 4.	
bdnzfz	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		24-20
bdnzfa		<i>Extended mnemonic for</i> bca 8,cr_bit,target		
bdnzfl		<i>Extended mnemonic for</i> bcl 8,cr_bit,target	(LR) \leftarrow CIA + 4.	
bdnzfla		<i>Extended mnemonic for</i> bcla 8,cr_bit,target	(LR) \leftarrow CIA + 4.	
bdnzflr	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		24-30
bdnzflrl		<i>Extended mnemonic for</i> bclrl 8,cr_bit	(LR) \leftarrow CIA + 4.	
bdz	target	Decrement CTR. Branch if CTR = 0. <i>Extended mnemonic for</i> bc 18,0,target		24-20
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.	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bdzlr		Decrement CTR. Branch if CTR = 0 to address in LR. <i>Extended mnemonic for</i> bclr 18,0		24-30
bdzlr		<i>Extended mnemonic for</i> bclr 18,0	(LR) ← CIA + 4.	
bdzfb	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR _{cr_bit} = 0. <i>Extended mnemonic for</i> bc 2,cr_bit,target		24-20
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		24-30
bdzflr		<i>Extended mnemonic for</i> bclr 2,cr_bit	(LR) ← CIA + 4.	
bdztd	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR _{cr_bit} = 1. <i>Extended mnemonic for</i> bc 10,cr_bit,target		24-20
bdztda		<i>Extended mnemonic for</i> bca 10,cr_bit,target		
bdztdl		<i>Extended mnemonic for</i> bcl 10,cr_bit,target	(LR) ← CIA + 4.	
bdztdla		<i>Extended mnemonic for</i> bcla 10,cr_bit,target	(LR) ← CIA + 4.	
bdztdlr	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		24-30
bdztdlr		<i>Extended mnemonic for</i> bclr 10,cr_bit	(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		24-20
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.	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
beqctr	[cr_field]	Branch if equal to address in CTR. Use CRO if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+2		24-26
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 CRO if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+2		24-30
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		24-20
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		24-26
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		24-30
bflrl		<i>Extended mnemonic for</i> bclrl 4,cr_bit	(LR) ← CIA + 4.	
bge	[cr_field], target	Branch if greater than or equal. Use CRO if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+0,target		24-20
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.	
bgectr	[cr_field]	Branch if greater than or equal to address in CTR. Use CRO if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+0		24-26
bgectrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+0	(LR) ← CIA + 4.	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bgelr	[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		24-30
bgelrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+0	(LR) ← CIA + 4.	
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		24-20
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+1		24-26
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+1		24-30
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+1,target		24-20
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+1		24-26
blectrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+1	(LR) ← CIA + 4.	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
blelr	[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		24-30
blelrl		<i>Extended mnemonic for</i> bclr 4,4*cr_field+1	(LR) ← CIA + 4.	
blr		Branch unconditionally to address in LR. <i>Extended mnemonic for</i> bclr 20,0		24-30
blrl		<i>Extended mnemonic for</i> bclr 20,0	(LR) ← CIA + 4.	
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		24-20
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+0		24-26
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+0		24-30
bltlrl		<i>Extended mnemonic for</i> bclr 12,4*cr_field+0	(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		24-20
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.	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
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		24-26
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+2		24-30
bnelrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+2	(LR) ← CIA + 4.	
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		24-20
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+1		24-26
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+1		24-30
bnglrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+1	(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		24-20
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.	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
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		24-26
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+0		24-30
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bc 4,4*cr_field+3,target		24-20
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.	
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		24-26
bnsctrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+3	(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		24-30
bnsrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+3	(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		24-20
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.	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bnuctr	[cr_field]	Branch if not unordered to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bcctr 4,4*cr_field+3</i>		24-26
bnuctrl		<i>Extended mnemonic for bcctrl 4,4*cr_field+3</i>	(LR) ← CIA + 4.	
bnulr	[cr_field]	Branch if not unordered to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bclr 4,4*cr_field+3</i>		24-30
bnulrl		<i>Extended mnemonic for bctrl 4,4*cr_field+3</i>	(LR) ← CIA + 4.	
bso	[cr_field], target	Branch if summary overflow. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bc 12,4*cr_field+3,target</i>		24-20
booa		<i>Extended mnemonic for bca 12,4*cr_field+3,target</i>		
bsol		<i>Extended mnemonic for bcl 12,4*cr_field+3,target</i>	(LR) ← CIA + 4.	
bsola		<i>Extended mnemonic for bcla 12,4*cr_field+3,target</i>	(LR) ← CIA + 4.	
bsoctr	[cr_field]	Branch if summary overflow to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bcctr 12,4*cr_field+3</i>		24-26
bsoctrl		<i>Extended mnemonic for bcctrl 12,4*cr_field+3</i>	(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 bclr 12,4*cr_field+3</i>		24-30
bsolrl		<i>Extended mnemonic for bctrl 12,4*cr_field+3</i>	(LR) ← CIA + 4.	
bt	cr_bit, target	Branch if CR _{cr_bit} = 1. <i>Extended mnemonic for bc 12,cr_bit,target</i>		24-20
bta		<i>Extended mnemonic for bca 12,cr_bit,target</i>		
btl		<i>Extended mnemonic for bcl 12,cr_bit,target</i>	(LR) ← CIA + 4.	
btla		<i>Extended mnemonic for bcla 12,cr_bit,target</i>	(LR) ← CIA + 4.	
btctr	cr_bit	Branch if CR _{cr_bit} = 1 to address in CTR. <i>Extended mnemonic for bcctr 12,cr_bit</i>		24-26
btctrl		<i>Extended mnemonic for bcctrl 12,cr_bit</i>	(LR) ← CIA + 4.	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
btlr	cr_bit	Branch if $CR_{cr_bit} = 1$, to address in LR. <i>Extended mnemonic for</i> bclr 12,cr_bit		24-30
btrl		<i>Extended mnemonic for</i> bclr 12,cr_bit	(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		24-20.
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+3		24-26
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+3		24-30
bunlrl		<i>Extended mnemonic for</i> bclr 12,4*cr_field+3	(LR) ← CIA + 4.	
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		24-147
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		24-147
clrlslwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,n,b-n,31-n	CR[CR0]	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
clrrwi	RA, RS, n	Clear right immediate. ($n < 32$) $(RA)_{32-n:31} \leftarrow ^n0$ <i>Extended mnemonic for</i> rlwinm RA,RS,0,0,31-n		24-147
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 $n = BF$.		24-34
cmpl	BF, 0, RA, IM	Compare (RA) to EXTS(IM), signed. Results in CR[CRn], where $n = BF$.		24-35
cmpl	BF, 0, RA, RB	Compare (RA) to (RB), unsigned. Results in CR[CRn], where $n = BF$.		24-36
cmpli	BF, 0, RA, IM	Compare (RA) to (¹⁶ 0 IM), unsigned. Results in CR[CRn], where $n = BF$.		24-37
cmplw	[BF,] RA, RB	Compare Logical Word. Use CR0 if BF is omitted. <i>Extended mnemonic for</i> cmpl BF,0,RA,RB		24-36
cmplwi	[BF,] RA, IM	Compare Logical Word Immediate. Use CR0 if BF is omitted. <i>Extended mnemonic for</i> cmpli BF,0,RA,IM		24-37
cmpw	[BF,] RA, RB	Compare Word. Use CR0 if BF is omitted. <i>Extended mnemonic for</i> cmp BF,0,RA,RB		24-34
cmpwi	[BF,] RA, IM	Compare Word Immediate. Use CR0 if BF is omitted. <i>Extended mnemonic for</i> cmpl BF,0,RA,IM		24-35
cntlzw	RA, RS	Count leading zeros in RS. Place result in RA.		24-38
cntlzw.			CR[CR0]	
crand	BT, BA, BB	AND bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT} .		24-39
crandc	BT, BA, BB	AND bit (CR_{BA}) with $\neg(CR_{BB})$. Place result in CR_{BT} .		24-40
crclr	bx	Condition register clear. <i>Extended mnemonic for</i> crxor bx,bx,bx		24-46
creqv	BT, BA, BB	Equivalence of bit CR_{BA} with CR_{BB} . $CR_{BT} \leftarrow \neg(CR_{BA} \oplus CR_{BB})$		24-41
crmove	bx, by	Condition register move. <i>Extended mnemonic for</i> cror bx,by,by		24-44
crnand	BT, BA, BB	NAND bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT} .		24-42

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
crnor	BT, BA, BB	NOR bit (CR _{BA}) with (CR _{BB}). Place result in CR _{BT} .		24-43
crnot	bx, by	Condition register not. <i>Extended mnemonic for crnor bx,by,by</i>		24-43
cror	BT, BA, BB	OR bit (CR _{BA}) with (CR _{BB}). Place result in CR _{BT} .		24-44
crorc	BT, BA, BB	OR bit (CR _{BA}) with \neg (CR _{BB}). Place result in CR _{BT} .		24-45
crset	bx	Condition register set. <i>Extended mnemonic for creqv bx,bx,bx</i>		24-41
crxor	BT, BA, BB	XOR bit (CR _{BA}) with (CR _{BB}). Place result in CR _{BT} .		24-46
dcba	RA, RB	Speculatively establish the data cache block which contains the effective address (RAI0) + (RB).		24-47
dcbf	RA, RB	Flush (store, then invalidate) the data cache block which contains the effective address (RAI0) + (RB).		24-49
dcbi	RA, RB	Invalidate the data cache block which contains the effective address (RAI0) + (RB).		24-50
dcbst	RA, RB	Store the data cache block which contains the effective address (RAI0) + (RB).		24-51
dcbt	RA, RB	Load the data cache block which contains the effective address (RAI0) + (RB).		24-52
dcbtst	RA, RB	Load the data cache block which contains the effective address (RAI0) + (RB).		24-53
dcbz	RA, RB	Zero the data cache block which contains the effective address (RAI0) + (RB).		24-54
dccci	RA, RB	Invalidate the data cache congruence class associated with the effective address (RAI0) + (RB).		24-56
dcread	RT, RA, RB	Read either tag or data information from the data cache congruence class associated with the effective address (RAI0) + (RB). Place the results in RT.		24-57
divw	RT, RA, RB	Divide (RA) by (RB), signed. Place result in RT.		24-59
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.		24-60
divwu.			CR[CR0]	
divwuo			XER[SO, OV]	
divwuo.			CR[CR0] XER[SO, OV]	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
eieio		Storage synchronization. All loads and stores that precede the eieio instruction complete before any loads and stores that follow the instruction access main storage. Implemented as sync , which is more restrictive.		24-61
eqv	RA, RS, RB	Equivalence of (RS) with (RB). $(RA) \leftarrow \neg((RS) \oplus (RB))$		24-62
eqv.			CR[CR0]	
extlwi	RA, RS, n, b	Extract and left justify immediate. ($n > 0$) $(RA)_{0:n-1} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{n:31} \leftarrow 32-n_0$ <i>Extended mnemonic for</i> rlwinm RA,RS,b,0,n-1		24-147
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		24-147
extrwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,b+n,32-n,31	CR[CR0]	
extsb	RA, RS	Extend the sign of byte (RS) _{24:31} .		24-63
extsb.		Place the result in RA.	CR[CR0]	
extsh	RA, RS	Extend the sign of halfword (RS) _{16:31} .		24-64
extsh.		Place the result in RA.	CR[CR0]	
icbi	RA, RB	Invalidate the instruction cache block which contains the effective address (RAI0) + (RB).		24-65
icbt	RA, RB	Load the instruction cache block which contains the effective address (RAI0) + (RB).		24-63
iccci	RA, RB	Invalidate instruction cache.		24-67
icread	RA, RB	Read either tag or data information from the instruction cache congruence class associated with the effective address (RAI0) + (RB). Place the results in ICDBDR.		24-68
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		24-146
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		24-146
insrwi.		<i>Extended mnemonic for</i> rlwimi. RA,RS,32-b-n,b,b+n-1	CR[CR0]	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
isync		Synchronize execution context by flushing the prefetch queue.		24-70
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) <i>Extended mnemonic for addi RT,RA,D</i>		24-9
lbz	RT, D(RA)	Load byte from EA = (RAI0) + EXTS(D) and pad left with zeroes, (RT) ← ²⁴ 0 MS(EA,1).		24-71
lbzu	RT, D(RA)	Load byte from EA = (RAI0) + EXTS(D) and pad left with zeroes, (RT) ← ²⁴ 0 MS(EA,1). Update the base address, (RA) ← EA.		24-72
lbzux	RT, RA, RB	Load byte from EA = (RAI0) + (RB) and pad left with zeroes, (RT) ← ²⁴ 0 MS(EA,1). Update the base address, (RA) ← EA.		24-73
lbzx	RT, RA, RB	Load byte from EA = (RAI0) + (RB) and pad left with zeroes, (RT) ← ²⁴ 0 MS(EA,1).		24-74
lha	RT, D(RA)	Load halfword from EA = (RAI0) + EXTS(D) and sign extend, (RT) ← EXTS(MS(EA,2)).		24-75
lhau	RT, D(RA)	Load halfword from EA = (RAI0) + EXTS(D) and sign extend, (RT) ← EXTS(MS(EA,2)). Update the base address, (RA) ← EA.		24-76
lhaux	RT, RA, RB	Load halfword from EA = (RAI0) + (RB) and sign extend, (RT) ← EXTS(MS(EA,2)). Update the base address, (RA) ← EA.		24-77
lhax	RT, RA, RB	Load halfword from EA = (RAI0) + (RB) and sign extend, (RT) ← EXTS(MS(EA,2)).		24-78
lbrx	RT, RA, RB	Load halfword from EA = (RAI0) + (RB), then reverse byte order and pad left with zeroes, (RT) ← ¹⁶ 0 MS(EA+1,1) MS(EA,1).		24-79
lhz	RT, D(RA)	Load halfword from EA = (RAI0) + EXTS(D) and pad left with zeroes, (RT) ← ¹⁶ 0 MS(EA,2).		24-80

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
lhzu	RT, D(RA)	Load halfword from EA = (RAI0) + EXTS(D) and pad left with zeroes, (RT) \leftarrow $^{16}0$ MS(EA,2). Update the base address, (RA) \leftarrow EA.		24-81
lhzux	RT, RA, RB	Load halfword from EA = (RAI0) + (RB) and pad left with zeroes, (RT) \leftarrow $^{16}0$ MS(EA,2). Update the base address, (RA) \leftarrow EA.		24-82
lhzx	RT, RA, RB	Load halfword from EA = (RAI0) + (RB) and pad left with zeroes, (RT) \leftarrow $^{16}0$ MS(EA,2).		24-83
li	RT, IM	Load immediate. (RT) \leftarrow EXTS(IM) <i>Extended mnemonic for addi RT,0,value</i>		24-9
lis	RT, IM	Load immediate shifted. (RT) \leftarrow (IM $^{16}0$) <i>Extended mnemonic for addis RT,0,value</i>		24-12
lmw	RT, D(RA)	Load multiple words starting from EA = (RAI0) + EXTS(D). Place into consecutive registers RT through GPR(31). RA is not altered unless RA = GPR(31).		24-84
lswi	RT, RA, NB	Load consecutive bytes from EA=(RAI0). 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} .		24-85
lswx	RT, RA, RB	Load consecutive bytes from EA=(RAI0)+(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.		24-87
lwarx	RT, RA, RB	Load word from EA = (RAI0) + (RB) and place in RT, (RT) \leftarrow MS(EA,4). Set the Reservation bit.		24-89
lwbrx	RT, RA, RB	Load word from EA = (RAI0) + (RB) then reverse byte order, (RT) \leftarrow MS(EA+3,1) MS(EA+2,1) MS(EA+1,1) MS(EA,1).		24-90

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
lwz	RT, D(RA)	Load word from EA = (RAI0) + EXTS(D) and place in RT, (RT) ← MS(EA,4).		24-91
lwzu	RT, D(RA)	Load word from EA = (RAI0) + EXTS(D) and place in RT, (RT) ← MS(EA,4). Update the base address, (RA) ← EA.		24-92
lwzux	RT, RA, RB	Load word from EA = (RAI0) + (RB) and place in RT, (RT) ← MS(EA,4). Update the base address, (RA) ← EA.		24-93
lwzx	RT, RA, RB	Load word from EA = (RAI0) + (RB) and place in RT, (RT) ← MS(EA,4).		24-94
macchw	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ signed $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ (RT) ← temp _{1:32}		24-95
macchw.			CR[CR0]	
macchwo			XER[SO, OV]	
macchwo.			CR[CR0] XER[SO, OV]	
macchws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ signed $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ if (($prod_0 = RT_0$) ∧ ($RT_0 \neq temp_1$)) then (RT) ← ($RT_0 \parallel^{31} (-RT_0)$) else (RT) ← temp _{1:32}		24-96
macchws.			CR[CR0]	
macchwso			XER[SO, OV]	
macchwso.			CR[CR0] XER[SO, OV]	
macchwsu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ unsigned $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ (RT) ← ($temp_{1:32} \vee^{32} temp_0$)		24-97
macchwsu.			CR[CR0]	
macchwsuo			XER[SO, OV]	
macchwsuo.			CR[CR0] XER[SO, OV]	
macchwu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ unsigned $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ (RT) ← temp _{1:32}		24-98
macchwu.			CR[CR0]	
macchwuo			XER[SO, OV]	
macchwuo.			CR[CR0] XER[SO, OV]	
machhw	RT, RA, RB	$prod_{0:15} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ signed $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ (RT) ← temp _{1:32}		24-99
machhw.			CR[CR0]	
machhwo			XER[SO, OV]	
machhwo.			CR[CR0] XER[SO, OV]	
machhws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ signed $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ if (($prod_0 = RT_0$) ∧ ($RT_0 \neq temp_1$)) then (RT) ← ($RT_0 \parallel^{31} (-RT_0)$) else (RT) ← temp _{1:32}		24-100
machhws.			CR[CR0]	
machhwso			XER[SO, OV]	
machhwso.			CR[CR0] XER[SO, OV]	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
machhwsu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ unsigned $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow (temp_{1:32} \vee {}^{32}temp_0)$		24-101
machhwsu.			CR[CR0]	
machhwsuo			XER[SO, OV]	
machhwsuo.			CR[CR0] XER[SO, OV]	
machhwu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ unsigned $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-102
machhwu.			CR[CR0]	
machhwuo			XER[SO, OV]	
machhwuo.			CR[CR0] XER[SO, OV]	
machlw	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ signed $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-103
machlw.			CR[CR0]	
machlwo			XER[SO, OV]	
machlwo.			CR[CR0] XER[SO, OV]	
machws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ 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 $(RT) \leftarrow temp_{1:32}$		24-104
machws.			CR[CR0]	
machwso			XER[SO, OV]	
machwso.			CR[CR0] XER[SO, OV]	
machwsu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ unsigned $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow (temp_{1:32} \vee {}^{32}temp_0)$		24-105
machwsu.			CR[CR0]	
machwsuo			XER[SO, OV]	
machwsuo.			CR[CR0] XER[SO, OV]	
machwuo	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ unsigned $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-106
machwuo.			CR[CR0]	
machwuo			XER[SO, OV]	
machwuo.			CR[CR0] XER[SO, OV]	
mcrf	BF, BFA	Move CR field, $(CR[CRn]) \leftarrow (CR[CRm])$ where $m \leftarrow BFA$ and $n \leftarrow BF$.		24-107
mcrxr	BF	Move XER[0:3] into field CRn, where $n \leftarrow BF$. $CR[CRn] \leftarrow (XER[SO, OV, CA])$. $(XER[SO, OV, CA]) \leftarrow {}^30$.		24-108
mfcr	RT	Move from CR to RT, $(RT) \leftarrow (CR)$.		24-109
mfcdcr	RT, DCRN	Move from DCR to RT, $(RT) \leftarrow (DCR(DCRN))$.		24-110
mfmsr	RT	Move from MSR to RT, $(RT) \leftarrow (MSR)$.		24-111

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mfccr0 mfctr mfdac1 mfdac2 mfdear mfdbcr0 mfdbcr1 mfdbsr mfdccr mfdcwr mfdvc1 mfdvc2 mfesr mfevpr mfiac1 mfiac2 mfiac3 mfiac4 mficcr mficdbdr mfir mfpld mfplt mfpv mfsvr mfsler mfsprg0 mfsprg1 mfsprg2 mfsprg3 mfsprg4 mfsprg5 mfsprg6 mfsprg7 mfsrr0 mfsrr1 mfsrr2 mfsrr3 mfsu0r mftr mftr mfxr mfzpr	RT	Move from special purpose register (SPR) SPRN. <i>Extended mnemonic for mfspr RT,SPRN</i> See Table 25-2, "Special Purpose Registers," on page 25-2 for listing of valid SPRN values.		24-112
mfspr	RT, SPRN	Move from SPR to RT, (RT) ← (SPR(SPRN)).		24-112
mftb	RT, TBRN	Move from TBR to RT, (RT) ← (TBR(TBRN)).		24-114
mftb	RT	Move the contents of TBL into RT, (RT) ← (TBL) <i>Extended mnemonic for mftb RT,TBL</i>		24-114

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mftbu	RT	Move the contents of TBU into RT, (RT) ← (TBU) <i>Extended mnemonic for mftb RT,TBU</i>		24-114
mr	RT, RS	Move register. (RT) ← (RS) <i>Extended mnemonic for or RT,RS,RS</i>		24-140
mr.		<i>Extended mnemonic for or. RT,RS,RS</i>	CR[CR0]	
mtcr	RS	Move to Condition Register. <i>Extended mnemonic for mtcrf 0xFF,RS</i>		24-116
mtcrf	FXM, RS	Move some or all of the contents of RS into CR as specified by FXM field, mask ← ${}^4(\text{FXM}_0) \parallel {}^4(\text{FXM}_1) \parallel \dots \parallel$ ${}^4(\text{FXM}_6) \parallel {}^4(\text{FXM}_7)$. (CR) ← ((RS) ∧ mask) ∨ (CR) ∧ ¬mask).		24-116
mtdcr	DCRN, RS	Move to DCR from RS, (DCR(DCRN)) ← (RS).		24-117
mtmsr	RS	Move to MSR from RS, (MSR) ← (RS).		24-118

