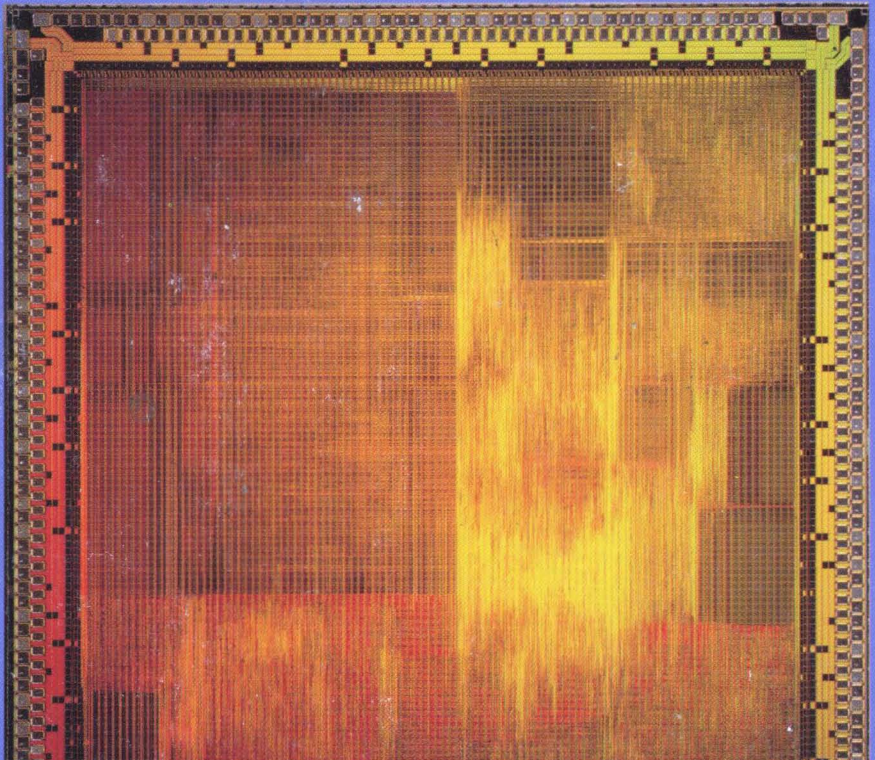
PowerPC 405GP Embedded Processor

User's Manual, Volume 2

Preliminary



IBM

PowerPC

IBM

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.

Patents and Trademarks

Contents

Figures

Tables

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:

This book contains the following appendixes:

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

Conventions

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

$\overline{\text{ActiveLow}}$	An overbar indicates an active-low signal.
n	A decimal number
0x*n*	A hexadecimal number
0b*n*	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

$\div$	Division yielding a quotient
%	Remainder of an integer division; (33 % 32) = 1.
||	Concatenation
$=, \neq$	Equal, not equal relations
$<, >$	Signed comparison relations
$\overset{u}{<}, \overset{u}{>}$	Unsigned comparison relations
if...then...else...	Conditional execution; if *condition* then *a* else *b*, where *a* and *b* represent one or more pseudocode statements. Indenting indicates the ranges of *a* and *b*. If *b* 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(r)	General Purpose Register (GPR) r, where $0 \leq r \leq 31$.
(GPR(r))	The contents of GPR r, where $0 \leq r \leq 31$.
DCR(DCRN)	A Device Control Register (DCR) specified by the DCRF field in an **mfdcr** or **mtdcr** instruction
SPR(SPRN)	An SPR specified by the SPRF field in an **mfspr** or **mtspr** instruction
TBR(TBRN)	A Time Base Register (TBR) specified by the TBRF field in an **mftb** instruction
GPRs	RA, RB, $_{...}$
(Rx)	The contents of a GPR, where *x* is A, B, S, or T
(RA|0)	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.
^{n}b	The bit or bit value *b* is replicated *n* times.

$\div$	Division yielding a quotient
%	Remainder of an integer division; (33 % 32) = 1.
$\|\|$	Concatenation
$=, \neq$	Equal, not equal relations
$<, >$	Signed comparison relations
$\overset{u}{<}, \overset{u}{>}$	Unsigned comparison relations
if...then...else...	Conditional execution; if *condition* then *a* else *b*, where *a* and *b* represent one or more pseudocode statements. Indenting indicates the ranges of *a* and *b*. If *b* 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
$FLD_{b,b,\ldots}$	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
$REG_{b,b,\ldots}$	A list of bits, by number or name, in a named register
REG[FLD]	A field in a named register
REG[FLD, FLD $_{\ldots}$]	A list of fields in a named register
REG[FLD:FLD]	A range of fields in a named register
GPR(r)	General Purpose Register (GPR) r, where $0 \le r \le 31$.
(GPR(r))	The contents of GPR r, where $0 \le r \le 31$.
DCR(DCRN)	A Device Control Register (DCR) specified by the DCRF field in an **mfdcr** or **mtdcr** instruction
SPR(SPRN)	An SPR specified by the SPRF field in an **mfspr** or **mtspr** instruction
TBR(TBRN)	A Time Base Register (TBR) specified by the TBRF field in an **mftb** instruction
GPRs	RA, RB, $_{\ldots}$
(Rx)	The contents of a GPR, where *x* is A, B, S, or T
(RA|0)	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.
^{n}b	The bit or bit value *b* is replicated *n* times.

xx	Bit positions which are don't-cares.
CEIL(x)	Least integer ≥ 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 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[.]**	**nmaclhw[o][.]**	**wrtee**
	machhws[o][.]	**mulhhwu[.]**	**nmaclhws[o][.]**	**wrteei**
	machhwsu[o][.]	**mullhw[.]**		
	machhwu[o][.]	**mullhwu[.]**		
	maclhw[o][.]			
	maclhws[o][.]			
	maclhwsu[o][.]			
	maclhwu[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
$\wedge$	AND logical operator
$\neg$	NOT logical operator
$\vee$	OR logical operator
$\oplus$	Exclusive-OR (XOR) logical operator
+	Twos complement addition
–	Twos complement subtraction, unary minus
$\times$	Multiplication
$\div$	Division yielding a quotient
%	Remainder of an integer division; (33 % 32) = 1.

$\|\|$	Concatenation
$=, \neq$	Equal, not equal relations
$<, >$	Signed comparison relations
$\overset{u}{<}, \overset{u}{>}$	Unsigned comparison relations
if...then...else...	Conditional execution; if *condition* then *a* else *b*, where *a* and *b* represent one or more pseudocode statements. Indenting indicates the ranges of *a* and *b*. If *b* 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
$FLD_{b,b,\ldots}$	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
$REG_{b,b,\ldots}$	A list of bits, by number or name, in a named register
REG[FLD]	A field in a named register
REG[FLD, FLD $_{\ldots}$]	A list of fields in a named register
REG[FLD:FLD]	A range of fields in a named register
GPR(r)	General Purpose Register (GPR) r, where $0 \leq r \leq 31$.
(GPR(r))	The contents of GPR r, where $0 \leq r \leq 31$.
DCR(DCRN)	A Device Control Register (DCR) specified by the DCRF field in an **mfdcr** or **mtdcr** instruction
SPR(SPRN)	An SPR specified by the SPRF field in an **mfspr** or **mtspr** instruction
TBR(TBRN)	A Time Base Register (TBR) specified by the TBRF field in an **mftb** instruction
GPRs	RA, RB, $_{\ldots}$
(Rx)	The contents of a GPR, where *x* is A, B, S, or T
(RA|0)	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 *b* is replicated *n* 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
=, ≠, <, >, $\overset{u}{<}$, $\overset{u}{>}$	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

31	RT	RA	RB	OE	266	Rc
0	6	11	16	21	22	31

(RT) ← (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

31	RT	RA	RB	OE	10	Rc
0	6	11	16	21	22	31

```
(RT) ← (RA) + (RB)
if (RA) + (RB) $\overset{u}{>}$ $2^{32} - 1$ then
   XER[CA] ← 1
else
   XER[CA] ← 0
```

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

XER[CA] is set to a value determined by the unsigned magnitude of the result of the add operation.