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mtccr0 mtctr mtdac1 mtdac2 mtdbcr0 mtdbcr1 mtdbsr mtdccr mtdear mtdcwr mtdvc1 mtdvc2 mtesr mtevpr mtiac1 mtiac2 mtiac3 mtiac4 mtlccr mtlcdbdr mtlr mtpld mtpit mtpvr mtsgr mtsler metsprg0 metsprg1 metsprg2 metsprg3 metsprg4 metsprg5 metsprg6 metsprg7 mtsrr0 mtsrr1 mtsrr2 mtsrr3 mtsu0r mttbl mttbu mttcr mttsr mtxer mtzpr	RS	Move to SPR SPRN. <i>Extended mnemonic for mtspr SPRN,RS</i> See Table 25-2, "Special Purpose Registers," on page 25-2 for listing of valid SPRN values.		24-119
mtspr	SPRN, RS	Move to SPR from RS, (SPR(SPRN)) ← (RS).		24-119
mulchw	RT, RA, RB	(RT) _{0:31} ← (RA) _{16:31} × (RB) _{0:15} signed		24-121
mulchw.			CR[CR0]	
mulchwu	RT, RA, RB	(RT) _{0:31} ← (RA) _{16:31} × (RB) _{0:15} unsigned		24-122
mulchwu.			CR[CR0]	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mulhww	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ signed		24-123
mulhww.			CR[CR0]	
mulhwu	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ unsigned		24-124
mulhwu.			CR[CR0]	
mulhw	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ signed		24-127
mulhw.			CR[CR0]	
mulhwu	RT, RA, RB	$(RT)_{16:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ unsigned		24-128
mulhwu.			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). $(RT) \leftarrow prod_{0:31}$.	CR[CR0]	24-125
mulhw.				
mulhwu	RT, RA, RB	Multiply (RA) and (RB), unsigned. Place high-order result in RT. $prod_{0:63} \leftarrow (RA) \times (RB)$ (unsigned). $(RT) \leftarrow prod_{0:31}$.	CR[CR0]	24-126
mulhwu.				
mulli	RT, RA, IM	Multiply (RA) and IM, signed. Place low-order result in RT. $prod_{0:47} \leftarrow (RA) \times IM$ (signed) $(RT) \leftarrow prod_{16:47}$		24-129
mulw	RT, RA, RB	Multiply (RA) and (RB), signed. Place low-order result in RT. $prod_{0:63} \leftarrow (RA) \times (RB)$ (signed). $(RT) \leftarrow prod_{32:63}$.		24-130
mulw.			CR[CR0]	
mulwo			XER[SO, OV]	
mulwo.			CR[CR0] XER[SO, OV]	
nand	RA, RS, RB	NAND (RS) with (RB). Place result in RA.		24-131
nand.			CR[CR0]	
neg	RT, RA	Negative (twos complement) of RA. $(RT) \leftarrow \neg(RA) + 1$		24-132
neg.			CR[CR0]	
nego			XER[SO, OV]	
nego.			CR[CR0] XER[SO, OV]	
nmacchw	RT, RA, RB	$nprod_{0:31} \leftarrow \neg((RA)_{16:31} \times (RB)_{0:15})$ signed $temp_{0:32} \leftarrow nprod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-133
nmacchw.			CR[CR0]	
nmacchwo			XER[SO, OV]	
nmacchwo.			CR[CR0] XER[SO, OV]	
nmacchws	RT, RA, RB	$nprod_{0:31} \leftarrow \neg((RA)_{16:31} \times (RB)_{0:15})$ 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 $(RT) \leftarrow temp_{1:32}$		24-134
nmacchws.			CR[CR0]	
nmacchwso			XER[SO, OV]	
nmacchwso.			CR[CR0] XER[SO, OV]	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
nmachhw	RT, RA, RB	$nprod_{0:31} \leftarrow -((RA)_{0:15} \times (RB)_{0:15})$ signed $temp_{0:32} \leftarrow nprod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-135
nmachhw.			CR[CR0]	
nmachhwo			XER[SO, OV]	
nmachhwo.			CR[CR0] XER[SO, OV]	
nop		Preferred no-op, triggers optimizations based on no-ops. <i>Extended mnemonic for ori 0,0,0</i>		24-142
nor	RA, RS, RB	NOR (RS) with (RB). Place result in RA.		24-139
nor.			CR[CR0]	
not	RA, RS	Complement register. $(RA) \leftarrow \neg(RS)$ <i>Extended mnemonic for nor RA,RS,RS</i>		24-139
not.			CR[CR0] <i>Extended mnemonic for nor. RA,RS,RS</i>	
or	RA, RS, RB	OR (RS) with (RB). Place result in RA.		24-140
or.			CR[CR0]	
orc	RA, RS, RB	OR (RS) with \neg (RB). Place result in RA.		24-141
orc.			CR[CR0]	
ori	RA, RS, IM	OR (RS) with (¹⁶ 0 IM). Place result in RA.		24-142
oris	RA, RS, IM	OR (RS) with (IM ¹⁶ 0). Place result in RA.		24-143
rftci		Return from critical interrupt $(PC) \leftarrow (SRR2)$. $(MSR) \leftarrow (SRR3)$.		24-144
rfti		Return from interrupt. $(PC) \leftarrow (SRR0)$. $(MSR) \leftarrow (SRR1)$.		24-145
rlwimi	RA, RS, SH, MB, ME	Rotate left word immediate, then insert according to mask. $r \leftarrow ROTL((RS), SH)$ $m \leftarrow MASK(MB, ME)$ $(RA) \leftarrow (r \wedge m) \vee ((RA) \wedge \neg m)$		24-146
rlwimi.			CR[CR0]	
rlwinm	RA, RS, SH, MB, ME	Rotate left word immediate, then AND with mask. $r \leftarrow ROTL((RS), SH)$ $m \leftarrow MASK(MB, ME)$ $(RA) \leftarrow (r \wedge m)$		24-147
rlwinm.			CR[CR0]	
rlwnm	RA, RS, RB, MB, ME	Rotate left word, then AND with mask. $r \leftarrow ROTL((RS), (RB)_{27:31})$ $m \leftarrow MASK(MB, ME)$ $(RA) \leftarrow (r \wedge m)$		24-150
rlwnm.			CR[CR0]	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
rotlw	RA, RS, RB	Rotate left. (RA) \leftarrow ROTL((RS), (RB) _{27:31}) <i>Extended mnemonic for</i> rlwnm RA,RS,RB,0,31		24-150
rotlw.		<i>Extended mnemonic for</i> rlwnm. RA,RS,RB,0,31	CR[CR0]	
rotlwi	RA, RS, n	Rotate left immediate. (RA) \leftarrow ROTL((RS), n) <i>Extended mnemonic for</i> rlwinm RA,RS,n,0,31		24-147
rotlwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,n,0,31	CR[CR0]	
rotrwi	RA, RS, n	Rotate right immediate. (RA) \leftarrow ROTL((RS), 32-n) <i>Extended mnemonic for</i> rlwinm RA,RS,32-n,0,31		24-147
rotrwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,32-n,0,31	CR[CR0]	
sc		System call exception is generated. (SRR1) \leftarrow (MSR) (SRR0) \leftarrow (PC) PC \leftarrow EVPR _{0:15} x'0C00' (MSR[WE, PR, EE, PE, DR, IR]) \leftarrow 0		24-151
slw	RA, RS, RB	Shift left (RS) by (RB) _{27:31} . n \leftarrow (RB) _{27:31} . r \leftarrow ROTL((RS), n). if (RB) ₂₆ = 0 then m \leftarrow MASK(0, 31 - n) else m \leftarrow ³² 0. (RA) \leftarrow r \wedge m.	CR[CR0]	24-152
slw.				
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		24-147
slwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,n,0,31-n	CR[CR0]	
sraw	RA, RS, RB	Shift right algebraic (RS) by (RB) _{27:31} . n \leftarrow (RB) _{27:31} . r \leftarrow ROTL((RS), 32 - n). if (RB) ₂₆ = 0 then m \leftarrow MASK(n, 31) else m \leftarrow ³² 0. s \leftarrow (RS) ₀ . (RA) \leftarrow (r \wedge m) \vee (³² s \wedge \neg m). XER[CA] \leftarrow s \wedge ((r \wedge \neg m) \neq 0).	CR[CR0]	24-153
sraw.				

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
srwi	RA, RS, SH	Shift right algebraic (RS) by SH. $n \leftarrow SH$. $r \leftarrow ROTL((RS), 32 - n)$. $m \leftarrow MASK(n, 31)$. $s \leftarrow (RS)_0$. $(RA) \leftarrow (r \wedge m) \vee ({}^{32}s \wedge \neg m)$. $XER[CA] \leftarrow s \wedge ((r \wedge \neg m) \neq 0)$.		24-154
srwi.			CR[CR0]	
srw	RA, RS, RB	Shift right (RS) by (RB) _{27:31} . $n \leftarrow (RB)_{27:31}$. $r \leftarrow ROTL((RS), 32 - n)$. if (RB) ₂₆ = 0 then $m \leftarrow MASK(n, 31)$ else $m \leftarrow {}^{32}0$. $(RA) \leftarrow r \wedge m$.		24-155
srw.			CR[CR0]	
srwi	RA, RS, n	Shift right immediate. ($n < 32$) $(RA)_{n:31} \leftarrow (RS)_{0:31-n}$ $(RA)_{0:n-1} \leftarrow {}^n0$ <i>Extended mnemonic for</i> rlwinm RA,RS,32-n,n,31		24-147
srwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,32-n,n,31	CR[CR0]	
stb	RS, D(RA)	Store byte (RS) _{24:31} in memory at $EA = (RAI0) + EXTS(D)$.		24-156
stbu	RS, D(RA)	Store byte (RS) _{24:31} in memory at $EA = (RAI0) + EXTS(D)$. Update the base address, $(RA) \leftarrow EA$.		24-157
stbux	RS, RA, RB	Store byte (RS) _{24:31} in memory at $EA = (RAI0) + (RB)$. Update the base address, $(RA) \leftarrow EA$.		24-158
stbx	RS, RA, RB	Store byte (RS) _{24:31} in memory at $EA = (RAI0) + (RB)$.		24-159
sth	RS, D(RA)	Store halfword (RS) _{16:31} in memory at $EA = (RAI0) + EXTS(D)$.		24-160
sthbrx	RS, RA, RB	Store halfword (RS) _{16:31} byte-reversed in memory at $EA = (RAI0) + (RB)$. $MS(EA, 2) \leftarrow (RS)_{24:31} \parallel (RS)_{16:23}$		24-161
sthu	RS, D(RA)	Store halfword (RS) _{16:31} in memory at $EA = (RAI0) + EXTS(D)$. Update the base address, $(RA) \leftarrow EA$.		24-162
sthux	RS, RA, RB	Store halfword (RS) _{16:31} in memory at $EA = (RAI0) + (RB)$. Update the base address, $(RA) \leftarrow EA$.		24-163
sthx	RS, RA, RB	Store halfword (RS) _{16:31} in memory at $EA = (RAI0) + (RB)$.		24-164

Table A-1. PPC405GP 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 = (RAI0) + EXTS(D).		24-165
stswi	RS, RA, NB	Store consecutive bytes in memory starting at EA=(RAI0). 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).		24-166
stswx	RS, RA, RB	Store consecutive bytes in memory starting at EA=(RAI0)+(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).		24-167
stw	RS, D(RA)	Store word (RS) in memory at EA = (RAI0) + EXTS(D).		24-169
stwbrx	RS, RA, RB	Store word (RS) byte-reversed in memory at EA = (RAI0) + (RB). $MS(EA, 4) \leftarrow (RS)_{24:31} \parallel (RS)_{16:23} \parallel (RS)_{8:15} \parallel (RS)_{0:7}$		24-170
stwcx.	RS, RA, RB	Store word (RS) in memory at EA = (RAI0) + (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 $(CR[CR0]) \leftarrow {}^2_0 \parallel 0 \parallel XER_{so}$.		24-171
stwu	RS, D(RA)	Store word (RS) in memory at EA = (RAI0) + EXTS(D). Update the base address, (RA) \leftarrow EA.		24-173
stwux	RS, RA, RB	Store word (RS) in memory at EA = (RAI0) + (RB). Update the base address, (RA) \leftarrow EA.		24-174
stwx	RS, RA, RB	Store word (RS) in memory at EA = (RAI0) + (RB).		24-175
sub	RT, RA, RB	Subtract (RB) from (RA). (RT) \leftarrow $\neg(RB) + (RA) + 1$. <i>Extended mnemonic for subf RT,RB,RA</i>		24-176
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]	

Table A-1. PPC405GP 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 subfc RT, RB, RA</i>		24-177
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]	
subf	RT, RA, RB	Subtract (RA) from (RB). (RT) \leftarrow \neg (RA) + (RB) + 1.		24-176
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].		24-177
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].		24-178
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].		24-179
subfme	RT, RA, RB	Subtract (RA) from (-1) with carry-in. (RT) \leftarrow \neg (RA) + (-1) + XER[CA]. Place carry-out in XER[CA].		24-180
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].		24-181
subfze.			CR[CR0]	
subfzeo			XER[SO, OV]	
subfzeo.			CR[CR0] XER[SO, OV]	
subi	RT, RA, IM	Subtract EXTS(IM) from (RAI0). Place result in RT. <i>Extended mnemonic for addi RT, RA, -IM</i>		24-9

Table A-1. PPC405GP 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 addic RT,RA,-IM</i>		24-10
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]	24-11
subis	RT, RA, IM	Subtract (IM ¹⁶ 0) from (RAI0). Place result in RT. <i>Extended mnemonic for addis RT,RA,-IM</i>		24-12
sync		Synchronization. All instructions that precede sync complete before any instructions that follow sync begin. When sync completes, all storage accesses initiated prior to sync will have completed.		24-182
tlbia		All TLB entries are invalidated and become unavailable for translation by clearing the valid (V) bit in the TLBHI portion of each TLB entry. The rest of the TLB fields unmodified.		24-183
tlbre	RT, RA, WS	If WS = 0: Load TLBHI of the selected TLB entry into RT. Load PID with the contents of the TID field of the selected TLB entry. (RT) ← TLBHI[(RA)] (PID) ← TLB[(RA)] _{TID} If WS = 1: Load TLBLO portion of the selected TLB entry into RT. (RT) ← TLBLO[(RA)]		24-184
tlbrehi	RT, RA	Load TLBHI of the selected TLB entry into RT. Load PID with the contents of the TID field of the selected TLB entry. (RT) ← TLBHI[(RA)] (PID) ← TLB[(RA)] _{TID} <i>Extended mnemonic for tlbre RT,RA,0</i>		24-184
tlbrelo	RT, RA	Load TLBLO of the selected TLB entry into RT. (RT) ← TLBLO[(RA)] <i>Extended mnemonic for tlbre RT,RA,1</i>		24-184

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
tlbsx	RT,RA,RB	Search the TLB for a valid entry that translates the EA. EA = (RA10) + (RB). If found, (RT) ← Index of TLB entry. If not found, (RT) Undefined.		24-186
tlbsx.		If found, (RT) ← Index of TLB entry. CR[CR0] _{EQ} ← 1. If not found, (RT) Undefined. CR[CR0] _{EQ} ← 1.	CR[CR0] _{LT,GT,SO}	
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 PPC405GP, tlbsync is a no-op.		24-187
tlbwe	RS, RA,WS	If WS = 0: Write TLBHI of the selected TLB entry from RS. Write the TID field of the selected TLB entry from the PID register. TLBHI[(RA)] ← (RS) TLB[(RA)] _{TID} ← (PID) _{24:31} If WS = 1: Write TLBLO portion of the selected TLB entry from RS. TLBLO[(RA)] ← (RS)		24-188
tlbwehl	RS, RA	Write TLBHI of the selected TLB entry from RS. Write the TID field of the selected TLB entry from the PID register. TLBHI[(RA)] ← (RS) TLB[(RA)] _{TID} ← (PID) _{24:31} <i>Extended mnemonic for</i> tlbwe RS,RA,0		24-188
tlbewlo	RS, RA	Write TLBLO of the selected TLB entry from RS. TLBLO[(RA)] ← (RS) <i>Extended mnemonic for</i> tlbwe RS,RA,1		24-188

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
trap		Trap unconditionally. <i>Extended mnemonic for tw 31,0,0</i>		24-190
tw _{eq}	RA, RB	Trap if (RA) equal to (RB). <i>Extended mnemonic for tw 4,RA,RB</i>		
tw _{ge}		Trap if (RA) greater than or equal to (RB). <i>Extended mnemonic for tw 12,RA,RB</i>		
tw _{gt}		Trap if (RA) greater than (RB). <i>Extended mnemonic for tw 8,RA,RB</i>		
tw _{le}		Trap if (RA) less than or equal to (RB). <i>Extended mnemonic for tw 20,RA,RB</i>		
tw _{lge}		Trap if (RA) logically greater than or equal to (RB). <i>Extended mnemonic for tw 5,RA,RB</i>		
tw _{lgt}		Trap if (RA) logically greater than (RB). <i>Extended mnemonic for tw 1,RA,RB</i>		
tw _{lle}		Trap if (RA) logically less than or equal to (RB). <i>Extended mnemonic for tw 6,RA,RB</i>		
tw _{llt}		Trap if (RA) logically less than (RB). <i>Extended mnemonic for tw 2,RA,RB</i>		
tw _{lng}		Trap if (RA) logically not greater than (RB). <i>Extended mnemonic for tw 6,RA,RB</i>		
tw _{lnl}		Trap if (RA) logically not less than (RB). <i>Extended mnemonic for tw 5,RA,RB</i>		
tw _{lt}		Trap if (RA) less than (RB). <i>Extended mnemonic for tw 16,RA,RB</i>		
tw _{ne}		Trap if (RA) not equal to (RB). <i>Extended mnemonic for tw 24,RA,RB</i>		
tw _{ng}		Trap if (RA) not greater than (RB). <i>Extended mnemonic for tw 20,RA,RB</i>		
tw _{nl}		Trap if (RA) not less than (RB). <i>Extended mnemonic for tw 12,RA,RB</i>		
tw		TO, RA, RB	Trap exception is generated if, comparing (RA) with (RB), any condition specified by TO is true.	

Table A-1. PPC405GP 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		24-193
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.		24-193
wrtee	RS	Write value of RS ₁₆ to MSR[EE].		24-196
wrteei	E	Write value of E to MSR[EE].		24-197

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
xor	RA, RS, RB	XOR (RS) with (RB). Place result in RA.		24-198
xor.			CR[CR0]	
xori	RA, RS, IM	XOR (RS) with (¹⁶ 0 IM). Place result in RA.		24-199
xoris	RA, RS, IM	XOR (RS) with (IM ¹⁶ 0). Place result in RA.		24-200

A.2 Instructions Sorted by Opcode

All 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 A-44) in bits 0:5. Some instructions also have a secondary opcode field (shown as field XO in Figure A-1 through Figure A-9). PPC405GP instructions, sorted by primary and secondary opcode, are listed in Table A-2.

The “Form” indicated in the table refers to the arrangement of valid field combinations within the four-byte instruction. See “Instruction Formats,” on page A-41, 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-2. PPC405GP Instructions by Opcode

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
3		D	twi	TO, RA, IM	24-193
4	8	X	mulhhu	RT, RA, RB	24-124
			mulhhu.		
4	12 (524)	XO	machhu	RT, RA, RB	24-102
			machhu.		
			machhuo		
			machhuo.		
4	40	X	mulhu	RT, RA, RB	24-123
			mulhu.		
4	44 (556)	XO	machu	RT, RA, RB	24-99
			machu.		
			machuo		
			machuo.		
4	46 (558)	XO	nmachu	RT, RA, RB	24-135
			nmachu.		
			nmachuo		
			nmachuo.		

Table A-2. PPC405GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
4	76 (588)	XO	machwsu	RT, RA, RB	24-101
			machwsu.		
			machwsuo		
			machwsuo.		
4	108 (620)	XO	machws	RT, RA, RB	24-100
			machws.		
			machwso		
			machwso.		
4	110 (622)	XO	nmachws	RT, RA, RB	24-136
			nmachws.		
			nmachwso		
			nmachwso.		
4	136	X	mulchw	RT, RA, RB	24-122
			mulchw.		
4	140 (652)	XO	macchw	RT, RA, RB	24-98
			macchw.		
			macchwuo		
			macchwuo.		
4	168	X	mulchw	RT, RA, RB	24-121
			mulchw.		
4	172 (684)	XO	macchw	RT, RA, RB	24-95
			macchw.		
			macchwuo		
			macchwuo.		
4	174 (686)	XO	nmacchw	RT, RA, RB	24-133
			nmacchw.		
			nmacchwuo		
			nmacchwuo.		
4	204 (716)	XO	macchwsu	RT, RA, RB	24-101
			macchwsu.		
			macchwsuo		
			macchwsuo.		
4	236 (748)	XO	macchws	RT, RA, RB	24-96
			macchws.		
			macchwso		
			macchwso.		
4	238 (750)	XO	nmacchws	RT, RA, RB	24-134
			nmacchws.		
			nmacchwso		
			nmacchwso.		

Table A-2. PPC405GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
4	392	X	mulhwu	RT, RA, RB	24-128
			mulhwu.		
4	396 (908)	XO	macchw	RT, RA, RB	24-106
			macchw.		
			macchwuo		
			macchwuo.		
4	424	X	mulhw	RT, RA, RB	24-126
			mulhw.		
4	428 (940)	XO	macchw	RT, RA, RB	24-103
			macchw.		
			macchwuo		
			macchwuo.		
4	430 (942)	XO	nmacchw	RT, RA, RB	24-137
			nmacchw.		
			nmacchwuo		
			nmacchwuo.		
4	492 (972)	XO	macchw	RT, RA, RB	24-104
			macchw.		
			macchwso		
			macchwso.		
4	460 (1004)	XO	macchw	RT, RA, RB	24-105
			macchw.		
			macchwso		
			macchwso.		
4	494 (1006)	XO	nmacchw	RT, RA, RB	24-138
			nmacchw.		
			nmacchwso		
			nmacchwso.		
7		D	mulli	RT, RA, IM	24-129
8		D	subflic	RT, RA, IM	24-179
10		D	cmpli	BF, 0, RA, IM	24-37
11		D	cmpi	BF, 0, RA, IM	24-35
12		D	addic	RT, RA, IM	24-10
13		D	addic.	RT, RA, IM	24-11
14		D	addi	RT, RA, IM	24-9
15		D	addis	RT, RA, IM	24-12
16		B	bc	BO, BI, target	24-20
			bca		
			bcl		
			bcla		
17		SC	sc		24-151

Table A-2. PPC405GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
18		I	b	target	24-19
			ba		
			bl		
			bla		
19	0	XL	mcrf	BF, BFA	24-107
19	16	XL	bclr	BO, BI	24-30
			bclrl		
19	33	XL	crnor	BT, BA, BB	24-43
19	50	XL	rfi		24-145
19	51	XL	rfcl		24-144
19	129	XL	crandc	BT, BA, BB	24-40
19	150	XL	isync		24-70
19	193	XL	crxor	BT, BA, BB	24-46
19	225	XL	crnand	BT, BA, BB	24-42
19	257	XL	crand	BT, BA, BB	24-39
19	289	XL	creqv	BT, BA, BB	24-41
19	417	XL	crorc	BT, BA, BB	24-45
19	449	XL	cror	BT, BA, BB	24-44
19	528	XL	bcctr	BO, BI	24-26
			bcctrl		
20		M	rlwiml	RA, RS, SH, MB, ME	24-146
			rlwiml.		
21		M	rlwinm	RA, RS, SH, MB, ME	24-147
			rlwinm.		
23		M	rlwnm	RA, RS, RB, MB, ME	24-150
			rlwnm.		
24		D	ori	RA, RS, IM	24-142
25		D	oris	RA, RS, IM	24-143
26		D	xori	RA, RS, IM	24-199
27		D	xoris	RA, RS, IM	24-200
28		D	andl.	RA, RS, IM	24-17
29		D	andis.	RA, RS, IM	24-18
31	0	X	cmp	BF, 0, RA, RB	24-34
31	4	X	tw	TO, RA, RB	24-190
31	8 (520)	XO	subfc	RT, RA, RB	24-177
			subfc.		
			subfco		
			subfco.		

Table A-2. PPC405GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	10 (522)	XO	addc	RT, RA, RB	24-7
			addc.		
			addco		
			addco.		
31	11	XO	mulhwu	RT, RA, RB	24-126
			mulhwu.		
31	19	X	mfcrr	RT	24-109
31	20	X	lwarx	RT, RA, RB	24-89
31	23	X	lwz	RT, RA, RB	24-94
31	24	X	slw	RA, RS, RB	24-152
			slw.		
31	26	X	cntlzw	RA, RS	24-38
			cntlzw.		
31	28	X	and	RA, RS, RB	24-15
			and.		
31	32	X	cmpl	BF, 0, RA, RB	24-36
31	40 (552)	XO	subf	RT, RA, RB	24-176
			subf.		
			subfo		
			subfo.		
31	54	X	dcbst	RA, RB	24-53
31	55	X	lwzux	RT, RA, RB	24-93
31	60	X	andc	RA, RS, RB	24-16
			andc.		
31	75	XO	mulhw	RT, RA, RB	24-125
			mulhw.		
31	83	X	mfmsr	RT	24-111
31	86	X	dcbf	RA, RB	24-49
31	87	X	lbzx	RT, RA, RB	24-74
31	104 (616)	XO	neg	RT, RA	24-132
			neg.		
			nego		
			nego.		
31	119	X	lbzux	RT, RA, RB	24-73
31	124	X	nor	RA, RS, RB	24-139
			nor.		
31	131	X	wrtee	RS	24-196
31	136 (648)	XO	subfe	RT, RA, RB	24-178
			subfe.		
			subfeo		
			subfeo.		

Table A-2. PPC405GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	138 (650)	XO	adde	RT, RA, RB	24-8
			adde.		
			addeo		
			addeo.		
31	144	XFX	mtcrf	FXM, RS	24-116
31	146	X	mtmsr	RS	24-118
31	150	X	stwcx.	RS, RA, RB	24-171
31	151	X	stwx	RS, RA, RB	24-175
31	163	X	wrteel	E	24-197
31	183	X	stwux	RS, RA, RB	24-174
31	200 (712)	XO	subfze	RT, RA, RB	24-181
			subfze.		
			subfzeo		
			subfzeo.		
31	202 (714)	XO	addze	RT, RA	24-14
			addze.		
			addzeo		
			addzeo.		
31	215	X	stbx	RS, RA, RB	24-159
31	232 (744)	XO	subfme	RT, RA, RB	24-180
			subfme.		
			subfmeo		
			subfmeo.		
31	234 (746)	XO	addme	RT, RA	24-13
			addme.		
			addmeo		
			addmeo.		
31	235 (747)	XO	mullw	RT, RA, RB	24-130
			mullw.		
			mullwo		
			mullwo.		
31	246	X	dcbst	RA, RB	24-53
31	247	X	stbux	RS, RA, RB	24-157
31	262	X	icbt	RA, RB	24-66
31	266 (778)	XO	add	RT, RA, RB	24-6
			add.		
			addo		
			addo.		
31	278	X	dcbt	RA, RB	24-51
31	279	X	lhzx	RT, RA, RB	24-83

Table A-2. PPC405GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	284	X	eqv	RA, RS, RB	24-62
			eqv.		
31	311	X	lhzux	RT, RA, RB	24-82
31	316	X	xor	RA, RS, RB	24-198
			xor.		
31	323	XFX	mfdcr	RT, DCRN	24-110
31	339	XFX	mfspr	RT, SPRN	24-112
31	343	X	lhax	RT, RA, RB	24-78
31	370	X	tlbia		24-183
31	371	XFX	mftb	RT, TBRN	24-114
31	375	X	lhaux	RT, RA, RB	24-77
31	407	X	sthx	RS, RA, RB	24-164
31	412	X	orc	RA, RS, RB	24-141
			orc.		
31	439	X	sthux	RS, RA, RB	24-163
31	444	X	or	RA, RS, RB	24-140
			or.		
31	451	XFX	mtdcr	DCRN, RS	24-117
31	454	X	dccci	RA, RB	24-56
31	459 (971)	XO	divwu	RT, RA, RB	24-60
			divwu.		
			divwuo		
			divwuo.		
31	467	XFX	mtspr	SPRN, RS	24-119
31	470	X	dcbi	RA, RB	24-50
31	476	X	nand	RA, RS, RB	24-131
			nand.		
31	486	X	dcread	RT, RA, RB	24-57
31	491 (1003)	XO	divw	RT, RA, RB	24-59
			divw.		
			divwo		
			divwo.		
31	512	X	mcrxr	BF	24-108
31	533	X	lswx	RT, RA, RB	24-87
31	534	X	lwbrx	RT, RA, RB	24-90
31	536	X	srw	RA, RS, RB	24-155
			srw.		
31	566	X	tlbsync		24-187
31	597	X	lswi	RT, RA, NB	24-85
31	598	X	sync		24-182
31	661	X	stswx	RS, RA, RB	24-167

Table A-2. PPC405GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
31	662	X	stwbrx	RS, RA, RB	24-170
31	725	X	stswi	RS, RA, NB	24-166
31	758	X	dcbz	RA, RB	24-47
31	790	X	lhbrx	RT, RA, RB	24-79
31	792	X	sraw	RA, RS, RB	24-153
			sraw.		
31	824	X	srawi	RA, RS, SH	24-154
			srawi.		
31	854	X	eleio		24-61
31	914	X	tlbsx	RT,RA,RB	24-186
			tlbsx.		
31	918	X	sthbrx	RS, RA, RB	24-161
31	922	X	extsh	RA, RS	24-64
			extsh.		
31	946	X	tlbre	RT, RA,WS	24-184
31	954	X	extsb	RA, RS	24-63
			extsb.		
31	966	X	iccci	RA, RB	24-67
31	978	X	tlbwe	RS, RA,WS	24-188
31	982	X	icbi	RA, RB	24-65
31	998	X	icread	RA, RB	24-68
31	1014	X	dcbz	RA, RB	24-54
32		D	lwz	RT, D(RA)	24-91
33		D	lwzu	RT, D(RA)	24-92
34		D	lbz	RT, D(RA)	24-71
35		D	lbzu	RT, D(RA)	24-72
36		D	stw	RS, D(RA)	24-169
37		D	stwu	RS, D(RA)	24-173
38		D	stb	RS, D(RA)	24-156
39		D	stbu	RS, D(RA)	24-157
40		D	lhz	RT, D(RA)	24-80
41		D	lhzu	RT, D(RA)	24-81
42		D	lha	RT, D(RA)	24-75
43		D	lhau	RT, D(RA)	24-76
44		D	sth	RS, D(RA)	24-160
45		D	sthu	RS, D(RA)	24-162
46		D	lmw	RT, D(RA)	24-84
47		D	stmw	RS, D(RA)	24-165

A.3 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 PPC405GP executes all invalid instruction forms without causing an illegal instruction exception.

A.3.1 Instruction Fields

PPC405GP 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)	Absolute address bit. 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. 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.
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 “BO Field on Conditional Branches” on page 3-35.
- 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 twos-complement integer displacement for load/store instructions.
- DCRN (11:20) Specifies a device control register (DCR).
- 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) Link bit.
 0 Do not update the link register (LR).
 1 Update the LR with the address of the next instruction.
- MB (21:25) Mask begin.
 Used in rotate-and-mask instructions to specify the beginning bit of a mask.
- ME (26:30) Mask end.
 Used in rotate-and-mask instructions to specify the ending bit of a mask.
- 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.
- Rc (31) Record bit.
 0 Do not set the CR.
 1 Set the CR to reflect the result of an operation.
 See “Condition Register (CR)” on page 3-12 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).
- TO (6:10) Specifies the conditions on which to trap, as described under **tw** and **twi** instructions.
- 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.3.2 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-2 on page A-33 indicates which “form” is utilized by each PPC405GP opcode. Fields indicated by slashes (/, //, or ///) are reserved. The figures are adapted from the PowerPC User Instruction Set Architecture.

A.3.2.1 I-Form

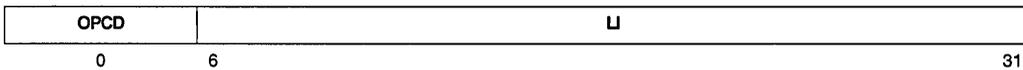


Figure A-1. I Instruction Format

A.3.2.2 B-Form

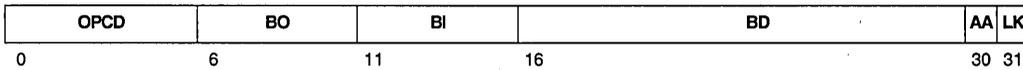


Figure A-2. B Instruction Format

A.3.2.3 SC-Form

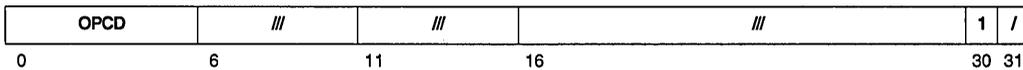


Figure A-3. SC Instruction Format

A.3.2.4 D-Form

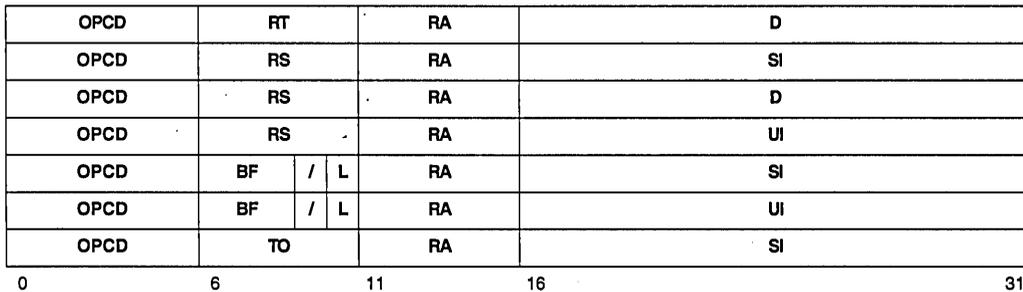


Figure A-4. D Instruction Format

A.3.2.5 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		Rc
OPCD	BF	//		///		///		XO		/
OPCD	BF	//		///		U		XO		Rc
OPCD	BF	//		///		///		XO		/
OPCD	TO		RA		RB		XO		/	
OPCD	BT		///		///		XO		Rc	
OPCD	///		RA		RB		XO		/	
OPCD	///		///		///		XO		/	
OPCD	///		///		E	//		XO		/
0	6	11	16	21	31					

Figure A-5. X Instruction Format

A.3.2.6 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.3.2.7 XFX-Form

OPCD	RT	SPRF			XO	/
OPCD	RT	DCRF			XO	/
OPCD	RT	/	FXM	/	XO	/
OPCD	RS	SPRF			XO	/
OPCD	RS	DCRF			XO	/
0	6	11	16	21		31

Figure A-7. XFX Instruction Format

A.3.2.8 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.3.2.9 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

Appendix B. Instructions by Category

Chapter 24, "Instruction Set," contains detailed descriptions of the instructions, their operands, and notation.

Table B-1 summarizes the instruction categories in the PPC405GP instruction set. The instructions within each category are listed in subsequent tables.

Table B-1. PPC405GP Instruction Set Functional Summary

Storage Reference	load, store
Arithmetic and Logical	add, subtract, negate, multiply, divide, and, andc, or, orc, xor, nand, nor, xnor, sign extension, count leading zeros, multiply accumulate
Comparison	compare, compare logical, compare immediate
Branch	branch, branch conditional, branch to LR, branch to CTR
CR Logical	crand, crandc, cror, crorc, crnand, crnor, crxor, crxnor, move CR field
Rotate/Shift	rotate and insert, rotate and mask, shift left, shift right
Cache Control	invalidate, touch, zero, flush, store, read
Interrupt Control	write to external interrupt enable bit, move to/from MSR, return from interrupt, return from critical interrupt
Processor Management	system call, synchronize, trap, move to/from DCRs, move to/from SPRs, move to/from CR

B.1 Implementation-Specific Instructions

To meet the functional requirements of processors for embedded systems and real-time applications, the PPC405GP defines the implementation-specific instructions summarized in Table B-2.

Table B-2. Implementation-specific Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
dccci	RA, RB	Invalidate the data cache congruence class associated with the effective address (EA) (RAI0) + (RB).		24-56
dcread	RT, RA, RB	Read either tag or data information from the data cache congruence class associated with the EA (RAI0) + (RB). Place the results in RT.		24-57
iccci	RA, RB	Invalidate instruction cache.		24-67
icread	RA, RB	Read either tag or data information from the instruction cache congruence class associated with the EA (RAI0) + (RB). Place the results in ICDBDR.		24-68

Table B-2. Implementation-specific Instructions (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
macchw	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ signed $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-95
macchw.			CR[CR0]	
macchw0			XER[SO, OV]	
macchw0.			CR[CR0] XER[SO, OV]	
macchws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ 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 $(RT) \leftarrow temp_{1:32}$		24-96
macchws.			CR[CR0]	
macchwso			XER[SO, OV]	
macchwso.			CR[CR0] XER[SO, OV]	
macchwsu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ unsigned $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow (temp_{1:32} \vee^{32} temp_0)$		24-97
macchwsu.			CR[CR0]	
macchwsuo			XER[SO, OV]	
macchwsuo.			CR[CR0] XER[SO, OV]	
macchwu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ unsigned $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-98
macchwu.			CR[CR0]	
macchwuo			XER[SO, OV]	
macchwuo.			CR[CR0] XER[SO, OV]	
machhw	RT, RA, RB	$prod_{0:15} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ signed $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-99
machhw.			CR[CR0]	
machhwo			XER[SO, OV]	
machhwo.			CR[CR0] XER[SO, OV]	
machhws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ 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 $(RT) \leftarrow temp_{1:32}$		24-100
machhws.			CR[CR0]	
machhwso			XER[SO, OV]	
machhwso.			CR[CR0] XER[SO, OV]	
machhwsu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ unsigned $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow (temp_{1:32} \vee^{32} temp_0)$		24-101
machhwsu.			CR[CR0]	
machhwsuo			XER[SO, OV]	
machhwsuo.			CR[CR0] XER[SO, OV]	

Table B-2. Implementation-specific Instructions (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
machhwu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ unsigned $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-102
machhwu.			CR[CR0]	
machhwuo			XER[SO, OV]	
machhwuo.			CR[CR0] XER[SO, OV]	
maclhw	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ signed $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-103
maclhw.			CR[CR0]	
maclhwo			XER[SO, OV]	
maclhwo.			CR[CR0] XER[SO, OV]	
maclhws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ 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 $(RT) \leftarrow temp_{1:32}$		24-104
maclhws.			CR[CR0]	
maclhwso			XER[SO, OV]	
maclhwso.			CR[CR0] XER[SO, OV]	
maclhwsu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ unsigned $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow (temp_{1:32} \vee^{32} temp_0)$		24-105
maclhwsu.			CR[CR0]	
maclhwsuo			XER[SO, OV]	
maclhwsuo.			CR[CR0] XER[SO, OV]	
maclhwu	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ unsigned $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-106
maclhwu.			CR[CR0]	
maclhwuo			XER[SO, OV]	
maclhwuo.			CR[CR0] XER[SO, OV]	
mulchw	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ signed		24-121
mulchw.			CR[CR0]	
mulchwu	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ unsigned		24-122
mulchwu.			CR[CR0]	
mulhhw	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ signed		24-123
mulhhw.			CR[CR0]	
mulhhwu	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ unsigned		24-124
mulhhwu.			CR[CR0]	
mullhw	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ signed		24-127
mullhw.			CR[CR0]	

Table B-2. Implementation-specific Instructions (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mullhwu	RT, RA, RB	$(RT)_{16:31} \leftarrow (RA)_{0:15} \times (RB)_{16:31}$ unsigned		24-128
mullhwu.			CR[CR0]	
nmacchw	RT, RA, RB	$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{0:15})$ signed $temp_{0:32} \leftarrow nprod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-133
nmacchw.			CR[CR0]	
nmacchwo			XER[SO, OV]	
nmacchwo.			CR[CR0] XER[SO, OV]	
nmacchws	RT, RA, RB	$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{0:15})$ signed $temp_{0:32} \leftarrow nprod_{0:31} + (RT)$ if $((nprod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \parallel^{31} (-RT_0))$ else $(RT) \leftarrow temp_{1:32}$		24-134
nmacchws.			CR[CR0]	
nmacchwso			XER[SO, OV]	
nmacchwso.			CR[CR0] XER[SO, OV]	
nmachhw	RT, RA, RB	$nprod_{0:31} \leftarrow -((RA)_{0:15} \times (RB)_{0:15})$ signed $temp_{0:32} \leftarrow nprod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-135
nmachhw.			CR[CR0]	
nmachhwo			XER[SO, OV]	
nmachhwo.			CR[CR0] XER[SO, OV]	
nmachhws	RT, RA, RB	$nprod_{0:31} \leftarrow -((RA)_{0:15} \times (RB)_{0:15})$ signed $temp_{0:32} \leftarrow nprod_{0:31} + (RT)$ if $((nprod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \parallel^{31} (-RT_0))$ else $(RT) \leftarrow temp_{1:32}$		24-136
nmachhws.			CR[CR0]	
nmachhwso			XER[SO, OV]	
nmachhwso.			CR[CR0] XER[SO, OV]	
nmachlw	RT, RA, RB	$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{16:31})$ signed $temp_{0:32} \leftarrow nprod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-137
nmachlw.			CR[CR0]	
nmachlwo			XER[SO, OV]	
nmachlwo.			CR[CR0] XER[SO, OV]	
nmachlws	RT, RA, RB	$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{16:31})$ signed $temp_{0:32} \leftarrow nprod_{0:31} + (RT)$ if $((nprod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \parallel^{31} (-RT_0))$ else $(RT) \leftarrow temp_{1:32}$		24-138
nmachlws.			CR[CR0]	
nmachlwso			XER[SO, OV]	
nmachlwso.			CR[CR0] XER[SO, OV]	

B.2 Instructions in the IBM PowerPC Embedded Environment

To meet the functional requirements of processors for embedded systems and real-time applications, the IBM PowerPC Embedded Environment defines instructions that are not part of the PowerPC Architecture.

Table B-3 summarizes the PPC405GP instructions in the PowerPC Embedded Environment.

Table B-3. Instructions in the IBM PowerPC Embedded Environment

Mnemonic	Operands	Function	Other Registers Changed	Page
dcb	RA, RB	Speculatively establish the data cache block which contains the EA (RAI0) + (RB).		24-47
dcbf	RA, RB	Flush (store, then invalidate) the data cache block which contains the EA (RAI0) + (RB).		24-49
dcbi	RA, RB	Invalidate the data cache block which contains the EA (RAI0) + (RB).		24-50
dcbst	RA, RB	Store the data cache block which contains the EA (RAI0) + (RB).		24-51
dcbt	RA, RB	Load the data cache block which contains the EA (RAI0) + (RB).		24-52
dcbtst	RA, RB	Load the data cache block which contains the EA (RAI0) + (RB).		24-53
dcbz	RA, RB	Zero the data cache block which contains the EA (RAI0) + (RB).		24-54
eieio		Storage synchronization. All loads and stores that precede the eieio instruction complete before any loads and stores that follow the instruction access main storage. Implemented as sync , which is more restrictive.		24-61
icbi	RA, RB	Invalidate the instruction cache block which contains the EA (RAI0) + (RB).		24-65
icbt	RA, RB	Load the instruction cache block which contains the EA (RAI0) + (RB).		24-66
isync		Synchronize execution context by flushing the prefetch queue.		24-70
mfdcr	RT, DCRN	Move from DCR to RT, (RT) ← (DCR(DCRN)).		24-110
mftb	RT	Move the contents of a Time Base Register (TBR) into RT, TBRN ← TBRF _{5:9} TBRF _{0:4} (RT) ← (TBR(TBRN))		24-114
mfmsr	RT	Move from MSR to RT, (RT) ← (MSR).		24-118

Table B-3. Instructions in the IBM PowerPC Embedded Environment (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mfspr	RT, SPRN	Move from SPR to RT, (RT) ← (SPR(SPRN)). Privileged for all SPRs except LR, CTR, TBHU, TBLU, and XER.		24-112
mtdcr	DCRN, RS	Move to DCR from RS, (DCR(DCRN)) ← (RS).		24-117
mtmsr	RS	Move to MSR from RS, (MSR) ← (RS).		24-118
mtspr	SPRN, RS	Move to SPR from RS, (SPR(SPRN)) ← (RS). Privileged for all SPRs except LR, CTR, and XER.		24-119
rfdi		Return from critical interrupt (PC) ← (SRR2). (MSR) ← (SRR3).		24-144
rfdi		Return from interrupt. (PC) ← (SRR0). (MSR) ← (SRR1).		24-145
tlbia		All of the entries in the TLB are invalidated and become unavailable for translation by clearing the valid (V) bit in the TLBHI portion of each TLB entry. The rest of the fields in the TLB entries are unmodified.		24-183
tlbre	RT, RA, WS	If WS = 0: Load TLBHI portion of the selected TLB entry into RT. Load the PID register with the contents of the TID field of the selected TLB entry. (RT) ← TLBHI[(RA)] (PID) ← TLB[(RA)] _{TID} If WS = 1: Load TLBLO portion of the selected TLB entry into RT. (RT) ← TLBLO[(RA)]		24-184

Table B-3. Instructions in the IBM PowerPC Embedded Environment (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
tlbsx	RT,RA,RB	Search the TLB array for a valid entry which translates the EA EA = (RA10) + (RB). If found, (RT) ← Index of TLB entry. If not found, (RT) Undefined.		24-186
tlbsx.		If found, (RT) ← Index of TLB entry. CR[CR0] _{EQ} ← 1. If not found, (RT) Undefined. CR[CR0] _{EQ} ← 1.	CR[CR0] _{LT,GT,SO}	
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 PPC405GP, tlbsync is a no-op.		24-187
tlbwe	RS, RA,WS	If WS = 0: Write TLBHI portion of the selected TLB entry from RS. Write the TID field of the selected TLB entry from the PID register. TLBHI[(RA)] ← (RS) TLB[(RA)] _{TID} ← (PID) _{24:31} If WS = 1: Write TLBLO portion of the selected TLB entry from RS. TLBLO[(RA)] ← (RS)		24-188
wrtee	RS	Write value of RS ₁₆ to MSR[EE].		24-196
wrteei	E	Write value of E to MSR[EE].		24-197

B.3 Privileged Instructions

Table B-4 lists instructions that are under control of the MSR[PR] bit. These instructions are not allowed to be executed when MSR[PR] = 1:

Table B-4. Privileged Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
dcbi	RA, RB	Invalidate the data cache block which contains the EA (RA10) + (RB).		24-50
dccci	RA, RB	Invalidate the data cache congruence class associated with the EA (RA10) + (RB).		24-56

Table B-4. Privileged Instructions (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
dcread	RT, RA, RB	Read either tag or data information from the data cache congruence class associated with the EA (RAI0) + (RB). Place the results in RT.		24-57
iccci	RA, RB	Invalidate instruction cache.		24-67
lcread	RA, RB	Read either tag or data information from the instruction cache congruence class associated with the EA (RAI0) + (RB). Place the results in ICDBDR.		24-68
mfocr	RT, DCRN	Move from DCR to RT, (RT) ← (DCR(DCRN)).		24-110
mfmsr	RT	Move from MSR to RT, (RT) ← (MSR).		24-118
mfspr	RT, SPRN	Move from SPR to RT, (RT) ← (SPR(SPRN)). Privileged for all SPRs except LR, CTR, TBHU, TBLU, and XER.		24-112
mtocr	DCRN, RS	Move to DCR from RS, (DCR(DCRN)) ← (RS).		24-117
mtmsr	RS	Move to MSR from RS, (MSR) ← (RS).		24-118
mtspr	SPRN, RS	Move to SPR from RS, (SPR(SPRN)) ← (RS). Privileged for all SPRs except LR, CTR, and XER.		24-119
rfci		Return from critical interrupt (PC) ← (SRR2). (MSR) ← (SRR3).		24-144
rfi		Return from interrupt. (PC) ← (SRR0). (MSR) ← (SRR1).		24-145
tlbre	RT, RA, WS	If WS = 0: Load TLBHI portion of the selected TLB entry into RT. Load the PID register with the contents of the TID field of the selected TLB entry. (RT) ← TLBHI[(RA)] (PID) ← TLB[(RA)] _{TID} If WS = 1: Load TLBLO portion of the selected TLB entry into RT. (RT) ← TLBLO[(RA)]		24-184

Table B-4. Privileged Instructions (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
tlbsx	RT,RA,RB	Search the TLB array for a valid entry which translates the EA EA = (RA)0 + (RB). If found, (RT) ← Index of TLB entry. If not found, (RT) Undefined.		24-186
tlbsx.		If found, (RT) ← Index of TLB entry. CR[CR0] _{EQ} ← 1. If not found, (RT) Undefined. CR[CR0] _{EQ} ← 1.	CR[CR0] _{LT,GT,SO}	
tlbwe	RS, RA,WS	If WS = 0: Write TLBHI portion of the selected TLB entry from RS. Write the TID field of the selected TLB entry from the PID register. TLBHI[(RA)] ← (RS) TLB[(RA)] _{TID} ← (PID) _{24:31} If WS = 1: Write TLBLO portion of the selected TLB entry from RS. TLBLO[(RA)] ← (RS)		24-188
wrtee	RS	Write value of RS ₁₆ to the External Enable bit (MSR[EE]).		24-196
wrteei	E	Write value of E to the External Enable bit (MSR[EE]).		24-197

B.4 Assembler Extended Mnemonics

In the appendix “Assembler Extended Mnemonics” of the PowerPC Architecture, it is required that a PowerPC assembler support at least a minimal set of extended mnemonics. These mnemonics encode to the opcodes of other instructions; the only benefit of extended mnemonics is improved usability. Code using extended mnemonics can be easier to write and to understand. Table B-5 lists the extended mnemonics required for the PPC405GP.

Note for every Branch Conditional mnemonic:

Bit 4 of the BO field provides a hint about the most likely outcome of a conditional branch. (“Branch Prediction” on page 3-36 describes branch prediction). Assemblers should set BO₄ = 0 unless a specific reason exists otherwise. In the BO field values specified in the following table, BO₄ = 0 has always been assumed. The assembler must allow the programmer to specify branch prediction. To do this, the assembler will support a suffix to every conditional branch mnemonic, as follows:

+Predict branch to be taken.

–Predict branch not to be taken.

As specific examples, **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 3-36).

Table B-5. Extended Mnemonics for PPC405GP

Mnemonic	Operands	Function	Other Registers Changed	Page
bctr		Branch unconditionally to address in CTR. <i>Extended mnemonic for</i> bcctr 20,0		24-26
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		24-20
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.	
bdnzlr		Decrement CTR. Branch, if CTR ≠ 0, to address in LR. <i>Extended mnemonic for</i> bclr 16,0		24-30
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		24-20
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> bclr 0,cr_bit		24-30
bdnzflrl		<i>Extended mnemonic for</i> bclrl 0,cr_bit	(LR) ← CIA + 4.	