Registers Altered

- RT
- XER[CA]
- CR[CR0]$_{LT, GT, EQ, SO}$ 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

31	RT	RA	RB	OE	138	Rc
0	6	11	16	21	22	31

```
(RT) $\leftarrow$ (RA) + (RB) + XER[CA]
if (RA) + (RB) + XER[CA] $\overset{u}{>}$ $2^{32} - 1$ then
   XER[CA] $\leftarrow$ 1
else
   XER[CA] $\leftarrow$ 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

14	RT	RA	IM	
0	6	11	16	31

(RT) ← (RA|0) + 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) *Extended mnemonic for* **addi RT,RA,D**	
li	RT, IM	Load immediate. (RT) ← EXTS(IM) *Extended mnemonic for* **addi RT,0,IM**	
subi	RT, RA, IM	Subtract EXTS(IM) from (RA|0). Place result in RT. *Extended mnemonic for* **addi RT,RA,–IM**	

addic

Add Immediate Carrying

addic RT, RA, IM

12	RT	RA	IM	
0	6	11	16	31

```
(RT) ← (RA) + EXTS(IM)
if (RA) + EXTS(IM) $\overset{u}{>}$ $2^{32} - 1$ 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]. *Extended mnemonic for* **addic RT,RA,–IM**	

addic. RT, RA, IM

13	RT	RA	IM
0	6	11	16 … 31

```
(RT) ← (RA) + EXTS(IM)
if (RA) + EXTS(IM) $\overset{u}{>}$ $2^{32} - 1$ 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]. *Extended mnemonic for* **addic. RT,RA,–IM**	CR[CR0]

addis

Add Immediate Shifted

addis RT, RA, IM

15	RT	RA	IM
0	6	11	16 31

$(RT) \leftarrow (RA|0) + (IM \parallel {}^{16}0)$

If the RA field is 0, the IM field is concatenated on its right with sixteen 0-bits and placed into register RT.

If the RA field is nonzero, the contents of register RA are added to the contents of the extended IM field. The sum is stored into register RT.

Registers Altered

- RT

Programming Note

An **addi** instruction stores a sign-extended 16-bit value in a GPR. An **addis** instruction followed by an **ori** instruction stores an arbitrary 32-bit value in a GPR, as shown in the following example:

```
addis    RT, 0, high 16 bits of value
ori      RT, RT, low 16 bits of value
```

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. $(RT) \leftarrow (IM \parallel {}^{16}0)$ *Extended mnemonic for* **addis RT,0,IM**	
subis	RT, RA, IM	Subtract $(IM \parallel {}^{16}0)$ from (RA|0). Place result in RT. *Extended mnemonic for* **addis RT,RA,–IM**	

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

31	RT	RA		OE	234	Rc
0	6	11	16	21	22	31

```
(RT) ← (RA) + XER[CA] + (–1)
if (RA) + XER[CA] + 0xFFFF FFFF $\overset{u}{>}$ $2^{32} – 1$ then
   XER[CA] ← 1
else
   XER[CA] ← 0
```

The sum of the contents of register RA, XER[CA], and –1 is placed into register RT.

XER[CA] is set to a value determined by the unsigned magnitude of the result of the add operation.

Registers Altered

- RT
- XER[CA]
- CR[CR0]$_{LT, GT, EQ, SO}$ 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

31	RT	RA		OE	202	Rc
0	6	11	16	21	22	31

```
(RT) $\leftarrow$ (RA) + XER[CA]
if (RA) + XER[CA] $\overset{u}{>} 2^{32} - 1$ then
   XER[CA] $\leftarrow$ 1
else
   XER[CA] $\leftarrow$ 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

31	RS	RA	RB	28	Rc
0	6	11	16	21	31

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

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.

andc

AND with Complement

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

31	RS	RA	RB	60	Rc
0	6	11	16	21 2	31

$(RA) \leftarrow (RS) \wedge \neg(RB)$

The contents of register RS are ANDed with the ones complement of the contents of register RB; the result is placed into register RA.

Registers Altered

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

Architecture Note

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

andi. RA, RS, IM

28	RS	RA	IM
0	6	11	16 31

$(RA) \leftarrow (RS) \wedge ({}^{16}0 \,||\, IM)$

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

29	RS	RA	IM
0	6	11	16 31

$(RA) \leftarrow (RS) \wedge (IM \parallel {}^{16}0)$

The IM field is extended to 32 bits by concatenating 16 0-bits on its right. The contents of register RS are ANDed with the extended IM field; the result is placed into register RA.

Registers Altered

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

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

18	LI	AA	LK
0	6	30	31

```
If AA = 1 then
   LI ← $target_{6:29}$
   NIA ← EXTS(LI || $^{2}0$)
else
   LI ← $(target - CIA)_{6:29}$
   NIA ← CIA + EXTS(LI || $^{2}0$)
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

16	BO	BI	BD	AA	LK
0	6	11	16	30	31

```
if $BO_2$ = 0 then
   CTR $\leftarrow$ CTR – 1
if ($BO_2$ = 1 $\vee$ ((CTR = 0) = $BO_3$)) $\wedge$ ($BO_0$ = 1 $\vee$ ($CR_{BI}$ = $BO_1$)) then
   if AA = 1 then
      BD $\leftarrow$ $target_{16:29}$
      NIA $\leftarrow$ EXTS(BD || $^20$)
   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. *Extended mnemonic for* **bc 16,0,target**	
bdnza		*Extended mnemonic for* **bca 16,0,target**	
bdnzl		*Extended mnemonic for* **bcl 16,0,target**	(LR) $\leftarrow$ CIA + 4.
bdnzla		*Extended mnemonic for* **bcla 16,0,target**	(LR) $\leftarrow$ CIA + 4.
bdnzf	cr_bit, target	Decrement CTR. Branch if CTR $\neq$ 0 AND CR_{cr_bit} = 0. *Extended mnemonic for* **bc 0,cr_bit,target**	
bdnzfa		*Extended mnemonic for* **bca 0,cr_bit,target**	
bdnzfl		*Extended mnemonic for* **bcl 0,cr_bit,target**	(LR) $\leftarrow$ CIA + 4.
bdnzfla		*Extended mnemonic for* **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. *Extended mnemonic for* **bc 8,cr_bit,target**	
bdnzta		*Extended mnemonic for* **bca 8,cr_bit,target**	
bdnztl		*Extended mnemonic for* **bcl 8,cr_bit,target**	(LR) $\leftarrow$ CIA + 4.
bdnztla		*Extended mnemonic for* **bcla 8,cr_bit,target**	(LR) $\leftarrow$ CIA + 4.
bdz	target	Decrement CTR; branch if CTR = 0. *Extended mnemonic for* **bc 18,0,target**	
bdza		*Extended mnemonic for* **bca 18,0,target**	
bdzl		*Extended mnemonic for* **bcl 18,0,target**	(LR) $\leftarrow$ CIA + 4.
bdzla		*Extended mnemonic for* **bcla 18,0,target**	(LR) $\leftarrow$ CIA + 4.

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. *Extended mnemonic for* **bc 2,cr_bit,target**	
bdzfa		*Extended mnemonic for* **bca 2,cr_bit,target**	
bdzfl		*Extended mnemonic for* **bcl 2,cr_bit,target**	(LR) ← CIA + 4.
bdzfla		*Extended mnemonic for* **bcla 2,cr_bit,target**	(LR) ← CIA + 4.
bdzt	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 1. *Extended mnemonic for* **bc 10,cr_bit,target**	
bdzta		*Extended mnemonic for* **bca 10,cr_bit,target**	
bdztl		*Extended mnemonic for* **bcl 10,cr_bit,target**	(LR) ← CIA + 4.
bdztla		*Extended mnemonic for* **bcla 10,cr_bit,target**	(LR) ← CIA + 4.
beq	[cr_field,] target	Branch if equal. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+2,target**	
beqa		*Extended mnemonic for* **bca 12,4∗cr_field+2,target**	
beql		*Extended mnemonic for* **bcl 12,4∗cr_field+2,target**	(LR) ← CIA + 4.
beqla		*Extended mnemonic for* **bcla 12,4∗cr_field+2,target**	(LR) ← CIA + 4.
bf	cr_bit, target	Branch if CR_{cr_bit} = 0. *Extended mnemonic for* **bc 4,cr_bit,target**	
bfa		*Extended mnemonic for* **bca 4,cr_bit,target**	
bfl		*Extended mnemonic for* **bcl 4,cr_bit,target**	LR
bfla		*Extended mnemonic for* **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. *Extended mnemonic for* **bc 4,4∗cr_field+0,target**	
bgea		*Extended mnemonic for* **bca 4,4∗cr_field+0,target**	
bgel		*Extended mnemonic for* **bcl 4,4∗cr_field+0,target**	LR
bgela		*Extended mnemonic for* **bcla 4,4∗cr_field+0,target**	LR
bgt	[cr_field,] target	Branch if greater than. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+1,target**	
bgta		*Extended mnemonic for* **bca 12,4∗cr_field+1,target**	
bgtl		*Extended mnemonic for* **bcl 12,4∗cr_field+1,target**	LR
bgtla		*Extended mnemonic for* **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. *Extended mnemonic for* **bc 4,4∗cr_field+1,target**	
blea		*Extended mnemonic for* **bca 4,4∗cr_field+1,target**	
blel		*Extended mnemonic for* **bcl 4,4∗cr_field+1,target**	LR
blela		*Extended mnemonic for* **bcla 4,4∗cr_field+1,target**	LR
blt	[cr_field,] target	Branch if less than Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+0,target**	
blta		*Extended mnemonic for* **bca 12,4∗cr_field+0,target**	
bltl		*Extended mnemonic for* **bcl 12,4∗cr_field+0,target**	(LR) ← CIA + 4.
bltla		*Extended mnemonic for* **bcla 12,4∗cr_field+0,target**	(LR) ← CIA + 4.

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. *Extended mnemonic for* **bc 4,4∗cr_field+2,target**	
bnea		*Extended mnemonic for* **bca 4,4∗cr_field+2,target**	
bnel		*Extended mnemonic for* **bcl 4,4*cr_field+2,target**	(LR) ← CIA + 4.
bnela		*Extended mnemonic for* **bcla 4,4∗cr_field+2,target**	(LR) ← CIA + 4.
bng	[cr_field,] target	Branch if not greater than. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+1,target**	
bnga		*Extended mnemonic for* **bca 4,4∗cr_field+1,target**	
bngl		*Extended mnemonic for* **bcl 4,4∗cr_field+1,target**	(LR) ← CIA + 4.
bngla		*Extended mnemonic for* **bcla 4,4∗cr_field+1,target**	(LR) ← CIA + 4.
bnl	[cr_field,] target	Branch if not less than; use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+0,target**	
bnla		*Extended mnemonic for* **bca 4,4∗cr_field+0,target**	
bnll		*Extended mnemonic for* **bcl 4,4∗cr_field+0,target**	(LR) ← CIA + 4.
bnlla		*Extended mnemonic for* **bcla 4,4∗cr_field+0,target**	(LR) ← CIA + 4.
bns	[cr_field,] target	Branch if not summary overflow. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+3,target**	
bnsa		*Extended mnemonic for* **bca 4,4∗cr_field+3,target**	
bnsl		*Extended mnemonic for* **bcl 4,4∗cr_field+3,target**	(LR) ← CIA + 4.
bnsla		*Extended mnemonic for* **bcla 4,4∗cr_field+3,target**	(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. *Extended mnemonic for* **bc 4,4*cr_field+3,target**	
bnua		*Extended mnemonic for* **bca 4,4*cr_field+3,target**	
bnul		*Extended mnemonic for* **bcl 4,4*cr_field+3,target**	(LR) ← CIA + 4.
bnula		*Extended mnemonic for* **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. *Extended mnemonic for* **bc 12,4*cr_field+3,target**	
bsoa		*Extended mnemonic for* **bca 12,4*cr_field+3,target**	
bsol		*Extended mnemonic for* **bcl 12,4*cr_field+3,target**	(LR) ← CIA + 4.
bsola		*Extended mnemonic for* **bcla 12,4*cr_field+3,target**	(LR) ← CIA + 4.
bt	cr_bit, target	Branch if $CR_{cr\_bit} = 1$. *Extended mnemonic for* **bc 12,cr_bit,target**	
bta		*Extended mnemonic for* **bca 12,cr_bit,target**	
btl		*Extended mnemonic for* **bcl 12,cr_bit,target**	(LR) ← CIA + 4.
btla		*Extended mnemonic for* **bcla 12,cr_bit,target**	(LR) ← CIA + 4.
bun	[cr_field], target	Branch if unordered. Use CR0 if *cr_field* is omitted. *Extended mnemonic for* **bc 12,4*cr_field+3,target**	
buna		*Extended mnemonic for* **bca 12,4*cr_field+3,target**	
bunl		*Extended mnemonic for* **bcl 12,4*cr_field+3,target**	(LR) ← CIA + 4.
bunla		*Extended mnemonic for* **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

19	BO	BI	▨	528	LK
0	6	11	16	21	31

```
if $BO_2$ = 0 then
   CTR ← CTR – 1
if ($BO_2$ = 1 ∨ ((CTR = 0) = $BO_3$)) ∧ ($BO_0$ = 1 ∨ ($CR_{BI}$ = $BO_1$)) then
   NIA ← $CTR_{0:29}$ || $^{2}0$
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 BO_2 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. *Extended mnemonic for* **bcctr 20,0**	
bctrl		*Extended mnemonic for* **bcctrl 20,0**	(LR) ← CIA + 4.
beqctr	[cr_field]	Branch, if equal, to address in CTR Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+2**	
beqctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+2**	(LR) ← CIA + 4.
bfctr	cr_bit	Branch, if CR_{cr_bit} = 0, to address in CTR. *Extended mnemonic for* **bcctr 4,cr_bit**	
bfctrl		*Extended mnemonic for* **bcctrl 4,cr_bit**	(LR) ← CIA + 4.
bgectr	[cr_field]	Branch, if greater than or equal, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+0**	
bgectrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+0**	(LR) ← CIA + 4.
bgtctr	[cr_field]	Branch, if greater than, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+1**	
bgtctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+1**	(LR) ← CIA + 4.
blectr	[cr_field]	Branch, if less than or equal, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+1**	
blectrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+1**	(LR) ← CIA + 4.
bltctr	[cr_field]	Branch, if less than, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+0**	
bltctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+0**	(LR) ← CIA + 4.

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

Mnemonic	Operands	Function	Other Registers Altered
bnectr	[cr_field]	Branch, if not equal, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+2**	
bnectrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+2**	(LR) ← CIA + 4.
bngctr	[cr_field]	Branch, if not greater than, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+1**	
bngctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+1**	(LR) ← CIA + 4.
bnlctr	[cr_field]	Branch, if not less than, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+0**	
bnlctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+0**	(LR) ← CIA + 4.
bnsctr	[cr_field]	Branch, if not summary overflow, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+3**	
bnsctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+3**	(LR) ← CIA + 4.
bnuctr	[cr_field]	Branch, if not unordered, to address in CTR; use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+3**	
bnuctrl		*Extended mnemonic for* **bcctrl 4,4∗cr_field+3**	(LR) ← CIA + 4.
bsoctr	[cr_field]	Branch, if summary overflow, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+3**	
bsoctrl		*Extended mnemonic for* **bcctrl 12,4∗cr_field+3**	(LR) ← CIA + 4.
btctr	cr_bit	Branch if CR_{cr_bit} = 1 to address in CTR. *Extended mnemonic for* **bcctr 12,cr_bit**	
btctrl		*Extended mnemonic for* **bcctrl 12,cr_bit**	(LR) ← CIA + 4.

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

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

bclr

Branch Conditional to Link Register

bclr	BO, BI	LK=0
bclrl	BO, BI	LK=1

19	BO	BI	▨	16	LK
0	6	11	16	21	31

```
if $BO_2$ = 0 then
    CTR ← CTR − 1
if ($BO_2$ = 1 ∨ ((CTR = 0) = $BO_3$)) ∧ ($BO_0$ = 1 ∨ ($CR_{BI}$ = $BO_1$)) then
    NIA ← $LR_{0:29}$ || $^2$0
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 BO_2 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. *Extended mnemonic for* **bclr 20,0**	
blrl		*Extended mnemonic for* **bclrl 20,0**	(LR) ← CIA + 4.