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
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		24-20
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.	
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		24-30
bdnztlrl		<i>Extended mnemonic for</i> bclrl 8,cr_bit	(LR) \leftarrow CIA + 4.	
bdz	target	Decrement CTR. Branch if CTR = 0. <i>Extended mnemonic for</i> bc 18,0,target		24-20
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.	
bdzlr		Decrement CTR. Branch, if CTR = 0, to address in LR. <i>Extended mnemonic for</i> bclr 18,0		24-30
bdzlrll		<i>Extended mnemonic for</i> bclrl 18,0	(LR) \leftarrow CIA + 4.	
bdzlf	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR _{cr_bit} = 0. <i>Extended mnemonic for</i> bc 2,cr_bit,target		24-20
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.	

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
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		24-30
bdzflrl		<i>Extended mnemonic for</i> bclr 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		24-20
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.	
bdztlr		<i>Extended mnemonic for</i> bcla 10,cr_bit,target	(LR) ← CIA + 4.	
bdztlrl	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		24-30
bdztlrl		<i>Extended mnemonic for</i> bclr 10,cr_bit	(LR) ← CIA + 4.	
beq	[cr_field,] target	Branch if equal. Use CRO if cr_field is omitted. <i>Extended mnemonic for</i> bc 12,4*cr_field+2,target		24-20
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.	
beqlr		<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 CRO if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+2		24-26
beqctrl		<i>Extended mnemonic for</i> bcctrl 12,4*cr_field+2	(LR) ← CIA + 4.	

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
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		24-30
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		24-20
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		24-26
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		24-30
bflrl		<i>Extended mnemonic for</i> bclrl 4,cr_bit	(LR) ← CIA + 4.	
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		24-20
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.	
bgela		<i>Extended mnemonic for</i> bcla 4,4*cr_field+0,target	(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		24-26
bgctrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+0	(LR) ← CIA + 4.	

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bgelr	[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		24-30
bgelrl		<i>Extended mnemonic for</i> bclr1 4,4*cr_field+0	(LR) ← CIA + 4.	
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		24-20
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.	
bgctr	[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		24-26
bgctr1		<i>Extended mnemonic for</i> bcctrl 12,4*cr_field+1	(LR) ← CIA + 4.	
bgtr	[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		24-30
bgtr1		<i>Extended mnemonic for</i> bclr1 12,4*cr_field+1	(LR) ← CIA + 4.	
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		24-20
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.	

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
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		24-26
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+1		24-30
blelrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+1	(LR) ← CIA + 4.	
blr		Branch, unconditionally, to address in LR. <i>Extended mnemonic for</i> bclr 20,0		24-30
blrl		<i>Extended mnemonic for</i> bclrl 20,0	(LR) ← CIA + 4.	
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		24-20
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+0		24-26
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+0		24-30
bltlrl		<i>Extended mnemonic for</i> bclrl 12,4*cr_field+0	(LR) ← CIA + 4.	

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
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		24-20
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+2		24-26
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+2		24-30
bnelrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+2	(LR) ← CIA + 4.	
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		24-20
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 4,4*cr_field+1		24-26
bngctrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+1	(LR) ← CIA + 4.	

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
bnglr	[cr_field]	Branch, if not greater than, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bclr 4,4*cr_field+1</i>		24-30
bnglrl		<i>Extended mnemonic for bclrl 4,4*cr_field+1</i>	(LR) ← CIA + 4.	
bnl	[cr_field,] target	Branch if not less than. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bc 4,4*cr_field+0,target</i>		24-20
bnla		<i>Extended mnemonic for bca 4,4*cr_field+0,target</i>		
bnll		<i>Extended mnemonic for bcl 4,4*cr_field+0,target</i>	(LR) ← CIA + 4.	
bnlla		<i>Extended mnemonic for bcla 4,4*cr_field+0,target</i>	(LR) ← CIA + 4.	
bnlctr	[cr_field]	Branch, if not less than, to address in CTR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bcctr 4,4*cr_field+0</i>		24-26
bnlctrl		<i>Extended mnemonic for bcctrl 4,4*cr_field+0</i>	(LR) ← CIA + 4.	
bnllr	[cr_field]	Branch, if not less than, to address in LR. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bclr 4,4*cr_field+0</i>		24-30
bnllrl		<i>Extended mnemonic for bclrl 4,4*cr_field+0</i>	(LR) ← CIA + 4.	
bns	[cr_field,] target	Branch if not summary overflow. Use CR0 if cr_field is omitted. <i>Extended mnemonic for bc 4,4*cr_field+3,target</i>		24-20
bnsa		<i>Extended mnemonic for bca 4,4*cr_field+3,target</i>		
bnsll		<i>Extended mnemonic for bcl 4,4*cr_field+3,target</i>	(LR) ← CIA + 4.	
bnslla		<i>Extended mnemonic for bcla 4,4*cr_field+3,target</i>	(LR) ← CIA + 4.	

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
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		24-26
bnsctrl		<i>Extended mnemonic for</i> bcctrl 4,4*cr_field+3	(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		24-30
bnsrlr		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+3	(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		24-20
bnuu		<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> bclal 4,4*cr_field+3,target	(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		24-26
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 4,4*cr_field+3		24-30
bnulrl		<i>Extended mnemonic for</i> bclrl 4,4*cr_field+3	(LR) ← CIA + 4.	

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
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		24-20
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.	
bsoctr	[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		24-26
bsoctrl		<i>Extended mnemonic for</i> bcctrl 12,4*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		24-30
bsolrl		<i>Extended mnemonic for</i> bctrl 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		24-20
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		24-26
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		24-30
btlrl		<i>Extended mnemonic for</i> bctrl 12,cr_bit	(LR) ← CIA + 4.	

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
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		24-20
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bcctr 12,4*cr_field+3		24-26
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 CR0 if cr_field is omitted. <i>Extended mnemonic for</i> bclr 12,4*cr_field+3		24-30
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		24-147
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 ≤ b < 32) (RA) _{b-n:31-n} ← (RS) _{b:31} (RA) _{32-n:31} ← ⁿ 0 (RA) _{0:b-n-1} ← ^{b-n} 0 <i>Extended mnemonic for</i> rlwinm RA,RS,n,b-n,31-n		24-147
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} ← ⁿ 0 <i>Extended mnemonic for</i> rlwinm RA,RS,0,0,31-n		24-147
clrrwi.		<i>Extended mnemonic for</i> rlwinm. RA,RS,0,0,31-n	CR[CR0]	

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
cmplw	[BF,] RA, RB	Compare Logical Word. Use CR0 if BF is omitted. <i>Extended mnemonic for cmpl BF,0,RA,RB</i>		24-36
cmplwi	[BF,] RA, IM	Compare Logical Word Immediate. Use CR0 if BF is omitted. <i>Extended mnemonic for cmpli BF,0,RA,IM</i>		24-37
cmpw	[BF,] RA, RB	Compare Word. Use CR0 if BF is omitted. <i>Extended mnemonic for cmp BF,0,RA,RB</i>		24-34
cmpwi	[BF,] RA, IM	Compare Word Immediate. Use CR0 if BF is omitted. <i>Extended mnemonic for cmpi BF,0,RA,IM</i>		24-35
crclr	bx	Condition register clear. <i>Extended mnemonic for crxor bx,bx,bx</i>		24-46
crmve	bx, by	Condition register move. <i>Extended mnemonic for cror bx,by,by</i>		24-44
crnot	bx, by	Condition register not. <i>Extended mnemonic for crnor bx,by,by</i>		24-43
crset	bx	Condition register set. <i>Extended mnemonic for creqv bx,bx,bx</i>		24-41
extlwi	RA, RS, n, b	Extract and left justify immediate. ($n > 0$) $(RA)_{0:n-1} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{n:31} \leftarrow 32-n0$ <i>Extended mnemonic for rlwinm RA,RS,b,0,n-1</i>		24-147
extlwi.		<i>Extended mnemonic for rlwinm. RA,RS,b,0,n-1</i>	CR[CR0]	
extrwi	RA, RS, n, b	Extract and right justify immediate. ($n > 0$) $(RA)_{32-n:31} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{0:31-n} \leftarrow 32-n0$ <i>Extended mnemonic for rlwinm RA,RS,b+n,32-n,31</i>		24-147
extrwi.		<i>Extended mnemonic for rlwinm. RA,RS,b+n,32-n,31</i>	CR[CR0]	

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
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> riwimi RA,RS,32-b,b,b+n-1		24-146
inslwi.		<i>Extended mnemonic for</i> riwimi. 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> riwimi RA,RS,32-b-n,b,b+n-1		24-146
insrwi.		<i>Extended mnemonic for</i> riwimi. RA,RS,32-b-n,b,b+n-1	CR[CR0]	
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		24-9
li	RT, IM	Load immediate. $(RT) \leftarrow EXTS(IM)$ <i>Extended mnemonic for</i> addi RT,0,value		24-9
lis	RT, IM	Load immediate shifted. $(RT) \leftarrow (IM \parallel 16_0)$ <i>Extended mnemonic for</i> addis RT,0,value		24-12

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mfccr0 mfctr mfdac1 mfdac2 mfdear mfdbcr0 mfdbcr1 mfdbsr mfdccr mfdcwr mfdvc1 mfdvc2 mfesr mfevpr mfiac1 mfiac2 mfiac3 mfiac4 mficcr mficbdr mflr mfpid mfpit mfpvr mfsgr mfsler mfsprg0 mfsprg1 mfsprg2 mfsprg3 mfsprg4 mfsprg5 mfsprg6 mfsprg7 mfsrr0 mfsrr1 mfsrr2 mfsrr3 mfsu0r mftcr mftsr mfxer mfzpr	RT	Move from special purpose register (SPR) SPRN. <i>Extended mnemonic for mfspr RT,SPRN</i> See Table 25-2, "Special Purpose Registers," on page 25-2 for listing of valid SPRN values.		24-112
mftb	RT	Move the contents of TBL into RT, (RT) ← (TBL) <i>Extended mnemonic for mftb RT,TBL</i>		24-114

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mftbu	RT	Move the contents of TBU into RT, (RT) ← (TBU) <i>Extended mnemonic for mftb RT,TBU</i>		24-114
mr	RT, RS	Move register. (RT) ← (RS) <i>Extended mnemonic for or RT,RS,RS</i>		24-140
mr.		<i>Extended mnemonic for or. RT,RS,RS</i>	CR[CR0]	
mtcr	RS	Move to Condition Register. <i>Extended mnemonic for mtcrf 0xFF,RS</i>		24-116

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mtccr0 mtctr mtdac1 mtdac2 mtdbc0 mtdbc1 mtdbsr mtdccr mtdear mtdcwr mtdvc1 mtdvc2 mtesr mtevpr mtiac1 mtiac2 mtiac3 mtiac4 mticcr mticdbdr mtr mtpid mtpit mtpvr mtsgr mtsler mtsprg0 mtsprg1 mtsprg2 mtsprg3 mtsprg4 mtsprg5 mtsprg6 mtsprg7 mtsrr0 mtsrr1 mtsrr2 mtsrr3 mtsu0r mttbl mttbu mttcr mttsr mtxer mtzpr	RS	Move to SPR SPRN. <i>Extended mnemonic for mtspr SPRN,RS</i> See Table 25-2, "Special Purpose Registers," on page 25-2 for listing of valid SPRN values.		24-119
nop		Preferred no-op; triggers optimizations based on no-ops. <i>Extended mnemonic for ori 0,0,0</i>		24-142

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
not	RA, RS	Complement register. (RA) ← ¬(RS) <i>Extended mnemonic for nor RA,RS,RS</i>		24-139
not.		<i>Extended mnemonic for nor. RA,RS,RS</i>	CR[CR0]	
rotlw	RA, RS, RB	Rotate left. (RA) ← ROTL((RS), (RB) _{27:31}) <i>Extended mnemonic for rlwnm RA,RS,RB,0,31</i>		24-150
rotlw.		<i>Extended mnemonic for rlwnm. RA,RS,RB,0,31</i>	CR[CR0]	
rotlwi	RA, RS, n	Rotate left immediate. (RA) ← ROTL((RS), n) <i>Extended mnemonic for rlwinm RA,RS,n,0,31</i>		24-147
rotlwi.		<i>Extended mnemonic for rlwinm. RA,RS,n,0,31</i>	CR[CR0]	
rotrwi	RA, RS, n	Rotate right immediate. (RA) ← ROTL((RS), 32-n) <i>Extended mnemonic for rlwinm RA,RS,32-n,0,31</i>		24-147
rotrwi.		<i>Extended mnemonic for rlwinm. RA,RS,32-n,0,31</i>	CR[CR0]	
slwi	RA, RS, n	Shift left immediate. (n < 32) (RA) _{0:31-n} ← (RS) _{n:31} (RA) _{32-n:31} ← n0 <i>Extended mnemonic for rlwinm RA,RS,n,0,31-n</i>		24-147
slwi.		<i>Extended mnemonic for rlwinm. RA,RS,n,0,31-n</i>	CR[CR0]	
srwi	RA, RS, n	Shift right immediate. (n < 32) (RA) _{n:31} ← (RS) _{0:31-n} (RA) _{0:n-1} ← n0 <i>Extended mnemonic for rlwinm RA,RS,32-n,n,31</i>		24-147
srwi.		<i>Extended mnemonic for rlwinm. RA,RS,32-n,n,31</i>	CR[CR0]	

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
sub	RT, RA, RB	Subtract (RB) from (RA). (RT) \leftarrow \neg (RB) + (RA) + 1. <i>Extended mnemonic for subf RT,RB,RA</i>		24-176
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]	
subc	RT, RA, RB	Subtract (RB) from (RA). (RT) \leftarrow \neg (RB) + (RA) + 1. Place carry-out in XER[CA]. <i>Extended mnemonic for subfc RT,RB,RA</i>		24-177
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]	
subi	RT, RA, IM	Subtract EXTS(IM) from (RAI0). Place result in RT. <i>Extended mnemonic for addi RT,RA,-IM</i>		24-9
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>		24-10
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]	24-11
subis	RT, RA, IM	Subtract (IM ¹⁶ 0) from (RAI0). Place result in RT. <i>Extended mnemonic for addis RT,RA,-IM</i>		24-12

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
tweqi	RA, IM	Trap if (RA) equal to EXTS(IM). <i>Extended mnemonic for twi 4,RA,IM</i>		
twgei		Trap if (RA) greater than or equal to EXTS(IM). <i>Extended mnemonic for twi 12,RA,IM</i>		
twgti		Trap if (RA) greater than EXTS(IM). <i>Extended mnemonic for twi 8,RA,IM</i>		
twlei		Trap if (RA) less than or equal to EXTS(IM). <i>Extended mnemonic for twi 20,RA,IM</i>		
twlgei		Trap if (RA) logically greater than or equal to EXTS(IM). <i>Extended mnemonic for twi 5,RA,IM</i>		
twlgti		Trap if (RA) logically greater than EXTS(IM). <i>Extended mnemonic for twi 1,RA,IM</i>		
twllei		Trap if (RA) logically less than or equal to EXTS(IM). <i>Extended mnemonic for twi 6,RA,IM</i>		
twliti		Trap if (RA) logically less than EXTS(IM). <i>Extended mnemonic for twi 2,RA,IM</i>		
twlengi		Trap if (RA) logically not greater than EXTS(IM). <i>Extended mnemonic for twi 6,RA,IM</i>		
twlnli		Trap if (RA) logically not less than EXTS(IM). <i>Extended mnemonic for twi 5,RA,IM</i>		
twliti		Trap if (RA) less than EXTS(IM). <i>Extended mnemonic for twi 16,RA,IM</i>		
twnei		Trap if (RA) not equal to EXTS(IM). <i>Extended mnemonic for twi 24,RA,IM</i>		
twngi		Trap if (RA) not greater than EXTS(IM). <i>Extended mnemonic for twi 20,RA,IM</i>		
twnli		Trap if (RA) not less than EXTS(IM). <i>Extended mnemonic for twi 12,RA,IM</i>		

B.5 Storage Reference Instructions

The PPC405GP uses load and store instructions to transfer data between memory and the general purpose registers. Load and store instructions operate on byte, halfword and word data. The storage reference instructions also support loading or storing multiple registers, character strings, and byte-reversed data. Table B-6 shows the storage reference instructions available for use in the PPC405GP.

Table B-6. Storage Reference Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
lbz	RT, D(RA)	Load byte from EA = (RAI0) + EXTS(D) and pad left with zeroes, (RT) \leftarrow $^{24}0$ MS(EA,1).		24-71
lbzu	RT, D(RA)	Load byte from EA = (RAI0) + EXTS(D) and pad left with zeroes, (RT) \leftarrow $^{24}0$ MS(EA,1). Update the base address, (RA) \leftarrow EA.		24-72
lbzux	RT, RA, RB	Load byte from EA = (RAI0) + (RB) and pad left with zeroes, (RT) \leftarrow $^{24}0$ MS(EA,1). Update the base address, (RA) \leftarrow EA.		24-73
lbzx	RT, RA, RB	Load byte from EA = (RAI0) + (RB) and pad left with zeroes, (RT) \leftarrow $^{24}0$ MS(EA,1).		24-74
lha	RT, D(RA)	Load halfword from EA = (RAI0) + EXTS(D) and sign extend, (RT) \leftarrow EXTS(MS(EA,2)).		24-75
lhau	RT, D(RA)	Load halfword from EA = (RAI0) + EXTS(D) and sign extend, (RT) \leftarrow EXTS(MS(EA,2)). Update the base address, (RA) \leftarrow EA.		24-76
lhaux	RT, RA, RB	Load halfword from EA = (RAI0) + (RB) and sign extend, (RT) \leftarrow EXTS(MS(EA,2)). Update the base address, (RA) \leftarrow EA.		24-77
lhax	RT, RA, RB	Load halfword from EA = (RAI0) + (RB) and sign extend, (RT) \leftarrow EXTS(MS(EA,2)).		24-78
lbrx	RT, RA, RB	Load halfword from EA = (RAI0) + (RB), then reverse byte order and pad left with zeroes, (RT) \leftarrow $^{16}0$ MS(EA+1,1) MS(EA,1).		24-79
lhz	RT, D(RA)	Load halfword from EA = (RAI0) + EXTS(D) and pad left with zeroes, (RT) \leftarrow $^{16}0$ MS(EA,2).		24-80

Table B-6. Storage Reference Instructions (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
lhzu	RT, D(RA)	Load halfword from EA = (RAI0) + EXTS(D) and pad left with zeroes, (RT) \leftarrow $^{16}0$ MS(EA,2). Update the base address, (RA) \leftarrow EA.		24-81
lhzux	RT, RA, RB	Load halfword from EA = (RAI0) + (RB) and pad left with zeroes, (RT) \leftarrow $^{16}0$ MS(EA,2). Update the base address, (RA) \leftarrow EA.		24-82
lhzx	RT, RA, RB	Load halfword from EA = (RAI0) + (RB) and pad left with zeroes, (RT) \leftarrow $^{16}0$ MS(EA,2).		24-83
lmw	RT, D(RA)	Load multiple words starting from EA = (RAI0) + EXTS(D). Place into consecutive registers, RT through GPR(31). RA is not altered unless RA = GPR(31).		24-84
lswi	RT, RA, NB	Load consecutive bytes from EA = (RAI0). 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} .		24-85
lswx	RT, RA, RB	Load consecutive bytes from EA=(RAI0)+(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.		24-87
lwarx	RT, RA, RB	Load word from EA = (RAI0) + (RB) and place in RT, (RT) \leftarrow MS(EA,4). Set the Reservation bit.		24-89
lwbrx	RT, RA, RB	Load word from EA = (RAI0) + (RB) then reverse byte order, (RT) \leftarrow MS(EA+3,1) MS(EA+2,1) MS(EA+1,1) MS(EA,1).		24-90
lwz	RT, D(RA)	Load word from EA = (RAI0) + EXTS(D) and place in RT, (RT) \leftarrow MS(EA,4).		24-91

Table B-6. Storage Reference Instructions (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
lwzu	RT, D(RA)	Load word from EA = (RAI0) + EXTS(D) and place in RT, (RT) ← MS(EA,4). Update the base address, (RA) ← EA.		24-92
lwzux	RT, RA, RB	Load word from EA = (RAI0) + (RB) and place in RT, (RT) ← MS(EA,4). Update the base address, (RA) ← EA.		24-93
lwzx	RT, RA, RB	Load word from EA = (RAI0) + (RB) and place in RT, (RT) ← MS(EA,4).		24-94
stb	RS, D(RA)	Store byte (RS) _{24:31} in memory at EA = (RAI0) + EXTS(D).		24-156
stbu	RS, D(RA)	Store byte (RS) _{24:31} in memory at EA = (RAI0) + EXTS(D). Update the base address, (RA) ← EA.		24-157
stbux	RS, RA, RB	Store byte (RS) _{24:31} in memory at EA = (RAI0) + (RB). Update the base address, (RA) ← EA.		24-158
stbx	RS, RA, RB	Store byte (RS) _{24:31} in memory at EA = (RAI0) + (RB).		24-159
sth	RS, D(RA)	Store halfword (RS) _{16:31} in memory at EA = (RAI0) + EXTS(D).		24-160
sthbrx	RS, RA, RB	Store halfword (RS) _{16:31} byte-reversed in memory at EA = (RAI0) + (RB). MS(EA, 2) ← (RS) _{24:31} (RS) _{16:23}		24-161
sthu	RS, D(RA)	Store halfword (RS) _{16:31} in memory at EA = (RAI0) + EXTS(D). Update the base address, (RA) ← EA.		24-162
sthux	RS, RA, RB	Store halfword (RS) _{16:31} in memory at EA = (RAI0) + (RB). Update the base address, (RA) ← EA.		24-163
sthx	RS, RA, RB	Store halfword (RS) _{16:31} in memory at EA = (RAI0) + (RB).		24-164
stmw	RS, D(RA)	Store consecutive words from RS through GPR(31) in memory starting at EA = (RAI0) + EXTS(D).		24-165

Table B-6. Storage Reference Instructions (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
stswi	RS, RA, NB	Store consecutive bytes in memory starting at EA=(RAI0). 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).		24-166
stswx	RS, RA, RB	Store consecutive bytes in memory starting at EA=(RAI0)+(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).		24-167
stw	RS, D(RA)	Store word (RS) in memory at EA = (RAI0) + EXTS(D).		24-169
stwbrx	RS, RA, RB	Store word (RS) byte-reversed in memory at EA = (RAI0) + (RB). $MS(EA, 4) \leftarrow (RS)_{24:31} \parallel (RS)_{16:23} \parallel (RS)_{8:15} \parallel (RS)_{0:7}$		24-170
stwcx.	RS, RA, RB	Store word (RS) in memory at EA = (RAI0) + (RB) only if the reservation bit is set. if RESERVE = 1 then MS(EA, 4) \leftarrow (RS) RESERVE \leftarrow 0 (CR[CR0]) \leftarrow ² 0 1 XER _{so} else (CR[CR0]) \leftarrow ² 0 0 XER _{so} .		24-171
stwu	RS, D(RA)	Store word (RS) in memory at EA = (RAI0) + EXTS(D). Update the base address, (RA) \leftarrow EA.		24-173
stwux	RS, RA, RB	Store word (RS) in memory at EA = (RAI0) + (RB). Update the base address, (RA) \leftarrow EA.		24-174
stwx	RS, RA, RB	Store word (RS) in memory at EA = (RAI0) + (RB).		24-175

B.6 Arithmetic and Logical Instructions

Table B-7 shows the set of arithmetic and logical instructions supported by the PPC405GP. Arithmetic operations are performed on integer or ordinal operands stored in registers. Instructions using two operands are defined in a three operand format where the operation is performed on the operands stored in two registers and the result is placed in a third register. Instructions using one operand are defined in a two operand format where the operation is performed on the operand in one register and the result is placed in another register. Several instructions also have immediate formats in which one operand is coded as part of the instruction itself. Most arithmetic and logical instructions can optionally set the condition code register based on the outcome of the instruction.