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

Mnemonic	Operands	Function	Other Registers Altered
bdnzlr		Decrement CTR. Branch if CTR ≠ 0 to address in LR. *Extended mnemonic for* **bclr 16,0**	
bdnzlrl		*Extended mnemonic for* **bclrl 16,0**	(LR) ← CIA + 4.
bdnzflr	cr_bit	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 0 to address in LR. *Extended mnemonic for* **bclr 0,cr_bit**	
bdnzflrl		*Extended mnemonic for* **bclrl 0,cr_bit**	(LR) ← CIA + 4.
bdnztlr	cr_bit	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 1 to address in LR. *Extended mnemonic for* **bclr 8,cr_bit**	
bdnztlrl		*Extended mnemonic for* **bclrl 8,cr_bit**	(LR) ← CIA + 4.
bdzlr		Decrement CTR. Branch if CTR = 0 to address in LR. *Extended mnemonic for* **bclr 18,0**	
bdzlrl		*Extended mnemonic for* **bclrl 18,0**	(LR) ← CIA + 4.
bdzflr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 0 to address in LR. *Extended mnemonic for* **bclr 2,cr_bit**	
bdzflrl		*Extended mnemonic for* **bclrl 2,cr_bit**	(LR) ← CIA + 4.
bdztlr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 1 to address in LR. *Extended mnemonic for* **bclr 10,cr_bit**	
bdztlrl		*Extended mnemonic for* **bclrl 10,cr_bit**	(LR) ← CIA + 4.
beqlr	[cr_field]	Branch if equal to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 12,4∗cr_field+2**	
beqlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+2**	(LR) ← CIA + 4.

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

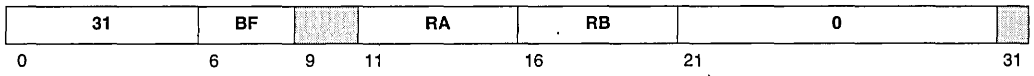
Mnemonic	Operands	Function	Other Registers Altered
bflr	cr_bit	Branch if CR_{cr_bit} = 0 to address in LR. *Extended mnemonic for* **bclr 4,cr_bit**	
bflrl		*Extended mnemonic for* **bclrl 4,cr_bit**	(LR) ← CIA + 4.
bgelr	[cr_field]	Branch, if greater than or equal, to address in LR. Use CR0 if cr_field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

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

Mnemonic	Operands	Function	Other Registers Altered
bnllr	[cr_field]	Branch, if not less than, to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 4,4*cr_field+0**	
bnllrl		*Extended mnemonic for* **bclrl 4,4*cr_field+0**	(LR) ← CIA + 4.
bnslr	[cr_field]	Branch if not summary overflow to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 4,4*cr_field+3**	
bnslrl		*Extended mnemonic for* **bclrl 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. *Extended mnemonic for* **bclr 4,4*cr_field+3**	
bnulrl		*Extended mnemonic for* **bclrl 4,4*cr_field+3**	(LR) ← CIA + 4.
bsolr	[cr_field]	Branch if summary overflow to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 12,4*cr_field+3**	
bsolrl		*Extended mnemonic for* **bclrl 12,4*cr_field+3**	(LR) ← CIA + 4.
btlr	cr_bit	Branch if CR_{cr_bit} = 1 to address in LR. *Extended mnemonic for* **bclr 12,cr_bit**	
btlrl		*Extended mnemonic for* **bclrl 12,cr_bit**	(LR) ← CIA + 4.
bunlr	[cr_field]	Branch if unordered to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 12,4*cr_field+3**	
bunlrl		*Extended mnemonic for* **bclrl 12,4*cr_field+3**	(LR) ← CIA + 4.

cmp
Compare

cmp BF, 0, RA, RB



```
$c_{0:3} \leftarrow {}^{4}0$
if (RA) < (RB) then $c_0 \leftarrow 1$
if (RA) > (RB) then $c_1 \leftarrow 1$
if (RA) = (RB) then $c_2 \leftarrow 1$
$c_3 \leftarrow$ XER[SO]
n ← BF
CR[CRn] ← $c_{0:3}$
```

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[CR*n*] 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. *Extended mnemonic for* **cmp BF,0,RA,RB**	

cmpi BF, 0, RA, IM

11	BF		RA	IM
0	6	9	11	16 31

```
$c_{0:3} \leftarrow {}^{4}0$
if (RA) < EXTS(IM) then $c_0 \leftarrow 1$
if (RA) > EXTS(IM) then $c_1 \leftarrow 1$
if (RA) = EXTS(IM) then $c_2 \leftarrow 1$
$c_3 \leftarrow$ XER[SO]
n ← BF
CR[CRn] ← $c_{0:3}$
```

The IM field is sign-extended to 32 bits. The contents of register RA are compared with the extended IM field, using a 32-bit signed compare.

The CR field specified by the BF field is updated to reflect the results of the compare and the value of XER[SO] is placed into the same CR field.

Registers Altered

- CR[CR*n*] where *n* is specified by the BF field

Invalid Instruction Forms

- Reserved fields

Programming Note

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. *Extended mnemonic for* **cmpi BF,0,RA,IM**	

cmpl

Compare Logical

cmpl BF, 0, RA, RB

31	BF		RA	RB	32	
0	6	9	11	16	21	31

```
$c_{0:3} \leftarrow {}^{4}0$
if (RA) $\overset{u}{<}$ (RB) then $c_0 \leftarrow 1$
if (RA) $\overset{u}{>}$ (RB) then $c_1 \leftarrow 1$
if (RA) $=$ (RB) then $c_2 \leftarrow 1$
$c_3 \leftarrow$ XER[SO]
n $\leftarrow$ BF
CR[CRn] $\leftarrow c_{0:3}$
```

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[CR*n*] 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. *Extended mnemonic for* **cmpl BF,0,RA,RB**	

cmpli BF, 0, RA, IM

10	BF		RA	IM
0	6	9	11	16 31

```
$c_{0:3} \leftarrow {}^{4}0$
if (RA) $\overset{u}{<}$ ($^{16}0$ || IM) then $c_0 \leftarrow 1$
if (RA) $\overset{u}{>}$ ($^{16}0$ || IM) then $c_1 \leftarrow 1$
if (RA) $=$ ($^{16}0$ || IM) then $c_2 \leftarrow 1$
$c_3 \leftarrow$ XER[SO]
n $\leftarrow$ BF
CR[CRn] $\leftarrow c_{0:3}$
```

The IM field is extended to 32 bits by concatenating 16 0-bits to its left. The contents of register RA are compared with IM using a 32-bit unsigned compare.

The CR field specified by the BF field is updated to reflect the results of the compare and the value of XER[SO] is placed into the same CR field.

Registers Altered

- CR[CR*n*] where *n* is specified by the BF field

Invalid Instruction Forms

- Reserved fields

Programming Note

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. *Extended mnemonic for* **cmpli BF,0,RA,IM**	

cntlzw

Count Leading Zeros Word

cntlzw	RA, RS	Rc=0
cntlzw.	RA, RS	Rc=1

31	RS	RA	▨	26	Rc
0	6	11	16	21	31

```
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

Condition Register AND

crand BT, BA, BB

19	BT	BA	BB	257	
0	6	11	16	21	31

$CR_{BT} \leftarrow CR_{BA} \wedge CR_{BB}$

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

19	BT	BA	BB	129	▨
0	6	11	16	21	31

$CR_{BT} \leftarrow CR_{BA} \wedge \neg CR_{BB}$

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

19	BT	BA	BB	289	
0	6	11	16	21	31

$CR_{BT} \leftarrow \neg(CR_{BA} \oplus CR_{BB})$

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. *Extended mnemonic for* **creqv bx,bx,bx**	

crnand

Condition Register NAND

crnand BT, BA, BB

19	BT	BA	BB	225	▨
0	6	11	16	21	31

$CR_{BT} \leftarrow \neg(CR_{BA} \wedge CR_{BB})$

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

19	BT	BA	BB	33	▨
0	6	11	16	21	31

$CR_{BT} \leftarrow \neg(CR_{BA} \vee CR_{BB})$

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. *Extended mnemonic for* **crnor bx,by,by**	

cror

Condition Register OR

cror BT, BA, BB

19	BT	BA	BB	449	▨
0	6	11	16	21	31

$CR_{BT} \leftarrow CR_{BA} \vee CR_{BB}$

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. *Extended mnemonic for* **cror bx,by,by**	

crorc BT, BA, BB

19	BT	BA	BB	417	
0	6	11	16	21	31

$CR_{BT} \leftarrow CR_{BA} \vee \neg CR_{BB}$

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

19	BT	BA	BB	193	▨
0	6	11	16	21	31

$CR_{BT} \leftarrow CR_{BA} \oplus CR_{BB}$

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. *Extended mnemonic for* **crxor bx,bx,bx**	

dcba RA, RB

31		RA	RB	758	
0	6	11	16	21	31

```
EA ← (RA|0) + (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

31	▨	RA	RB	86	▨
0	6	11	16	21	31

```
EA ← (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

31		RA	RB	470	
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
DCBI(EA)
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the data block at the EA is in the data cache, the data block is marked invalid, regardless of whether or not the 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

31		RA	RB	54	
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
DCBST(EA)
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0, and is the contents of register RA otherwise.

If the data block at the EA is in the data cache and marked as modified, the data block is copied back to main storage and marked as unmodified in the data cache.

If the data block at the EA is in the data cache, and is not marked as modified, or if the data block at the EA is not in the data cache, no operation is performed.

The operation specified by this instruction is performed whether or not the 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

31	▨	RA	RB	278	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
DCBT(EA)
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

If the data block at the EA is not in the data cache and the 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

31		RA	RB	246	
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
DCBTST(EA)
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the data block at the EA is not in the data cache and the 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

31	▨	RA	RB	1014	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
DCBZ(EA)
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the data block at the EA is in the data cache and the 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

31	▨	RA	RB	454	▨
0	6	11	16	21	31

```
EA ← (RA|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 **dccci** 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 **dccci** 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

31	RT	RA	RB	486	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
if ((CCR0[CIS] = 0) ∧ (CCR0[CWS] = 0)) then (RT) ← (d-cache data, way A)
if ((CCR0[CIS] = 0) ∧ (CCR0[CWS] = 1)) then (RT) ← (d-cache data, way B)
if ((CCR0[CIS] = 1) ∧ (CCR0[CWS] = 0)) then (RT) ← (d-cache tag, way A)
if ((CCR0[CIS] = 1) ∧ (CCR0[CWS] = 1)) then (RT) ← (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

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

31	**RT**	**RA**	**RB**	**OE**	**491**	**Rc**
0	6	11	16	21	22	31

(RT) ← (RA) ÷ (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:

dividend = (quotient × divisor) + 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 ÷ −1) or (n ÷ 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

31	RT	RA	RB	OE	459	Rc
0	6	11	16	21	22	31

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

The contents of register RA are divided by the contents of register RB. The quotient is placed into register RT.

The dividend and the divisor are interpreted as unsigned integers. The quotient is the unique unsigned integer that satisfies:

dividend = (quotient $\times$ divisor) + 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

31		854	
0	6	21	31

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.

eqv

Equivalent

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

31	RS	RA	RB	284	Rc
0	6	11	16	21	31

$(RA) \leftarrow \neg((RS) \oplus (RB))$

The contents of register RS are XORed with the contents of register RB; the ones complement of the result is placed into register RA.

Registers Altered

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

Architecture Note

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

extsb	RA, RS	Rc=0
extsb.	RA, RS	Rc=1

31	RS	RA		954	Rc
0	6	11	16	21	31

(RA) $\leftarrow$ EXTS(RS)$_{24:31}$

The least significant byte of register RS is sign-extended to 32 bits by replicating bit 24 of the register into bits 0 through 23 of the result. The result is placed into register RA.

Registers Altered

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

Invalid Instruction Forms

- Reserved fields

Architecture Note

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

extsh

Extend Sign Halfword

extsh	RA, RS	Rc=0
extsh.	RA, RS	Rc=1

31	RS	RA	▨	922	Rc
0	6	11	16	21	31

(RA) ← EXTS(RS)$_{16:31}$

The least significant halfword of register RS is sign-extended to 32 bits by replicating bit 16 of the register into bits 0 through 15 of the result. The result is placed into register RA.

Registers Altered

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

Invalid Instruction Forms

- Reserved fields

Architecture Note

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

icbi RA, RB

31		RA	RB	982	
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
ICBI(EA)
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the instruction block at the EA is in the instruction cache, the cache block is marked invalid.

If the instruction block at the EA is not in the instruction cache, no additional operation is performed.

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

31	▨	RA	RB	262	▨
0	6	11	16	21	31

```
EA← (RA|0) + (RB)
ICBT(EA)
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

If the instruction block at the EA is not in the instruction cache, and 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.

31		RA	RB	966	
0	6	11	16	21	31

EA ← (RA|0) + (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

31	▨	RA	RB	998	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
if ((CCR0[CIS] = 0) ∧ (CCR0[CWS] = 0)) then (ICDBDR) ← (i-cache data, way A)
if ((CCR0[CIS] = 0) ∧ (CCR0[CWS] = 1)) then (ICDBDR) ← (i-cache data, way B)
if ((CCR0[CIS] = 1) ∧ (CCR0[CWS] = 0)) then (ICDBDR) ← (i-cache tag, way A)
if ((CCR0[CIS] = 1) ∧ (CCR0[CWS] = 1)) then (ICDBDR) ← (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 **mficdbdr**.

```
icread r5,r6  # read cache information
isync         # ensure completion of icread
mficdbdr 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.

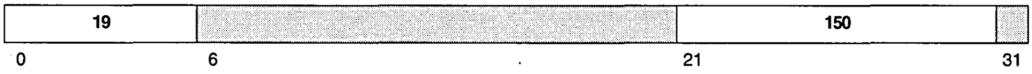
Architecture Note

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 cachable.

```
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)

34	RT	RA	D	
0	6	11	16	31

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

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)

35	RT	RA	D	
0	6	11	16	31

```
EA ← (RA|0) + EXTS(D)
(RA) ← EA
(RT) ← $^{24}0$ || 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

31	RT	RA	RB	119	▨
0	6	11	16	21	31

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

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The byte at the EA is extended to 32 bits by concatenating 24 0-bits to its left. The result is placed into register RT.