Table B-7. Arithmetic and Logical Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
add	RT, RA, RB	Add (RA) to (RB). Place result in RT.		24-6
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].		24-7
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].		24-8
adde.			CR[CR0]	
addeo			XER[SO, OV]	
addeo.			CR[CR0] XER[SO, OV]	
addi	RT, RA, IM	Add EXTS(IM) to (RAI0). Place result in RT.		24-9
addic	RT, RA, IM	Add EXTS(IM) to (RAI0). Place result in RT. Place carry-out in XER[CA].		24-10
addic.	RT, RA, IM	Add EXTS(IM) to (RAI0). Place result in RT. Place carry-out in XER[CA].	CR[CR0]	24-11
addis	RT, RA, IM	Add (IM ¹⁶ 0) to (RAI0). Place result in RT.		24-12

Table B-7. Arithmetic and Logical Instructions (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].		24-13
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].		24-14
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.		24-15
and.			CR[CR0]	
andc	RA, RS, RB	AND (RS) with \neg (RB). Place result in RA.		24-16
andc.			CR[CR0]	
andi.	RA, RS, IM	AND (RS) with ($^{16}0 \parallel$ IM). Place result in RA.	CR[CR0]	24-17
andis.	RA, RS, IM	AND (RS) with (IM \parallel $^{16}0$). Place result in RA.	CR[CR0]	24-18
cntlzw	RA, RS	Count leading zeros in RS. Place result in RA.		24-38
cntlzw.			CR[CR0]	
divw	RT, RA, RB	Divide (RA) by (RB), signed. Place result in RT.		24-59
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.		24-60
divwu.			CR[CR0]	
divwuo			XER[SO, OV]	
divwuo.			CR[CR0] XER[SO, OV]	
eqv	RA, RS, RB	Equivalence of (RS) with $\overline{(RB)}$. (RA) $\leftarrow \neg((RS) \oplus (RB))$		24-62
eqv.			CR[CR0]	
extsb	RA, RS	Extend the sign of byte (RS) _{24:31} . Place the result in RA.		24-63
extsb.			CR[CR0]	

Table B-7. Arithmetic and Logical Instructions (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
extsh	RA, RS	Extend the sign of halfword (RS) _{16:31} . Place the result in RA.		24-64
extsh.			CR[CR0]	
mulhw	RT, RA, RB	Multiply (RA) and (RB), signed. Place hi-order result in RT. $prod_{0:63} \leftarrow (RA) \times (RB)$ (signed). (RT) $\leftarrow prod_{0:31}$.		24-125
mulhw.			CR[CR0]	
mulhwu	RT, RA, RB	Multiply (RA) and (RB), unsigned. Place hi-order result in RT. $prod_{0:63} \leftarrow (RA) \times (RB)$ (unsigned). (RT) $\leftarrow prod_{0:31}$.		24-126
mulhwu.			CR[CR0]	
mulli	RT, RA, IM	Multiply (RA) and IM, signed. Place lo-order result in RT. $prod_{0:47} \leftarrow (RA) \times IM$ (signed) (RT) $\leftarrow prod_{16:47}$		24-129
mulw	RT, RA, RB	Multiply (RA) and (RB), signed. Place lo-order result in RT. $prod_{0:63} \leftarrow (RA) \times (RB)$ (signed). (RT) $\leftarrow prod_{32:63}$.		24-130
mulw.			CR[CR0]	
mulwo			XER[SO, OV]	
mulwo.			CR[CR0] XER[SO, OV]	
nand	RA, RS, RB	NAND (RS) with (RB). Place result in RA.		24-131
nand.			CR[CR0]	
neg	RT, RA	Negative (two's complement) of RA. (RT) $\leftarrow \neg(RA) + 1$		24-132
neg.			CR[CR0]	
nego			XER[SO, OV]	
nego.			CR[CR0] XER[SO, OV]	
nor	RA, RS, RB	NOR (RS) with (RB). Place result in RA.		24-139
nor.			CR[CR0]	
or	RA, RS, RB	OR (RS) with (RB). Place result in RA.		24-140
or.			CR[CR0]	
orc	RA, RS, RB	OR (RS) with $\neg(RB)$. Place result in RA.		24-141
orc.			CR[CR0]	
ori	RA, RS, IM	OR (RS) with (¹⁶ 0 IM). Place result in RA.		24-142
oris	RA, RS, IM	OR (RS) with (IM ¹⁶ 0). Place result in RA.		24-143

Table B-7. Arithmetic and Logical Instructions (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
subf	RT, RA, RB	Subtract (RA) from (RB). (RT) ← ¬(RA) + (RB) + 1.		24-176
subf.			CR[CR0]	
subfo			XER[SO, OV]	
subfo.			CR[CR0] XER[SO, OV]	
subfc	RT, RA, RB	Subtract (RA) from (RB). (RT) ← ¬(RA) + (RB) + 1. Place carry-out in XER[CA].		24-177
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) ← ¬(RA) + (RB) + XER[CA]. Place carry-out in XER[CA].		24-178
subfe.			CR[CR0]	
subfeo			XER[SO, OV]	
subfeo.			CR[CR0] XER[SO, OV]	
subfic	RT, RA, IM	Subtract (RA) from EXTS(IM). (RT) ← ¬(RA) + EXTS(IM) + 1. Place carry-out in XER[CA].		24-179
subfme	RT, RA, RB	Subtract (RA) from (-1) with carry-in. (RT) ← ¬(RA) + (-1) + XER[CA]. Place carry-out in XER[CA].		24-180
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) ← ¬(RA) + XER[CA]. Place carry-out in XER[CA].		24-181
subfze.			CR[CR0]	
subfzeo			XER[SO, OV]	
subfzeo.			CR[CR0] XER[SO, OV]	
xor	RA, RS, RB	XOR (RS) with (RB). Place result in RA.		
xor.			CR[CR0]	
xori	RA, RS, IM	XOR (RS) with (¹⁶ 0 IM). Place result in RA.		
xoris	RA, RS, IM	XOR (RS) with (IM ¹⁶ 0). Place result in RA.		

B.7 Condition Register Logical Instructions

Condition Register (CR) logical instructions allow the user to combine the results of several comparisons without incurring the overhead of conditional branching. These instructions can significantly improve code performance if multiple conditions are tested prior to making a branch decision. Table B-8 summarizes the CR logical instructions.

Table B-8. Condition Register Logical Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
crand	BT, BA, BB	AND bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT} .		24-39
crandc	BT, BA, BB	AND bit (CR_{BA}) with $\neg(CR_{BB})$. Place result in CR_{BT} .		24-40
creqv	BT, BA, BB	Equivalence of bit CR_{BA} with CR_{BB} . $CR_{BT} \leftarrow \neg(CR_{BA} \oplus CR_{BB})$		24-41
crnand	BT, BA, BB	NAND bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT} .		24-42
crnor	BT, BA, BB	NOR bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT} .		24-43
cror	BT, BA, BB	OR bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT} .		24-44
crorc	BT, BA, BB	OR bit (CR_{BA}) with $\neg(CR_{BB})$. Place result in CR_{BT} .		24-45
crxor	BT, BA, BB	XOR bit (CR_{BA}) with (CR_{BB}). Place result in CR_{BT} .		24-46
mcrf	BF, BFA	Move CR field, ($CR[CRn] \leftarrow CR[CRm]$) where $m \leftarrow BFA$ and $n \leftarrow BF$.		24-107

B.8 Branch Instructions

The architecture provides conditional and unconditional branches to any storage location. The conditional branch instructions test condition codes set previously and branch accordingly. Conditional branch instructions may decrement and test the Count Register (CTR) as part of determination of the branch condition and may save the return address in the Link Register (LR). The target address for a branch may be a displacement from the current instruction address (CIA), or may be contained in the LR or CTR, or may be an absolute address.

Table B-9. Branch Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
b	target	Branch unconditional relative. $LI \leftarrow (target - CIA)_{6:29}$ $NIA \leftarrow CIA + EXTS(LI \parallel 20)$		24-19
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 $BO_2 = 0.$	24-20
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 $BO_2 = 0.$	24-26
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 $BO_2 = 0.$	24-30
bctrl			CTR if $BO_2 = 0.$ (LR) $\leftarrow CIA + 4.$	

B.9 Comparison Instructions

Comparison instructions perform arithmetic and logical comparisons between two operands and set one of the eight condition code register fields based on the outcome of the comparison. Table B-10 shows the comparison instructions supported by the PPC405GP.

Table B-10. Comparison Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
cmp	BF, 0, RA, RB	Compare (RA) to (RB), signed. Results in CR[CRn], where $n = BF$.		24-34
cmpi	BF, 0, RA, IM	Compare (RA) to EXTS(IM), signed. Results in CR[CRn], where $n = BF$.		24-35
cmpl	BF, 0, RA, RB	Compare (RA) to (RB), unsigned. Results in CR[CRn], where $n = BF$.		24-36
cmpli	BF, 0, RA, IM	Compare (RA) to (¹⁶ 0 IM), unsigned. Results in CR[CRn], where $n = BF$.		24-37

B.10 Rotate and Shift Instructions

Rotate and shift instructions rotate and shift operands which are stored in the general purpose registers. Rotate instructions can also mask rotated operands. Table B-11 shows the PPC405GP rotate and shift instructions.

Table B-11. Rotate and Shift Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
rlwimi	RA, RS, SH, MB, ME	Rotate left word immediate, then insert according to mask.		24-146
rlwimi.		$r \leftarrow \text{ROTL}((RS), SH)$ $m \leftarrow \text{MASK}(MB, ME)$ $(RA) \leftarrow (r \wedge m) \vee ((RA) \wedge \neg m)$	CR[CR0]	
rlwinm	RA, RS, SH, MB, ME	Rotate left word immediate, then AND with mask.		24-147
rlwinm.		$r \leftarrow \text{ROTL}((RS), SH)$ $m \leftarrow \text{MASK}(MB, ME)$ $(RA) \leftarrow (r \wedge m)$	CR[CR0]	
rlwnm	RA, RS, RB, MB, ME	Rotate left word, then AND with mask.		24-150
rlwnm.		$r \leftarrow \text{ROTL}((RS), (RB)_{27:31})$ $m \leftarrow \text{MASK}(MB, ME)$ $(RA) \leftarrow (r \wedge m)$	CR[CR0]	
slw	RA, RS, RB	Shift left (RS) by (RB) _{27:31} .		24-152
slw.		$n \leftarrow (RB)_{27:31}$ $r \leftarrow \text{ROTL}((RS), n)$ if $(RB)_{26} = 0$ then $m \leftarrow \text{MASK}(0, 31 - n)$ else $m \leftarrow {}^{32}0$. $(RA) \leftarrow r \wedge m$.	CR[CR0]	
sraw	RA, RS, RB	Shift right algebraic (RS) by (RB) _{27:31} .		24-153
sraw.		$n \leftarrow (RB)_{27:31}$ $r \leftarrow \text{ROTL}((RS), 32 - n)$ 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]	
srawi	RA, RS, SH	Shift right algebraic (RS) by SH.		24-154
srawi.		$n \leftarrow SH$. $r \leftarrow \text{ROTL}((RS), 32 - n)$. $m \leftarrow \text{MASK}(n, 31)$. $s \leftarrow (RS)_0$. $(RA) \leftarrow (r \wedge m) \vee ({}^{32}s \wedge \neg m)$. $\text{XER}[CA] \leftarrow s \wedge ((r \wedge \neg m) \neq 0)$.	CR[CR0]	
srw	RA, RS, RB	Shift right (RS) by (RB) _{27:31} .		24-155
srw.		$n \leftarrow (RB)_{27:31}$. $r \leftarrow \text{ROTL}((RS), 32 - n)$. if $(RB)_{26} = 0$ then $m \leftarrow \text{MASK}(n, 31)$ else $m \leftarrow {}^{32}0$. $(RA) \leftarrow r \wedge m$.	CR[CR0]	

B.11 Cache Control Instructions

Cache control instructions allow the user to indirectly control the contents of the data and instruction caches. The user may fill, flush, invalidate and zero blocks (16-byte lines) in the data cache. The user may also invalidate congruence classes in both caches and invalidate individual lines in the instruction cache.

Table B-12. Cache Control Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
dcba	RA, RB	Speculatively establish the data cache block which contains the EA (RAI0) + (RB).		24-47
dcbf	RA, RB	Flush (store, then invalidate) the data cache block which contains the EA (RAI0) + (RB).		24-49
dcbl	RA, RB	Invalidate the data cache block which contains the EA (RAI0) + (RB).		24-50
dcbst	RA, RB	Store the data cache block which contains the EA (RAI0) + (RB).		24-51
dcbt	RA, RB	Load the data cache block which contains the EA (RAI0) + (RB).		24-52
dcbstst	RA, RB	Load the data cache block which contains the EA (RAI0) + (RB).		24-53
dcbz	RA, RB	Zero the data cache block which contains the EA (RAI0) + (RB).		24-54
dccl	RA, RB	Invalidate the data cache congruence class associated with the EA (RAI0) + (RB).		24-56
dcread	RT, RA, RB	Read either tag or data information from the data cache congruence class associated with the EA (RAI0) + (RB). Place the results in RT.		24-57
icbi	RA, RB	Invalidate the instruction cache block which contains the EA (RAI0) + (RB).		24-65
icbt	RA, RB	Load the instruction cache block which contains the EA (RAI0) + (RB).		24-66
iccci	RA, RB	Invalidate instruction cache.		24-67
icread	RA, RB	Read either tag or data information from the instruction cache congruence class associated with the EA (RAI0) + (RB). Place the results in ICDBDR.		24-68

B.12 Interrupt Control Instructions

The interrupt control instructions allow the user to move data between general purpose registers and the machine state register, return from interrupts and enable or disable maskable external interrupts. Table B-13 shows the interrupt control instruction set.

Table B-13. Interrupt Control Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
mfmsr	RT	Move from MSR to RT, (RT) ← (MSR).		24-111
mtmsr	RS	Move to MSR from RS, (MSR) ← (RS).		24-118
rfci		Return from critical interrupt (PC) ← (SRR2). (MSR) ← (SRR3).		24-144
rfi		Return from interrupt. (PC) ← (SRR0). (MSR) ← (SRR1).		24-145
wrtee	RS	Write value of RS ₁₆ to the External Enable bit (MSR[EE]).		
wrteei	E	Write value of E to the External Enable bit (MSR[EE]).		

B.13 Processor Management Instructions

The processor management instructions move data between GPRs and SPRs and DCRs in the PPC405GP; these instructions also provide traps, system calls and synchronization controls.

Table B-14. Processor Management Instructions

Mnemonic	Operands	Function	Other Registers Changed	Page
eieio		Storage synchronization. All loads and stores that precede the eieio instruction complete before any loads and stores that follow the instruction access main storage. Implemented as sync , which is more restrictive.		24-61
isync		Synchronize execution context by flushing the prefetch queue.		24-70
mcrxr	BF	Move XER[0:3] into field CR _n , where n ← BF. CR[CR _n] ← (XER[SO, OV, CA]). (XER[SO, OV, CA]) ← ³ 0.		24-108
mfcr	RT	Move from CR to RT, (RT) ← (CR).		24-109
mf dcr	RT, DCRN	Move from DCR to RT, (RT) ← (DCR(DCRN)).		24-110

Table B-14. Processor Management Instructions (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mfspr	RT, SPRN	Move from SPR to RT, (RT) ← (SPR(SPRN)).		24-112
mtcrf	FXM, RS	Move some or all of the contents of RS into CR as specified by FXM field, mask ← ⁴ (FXM ₀) ⁴ (FXM ₁) ... ⁴ (FXM ₆) ⁴ (FXM ₇). (CR) ← ((RS) ∧ mask) ∨ (CR) ∧ ¬mask.		24-116
mtdcr	DCRN, RS	Move to DCR from RS, (DCR(DCRN)) ← (RS).		24-117
mtspr	SPRN, RS	Move to SPR from RS, (SPR(SPRN)) ← (RS).		24-119
sc		System call exception is generated. (SRR1) ← (MSR) (SRR0) ← (PC) PC ← EVPR _{0:15} 0x0C00 (MSR[WE, PR, EE, PE, DR, IR]) ← 0		24-151
sync		Synchronization. All instructions that precede sync complete before any instructions that follow sync begin. When sync completes, all storage accesses initiated before sync will have completed.		24-182
tw	TO, RA, RB	Trap exception is generated if, comparing (RA) with (RB), any condition specified by TO is true.		
twi	TO, RA, IM	Trap exception is generated if, comparing (RA) with EXTS(IM), any condition specified by TO is true.		

Appendix C. Code Optimization and Instruction Timings

The code optimization guidelines in “Code Optimization Guidelines” and the information describing instruction timings in “Instruction Timings,” on page C-3 can help compiler, system, and application programmers produce high-performance code and determine accurate execution times.

C.1 Code Optimization Guidelines

The following guidelines can help to reduce program execution times.

C.1.1 Condition Register Bits for Boolean Variables

Compilers can use Condition Register (CR) bits to store boolean variables, where 0 and 1 represent False and True values, respectively. This generally improves performance, compared to using General Purpose Registers (GPRs) to store boolean variables. Most common operations on boolean variables can be accomplished using the CR Logical instructions.

C.1.2 CR Logical Instruction for Compound Branches

For example, consider the following pseudocode:

```
if (Var28 || Var29 || Var30 || Var31) branch to target
```

Var28–Var31 are boolean variables, maintained as bits in the CR[CR7] field (CR_{28:31}). The value 1 represents True; 0 represents False.

This could be coded with branches as:

```
bt      28, target
bt      29, target
bt      30, target
bt      31, target
```

Generally faster, functionally equivalent code, using CR Logical instructions, follows:

```
crcr   2, 28, 29
cror   2, 2, 30
cror   2, 2, 31
bt     2, target
```

C.1.3 Floating-Point Emulation

Two ways of handling floating-point emulation are available.

The preferred method is a call interface to subroutines in a floating-point emulation run-time library.

Alternatively, code can use the PowerPC floating point instructions. The PPC405GP, an integer processor, does not recognize these instructions and will take an illegal instruction interrupt. The interrupt handler can be written to determine the instruction opcode and execute appropriate (integer-based) library routines to provide the equivalent function.

Because this method adds interrupt context switching time to the execution time of library routines that would have been called directly by the preferred method, it is not preferred. However, this method supports code that contains PowerPC floating-point instructions.

C.1.4 Cache Usage

Code and data can be organized, based on the size and structure of the instruction and data cache arrays, to minimize cache misses.

In the cache arrays, any two addresses in which $A_{m:26}$ (the index) are the same, but which differ in $A_{0:m-1}$ (the tag), are called congruent. (This describes a two-way set-associative cache.) $A_{27:31}$ define the 32 bytes in a cache line, the smallest object that can be brought into the cache. Only two congruent lines can be in the cache simultaneously; accessing a third congruent line causes the removal from the cache of one of the two lines previously there.

Table C-1 illustrates the value of m and the index size for the various cache array sizes.

Table C-1. Cache Sizes, Tag Fields, and Lines

Array Size	Instruction Cache Array			Data Cache Array		
	m (Tag Field Bits)	n (Lines)	Index Bits	m (Tag Field Bits)	n (Lines)	Index Bits
0KB	—	—	—	—	—	—
4KB	22 (0:21)	64	21:26	21 (0:20)	64	21:26
8KB	22 (0:21)	128	20:26	20 (0:19)	128	20:26
16KB	22 (0:21)	256	19:26	19 (0:18)	256	19:26
32KB	22 (0:21)	512	18:26	18 (0:17)	512	18:26

Moving new code and data into the cache arrays occurs at the speed of external memory. Much faster execution is possible when all code and data is available in the cache. Organizing code to uniformly use $A_{m:26}$ minimizes the use of congruent addresses.

C.1.5 CR Dependencies

For CR-setting arithmetic, compare, CR-logical, and logical instructions, and the CR-setting **mcrf**, **mcrxr**, and **mtrcf** instructions, put two instructions between the CR-setting instruction and a Branch instruction that uses a bit in the CR field set by the CR-setting instruction.

C.1.6 Branch Prediction

Use the Y-bit in branch instructions to force proper branch prediction when there is a more likely prediction than the standard prediction. See "Branch Prediction" on page 3-36 for a more information about branch prediction.

C.1.7 Alignment

For speed, align all accesses on the appropriate operand-size boundary. For example, load/store word operands should be word-aligned, and so on. Hardware does not trap unaligned accesses; instead, two accesses are performed for a load or store of an unaligned operand that crosses a word boundary. Unaligned accesses that do not cross word boundaries are performed in one access.

Align branch targets that are unlikely to be hit by “fall-through” code on cache line boundaries (such as the address of functions such as **strcpy**), to minimize the number of unused instructions in cache line fills.

C.2 Instruction Timings

The following timing descriptions consider only “first order” effects of cache misses in the ICU (instruction-side) and DCU (data-side) arrays.

The timing descriptions *do not* provide complete descriptions of the performance penalty associated with cache misses; the timing descriptions do not consider bus contention between the instruction-side and the data-side, or the time associated with performing line fills or flushes. Unless specifically stated otherwise, the number of cycles apply to systems having zero-wait memory access.

C.2.1 General Rules

Instructions execute in order.

All instructions, assuming cache hits, execute in one cycle, except:

- Divide instructions execute in 35 clock cycles.
- Branches execute in one or three clock cycles, as described in “Branches.”
- MAC and multiply instructions execute in one to five cycles as described in “Multiplies.”
- Aligned load/store instructions that hit in the cache execute in one clock cycle/word. See “Alignment” for information on execution timings for unaligned load/stores.
- In isolation, a data cache control instruction takes two cycles in the processor pipeline. However, subsequent DCU accesses are stalled until a cache control instruction finishes accessing the data cache array.

Note: Note that subsequent DCU accesses do not remain stalled while transfers associated with previous data cache control instructions continue on the PLB.

C.2.2 Branches

Branch instructions are decoded in prefetch buffer 0 (PFB0) and the decode stage of the instruction pipeline. Branch targets, whether the branch is known or predicted taken, can be fetched from the PFB0 and DCD stages. Incorrectly predicted branches can be corrected from the DCD or EXE (execute) stages of the pipeline.

Branches can be known taken or known not taken, or can have address or condition dependencies. Branches having address dependencies are never predicted taken. The directions of conditional branches having no address dependencies are statically predicted.

Conditional branches may depend on the results of an instruction that is changing the CR or the CTR.

Address dependencies can occur when:

- A **bclr** instruction that is known taken, or unresolved, follows (immediately, or separated by only one instruction) a link updating instruction (**mtlr** or a branch and link).
- A **bcctr** instruction that is known taken, or unresolved, follows (immediately, or separated by only one instruction) a counter updating instruction (**mtctr** or a branch that decrements the counter).

Instruction timings for branch instructions follow:

- A branch known not taken (BKNT) executes in one clock cycle. By definition a BKNT does not have address or condition dependencies.
- A branch known taken (BKT) by definition has no condition dependencies, but can have address dependencies. A BKT without address dependencies can execute in one clock cycle if it is first decoded from the PFB0 stage, or in two clock cycles if it is first decoded in the DCD stage. A BKT having address dependencies can execute in two clock cycles if there is one instruction between the branch and the address dependency, or in three clock cycles if there are no instructions between the branch and address dependency.
- A branch predicted not taken (BPNT), which must have condition dependencies, executes in one clock cycle if the prediction is correct. If the prediction is incorrect, the branch can take two or three cycles. If there was one instruction between the branch and the instruction causing the condition dependency, the branch executes in two cycles. If there were no instructions between the branch and the instruction causing the condition dependency, the branch executes in three clock cycles.
- A branch that is correctly predicted taken (BPT), which must have condition dependencies, executes in one clock cycle, if it is first decoded from the PFB0 stage, or two clock cycles if it is first decoded in the DCD stage. If the prediction is incorrect, the branch can take two or three cycles. If there is one instruction between the branch and the instruction causing the condition dependency, the branch executes in two cycles. If there are no instructions between the branch and the instruction causing the condition dependency, the branch executes in three clock cycles.

C.2.3 Multiplies

For multiply instructions having two word operands, hardware internal to the core automatically detects smaller operand sizes (by examining sign bit extension) to reduce the number of cycles necessary to complete the multiplication.

The PPC405GP also supports multiply accumulate (MAC) instructions and multiply instructions having halfword operands.

Word and halfword multiply instructions are pipelined in the execution unit and use the same multiplication hardware. Because these instructions are pipelined in the execution stage they have latency and reissue rate cycle numbers. Under conditions to be described, a second multiply or MAC instruction can begin execution before the first multiply or MAC instruction completes. When these conditions are met, the reissue rate cycle numbers should be used; otherwise, the latency cycle numbers should be used. (A MAC or multiply instruction can follow another MAC or a multiply and still meet the conditions that support the use of the reissue rate cycle numbers.

Use reissue rate cycle numbers for multiply or MAC instructions that are followed by another multiply or MAC instruction, and do not have an operand dependency from a previous multiply or MAC instruction. However, one operand dependency is allowed for reissue rate cycle numbers. Internal forwarding logic allows the accumulate value of a first MAC instruction to be used as the accumulate value of a second MAC instruction without affecting the reissue rate.

Use latency cycle numbers for multiply or MAC instructions that are not followed by another multiply or MAC, or that have an operand dependency from a previous multiply or MAC instruction. However, accumulate-only dependencies between adjacent MAC instructions use reissue rate cycle numbers.