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

Registers Altered

- RA
- RT

Invalid Instruction Forms

- Reserved fields
- RA=RT
- RA=0

Architecture Note

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

lbzx

Load Byte and Zero Indexed

lbzx RT,RA, RB

31	RT	RA	RB	87	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
(RT) ← $^{24}0$ || MS(EA,1)
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The byte at the EA is extended to 32 bits by concatenating 24 0-bits to its left. The result is placed into register RT.

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

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

Architecture Note

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

lha RT, D(RA)

42	RT	RA	D	
0	6	11	16	31

```
EA ← (RA|0) + EXTS(D)
(RT) ← EXTS(MS(EA,2))
```

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 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)

43	RT	RA	D	
0	6	11	16	31

```
EA ← (RA|0) + EXTS(D)
(RA) ← EA
(RT) ← EXTS(MS(EA,2))
```

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The halfword at the EA is sign-extended to 32 bits and placed into register RT.

Registers Altered

- RA
- RT

Invalid Instruction Forms

- RA = RT
- RA = 0

Architecture Note

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

lhaux RT, RA, RB

31	RT	RA	RB	375	
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
(RA) ← EA
(RT) ← 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

31	RT	RA	RB	343	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
(RT) ← EXTS(MS(EA,2))
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The halfword at the EA is sign-extended to 32 bits and placed into register RT.

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

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

Architecture Note

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

lhbrx RT, RA, RB

31	RT	RA	RB	790	
0	6	11	16	21	31

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

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)

40	RT	RA	D	
0	6	11	16	31

```
$EA \leftarrow (RA|0) + EXTS(D)$
$(RT) \leftarrow {}^{16}0 \parallel MS(EA,2)$
```

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

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

Registers Altered

- RT

Architecture Note

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

lhzu RT, D(RA)

41	RT	RA	D	
0	6	11	16	31

```
$EA \leftarrow (RA|0) + EXTS(D)$
$(RA) \leftarrow EA$
$(RT) \leftarrow {}^{16}0 \parallel MS(EA,2)$
```

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

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

Registers Altered

- RA
- RT

Invalid Instruction Forms

- RA = RT
- RA = 0

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**

31	RT	RA	RB	311	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
(RA) ← EA
(RT) ← $^{16}0$ || MS(EA,2)
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The 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

31	RT	RA	RB	279	
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
(RT) ← $^{16}0$ || MS(EA,2)
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

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

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

Registers Altered

- RT

Invalid Instruction Forms

- Reserved fields

Architecture Note

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

lmw

Load Multiple Word

lmw RT, D(RA)

46	RT	RA	D	
0	6	11	16	31

```
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

31	RT	RA	NB	597	
0	6	11	16	21	31

```
EA ← (RA|0)
if NB = 0 then
   CNT ← 32
else
   CNT ← NB
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 = $R_{FINAL}$)) then
         (GPR(r)) ← 0
   if ((r ≠ RA) ∨ (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 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.

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

31	RT	RA	RB	533	▨
0	6	11	16	21	31

```
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 $(((r \neq RA) \wedge (r \neq RB)) \vee (r = R_{FINAL}))$ then
      (GPR(r)) ← 0
  if $(((r \neq RA) \wedge (r \neq RB)) \vee (r = R_{FINAL}))$ 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

31	RT	RA	RB	20	
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
RESERVE ← 1
(RT) ← 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 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

31	RT	RA	RB	534	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
(RT) ← MS(EA+3,1) || MS(EA+2,1) || MS(EA+1,1) || 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)

32	RT	RA	D	
0	6	11	16	31

```
EA ← (RA|0) + EXTS(D)
(RT) ← MS(EA,4)
```

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise.

The 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)

33	RT	RA	D
0	6	11	16 31

```
EA ← (RA|0) + EXTS(D)
(RA) ← EA
(RT) ← MS(EA,4)
```

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 if the RA field is 0 and is the contents of register RA otherwise. The EA is placed into register RA.

The word at the EA is placed into register RT.

Registers Altered

- RA
- RT

Invalid Instruction Forms

- RA = RT
- RA = 0

Architecture Note

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

lwzux RT, RA, RB

31	RT	RA	RB	55	
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
(RA) ← EA
(RT) ← 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

31	RT	RA	RB	23	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
(RT) ← 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

Multiply Accumulate Cross Halfword to Word Modulo Signed

macchw	RT, RA, RB	OE=0, Rc=0
macchw.	RT, RA, RB	OE=0, Rc=1
macchwo	RT, RA, RB	OE=1, Rc=0
macchwo.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	172	Rc
0	6	11	16	21	22	31

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

4	RT	RA	RB	OE	236	Rc
0	6	11	16	21	22	31

```
$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ signed
$temp_{0:32} \leftarrow prod_{0:31} + (RT)$
if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \,||\, {}^{31}(\neg RT_0))$
else $(RT) \leftarrow temp_{1:32}$
```

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

4	RT	RA	RB	OE	204	Rc
0	6	11	16	21	22	31

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

4	RT	RA	RB	OE	140	Rc
0	6	11	16	21	22	31

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

4	RT	RA	RB	OE	44	Rc
0	6	11	16	21	22	31

```
$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ 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
machhwso	RT, RA, RB	OE=1, Rc=0
machhwso.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	108	Rc
0	6	11	16	21	22	31

```
$prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ signed
$temp_{0:32} \leftarrow prod_{0:31} + (RT)$
if $((prod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \,\|\, {}^{31}(\neg RT_0))$
else $(RT) \leftarrow temp_{1:32}$
```

The high-order halfword of RA is multiplied by the high-order halfword of RB. The signed product is 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

4	RT	RA	RB	OE	76	Rc
0	6	11	16	21	22	31

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

4	RT	RA	RB	OE	12	Rc
0	6	11	16	21	22	31

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

4	RT	RA	RB	OE	428	Rc
0	6	11	16	21	22	31

```
$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ signed
$temp_{0:32} \leftarrow prod_{0:31} + (RT)$
$(RT) \leftarrow temp_{1:32}$
```

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
maclhwso	RT, RA, RB	OE=1, Rc=0
maclhwso.	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 $(RT) \leftarrow (RT_0 \,\|\, {}^{31}(\neg RT_0))$
else $(RT) \leftarrow temp_{1:32}$
```

The low-order halfword of RA is multiplied by the low-order halfword of RB. The signed product is 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.

maclhwsu	RT, RA, RB	OE=0, Rc=0
maclhwsu.	RT, RA, RB	OE=0, Rc=1
maclhwsuo	RT, RA, RB	OE=1, Rc=0
maclhwsuo.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	460	Rc
0	6	11	16	21	22	31

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

4	RT	RA	RB	OE	396	Rc
0	6	11	16	21	22	31

```
$prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ 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 BF, BFA

19	BF	▨	BFA	▨	0	▨
0	6	9	11	14	21	31

```
m ← BFA
n ← BF
(CR[CRn]) ← (CR[CRm])
```

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

Registers Altered

- CR[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

31	BF	▨	512	▨
0	6	9	21	31

```
$n \leftarrow BF$
$(CR[CRn]) \leftarrow XER_{0:3}$
$XER_{0:3} \leftarrow {}^{4}0$
```

The contents of $XER_{0:3}$ are placed into the CR field specified by the BF field. $XER_{0:3}$ 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 CR[CR0] are undefined.

Registers Altered

- CR[CR*n*] where *n* is specified by the BF field.
- XER[SO, OV, CA]

Invalid Instruction Forms

- Reserved fields

Architecture Note

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

mfcr RT

31	RT		19	
0	6	11	21	31

(RT) ← (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

31	RT	DCRF	323	▨
0	6	11	21	31

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

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

31	RT	▨	83	▨
0	6	11	21	31

(RT) ← (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

31	RT	SPRF	339	▨
0	6	11	21	31

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

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 **mfspr** 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 mfspr

Mnemonic	Operands	Function	Other Registers Changed
mfccr0 **mfctr** **mfdac1** **mfdac2** **mfdear** **mfdbcr0** **mfdbcr1** **mfdbsr** **mfdccr** **mfdcwr** **mfdvc1** **mfdvc2** **mfesr** **mfevpr** **mfiac1** **mfiac2** **mfiac3** **mfiac4** **mficcr** **mficdbdr** **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. *Extended mnemonic for* **mfspr 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

31	RT	TBRF	371	
0	6	11	21	31

$TBRN \leftarrow TBRF_{5:9} \parallel TBRF_{0:4}$
$(RT) \leftarrow (TBR(TBRN))$

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.

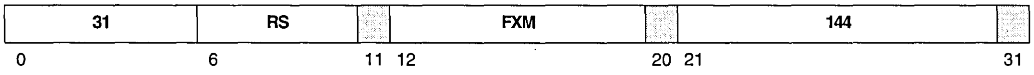
Table 24-21. Extended Mnemonics for mftb

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

mtcrf

Move to Condition Register Fields

mtcrf FXM, RS



```
mask ← $^4(FXM_0) \| ^4(FXM_1) \| ... \| ^4(FXM_6) \| ^4(FXM_7)$
(CR) ← $((RS) \land mask) \lor ((CR) \land \neg mask)$
```

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. *Extended mnemonic for* **mtcrf 0xFF,RS**	

mtdcr DCRN, RS

31	RS	DCRF	451	
0	6	11	21	31

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

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

31	RS	▨	146	▨
0	6	11	21	31

(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

31	RS	SPRF	467	▨
0	6	11	21	31

SPRN $\leftarrow$ $\text{SPRF}_{5:9}$ || $\text{SPRF}_{0:4}$
(SPR(SPRN)) $\leftarrow$ (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.

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** **mtlr** **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 special purpose register SPRN. *Extended mnemonic for* **mtspr SPRN,RS** See “Special Purpose Registers” on page 25-1 for a list of valid SPRN values.	

mulchw

Multiply Cross Halfword to Word Signed

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

4	RT	RA	RB	168	Rc
0	6	11	16	21	31

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

4	RT	RA	RB	136	Rc
0	6	11	16	21	31

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

4	RT	RA	RB	40	Rc
0	6	11	16	21	31

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

mulhhwu

Multiply High Halfword to Word Unsigned

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

4	RT	RA	RB	8	Rc
0	6	11	16	21	31

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

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

31	RT	RA	RB		75	Rc
0	6	11	16	21	22	31

```
prod$_{0:63}$ ← (RA) × (RB) signed
(RT) ← prod$_{0:31}$
```

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

Registers Altered

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

Architecture Note

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

mulhwu

Multiply High Word Unsigned

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

31	RT	RA	RB	▨	11	Rc
0	6	11	16	21		31

```
$prod_{0:63} \leftarrow (RA) \times (RB)$ 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	RT, RA, RB	Rc=0
mullhw.	RT, RA, RB	Rc=1

4	RT	RA	RB	424	Rc
0	6	11	16	21	31

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

4	RT	RA	RB	392	Rc
0	6	11	16	21	31

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

mulli RT, RA, IM

7	RT	RA	IM	
0	6	11	16	31

```
$prod_{0:47} \leftarrow (RA) \times EXTS(IM)$ 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

31	RT	RA	RB	OE	235	Rc
0	6	11	16	21	22	31

```
$prod_{0:63} \leftarrow$ (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
NAND

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

31	RT	RA	RB	476	Rc
0	6	11	16	21	31

(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

31	RT	RA	▨	OE	104	Rc
0	6	11	16	21	22	31

(RT) ← ¬(RA) + 1

The twos 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
nmacchwo	RT, RA, RB	OE=1, Rc=0
nmacchwo.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	174	Rc
0	6	11	16	21	22	31

```
$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{0:15})$ signed
$temp_{0:32} \leftarrow nprod_{0:31} + (RT)$
$(RT) \leftarrow temp_{1:32}$
```

The low-order halfword of RA is multiplied by the high-order halfword of RB. The 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

4	RT	RA	RB	OE	238	Rc
0	6	11	16	21	22	31

```
$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{0:15}$ signed
$temp_{0:32} \leftarrow nprod_{0:31} + (RT)$
if $((nprod_0 = RT_0) \land (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \,||\, ^{31}(\neg RT_0))$
else $(RT) \leftarrow temp_{1:32}$
```

The low-order halfword of RA is multiplied by the high-order halfword of RB. The 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	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

4	RT	RA	RB	OE	46	Rc
0	6	11	16	21	22	31

```
nprod$_{0:31}$ ← –((RA)$_{0:15}$ x (RB)$_{0:15}$) signed
temp$_{0:32}$ ← nprod$_{0:31}$ + (RT)
(RT) ← 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
nmachhwso	RT, RA, RB	OE=1, Rc=0
nmachhwso.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	110	Rc
0	6	11	16	21	22	31

```
$nprod_{0:31} \leftarrow -((RA)_{0:15} \times (RB)_{0:15})$ signed
$temp_{0:32} \leftarrow nprod_{0:31} + (RT)$
if $((nprod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \,\|\, ^{31}(\neg RT_0))$
else $(RT) \leftarrow temp_{1:32}$
```

The high-order halfword of RA is multiplied by the high-order halfword of RB. The 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.

nmaclhw	RT, RA, RB	OE=0, Rc=0
nmaclhw.	RT, RA, RB	OE=0, Rc=1
nmaclhwo	RT, RA, RB	OE=1, Rc=0
nmachlwo.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	430	Rc
0	6	11	16	21	22	31

```
$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{16:31})$ signed
$temp_{0:32} \leftarrow nprod_{0:31} + (RT)$
$(RT) \leftarrow temp_{1:32}$
```

The low-order halfword of RA is multiplied by the low-order halfword of RB. The 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
nmachlwso.	RT, RA, RB	OE=1, Rc=1

4	RT	RA	RB	OE	494	Rc
0	6	11	16	21	22	31

```
$nprod_{0:31} \leftarrow -((RA)_{16:31} \times (RB)_{16:31})$ signed
$temp_{0:32} \leftarrow nprod_{0:31} + (RT)$
if $((nprod_0 = RT_0) \wedge (RT_0 \neq temp_1))$ then $(RT) \leftarrow (RT_0 \,\|\, {}^{31}(\neg RT_0))$
else $(RT) \leftarrow temp_{1:32}$
```

The low-order halfword of RA is multiplied by the low-order halfword of RB. The 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

31	RT	RA	RB	124	Rc
0	6	11	16	21	31

(RA) ← ¬((RS) ∨ (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) *Extended mnemonic for* **nor RA,RS,RS**	
not.		*Extended mnemonic for* **nor. RA,RS,RS**	CR[CR0]

or

OR

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

31	RS	RA	RB	444	Rc
0	6	11	16	21	31

(RA) ← (RS) ∨ (RB)

The contents of register RS is ORed with the contents of register RB; the result is placed into register RA.

Registers Altered

- RA
- CR[CR0]$_{LT, GT, EQ, SO}$ 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) *Extended mnemonic for* **or RT,RS,RS**	
mr.		*Extended mnemonic for* **or. RT,RS,RS**	CR[CR0]

orc

OR with Complement

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

31	RT	RA	RB	412	Rc
0	6	11	16	21	31

(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]$_{\text{LT, GT, EQ, SO}}$ if Rc contains 1

Architecture Note

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

ori

OR Immediate

ori RA, RS, IM

24	RS	RA	IM	
0	6	11	16	31

$(RA) \leftarrow (RS) \vee (^{16}0 \parallel IM)$

The IM field is extended to 32 bits by concatenating 16 0-bits on the left. Register RS is ORed with the extended IM field; the result is placed into register RA.

Registers Altered

- RA

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. *Extended mnemonic for* **ori 0,0,0**	

oris RA, RS, IM

25	RS	RA	IM
0	6	11	16 31

$(RA) \leftarrow (RS) \lor (IM \parallel {}^{16}0)$

The IM Field is extended to 32 bits by concatenating 16 0-bits on the right. Register RS is ORed with the extended IM field and the result is placed into register RA.