An operand dependency exists when a second multiply or MAC instruction depends on the result of a first multiply or MAC instruction.

Table C-2 summarizes the multiply and MAC instruction timings. In the table, the syntax “[o]” indicates that the instruction has an “o” form that updates XER[SO,OV], and a “non-o” form. The syntax “[.]” indicates that the instruction has a “record” form that updates CR[CR0], and a “non-record” form.

Table C-2. Multiply and MAC Instruction Timing

Operation	Reissue Rate Cycles	Latency Cycles
MAC		
MAC and negative MAC instructions	1	2
Halfword × Halfword		
mullhw[.], mullhwu[.], mulhhw[.], mulhhwu[.], mulchw[.], mulchwu[.]	1	2
mullf[.], mullw[o][.], mulhw[.], mulhwu[.]	2	3
Halfword × Word		
mullf[.], mullw[o][.], mulhw[.], mulhwu[.]	2	3
Word × Word		
mullw[o][.], mulhw[.], mulhwu[.]	4	5

C.2.4 Scalar Load Instructions

Generally, the PPC405GP executes cachable load instructions that hit in the data cache array or fill buffer, or noncachable load instructions that hit in the fill buffer (when enabled), in one cycle. However, the pipelined nature of load instructions can even cause loads that hit in the cache or line buffer to appear to take extra cycles under some conditions.

If a load is followed by an instruction that uses the load target as an operand, a load-use dependency exists. When the load target is returned, it is forwarded to the operand register of the “using” instruction. This forwarding results in an additional cycle of latency to a load immediately followed by a “using” instruction, causing the load to appear to execute in two cycles.

Because the PPC405GP can execute instructions that follow load misses if no load-use dependency exists, the load and the “using” instruction should be separated by two “non-using” instructions when possible. If only one instruction can be placed between the load and the “using” instruction, the load appears to execute in two cycles.

C.2.5 Scalar Store Instructions

Cachable stores that miss in the DCU, and noncachable stores, are queued in the data cache so that the store appears to execute in a single cycle if operand-aligned. Under certain conditions, the DCU can pipeline up to three store instructions. (See Chapter 4, “Cache Operations,” for more information.) **stwcx.** instructions that do not cause alignment errors execute in two cycles.

C.2.6 Alignment in Scalar Load and Store Instructions

The PPC405GP requires an extra cycle to execute scalar loads and stores having unaligned big or little endian data (except for **lwarx** and **stwcx.**, which require word-aligned operands). If the target data is not operand aligned, and the sum of the least two significant bits of the effective address (EA)

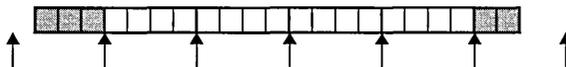
and the byte count is greater than four, the PPC405GP decomposes a load or store scalar into two load or store operations. That is, the PPC405GP never presents the DCU with a request for a transfer that crosses a word boundary. For example, a `lwz` with an EA of 0b11 causes the PPC405GP to decompose the `lwz` into two load operations. The first load operation is for a byte at the starting effective address; the second load operation is for three bytes, starting at the next word address.

C.2.7 String and Multiple Instructions

Calculating execution times for string and multiple instructions (`lmw` and `stmw`) instructions requires an understanding of data alignment, and of the behavior of the string instructions with respect to alignment.

In the following example, the string contains 21 bytes. The first three bytes do not begin on a word boundary, and the final two bytes do not end on a word boundary. The PPC405GP handles any unaligned leading bytes as a special case, then moves as many bytes as aligned words as possible, and finally handles any unaligned trailing bytes as a special case.

In the following example, arrows indicate word boundaries (the address is an exact multiple of four); shaded boxes represent unaligned bytes.



The execution time of the string instruction is the sum of the:

1. Cycles required to handle unaligned leading bytes; if any, add one clock cycle.
In the example, there are unaligned leading bytes; this transfer adds one clock cycle.
2. Cycles required to handle the number of word-aligned transfers required. Assuming data cache hits, each word-aligned transfer requires one clock cycle.
In the example, there are four aligned words; this transfer requires four clock cycles.
3. Cycles required to handle unaligned trailing bytes; if any, add one clock cycle.
In the example, there are unaligned trailing bytes; this transfer adds one clock cycle.

A string instruction operating on the example 21-byte string requires six clock cycles.

C.2.8 Loads and Store Misses

Cachable stores that miss in the DCU, and noncachable stores, are queued internally in the DCU so that the store instruction appears to execute in one cycle. Under certain conditions, the DCU can pipeline up to three store instructions. (See the Chapter 4, "Cache Operations," for more information.)

Because the PPC405GP can execute instructions that follow load misses if no load-use dependency exists, the load and the "using" instruction should be separated by "non-using" instructions whenever possible. The number of load miss penalty cycles incurred by a load that misses in the DCU or DCU fill buffer is reduced by one cycle for every non-use instruction following the load. When the number of non-use instructions following the load is equal to or greater than the number of cycles that it takes to obtain the load data, the load instruction appears to execute in a single cycle. The number of cycles that it takes to obtain load data when it misses in the data cache and fill buffer depends on whether operand forwarding is enabled or disabled and the system memory timing.

C.2.9 Instruction Cache Misses

Refer to “Instruction Processing” on page 3-33 for detailed information about the instruction queue and instruction fetching. Table C-4 illustrates instruction cache penalties for cachable and noncachable fetches that miss in the ICU array and fill buffer.

Table C-4. Instruction Cache Miss Penalties

Type of ICU Request	Miss Penalty Cycles
Sequential	3
Branch Taken from DCD	5
Branch Taken from PFB0	4

Table C-4 assumes that:

- The PPC405GP and processor local bus (PLB) run at the same frequency
- The PLB returns an address acknowledge during the first cycle in which the DCU asserts the PLB request
- The target instruction is returned in the cycle following the address acknowledge cycle

The penalty cycles shown for sequential ICU requests assume that the DCD stage and pre-fetch queue are filled with single-cycle nonbranching instructions or BKNT branch instructions. The penalty cycles for the remaining two rows are for taken branches from DCD and PFB0 respectively.

Index

A

- AA field
 - conditional branches 3-35
 - unconditional branches 3-34
- access protection
 - cache instructions 6-15
 - string instructions 6-16
 - virtual mode 6-12
- add 24-6
- add. 24-6
- addc 24-7
- addc. 24-7
- addco 24-7
- addco. 24-7
- addc. 24-8
- addc. 24-8
- addc. 24-8
- addi 24-9
- addic 24-10
- addic. 24-11
- addis 24-12
- addme 24-13
- addme. 24-13
- addmeo 24-13
- addmeo. 24-13
- addo 24-6
- addo. 24-6
- address map
 - illustrated 3-2
- address translation
 - illustrated 6-2
 - MMU 6-1
 - relationship between TLBs, illustrated 6-8
- addressing modes 1-11
- addze 24-14
- addze. 24-14
- addzeo 24-14
- addzeo. 24-14
- alignment
 - for cache control instructions 3-27
 - for storage reference instructions 3-27
 - of data types 3-26
- alignment interrupts
 - causes of 3-28
 - register settings 10-40
 - summary 10-39
- and 24-15
- and. 24-15
- andc 24-16
- andc. 24-16
- andi. 24-17
- andis. 24-18
- architecture, PowerPC 1-4
- arithmetic compares 3-13
- arithmetic instructions 3-49

asynchronous interrupts 10-22

B

- b 24-19
- ba 24-19
- bc 24-20
- bca 24-20
- bcctr 24-26
- bcctrl 24-26
- bcl 24-20
- bcla 24-20
- bclr 24-30
- bclrl 24-30
- bctr 24-27
- bctrl 24-27
- bdnz 24-21
- bdnza 24-21
- bdnzf 24-21
- bdnzfa 24-21
- bdnzfl 24-21
- bdnzfla 24-21
- bdnzflr 24-31
- bdnzflrl 24-31
- bdnzl 24-21
- bdnzla 24-21
- bdnzlr 24-31
- bdnzlrl 24-31
- bdnzt 24-21
- bdnzta 24-21
- bdnztl 24-21
- bdnztla 24-21
- bdnztlr 24-31
- bdnztlrl 24-31
- bdz 24-21
- bdza 24-21
- bdzf 24-22
- bdzfa 24-22
- bdzfl 24-22
- bdzfla 24-22
- bdzflr 24-31
- bdzflrl 24-31
- bdzl 24-21
- bdzla 24-21
- bdzlr 24-31
- bdzlr 24-31
- bdzlr 24-31
- bdzt 24-22
- bdzta 24-22
- bdztl 24-22
- bdztle 24-22
- bdztlr 24-31
- bdztlrl 24-31
- beq 24-22
- beqa 24-22
- beqctr 24-27
- beqctrl 24-27
- beql 24-22
- beqlr 24-31

beqrl 24-31
 bf 24-22
 bfa 24-22
 bfctr 24-27
 bfctrl 24-27
 bfl 24-22
 bfla 24-22
 bflr 24-32
 bflrl 24-32
 bge 24-23
 bgea 24-23
 bgectrl 24-27
 bgel 24-23
 bgela 24-23
 bgehr 24-32
 bgehr 24-32
 bgrctr 24-27
 bgt 24-23
 bgta 24-23
 bgtctr 24-27
 bgtctrl 24-27
 bgtl 24-23
 bgtla 24-23
 bgtlr 24-32
 bgtlrl 24-32
 BI field
 conditional branches 3-35
 big endian
 alignment 3-27
 defined 3-28
 mapping 3-29
 storage regions
 byte-reverse instructions 3-32- 3-33
 bl 24-19
 bla 24-19
 ble 24-23
 blea 24-23
 blectr 24-27
 blectrl 24-27
 blel 24-23
 blela 24-23
 blehr 24-32
 blehr 24-32
 blr 24-30
 blrl 24-30
 blt 24-23
 bita 24-23
 bitctr 24-27
 bitctrl 24-27
 bitl 24-23
 bitla 24-23
 bitlr 24-32
 bitlrl 24-32
 bne 24-24
 bnea 24-24
 bnectr 24-28
 bnectrl 24-28
 bnel 24-24
 bnela 24-24
 bnelr 24-32
 bnelrl 24-32
 bng 24-24
 bnga 24-24
 bngctr 24-28
 bngctrl 24-28
 bngl 24-24
 bngla 24-24
 bnglr 24-32
 bnglrl 24-32
 bnl 24-24
 bnla 24-24
 bnlctr 24-28
 bnlctrl 24-28
 bnll 24-24
 bnlla 24-24
 bnllr 24-33
 bnllrl 24-33
 bns 24-24
 bnsa 24-24
 bnsctr 24-28
 bnsctrl 24-28
 bnsl 24-24
 bnsla 24-24
 bnslr 24-33
 bnsrl 24-33
 bnu 24-25
 bnua 24-25
 bnuctr 24-28
 bnuctrl 24-28
 bnul 24-25
 bnula 24-25
 bnulr 24-33
 bnulrl 24-33
 BO field
 conditional branches 3-35- 3-36
 branch instructions 3-51
 branch prediction 3-36, A-1, B-9
 controlling through mnemonics 3-37
 branching control
 AA field on conditional branches 3-35
 AA field on unconditional branches 3-34
 BI field on conditional branches 3-35
 BO field on conditional branches 3-35- 3-36
 branch prediction 3-36
 bso 24-25
 bsoa 24-25
 bsoctr 24-28
 bsoctrl 24-28
 bsol 24-25
 bsola 24-25
 bsolr 24-33
 bsolrl 24-33
 bt 24-25
 bta 24-25
 btctr 24-28
 btctrl 24-28
 btl 24-25
 btla 24-25
 btlr 24-33
 btlrl 24-33
 bun 24-25
 buna 24-25

- bunctr 24-29
- bunctrl 24-29
- bunl 24-25
- bunla 24-25
- bunlr 24-33
- bunlrl 24-33
- bus timeout error 16-11
- byte ordering
 - big endian, defined 3-28
 - little endian
 - defined 3-28
 - supported 3-30
 - overview 3-28
- byte reversal
 - during load/store access 3-31
- byte-reverse instructions
 - augmented by endian (E) storage attribute 3-33
 - compare to endian (E) storage attribute 3-32

C

- cache
 - instructions
 - DAC debug events 12-20
 - cache block, defined 4-9
 - cache control instructions
 - access protection 6-15
 - causing data storage interrupts 6-15
 - cache line
 - dirty, defined 4-16
 - See also* cache block
 - cache line fills
 - DCU 4-6
 - defined 4-6
 - types 4-3
 - cache organization
 - DCU 4-5
 - ICU 4-2
 - caches. *See* ICU;DCU
 - caching inhibited (I) storage attribute
 - for data accesses, controlled by DCCR 6-19
 - for instruction fetches, controlled by ICCR 6-19
 - virtual mode 6-5
- CCR0 25-15
- character mode
 - NS16450 21-1
- clrslwi 24-147
- clrslwi. 24-147
- clrlwi 24-147
- clrlwi. 24-147
- clrrwi 24-148
- clrrwi. 24-148
- cmp 24-34
- cmpi 24-35
- cmpl 24-36
- cmpli 24-37
- cmplw 24-36
- cmplwi 24-37
- cmpw 24-34
- cmpwi 24-35
- cntlzw 24-38
- cntlzw. 24-38
- compare instructions

- arithmetic 3-13
 - in core, listed 3-50
 - effect on CR fields 3-13
 - logical 3-13
- Condition Register. *See* CR
- conditional branches
 - AA field 3-35
 - BI field 3-35
 - BO field 3-35- 3-36
 - mnemonics used to control prediction 3-37
- context synchronization
 - defined 3-44
 - for ITLB 6-8
 - limitations 3-44
- context, defined 3-44
- conventions
 - notational xlviii
- Count Register. *See* CTR
- CPC0_CR1 25-20
- CPC0_ER 25-21
- CPC0_FR 25-22
- CPC0_JTAGID 25-23
- CPC0_PLLMR 25-24
- CPC0_PSR 25-26
- CPC0_SR 25-28
- CR 25-29
- CR (Condition Register)
 - arithmetic and logical instructions 3-49
 - compare instructions 3-13, 3-50
 - CRO field 3-14
 - logical instructions 3-50
 - setting fields 3-12
 - summarized 1-11
- crand 24-39
- crandc 24-40
- crclr 24-46
- creqv 24-41
- critical input interrupts 10-34
 - register settings 10-35
- critical interrupts 10-23
 - defined 10-26
 - processing 10-27
- crmove 24-44
- crnand 24-42
- crnor 24-43
- crnot 24-43
- cror 24-44
- crorc 24-45
- crset 24-41
- crxor 24-46
- CTR 15-6, 15-14, 15-15, 15-16, 15-17, 15-20, 25-30, 25-209, 25-210, 25-218, 25-219, 25-220, 25-221
- CTR (Count Register)
 - branch instructions 3-51
 - functions 3-7
 - testing by branch instructions 3-35

D

- DAC1 12-14
- DAC1-DAC2 12-14, 14-7, 25-31, 25-48
- Data Address Compare Register (DAC1) 12-14

- data alignment
 - in little endian storage 3-32
 - overview 3-26
- Data Cache Cachability Register. *See* DCCR
- Data Cache Cachability Register. *See* DCCR
- data cache unit. *See* DCU
- Data Cache Write-through Register. *See* DCWR
- Data Cache Write-through Register. *See* DCWR
- Data Exception Address Register. *See* DEAR
- data machine check interrupts
 - register settings 10-36
- data storage interrupts
 - caused by cache control instructions 6-15
 - causes 10-36
 - described 6-9
 - programming note 10-37
 - register settings 10-37
- data TLB. *See* DTLB
- data types
 - illustrated 3-26
 - summarized 1-10
- DBCR 12-9
- DBCR0 25-32, 25-34
- DBCR0 (Debug Control Register 0)
 - resets 8-2
- DBSR 12-12, 25-36
- dcba
 - does not cause interrupts 6-15
 - functions 4-10
- dcbf 24-49
 - data storage interrupts 6-16
 - functions 4-10
- dcbi 24-50
 - data storage interrupts 6-15
 - functions 4-10
- dcbst 24-51
 - data storage interrupts 6-16
 - functions 4-10
- dcbt 24-52
 - data storage interrupts 6-16
 - functions 4-10
- dcbst
 - functions 4-10
- dcbz 24-54
 - data storage interrupts 6-15
 - functions 4-11
- dccci 24-56
 - data storage interrupts 6-16
 - functions 4-11
 - when use not recommended 6-16
- DCCR 25-38
- DCCR (Data Cache Cachability Register)
 - controlling cachability 4-8
 - controlling the caching inhibited (I) storage attribute 6-19
- DCP0_ADDR0–DCP0_ADDR1 25-40
- DCP0_ADDR0–DCP0_ADDR3 14-5, 25-40
- DCP0_CFG 14-6, 25-41
- DCP0_CFGADDR 25-42
- DCP0_CFGADDR (Decompression Controller Address Register)
 - accessing 3-21
- DCP0_CFGDATA 25-43
- DCP0_CFGDATA (Decompression Controller Data Register)
 - accessing 3-21
- DCP0_ESR 14-8, 25-44
- DCP0_ID 14-6, 25-46
- DCP0_ITOR0–DCP0_ITOR3 14-4, 25-47
- DCP0_MEMBEAR 14-7, 25-48
- DCP0_PLBBEAR 14-7, 25-49
- DCP0_RAM0–DCP0_RAM3FF 25-50
- DCP0_VER 14-7, 25-51
- dcread 24-57
 - controlled by CCR0 4-11
 - as debugging tool 4-15
 - functions 4-11
- DCRs (device control registers)
 - indirectly accessed 3-19
 - instructions for reading 3-43
 - summary 1-11
 - uses for 3-16
- DCU (data cache unit)
 - cachability control 4-8
 - cache line fills 4-6
 - coherency 4-9
 - debugging 4-15
 - features 4-1
 - instructions 4-10
 - load commands, accepting 4-17
 - load strategies 4-7
 - organization 4-5
 - performance 4-16
 - pipeline stalls 4-16
 - priority changes 4-17
 - priority signal 4-17
 - sequential caching 4-18
 - simultaneous cache operations 4-18
 - store commands 4-17
 - tag information in GPRs 4-16
 - write strategies 4-7
- DCWR 25-52
- DCWR (Data Cache Write-through Register)
 - controlling write strategies 4-7
 - write-through policy 6-18
- DEAR 25-54
- DEAR (Data Exception Address Register)
 - illustrated 10-34
- Debug Control Register (DBCR) 12-9
- Debug Control Register 0. *See* DBCR0
- debug interrupts
 - register settings 10-44
- debugging
 - boundary scan chain 12-3
 - DCU 4-15
 - debug events 12-16
 - debug interfaces 12-1
 - JTAG test access port 12-1
 - trace status port 12-5
 - development tools 12-1
 - ICDBDR 4-14
 - modes 12-6

- external 12-6
- internal 12-6
- real-time trace 12-7
- processor control 12-8
- processor status 12-8
- decompression controller
 - access procedures, overview 3-19
 - indirect access of DCRs 3-21
 - offsets for DCRs 3-21
- Decompression Controller Address Register. *See* DCP0_CFGADDR
- Decompression Controller Data Register. *See* DCP0_CFGDATA
- Device Control Registers. *See* DCRs
- device-paced transfers 16-11
 - bus timeout error 16-11
- dirty cache line, defined 4-16
- divw 24-59
- divw. 24-59
- divwo 24-59
- divwo. 24-59
- divwu 24-60
- divwu. 24-60
- divwuo 24-60
- divwuo. 24-60
- DMA
 - memory-to-memory mode transfers
 - initiated by software 18-21
- DMA operations
 - arbitration transfer priorities 18-13
 - data parity 18-14
 - errors 18-14
 - interrupts 18-15
- DMA0_CR0-DMA0_CR3 18-8, 25-55
- DMA0_CT0-DMA0_CT3 18-11, 25-58
- DMA0_DA0-DMA0_DA3 18-11, 25-59
- DMA0_POL 25-60
- DMA0_SA0-DMA0_SA3 18-10
- DMA0_SA0-DMA0_SA3 25-62
- DMA0_SG0-DMA0_SG3 18-12
- DMA0_SG0-DMA0_SG3 25-63
- DMA0_SGC 18-13, 25-64
- DMA0_SLP 25-65
- DMA0_SR 18-7, 25-66
- DTLB (data translation lookaside buffer)
 - accesses 6-7
 - miss interrupts 6-10, 10-43
 - summary 6-7

E

- EA (effective address)
 - forming 3-27
 - translation to RA, illustrated 6-2
 - when non-cachable 4-8
- EAs (effective addresses)
 - indexing the cache array 4-4
- EBC (external bus controller)
 - DCRs
 - access procedures, overview 3-19
 - indirect access 3-20
 - offsets 3-20
- EBC0_BEAR 25-68
- EBC0_BESR0 25-69
- EBC0_BESR1 25-71
- EBC0_BnAP 25-73
- EBC0_BnCR 25-75
- EBC0_CFG 25-76
- EBC0_CFGADDR 25-78
- EBC0_CFGADDR (Peripheral Controller Address Register)
 - accessing 3-20
- EBC0_CFGDATA 25-79
- EBC0_CFGDATA (Peripheral Controller Data Register)
 - accessing 3-20
- effective address. *See* EA
- effective addresses. *See* EAs
- eieio 24-61
 - storage synchronization 3-47
- EMAC_OIAHR 25-81
- EMAC0_GAHT1-EMAC0_GAHT4 25-80
- EMAC0_IAHT1-EMAC0_IAHT4 25-82
- EMAC0_TMR0 25-83
- EMAC0_TMR1 25-84
- EMAC0_TRTR 25-85
- EMAC0_VTCI 25-86
- EMAC0_VTPID 25-87
- embedded controllers
 - instruction set 3-48
- endian (E) storage attribute
 - and byte-reverse load/store instructions 3-33
 - controlled by SLER 6-19
 - and little endian 3-30
 - when controlled by TLB 6-6
- eqv 24-62
- eqv. 24-62
- ESR 25-88
- ESR (Exception Status Register)
 - usage for program interrupts 10-40
- ESR (Exception Syndrome Register)
 - clearing privileged exceptions 3-42
 - illustrated 10-31
 - MCI bit, behavior of 10-33
- EVPR 25-89
- EVPR (Exception Vector Prefix Register)
 - illustrated 10-31
- Exception Syndrome Register. *See* ESR
- Exception Vector Prefix Register. *See* EVPR
- exceptions
 - defined 10-22
 - handling, and MSR bits 3-41
 - privileged, clearing 3-42
 - registers during debug exceptions 10-45
- exceptions. *See also* interrupts
- execution mode
 - controlling by MSR 3-41
- execution synchronization, defined 3-46
- extended memonics
 - beqlr 24-31
- extended menmonics
 - blectrl 24-27
 - bnlctrl 24-28
- extended mnemonic
 - bnqla 24-24

extended mnemonics

alphabetical B-9

bctr 24-27
bctrl 24-27
bdnz 24-21
bdnza 24-21
bdnzf 24-21
bdnzfa 24-21
bdnzfkr 24-31
bdnzfl 24-21
bdnzfla 24-21
bdnzflrl 24-31
bdnzl 24-21
bdnzla 24-21
bdnzlr 24-31
bdnzlrl 24-31
bdnzt 24-21
bdnzta 24-21
bdnztl 24-21
bdnztla 24-21
bdnztlr 24-31
bdnztlrl 24-31
bdz 24-21
bdza 24-21
bdzf 24-22
bdzfa 24-22
bdzfl 24-22
bdzfla 24-22
bdzflr 24-31
bdzflrl 24-31
bdzl 24-21
bdzla 24-21
bdzlr 24-31
bdzlrl 24-31
bdzt 24-22
bdzta 24-22
bdztl 24-22
bdztle 24-22
bdztlr 24-31
bdztlrl 24-31
beq 24-22
beqa 24-22
beqctr 24-27
beqctrl 24-27
beql 24-22
beqlrl 24-31
bf 24-22
bfa 24-22
bfctr 24-27
bfctrl 24-27
bfl 24-22
bfla 24-22
bflr 24-32
bflrl 24-32
bge 24-23
bgea 24-23
bgectr 24-27
bgectrl 24-27
bgel 24-23
bgela 24-23
bgelr 24-32

bgelrl 24-32
bgt 24-23
bgta 24-23
bgctr 24-27
bgctrl 24-27
bgtl 24-23
bgta 24-23
bgtlr 24-32
bgtlrl 24-32
ble 24-23
blea 24-23
blectr 24-27
blel 24-23
blela 24-23
blelr 24-32
blelrl 24-32
blr 24-30
blrl 24-30
blt 24-23
blta 24-23
bltctr 24-27
bltctrl 24-27
bltl 24-23
bltla 24-23
bltlr 24-32
bltlrl 24-32
bne 24-24
bnea 24-24
bnectrl 24-28
bnel 24-24
bnela 24-24
bnelr 24-32
bnelrl 24-32
bng 24-24
bnga 24-24
bngctr 24-28
bngctrl 24-28
bngl 24-24
bnglr 24-32
bnglrl 24-32
bnl 24-24
bnla 24-24
bnlctr 24-28
bnll 24-24
bnlla 24-24
bnllr 24-33
bnllrl 24-33
bns 24-24
bnsa 24-24
bnsctr 24-28
bnsctrl 24-28
bnsll 24-24
bnslla 24-24
bnsllr 24-33
bnsllrl 24-33
bnu 24-25
bnua 24-25
bnuctr 24-28
bnuctrl 24-28
bnul 24-25
bnula 24-25

bnulr 24-33
 bnulrl 24-33
 bsalr 24-33
 bso 24-25
 bsoa 24-25
 bsoctr 24-28
 bsoctrl 24-28
 bsol 24-25
 bsola 24-25
 bsolrl 24-33
 bt 24-25
 bta 24-25
 bctr 24-28
 btctrl 24-28
 bit 24-25
 bitla 24-25
 bitr 24-33
 bitrl 24-33
 bun 24-25
 buna 24-25
 bunctr 24-29
 bunctrl 24-29
 bunl 24-25
 bunla 24-25
 bunlr 24-33
 bunlrl 24-33
 cirslwi 24-147
 cirslwi. 24-147
 cirliwi 24-147
 cirliwi. 24-147
 clrrwi 24-148
 clrrwi. 24-148
 cmplw 24-36
 cmplwi 24-37
 cmpw 24-34
 cmpwi 24-35
 crclr 24-46
 crmove 24-44
 crnot 24-43
 crset 24-41
 explained 3-53
 extlwi 24-148
 extlwi. 24-148
 extrwi 24-148
 extrwi. 24-148
 for addi 24-9
 for addic 24-10
 for addic. 24-11, 24-115
 for addis 24-12
 for bc, bca, bcl, bcla 24-21
 for bcctr, bcctrl 24-27
 for bclr, bclrl 24-30
 for cmp 24-34
 for cmpi 24-35
 for cmpl 24-36
 for cmpli 24-37
 for creqv 24-41
 for crnor 24-43
 for cror 24-44
 for crxor 24-46
 for mfspr 24-113
 for mtrcf 24-116
 for mtspr 24-120
 for nor, nor. 24-139
 for or, or. 24-140
 for ori 24-142
 for rlwimi, rlwimi. 24-146
 for rlwinm, rlwinm. 24-147
 for rlwnm, rlwnm. 24-150
 for subf, subf., subfo, subfo. 24-176
 for subfc, subfc., subfco, subfco. 24-177
 for tlbre 24-185
 for tw 24-191
 for twi 24-194
 inslwi 24-146
 inslwi. 24-146
 insrwi 24-146
 insrwi. 24-146
 li 24-9
 lis 24-12
 mftb 24-115
 mftbu 24-115
 mr 24-140
 mr. 24-140
 mtr 24-116
 nop 24-142
 not 24-139
 not. 24-139
 rotlw 24-150
 rotlw. 24-150
 rotlwi 24-148
 rotlwi. 24-148
 rotrwi 24-148
 rotrwi. 24-148
 slwi 24-148
 slwi. 24-148
 srwi 24-149
 srwi. 24-149
 sub 24-176
 sub. 24-176
 subc 24-177
 subc. 24-177
 subco 24-177
 subco. 24-177
 subi 24-9
 subic 24-10
 subic. 24-11
 subis 24-12
 subo 24-176
 subo. 24-176
 tblrehi 24-185
 tblrelo 24-185
 tblwehi 24-189
 tblwelo 24-189
 trap 24-191
 tweq 24-191
 tweqi 24-194
 twge 24-191
 twgei 24-194
 twgle 24-191
 twgt 24-191
 twgti 24-194

- twle 24-191
- twlei 24-194
- twigei 24-194
- twlgt 24-191
- twlgti 24-194
- twlle 24-192
- twllel 24-194
- twllt 24-192
- twllti 24-194
- twlng 24-192
- twlngi 24-194
- twlnl 24-192
- twlnli 24-195
- twlt 24-192
- twlti 24-195
- twne 24-192
- twnei 24-195
- twng 24-192
- twngi 24-195
- twnl 24-192
- twnli 24-195
- extended mnemonics for
 - tbre 24-189
- external bus controller. *See* EBC
- external bus master 16-17
 - arbitration 16-17
 - interface 16-17
- external interrupts
 - programming note 10-38
 - register settings 10-39
- extlwi 24-148
- extlwi. 24-148
- extrwi 24-148
- extrwi. 24-148
- extsb 24-63
- extsb. 24-63

F

- features
 - DCU 4-1
 - ICU 4-1
- fetcher, improved performance of 4-3
- FIFO 21-15
- FIFO control register 21-7
- FIFO operation
 - interrupt mode 21-15
 - polled mode 21-16
- FIT 11-5
- FIT (fixed interval timer)
 - interrupts, causes 10-42
 - interrupts, register settings 10-42
- fixed interval timer 11-5
- fixed interval timer. *See* FIT
- Fixed Point Exception Register. *See* XER

G

- general interrupt handling registers, listed 10-27
- general purpose registers. *See* GPRs
- GPIO0_IR 25-90
- GPIO0_IRO 23-1
- GPIO0_ODR 25-91
- GPIO0_OR 25-92

- GPIO0_TCR 25-93
- GPR0-GPR31 25-94
- GPRs (general purpose registers)
 - interrupt control instructions 3-52
 - overview 3-6
 - summary 1-10
- guarded (G) storage attribute
 - controlled by SGR 6-19
 - preventing speculative accesses 3-37
 - virtual mode 6-6

I

- I storage attribute. *See* caching inhibited (I) storage attribute
- IAC1-IAC4 25-67, 25-95
- IAC1-IAC4 12-14
- icbi 24-65
 - data storage interrupts 6-16
 - function 4-9
- icbt 24-66
 - data storage interrupts 6-16
 - function 4-9
- iccci 24-67
 - function 4-9
 - when use not recommended 6-16
- ICCR 25-96
- ICCR (Instruction Cache Cachability Register)
 - controlling the I storage attribute 6-19
 - controls cachability 4-4
- ICDBCR (Instruction Cache Debug Control Register)
 - programming note 4-15
- ICDBDR 25-98
- ICDBDR (Instruction Cache Debug Data Register)
 - illustrated 4-14, 25-98
- icread 24-68
 - controlled by CCR0 4-11
 - function 4-9
 - programming note 4-15
- ICU (instruction cache unit)
 - cachability control 4-4
 - cache line fills 4-3
 - coherency 4-5
 - features 4-1
 - instruction flow, illustrated 4-3
 - instructions 4-9
 - least-recently-used (LRU) policy 4-2
 - organization 4-2
 - synchronization 4-5
 - synonyms 4-4
- IIC 22-1
- IIC0_CLKDIV 25-99
- IIC0_CNTL 25-100
- IIC0_DIRECTCNTL 25-101
- IIC0_EXTSTS 25-102
- IIC0_HMADR 25-104
- IIC0_HSADR 25-105
- IIC0_INTRMSK 25-106
- IIC0_LMADR 25-107
- IIC0_LSADR 25-108
- IIC0_MDBUF 25-109
- IIC0_MDCNTL 25-110
- IIC0_SDBUF 25-111

IIC0_STS 25-112
 IIC0_XFRCNT 25-113
 IIC0_XTCNTLSS 25-114
 imprecise interrupts 10-22
 initialization
 code example 8-16
 of processor 8-14
 requirements 8-15
 sequence 8-15
 inslwi 24-146
 inslwi. 24-146
 insrwi 24-146
 insrwi. 24-146
 instruction
 add 24-6
 add. 24-6
 addc 24-7
 addc. 24-7
 addco 24-7
 addco. 24-7
 adde 24-8
 adde. 24-8
 addeo 24-8
 addeo. 24-8
 addi 24-9
 addic 24-10
 addic. 24-11
 addis 24-12
 addme 24-13
 addme. 24-13
 addmeo 24-13
 addmeo. 24-13
 addo 24-6
 addo. 24-6
 addze 24-14
 addze. 24-14
 addzeo 24-14
 addzeo. 24-14
 and 24-15
 and. 24-15
 andc 24-16
 andc. 24-16
 andi. 24-17
 andis. 24-18
 b 24-19
 ba 24-19
 bc 24-20
 bca 24-20
 bcctr 24-26
 bcctrl 24-26
 bcl 24-20
 bcla 24-20
 bclr 24-30
 bcrlr 24-30
 bl 24-19
 bla 24-19
 cmp 24-34
 cmpi 24-35
 cmpi. 24-36
 cmpli 24-37
 cntlzw 24-38
 cntlzw. 24-38
 crand 24-39
 crandc 24-40
 creqv 24-41
 crnand 24-42
 crnor 24-43
 cror 24-44
 crorc 24-45
 crxor 24-46
 dcbf 24-49
 dcbi 24-50
 dcbst 24-51
 dcbt 24-52
 dcbz 24-54
 dccci 24-56
 dcread 24-57
 diww 24-59
 divw. 24-59
 divwo 24-59
 divwo. 24-59
 divwu 24-60
 divwu. 24-60
 divwuo 24-60
 divwuo. 24-60
 eieio 24-61
 eqv 24-62
 eqv. 24-62
 extsb 24-63
 extsb. 24-63
 icbi 24-65
 icbt 24-66
 iccci 24-67
 icread 24-68
 isync 24-70
 lbz 24-71
 lbzu 24-72
 lbzx 24-74
 lha 24-75
 lhau 24-76
 lhax 24-78
 lhbrx 24-79
 lhz 24-80
 lhz. 24-81
 lhzux 24-82
 lhzx 24-83
 lmw 24-84
 lswi 24-85
 lswx 24-87
 lwarx 24-89
 lwz 24-91
 lwz. 24-92
 lwzux 24-93
 lwzx 24-94
 macchw 24-95
 macchws 24-96
 macchwsu 24-97
 macchwu 24-98
 machhw 24-99
 machhwsu 24-101
 machhwu 24-102
 maclhw 24-103

maclhws 24-104, 24-138
 maclhwu 24-106
 mcrf 24-107
 mcrr 24-108
 mfcrr 24-109
 mfdcr 24-110
 mfmsr 24-111
 mfspr 24-112
 mtrf 24-116
 mtdcr 24-117
 mtspr 24-119
 mulchw 24-121
 mulchwu 24-122
 mulhww 24-123
 mulhwwu 24-124
 mulhwwu 24-126
 mulhwwu. 24-126
 mullhw 24-127
 mullhww 24-128
 mulli 24-129
 mullw 24-130
 mullw. 24-130
 mullwo 24-130
 mullwo. 24-130
 nand 24-131
 nand. 24-131
 neg 24-132
 neg. 24-132
 nego 24-132
 nego. 24-132
 nmacchw 24-133
 nmacchws 24-134
 nmachhw 24-135
 nmachhws 24-136
 nmaclhw 24-137
 nmaclhws 24-138
 nor 24-139
 nor. 24-139
 or 24-140
 or. 24-140
 orc 24-141
 orc. 24-141
 ori 24-142
 oris 24-143
 rfc1 24-144
 rfi 24-145
 rlwimi 24-146
 rlwimi. 24-146
 rlwinm 24-147
 rlwinm. 24-147
 rlwnm 24-150
 rlwnm. 24-150
 sc 24-151
 slw 24-152
 slw. 24-152
 srw 24-153
 srw. 24-153
 srawi 24-154
 srawi. 24-154
 srw 24-155
 srw. 24-155

stb 24-156
 stbu 24-157
 stbux 24-158
 stbx 24-159
 sth 24-160
 sthbrx 24-161
 sthu 24-162
 sthux 24-163
 sthx 24-164
 stmw 24-165
 stswi 24-166
 stswx 24-167
 stw 24-169
 stwbrx 24-170
 stwcc. 24-171
 stwu 24-173
 stwux 24-174
 stwx 24-175
 subf 24-176
 subf. 24-176
 subfc 24-177
 subfc. 24-177
 subfco 24-177
 subfco. 24-177
 subfe 24-178
 subfe. 24-178
 subfeo 24-178
 subfeo. 24-178
 subfic 24-179
 subfme 24-180
 subfme. 24-180
 subfmeo 24-180
 subfmeo. 24-180
 subfo 24-176
 subfo. 24-176
 subfze 24-181
 subfze. 24-181
 subfzeo 24-181
 subfzeo. 24-181
 sync 24-182
 tibia 24-183
 tibre 24-184
 tibsx 24-186
 tibsx. 24-186
 tibsync 24-187
 tibwe 24-188
 tw 24-190
 twi 24-193
 wrtee 24-196
 wrteei 24-197
 xor 24-198
 xori 24-199
 instruction cache array, improved fetcher performance 4-3
 Instruction Cache Cachability Register. *See* ICCR
 Instruction Cache Cachability Register. *See* ICCR
 Instruction Cache Debug Data Register. *See* ICDBDR
 instruction cache synonym, defined 4-4
 instruction cache unit. *See* ICU
 instruction fetching
 from little endian storage 3-31

- instruction fields A-41
- instruction formats 24-2, A-41
 - diagrams A-43
- instruction forms A-41, A-43
- instruction queue
 - illustrated 3-34
 - role in processing instructions 3-33
- instruction set
 - brief summaries by category 3-47
 - for embedded controllers 3-48
- instruction set portability 24-1
- instruction set summary
 - cache control 3-52
 - CR logical 3-51
- instruction storage interrupts
 - causes 10-38
 - register settings 10-38
- instruction timings C-3
 - branches and cr logicals C-3
 - general rules C-3
 - instruction cache misses C-7
 - loads and stores C-6
 - strings C-6
- instruction TLB. *See* ITLB
- instructions
 - alphabetical, including extended mnemonics A-1
 - arithmetic and logical 3-49, B-33
 - arithmetic compares 3-13
 - branch 3-51, B-38
 - branch conditional, testing CTR 3-35
 - byte-reverse, usefulness of 3-32
 - cache
 - DAC debug events 12-20
 - cache control B-41
 - cache control, alignment of 3-27
 - compare 3-50
 - comparison B-39
 - condition register logical B-37
 - context synchronizing, defined 3-44
 - CR logical 3-50
 - extended mnemonics B-9
 - format diagrams A-43
 - formats A-41
 - forms A-41, A-43
 - ICU controlling 4-9
 - interrupt control 3-52, B-42
 - logical compares 3-13
 - opcodes A-33
 - privileged B-7
 - privileged, listed 3-42
 - processor management 3-53, B-42
 - for reading DCRs 3-43
 - for reading privileged SPRs 3-42
 - rotate and shift B-40
 - specific to PowerPC Embedded Controllers B-1, B-5
 - storage reference B-29
 - storage reference, alignment of 3-27
 - storage reference, in core 3-48
 - TLB management 3-53
- interrupt controller interface 1-9
- interrupt enable register
 - description 21-5
- interrupt identification register
 - description 21-6
- Interrupts 22-21
- interrupts
 - alignment 3-28
 - register settings 10-40
 - summary 10-39
 - asynchronous, defined 10-22
 - behavior 10-22
 - critical
 - defined 10-26
 - processing 10-27
 - critical input 10-34
 - data machine check 10-36
 - data storage 6-9, 10-36
 - register settings 10-37
 - debug, register settings 10-44
 - defined 10-22
 - DTLB miss 6-10
 - DTLB, register settings 10-43
 - external
 - programming note 10-38
 - register settings 10-39
 - fetching past, speculatively 3-39
 - FIT, causes 10-42
 - FIT, register settings 10-42
 - handling as critical 10-23
 - handling priorities 10-24
 - handling priorities, illustrated 10-25
 - imprecise, defined 10-22
 - instruction storage 6-10
 - causes 10-38
 - register settings 10-38
 - ITLB miss 6-10
 - ITLB miss, registers 10-44
 - machine check, causes of 10-35
 - machine check, defined 10-23
 - machine check—instruction
 - handling 10-35
 - register settings 10-36
 - synchronism 10-24
 - noncritical
 - defined 10-26
 - processing 10-26
 - PIT, register settings 10-41
 - precise handling 10-23
 - precise, defined 10-22
 - program
 - causes 10-40
 - ESR usage 10-40
 - register settings 10-41
 - register settings during critical 10-35
 - synchronous, defined 10-22
 - system call, register settings 10-41
 - TLB miss, preventing 6-10
 - TLB-related 6-9
 - vector offsets, illustrated 10-27
 - WDT, causes 10-43
 - WDT, register settings 10-43
- isync 24-70

- and ITLB 6-8
- context synchronization, example 3-45
- ITLB (instruction translation lookaside buffer)
 - accesses 6-6
 - consistency 6-8
 - defined 6-6
 - miss interrupts 6-10, 10-44
 - programming note 6-9

L

- lbz 24-71
- lbzu 24-72
- lbzx 24-74
- least-recently-used policy. *See* LRU
- lha 24-75
- lhau 24-76
- lhax 24-78
- lhbrx 24-79
- lhz 24-80
- lhzu 24-81
- lhzux 24-82
- lhzx 24-83
- li 24-9
- line control register
 - description 21-8
- line status register 21-11
- Link Register. *See* LR
- lis 24-12
- little endian
 - alignment 3-27
 - byte ordering supported 3-30
 - defined 3-28
 - mapping 3-30
 - storage attributes 3-30
 - storage regions
 - accessing data from 3-31
 - byte-reverse instructions 3-32- 3-33
 - fetching instructions from 3-31
- lmw 24-84
- load strategies, controlled by DCU 4-7
- logical compares 3-13
- logical instructions
 - CR 3-50
 - overview 3-49
- LR 25-116
- LR (Link Register)
 - branch instructions 3-51
 - function 3-8
- LRU (least-recently-used) policy
 - DCU 4-6
 - ICU 4-2
- lswi 24-85
- lswx 24-87
- lwarx 24-89
- lwz 24-91
- lwzu 24-92
- lwzux 24-93
- lwzx 24-94

M

- macchw 24-95
- macchws 24-96

- macchwsu 24-97
- macchwu 24-98
- machhw 24-99
- machhwsu 24-101
- machhwu 24-102
- machine check interrupts
 - causes 10-35
 - defined 10-23
- machine check—instruction interrupts
 - handling 10-35
 - register settings 10-36
 - synchronism 10-24
- Machine State Register. *See* MSR
- macihw 24-103
- macihws 24-104, 24-138
- macihwu 24-106
- MAL0_CFG 25-117
- MAL0_ESR 25-119
- MAL0_IER 25-121
- MAL0_RCBS0 25-122
- MAL0_RXCARR 25-123
- MAL0_RXCASR 25-124
- MAL0_RXCTPOR 25-125
- MAL0_RXDEIR 25-126
- MAL0_RXEOBISR 25-127
- MAL0_TXCASR 25-129
- MAL0_TXCTPOR 25-130
- MAL0_TXCTP1R 25-130
- MAL0_TXDEIR 25-131
- MAL0_TXEOBISR 25-132
- mapping
 - big endian 3-29
 - little endian 3-30
 - structure, examples 3-29
- mcrf 24-107
- mcrxr 24-108
- Memory Controller Address Register. *See* SDRAM0_CFGADDR
- Memory Controller Data Register. *See* SDRAM0_CFGDATA
- memory interface
 - bus attachment
 - alternative 16-3
 - external bus master 16-17
 - SRAM
 - burst mode 16-8
 - bus timeout error 16-11
 - device-paced transfers 16-11
 - timing 15-1, 16-2
- memory map
 - address space usage 3-2
 - PCI configuration registers 25-13
- memory mapping
 - of hardware 3-40
- memory models, non-supported 4-7, 6-5
- memory organization 3-1
- memory-mapped input/output registers. *See* MMIO registers
- mfcrr 24-109
- mfdcr 24-110
- mfmsr 24-111

- mfspir 24-112
- mftb 24-115
- mftbu 24-115
- misalignments, defined 3-27
- MMIO (memory-mapped input/output) registers
 - directly accessed 3-22
 - indirectly accessed 3-25
- MMU (memory management unit)
 - accesses, interrupts from 6-9
 - address translation 6-1
 - data storage interrupts 6-9
 - DTLB miss interrupts 6-10
 - execute permissions 6-13
 - general access protection 6-12
 - instruction storage interrupts 6-10
 - ITLB miss interrupts 6-10
 - MSR and access protection 6-12
 - overview 1-6
 - recording page references and changes 6-11
 - TLB management 6-10
 - zone protection 6-13
- mnemonics, extended. *See* extended mnemonics
- modes
 - execution 3-41
 - real, storage attribute control 6-17
- mr 24-140
- mr. 24-140
- MSR 3-15, 25-133
- MSR (Machine State Register)
 - bits and exception handling 3-41
 - contents after resets 8-3
 - controlling execution mode 3-41
 - DR bit 6-1
 - illustrated 10-28
 - interrupt control instructions 3-52
 - IR bit 6-1
 - programming note 10-28
 - summarized 1-10
- mtrcr 24-116
- mtrcf 24-116
- mtdcr 24-117
- mtmsr
 - execution synchronization 3-46
- mtspr 24-119
- mulchw 24-121
- mulchwu 24-122
- mulhww 24-123
- mulhwwu 24-124
- mulhwu 24-126
- mulhwu. 24-126
- mullhw 24-127
- mullhwu 24-128
- mulli 24-129
- mullw 24-130
- mullw. 24-130
- mullwo 24-130
- mullwo. 24-130

N

- nand 24-131
- nand. 24-131
- neg 24-132
- neg. 24-132
- nego 24-132
- nego. 24-132
- nmacchww 24-133
- nmacchws 24-134
- nmachhw 24-135
- nmachhws 24-136
- nmaclhw 24-137
- nmaclhws 24-138
- noncritical interrupts
 - defined 10-26
 - processing 10-26
- nop 24-142
- nor 24-139
- nor. 24-139
- not 24-139
- not. 24-139
- notation xlviii, 24-2, A-41
- notational conventions xlviii

O

- OCM0_DSARC 25-135
- OCM0_DSCNTL 25-136
- OCM0_ISARC 25-137
- OCM0_ISCNTL 25-138
- on-chip peripheral bus 2-11
 - features 2-11
- OPB 2-11
- OPB arbiter
 - registers 2-12
- OPBA0_CR 2-12, 25-139
- opcodes A-33
- optimization
 - coding guidelines C-1
 - alignment C-2
 - boolean variables C-1
 - branch prediction C-2
 - dependency upon CR C-2
 - floating point emulation C-1
- or 24-140
- or. 24-140
- orc 24-141
- orc. 24-141
- ori 24-142
- oris 24-143
- overlapped PLB transfers 2-4

P

- page identification fields, UTLB 6-3
- parallel-to-serial conversion 21-1
- PCI
 - configuration registers 25-13
- PCI configuration registers
 - accessing 3-25
 - offsets 3-25, 17-20, 25-13
- PCIC0_BAR0 25-141
- PCIC0_BIST 25-142
- PCIC0_BRDGOPT1 25-143
- PCIC0_BRDGOPT2 25-144
- PCIC0_CACHELS 25-145
- PCIC0_CAP 25-146
- PCIC0_CAPID 25-147

PCIC0_CFGADDR 25-148
 PCIC0_CFGDATA 25-149
 PCIC0_CLS 25-150
 PCIC0_CMD 25-151
 PCIC0_DATA 25-153
 PCIC0_DEVID 25-154
 PCIC0_ERREN 25-155
 PCIC0_ERRSTS 25-156
 PCIC0_HDTYPE 25-157
 PCIC0_ICS 25-158
 PCIC0_INTLN 25-159
 PCIC0_INTPN 25-160
 PCIC0_LATTIM 25-161
 PCIC0_MAXLTNCY 25-162
 PCIC0_MINGNT 25-163
 PCIC0_NEXTIPTR 25-164
 PCIC0_PLBBEAR 25-165
 PCIC0_PLBBESR0 25-166
 PCIC0_PMC 25-170
 PCIC0_PMCSR 25-171
 PCIC0_PMSCRR 25-173
 PCIC0_PTM1BAR 25-174
 PCIC0_PTM2BAR 25-175
 PCIC0_REVID 25-176
 PCIC0_SBSYSID 25-177
 PCIC0_SBSYSVID 25-178
 PCIC0_STATUS 25-179
 PCIC0_VENDID 25-181
 PCILO_PMM0LA 25-182
 PCILO_PMM0MA 25-183
 PCILO_PMM0PCIHA 25-184
 PCILO_PMM0PCILA 25-185
 PCILO_PMM1LA 25-186
 PCILO_PMM1MA 25-187
 PCILO_PMM1PCIHA 25-188
 PCILO_PMM1PCILA 25-189
 PCILO_PMM2LA 25-190
 PCILO_PMM2MA 25-191
 PCILO_PMM2PCIHA 25-192
 PCILO_PMM2PCILA 25-193
 PCILO_PTM1LA 25-194
 PCILO_PTM1MS 25-195
 PCILO_PTM2LA 25-196
 PCILO_PTM2MS 25-197
 PCIIPCIC0_PLBBESR1 25-168
 performance
 DCU
 improve with simultaneous caching 4-18
 limited by sequential caching 4-18
 overview 4-16
 improve
 through byte-writeability 4-6
 lower
 from cache-inhibited regions 4-4
 Peripheral Controller Address Register. *See*
 EBC0_CFGADDR
 Peripheral Controller Data Register. *See*
 EBC0_CFGDATA
 physical address map 3-2
 PID 25-198
 PID (process ID)
 illustrated 6-12
 PIT 11-4, 25-199
 PIT (programmable interval timer)
 interrupts, register settings 10-41
 PLB 2-1
 PLB arbiter
 registers 2-5
 PLB to OPB bridge
 registers 2-8
 PLB0_ACR 2-5, 25-200
 PLB0_BEAR 2-5, 25-201
 PLB0_BESR 2-6, 25-202
 POB0_BEAR 2-8, 25-204
 POB0_BESR0 25-205
 POB0_BESR1 25-207
 portability, instruction set 24-1
 PowerPC architecture 1-4
 precise interrupts 10-22
 pre-fetch
 branches to CTR 3-38
 branches to LR 3-38
 buffers 3-33
 past interrupts 3-39
 primary opcodes A-33
 priority signal
 DCU 4-17
 privileged mode
 defined 3-41
 instructions, listed 3-42
 registers 3-5
 privileged programming model 3-1
 privileged SPRs
 instructions for reading 3-42
 problem state. *See* user mode
 process ID. *See* PID
 processor
 management instructions 3-53
 processor local bus 2-1
 overlapped transfers 2-4
 transfer protocol 2-3
 Processor Version Register. *See* PVR
 program interrupts
 causes 10-40
 ESR usage 10-40
 register settings 10-41
 programmable interval timer 11-4
 programming model
 features 3-1
 programming models
 privileged 3-1
 user 3-1
 programming note
 data storage interrupts 10-37
 EA access in DCU 4-8
 external or timer interrupts 10-38
 instruction pipeline 4-15
 MSR affected by instructions 10-28
 non-supported memory models 4-7
 reserved fields 3-3
 RPN field 6-4
 synchronizing the ITLB 6-9

pseudocode 24-2
PVR 25-208
PVR (Processor Version Register)
illustrated 3-12

R

real mode

storage attribute control 6-17
register description 21-4
register set summary 1-10- 1-11
registers

categories 3-3, 25-1
CCRO 25-15
CPC0_CR1 25-20
CPC0_ER 25-21
CPC0_FR 25-22
CPC0_JTAGID 25-23
CPC0_PLLMR 25-24
CPC0_PSR 25-26
CPC0_SR 25-28
CR 25-1, 25-29
CTR 15-6, 15-14, 15-15, 15-16, 15-17, 15-20,
25-30, 25-209, 25-210, 25-218,
25-219, 25-220, 25-221

DAC1 12-14
DAC1-DAC2 12-14, 14-7, 25-31, 25-48
DBCR 12-9
DBCR0 25-32, 25-34
DBSR 12-12, 25-36
DCCR 25-38
DCP0_ADDR0-DCP0_ADDR1 25-40
DCP0_ADDR0-DCP0_ADDR3 14-5, 25-40
DCP0_CFG 14-6, 25-41
DCP0_CFGADDR 25-42
DCP0_CFGDATA 25-43
DCP0_ESR 14-8, 25-44
DCP0_ID 14-6, 25-46
DCP0_ITOR0-DCP0_ITOR3 14-4, 25-47
DCP0_MEMBEAR 14-7, 25-48
DCP0_PLBBEAR 14-7, 25-49
DCP0_RAM0-DCP0_RAM3FF 25-50
DCP0_VER 14-7, 25-51
DCR numbering 25-4

DCRs

indirectly accessed 3-19
summarized 1-11
uses for 3-16

DCWR 25-52

DEAR 25-54

descriptions of commonly used 3-3

DMA0_CR0-DMA0_CR3 18-8, 25-55
DMA0_CT0-DMA0_CT3 18-11, 25-58
DMA0_DA0-DMA0_DA3 18-11, 25-59
DMA0_POL 25-60
DMA0_SAO-DMA0_SA3 18-10
DMA0_SAO-DMA0_SA3 25-62
DMA0_SG0-DMA0_SG3 18-12
DMA0_SG0-DMA0_SG3 25-63
DMA0_SGC 18-13, 25-64
DMA0_SLP 25-65
DMA0_SR 18-7, 25-66
during debug exceptions 10-45

EBC0_BEAR 25-68
EBC0_BESR0 25-69
EBC0_BESR1 25-71
EBC0_BnAP 25-73
EBC0_BnCR 25-75
EBC0_CFG 25-76
EBC0_CFGADDR 25-78
EBC0_CFGDATA 25-79
EMAC0_GAHT1-EMAC0_GAHT4 25-80
EMAC0_IAHR 25-81
EMAC0_IAHT1-EMAC0_IAHT4 25-82
EMAC0_TMR0 25-83
EMAC0_TMR1 25-84
EMAC0_TRTR 25-85
EMAC0_VTCI 25-86
EMAC0_VTPID 25-87
ESR 25-88
EVPR 25-89
GPIO0_IR 25-90
GPIO0_ODR 25-91
GPIO0_OR 25-92
GPIO0_TCR 25-93
GPR 25-1
GPR0-GPR31 25-94
GPRs
overview 3-6
summary 1-10
IAC1-IAC4 25-67, 25-95
IAC1-IAC4 12-14
ICCR 25-96
ICDBDR 25-98
IIC0_CLKDIV 25-99
IIC0_CNTL 25-100
IIC0_DIRECTCNTL 25-101
IIC0_EXTSTS 25-102
IIC0_HMADR 25-104
IIC0_HSADR 25-105
IIC0_INTRMSK 25-106
IIC0_LMADR 25-107
IIC0_LSADR 25-108
IIC0_MDBUF 25-109
IIC0_MDCNTL 25-110
IIC0_SDBUF 25-111
IIC0_STS 25-112
IIC0_XFRONT 25-113
IIC0_XTCNTLSS 25-114
interrupt handling 10-27
LR 25-116
MAL0_CFG 25-117
MAL0_ESR 25-119
MAL0_IER 25-121
MAL0_RCBS0 25-122
MAL0_RXCARR 25-123
MAL0_RXCASR 25-124
MAL0_RXCTP0R 25-125
MAL0_RXDEIR 25-126
MAL0_RXEOBISR 25-127
MAL0_TXCASR 25-129
MAL0_TXCTP0R 25-130
MAL0_TXCTP1R 25-130
MAL0_TXDEIR 25-131

MAL0_TXEOBISR 25-132
 MMIO registers
 directly accessed 3-22
 indirectly accessed 3-25
 MSR 3-15, 25-1, 25-133
 OCM0_DSARC 25-135
 OCM0_DSCNTL 25-136
 OCM0_ISARC 25-137
 OCM0_ISCNTL 25-138
 OPB arbiter 2-12
 OPBA0_CR 2-12, 25-139
 PCI configuration 25-13
 PCIC0_BAR0 25-141
 PCIC0_BIST 25-142
 PCIC0_BRDGOPT1 25-143
 PCIC0_BRDGOPT2 25-144
 PCIC0_CACHELS 25-145
 PCIC0_CAP 25-146
 PCIC0_CAPID 25-147
 PCIC0_CFGADDR 25-148
 PCIC0_CFGDATA 25-149
 PCIC0_CLS 25-150
 PCIC0_CMD 25-151
 PCIC0_DATA 25-153
 PCIC0_DEVID 25-154
 PCIC0_ERREN 25-155
 PCIC0_ERRSTS 25-156
 PCIC0_HDTYPE 25-157
 PCIC0_ICS 25-158
 PCIC0_INTLN 25-159
 PCIC0_INTPN 25-160
 PCIC0_LATTIM 25-161
 PCIC0_MAXLTNCY 25-162
 PCIC0_MINGNT 25-163
 PCIC0_NEXTIPTR 25-164
 PCIC0_PLBBEAR 25-165
 PCIC0_PLBBESR0 25-166
 PCIC0_PMC 25-170
 PCIC0_PMCSR 25-171
 PCIC0_PMSCRR 25-173
 PCIC0_PTM1BAR 25-174
 PCIC0_PTM2BAR 25-175
 PCIC0_REVID 25-176
 PCIC0_SBSYSID 25-177
 PCIC0_SBSYSVID 25-178
 PCIC0_STATUS 25-179
 PCIC0_VENDID 25-181
 PCIL0_PMM0LA 25-182
 PCIL0_PMM0MA 25-183
 PCIL0_PMM0PCIHA 25-184
 PCIL0_PMM0PCILA 25-185
 PCIL0_PMM1LA 25-186
 PCIL0_PMM1MA 25-187
 PCIL0_PMM1PCIHA 25-188
 PCIL0_PMM1PCILA 25-189
 PCIL0_PMM2LA 25-190
 PCIL0_PMM2MA 25-191
 PCIL0_PMM2PCIHA 25-192
 PCIL0_PMM2PCILA 25-193
 PCIL0_PTM1LA 25-194
 PCIL0_PTM1MS 25-195

PCIL0_PTM2LA 25-196
 PCIL0_PTM2MS 25-197
 PCIPPCIC0_PLBBESR1 25-168
 PID 25-198
 PIT 11-4, 25-199
 PLB arbiter 2-5
 PLB to OPB bridge 2-8
 PLB0_ACR 2-5, 25-200
 PLB0_BEAR 2-5, 25-201
 PLB0_BESR 2-6, 25-202
 POB0_BEAR 2-8, 25-204
 POB0_BESR0 25-205
 POB0_BESR1 25-207
 PVR 25-208
 reserved 25-1
 reserved fields 3-3, 25-1
 SDRAM0_BOCR-SDRAM0_B3CR 25-209
 SDRAM0_BEAR 25-210
 SDRAM0_BESR0 25-211
 SDRAM0_BESR1 25-213
 SDRAM0_CFG 25-214
 SDRAM0_CFGADDR 25-216
 SDRAM0_CFGDATA 25-217
 SDRAM0_ECCCFG 25-218
 SDRAM0_ECCESR 25-219
 SDRAM0_PMIT 25-220
 SDRAM0_RTR 25-221
 SDRAM0_TR 25-222
 SGR 25-224
 SLER 25-226
 SPR numbering 25-1
 SPRG0-SPRG4 3-11
 SPRG0-SPRG7 25-228
 SPRs
 overview 3-6
 summary 1-10
 SRR0 25-229
 SRR1 25-230
 SRR2 25-231
 SRR3 25-232
 SUOR 25-233
 summary of sets 1-10- 1-11
 supervisor, illustrated 3-5
 TBL 25-235
 TBU 25-236
 TCR 11-5, 11-6, 11-9, 25-237
 TSR 11-6, 11-8, 25-238
 UARTx_DLL 25-239
 UARTx_DLM 25-240
 UARTx_FCR 25-241
 UARTx_IER 25-242
 UARTx_IIR 25-243
 UARTx_LCR 25-244
 UARTx_LSR 25-245
 UARTx_MCR 25-247
 UARTx_MSR 25-248
 UARTx_RBR 25-249
 UARTx_SCR 25-250
 UARTx_THR 25-251
 UIC0_CR 25-252
 UIC0_ER 25-255

- UIC0_MSR 25-258
- UIC0_PR 25-261
- UIC0_SR 25-264
- UIC0_TR 25-267
- UIC0_VCR 25-270
- UIC0_VR 25-271
- user, illustrated 3-5
- USPRG0 3-11, 25-272
- XER 25-273
- ZPR 25-274
- reservation bit 24-89, 24-171
- reserved fields 25-1
 - programming note 3-3
- reserved registers 25-1
- resets
 - effects on MSR 8-3
 - effects on SPRs 8-4
 - processor initialization 8-14
 - processor state after 8-2
- rftci 24-144
 - effect on MSR reserved fields 10-28
- rfti 24-145
 - effect on MSR reserved fields 10-28
- rlwimi 24-146
- rlwimi. 24-146
- rlwinm 24-147
- rlwinm. 24-147
- rlwnm 24-150
- rlwnm. 24-150
- rotlw 24-150
- rotlw. 24-150
- rotlwi 24-148
- rotlwi. 24-148
- rotrwi 24-148
- rotrwi. 24-148
- extended mnemonics
 - bnctr 24-28

S

- Save/Restore Registers 0-1. *See* SRR0-1
- sc 24-151
- scratchpad register 21-13
- SDRAM controller
 - DCRs
 - access procedures, overview 3-19
 - indirect access 3-19
 - offsets 3-19
- SDRAM0_B0CR-SDRAM0_B3CR 25-209
- SDRAM0_BEAR 25-210
- SDRAM0_BESR0 25-211
- SDRAM0_BESR1 25-213
- SDRAM0_CFG 25-214
- SDRAM0_CFGADDR 25-216
- SDRAM0_CFGADDR (Memory Controller Address Register)
 - accessing 3-19
- SDRAM0_CFGDATA 25-217
- SDRAM0_CFGDATA (Memory Controller Data Register)
 - accessing 3-19
- SDRAM0_ECCCFG 25-218
- SDRAM0_ECCESR 25-219
- SDRAM0_PMIT 25-220
- SDRAM0_RTR 25-221
- SDRAM0_TR 25-222
- secondary opcodes A-33
- serial-interface characteristic 21-2
- serial-to-parallel conversion 21-1
- SGR 25-224
- SGR (Storage Guarded Register)
 - controlling speculative accesses 3-38
 - controlling the guarded (G) storage attribute 6-19
- shadow TLB. *See* DTLB
- SLER 25-226
- SLER (Storage Little Endian Register)
 - controlling the endian (E) storage attribute 6-19
- slw 24-152
- slw. 24-152
- slwi 24-148
- slwi. 24-148
- Special Purpose Register General 0-7. *See* SPRG0-7
- special purpose registers. *See* SPRs
- speculative accesses
 - to CTR or LR 3-38
 - defined 3-37
 - down predicted path 3-38
 - fetching past interrupts 3-39
 - fetching past tw or twi 3-39
 - fetching past unconditional branches 3-40
 - preventing inappropriate 3-37- 3-41
- SPRG0-7 (Special Purpose Register General 0-7)
 - temporary storage to 3-11
- SPRG0-SPRG4 3-11
- SPRG0-SPRG7 25-228
- SPRs (special purpose registers)
 - contents after resets 8-4
 - listed, with page references 3-7
 - overview 3-6
 - privileged and non-privileged 3-6
 - privileged, instructions for reading 3-42
 - summary 1-10
- SRAM
 - timing 15-1, 16-2
- sraw 24-153
- sraw. 24-153
- srawi 24-154
- srawi. 24-154
- SRR0 25-229
- SRR0-1 (Save/Restore Registers 0-1)
 - illustrated 10-29
- SRR1 25-230
- SRR2 25-231
- SRR3 25-232
- srw 24-155
- srw. 24-155
- srwi 24-149
- srwi. 24-149
- stb 24-156
- stbu 24-157
- stbux 24-158
- stbx 24-159
- sth 24-160
- sthrbx 24-161
- sthu 24-162

- sthux 24-163
- sthx 24-164
- stmw 24-165
- storage attribute control registers
 - DCCR 6-19
 - DCWR 6-18
 - ICCR 6-19
 - SGR 6-19
 - SLER 6-19
 - SUOR 6-19
- storage attributes
 - caching inhibited (I)
 - real mode 6-19
 - virtual mode 6-5
 - endian (E)
 - and little endian 3-30
 - real mode 6-19
 - when controlled by TLB 6-6
 - guarded (G)
 - controlling speculative accesses 3-37
 - real mode 6-19
 - virtual mode 6-6
 - memory coherent (M)
 - not supported 6-6
 - overview 3-3
 - real mode 6-17
 - TLB control of
 - user-defined (U0)
 - real mode 6-19
 - virtual mode 6-5
 - write-through (W)
 - real mode 6-18
 - virtual mode 6-5
- Storage Guarded Register. *See* SGR
- Storage Guarded Register. *See* SGR
- Storage Little Endian Register. *See* SLER
- storage reference instructions 3-48
- storage regions
 - big endian
 - alignment 3-27
 - byte-reverse instructions 3-32- 3-33
 - little endian
 - accessing data from 3-31
 - alignment 3-27
 - byte reversal 3-31
 - byte-reverse instructions 3-32- 3-33
 - data alignment 3-32
 - fetching instructions from 3-31
- storage synchronization 3-46
- Storage User-Defined 0 Register. *See* SUOR
- string instructions
 - access protection 6-16
- structure mapping
 - examples 3-29
- stswi 24-166
- stswx 24-167
- stw 24-169
- stwbrx 24-170
- stwcx. 24-171
- stwu 24-173
- stwux 24-174
- stwx 24-175
- SUOR 25-233
- SUOR (Storage User-Defined 0 Register)
 - controlling the user-defined (U0) storage attribute 6-19
- sub 24-176
- sub. 24-176
- subc 24-177
- subc. 24-177
- subco 24-177
- subco. 24-177
- subf 24-176
- subf. 24-176
- subfc 24-177
- subfc. 24-177
- subfco 24-177
- subfco. 24-177
- subfe 24-178
- subfe. 24-178
- subfeo 24-178
- subfeo. 24-178
- subfic 24-179
- subfme 24-180
- subfme. 24-180
- subfmeo 24-180
- subfmeo. 24-180
- subfo 24-176
- subfo. 24-176
- subfze 24-181
- subfze. 24-181
- subfzco 24-181
- subfzco. 24-181
- subi 24-9
- subic 24-10
- subic. 24-11
- subis 24-12
- subo 24-176
- subo. 24-176
- supervisor state. *See* privileged mode
- sync 24-182
 - storage synchronization 3-46
- synchronization
 - context 3-44
 - execution, defined 3-46
 - ICU 4-5
 - references to PowerPC Architecture 3-43
 - storage 3-46
- synchronous interrupts 10-22
- system call interrupts
 - register settings 10-41

T

- TBL 25-235
- tblrehi 24-185
- tblrelo 24-185
- tblwehi 24-189
- tblwelo 24-189
- TBU 25-236
- TCR 11-9, 25-237
- TID (translation ID)
 - and MMU access protection
- time base 11-2

- implementation 3-15
- writing 3-15
- timer interrupts
 - programming note 10-38
- timers
 - FIT 11-5
 - fixed interval timer 11-5
 - PIT 11-4
 - programmable interval timer 11-4
 - TCR 11-9
 - timer control register 11-9
 - timer status register 11-8
 - TSR 11-8
 - watchdog 11-6
- timings
 - instruction C-3
 - branches and cr logicals C-3
 - general rules C-3
 - instruction cache misses C-7
 - loads and stores C-6
 - strings C-6
- TLB (translation lookaside buffer) 6-2
 - access protection 6-12, 6-15
 - and cacheability control 4-8
 - execute permissions 6-13
 - interrupts 6-9
 - invalidate instruction 6-11
 - management instructions 3-53
 - preventing miss interrupts 6-10
 - read/write instructions 6-11
 - search instructions 6-11
 - sync instruction 6-11
 - zone protection 6-13
 - See also* ITLB;UTLB;DTLB
- tliba 24-183
 - and TLB management 6-11
- tlibre 24-184
 - and TLB management 6-11
- tlibsx 24-186
 - and TLB management 6-11
- tlibsx. 24-186
 - and TLB management 6-11
- tlibsync 24-187
 - and TLB management 6-11
- tlibwe 24-188
 - and TLB management 6-11
- transfer protocol
 - processor local bus 2-3
- translation ID. *See* TID
- translation lookaside buffer. *See* TLB
- translation, address. *See* address translation
- trap 24-191
- TSR 11-8, 25-238
- tw 24-190
 - fetching past 3-39
- tweq 24-191
- tweqi 24-194
- twge 24-191
- twgei 24-194
- twgle 24-191
- twgt 24-191

- twgti 24-194
- twi 24-193
 - fetching past 3-39
- twle 24-191
- twlei 24-194
- twlgei 24-194
- twlgt 24-191
- twlgti 24-194
- twlle 24-192
- twllei 24-194
- twllt 24-192
- twllti 24-194
- twlng 24-192
- twlngi 24-194
- twlnl 24-192
- twlnli 24-195
- twlt 24-192
- twlti 24-195
- twne 24-192
- twnei 24-195
- twng 24-192
- twngi 24-195
- twnl 24-192
- twnli 24-195

U

- UART Reset and Sleep mode 21-16
- UARTx_DLL 25-239
- UARTx_DLM 25-240
- UARTx_FCR 25-241
- UARTx_IER 25-242
- UARTx_IIR 25-243
- UARTx_LCR 25-244
- UARTx_LSR 25-245
- UARTx_MCR 25-247
- UARTx_MSR 25-248
- UARTx_RBR 25-249
- UARTx_SCR 25-250
- UARTx_THR 25-251
- UIC0_CR 25-252
- UIC0_ER 25-255
- UIC0_MSR 25-258
- UIC0_PR 25-261
- UIC0_SR 25-264
- UIC0_TR 25-267
- UIC0_VCR 25-270
- UIC0_VR 25-271
- unconditional branches
 - AA field 3-34
 - speculative accesses 3-40
- unified TLB. *See* UTLB
- user mode
 - defined 3-41
 - registers 3-5
- user programming model 3-1
- user-defined (U0) storage attribute
 - controlled by SUOR 6-19
- USPRG0 3-11, 25-272
- UTLB (unified translation lookaside buffer)
 - access control fields 6-5
 - entry format, illustrated 6-3
 - EPN field 6-3

- EX field 6-5
- field categories 6-3
- functional overview 6-2
- page identification fields 6-3
- RPN field 6-4
- SIZE field 6-4
- TID field 6-4
- translation field 6-4
- V field 6-4
- WR field 6-5
- ZSEL field 6-5

V

- virtual mode
 - and TLB control of storage attributes 6-5

W

- watchdog timer 11-6
- WDT (watchdog timer)
 - interrupts, causes 10-43
 - interrupts, register settings 10-43
- write strategies
 - controlled by DCWR 4-7
 - used by DCU 4-7
- write-through (W) storage attribute
 - controlled by DCWR 6-18
 - when controlled by TLB 6-5
- wrttee 24-196
- wrtteei 24-197

X

- XER 25-273
- XER (Fixed Point Exception Register)
 - illustrated 3-8
- xor 24-198
- xori 24-199

Z

- zone fault 10-37
- Zone Protection Register. *See* ZPR
- zone, defined 6-13
- ZPR 25-274
- ZPR (Zone Protection Register)
 - illustrated 6-13



© International Business Machines Corporation 1996, 2000
Printed in the United States of America
6/19/00
All Rights Reserved

The information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change IBM's product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. All information contained in this document was obtained in specific environments, and is presented as illustration. The results obtained in other operating environments may vary.

While the information contained herein is believed to be accurate, such information is preliminary, and should not be relied upon for accuracy or completeness, and no representations or warranties of accuracy or completeness are made.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. In no event will IBM be liable for any damages arising directly or indirectly from any use of the information contained in this document.

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY
12533-6531

The IBM home page can be found at www.ibm.com

The IBM Microelectronics Division home page can be found at www.chips.ibm.com

Address technical queries about this product to pccsupp@us.ibm.com



© International Business Machines Corporation 2000
Printed in the United States of America
June 2000
All Rights Reserved

IBM and PowerPC are trademarks of International Business Machines Corporation in the United States and/or other countries.

While the information contained herein is believed to be accurate, such information is preliminary, and should not be relied upon for accuracy or completeness, and no representations or warranties of accuracy or completeness are made.

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change IBM's product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. All information contained in this document was obtained in specific environments, and is presented as illustration. The results obtained in other operating environments may vary.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. In no event will IBM be liable for any damages arising directly or indirectly from any use of the information contained in this document.

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY
12533-6531

The IBM home page can be found at
www.ibm.com

The IBM Microelectronics Division home page can be found at
www.chips.ibm.com



GK10-3118-03

PowerPC