Registers Altered

- RA

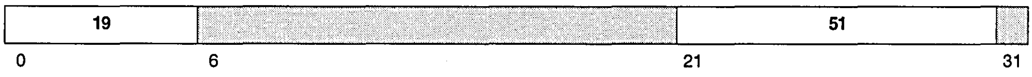
Architecture Note

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

rfci

Return From Critical Interrupt

rfci



(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

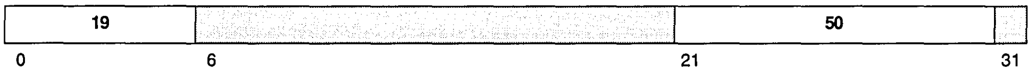
Programming Note

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

20	RS	RA	SH	MB	ME	Rc
0	6	11	16	21	26	31

```
$r \leftarrow$ ROTL((RS), SH)
$m \leftarrow$ 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 Rc 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). $(RA)_{b:b+n-1} \leftarrow (RS)_{0:n-1}$ *Extended mnemonic for* **rlwimi RA,RS,32–b,b,b+n–1**	
inslwi.		*Extended mnemonic for* **rlwimi. RA,RS,32–b,b,b+n–1**	CR[CR0]
insrwi	RA, RS, n, b	Insert from right immediate. (n > 0) $(RA)_{b:b+n-1} \leftarrow (RS)_{32-n:31}$ *Extended mnemonic for* **rlwimi RA,RS,32–b–n,b,b+n–1**	
insrwi.		*Extended mnemonic for* **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

21	RS	RA	SH	MB	ME	Rc
0	6	11	16	21	26	31

```
r ← ROTL((RS), SH)
m ← MASK(MB, ME)
(RA) ← r ∧ 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. ($n < 32$) $(RA)_{0:n-1} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,0,n,31**	
clrlwi.		*Extended mnemonic for* **rlwinm. RA,RS,0,n,31**	CR[CR0]
clrlslwi	RA, RS, b, n	Clear left and shift left immediate. ($n \leq b < 32$) $(RA)_{b-n:31-n} \leftarrow (RS)_{b:31}$ $(RA)_{32-n:31} \leftarrow {}^{n}0$ $(RA)_{0:b-n-1} \leftarrow {}^{b-n}0$ *Extended mnemonic for* **rlwinm RA,RS,n,b−n,31−n**	
clrlslwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,b−n,31−n**	CR[CR0]

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

Mnemonic	Operands	Function	Other Registers Altered
clrrwi	RA, RS, n	Clear right immediate. ($n < 32$) $(RA)_{32-n:31} \leftarrow {}^{n}0$ *Extended mnemonic* for **rlwinm RA,RS,0,0,31–n**	
clrrwi.		*Extended mnemonic for* **rlwinm. RA,RS,0,0,31–n**	CR[CR0]
extlwi	RA, RS, n, b	Extract and left justify immediate. ($n > 0$) $(RA)_{0:n-1} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{n:31} \leftarrow {}^{32-n}0$ *Extended mnemonic for* **rlwinm RA,RS,b,0,n–1**	
extlwi.		*Extended mnemonic for* **rlwinm. RA,RS,b,0,n–1**	CR[CR0]
extrwi	RA, RS, n, b	Extract and right justify immediate. ($n > 0$) $(RA)_{32-n:31} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{0:31-n} \leftarrow {}^{32-n}0$ *Extended mnemonic for* **rlwinm RA,RS,b+n,32–n,31**	
extrwi.		*Extended mnemonic for* **rlwinm. RA,RS,b+n,32–n,31**	CR[CR0]
rotlwi	RA, RS, n	Rotate left immediate. (RA) ← ROTL((RS), n) *Extended mnemonic for* **rlwinm RA,RS,n,0,31**	
rotlwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,0,31**	CR[CR0]
rotrwi	RA, RS, n	Rotate right immediate. (RA) ← ROTL((RS), 32–n) *Extended mnemonic for* **rlwinm RA,RS,32–n,0,31**	
rotrwi.		*Extended mnemonic for* **rlwinm. RA,RS,32–n,0,31**	CR[CR0]
slwi	RA, RS, n	Shift left immediate. ($n < 32$) $(RA)_{0:31-n} \leftarrow (RS)_{n:31}$ $(RA)_{32-n:31} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,n,0,31–n**	
slwi.		*Extended mnemonic for* **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. ($n < 32$) $(RA)_{n:31} \leftarrow (RS)_{0:31-n}$ $(RA)_{0:n-1} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,32–n,n,31**	
srwi.		*Extended mnemonic for* **rlwinm. RA,RS,32–n,n,31**	CR[CR0]

rlwnm

Rotate Left Word then AND with Mask

rlwnm	RA, RS, RB, MB, ME	Rc=0
rlwnm.	RA, RS, RB, MB, ME	Rc=1

23	RS	RA	RB	MB	ME	Rc
0	6	11	16	21	26	31

```
r ← ROTL((RS), (RB)$_{27:31}$)
m ← MASK(MB, ME)
(RA) ← 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)$_{27:31}$) *Extended mnemonic for* **rlwnm RA,RS,RB,0,31**	
rotlw.		*Extended mnemonic for* **rlwnm. RA,RS,RB,0,31**	CR[CR0]

sc

0	6	30	31
17	▨	1	▨

```
(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

31	RS	RA	RB	24	Rc
0	6	11	16	21	31

```
n ← $(RB)_{27:31}$
r ← ROTL((RS), n)
if $(RB)_{26}$ = 0 then
   m ← MASK(0, 31 – n)
else
   m ← $^{32}0$
(RA) ← r $\wedge$ 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 RB_{26} = 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

31	RS	RA	RB	792	Rc
0	6	11	16	21	31

```
$n \leftarrow (RB)_{27:31}$
$r \leftarrow$ ROTL((RS), 32 – n)
if $(RB)_{26}$ = 0 then
    $m \leftarrow$ MASK(n, 31)
else
    $m \leftarrow {}^{32}0$
$s \leftarrow (RS)_0$
$(RA) \leftarrow (r \wedge m) \vee ({}^{32}s \wedge \neg m)$
$XER[CA] \leftarrow s \wedge ((r \wedge \neg m) \neq 0)$
```

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 RS_0 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

0	6	11	16	21	31
31	RS	RA	SH	824	Rc

```
$n \leftarrow SH$
$r \leftarrow ROTL((RS), 32 - n)$
$m \leftarrow MASK(n, 31)$
$s \leftarrow (RS)_0$
$(RA) \leftarrow (r \wedge m) \vee (^{32}s \wedge \neg m)$
$XER[CA] \leftarrow s \wedge ((r \wedge \neg m) \neq 0)$
```

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 RS_0 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

31	RS	RA	RB	536	Rc
0	6	11	16	21	31

```
n ← $(RB)_{27:31}$
r ← ROTL((RS), 32 – n)
if $(RB)_{26}$ = 0 then
   m ← MASK(n, 31)
else
   m ← $^{32}0$
(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)

38	RS	RA	D	
0	6	11	16	31

```
$EA \leftarrow (RA|0) + EXTS(D)$
$MS(EA, 1) \leftarrow (RS)_{24:31}$
```

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

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

Registers Altered

- None

Architecture Note

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

stbu RS, D(RA)

39	RS	RA	D
0	6	11	16 … 31

```
$EA \leftarrow (RA|0) + EXTS(D)$
$MS(EA, 1) \leftarrow (RS)_{24:31}$
$(RA) \leftarrow EA$
```

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

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

The EA is placed into register RA.

Registers Altered

- RA

Invalid Instruction Forms

RA = 0

Architecture Note

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

stbux

Store Byte with Update Indexed

stbux RS, RA, RB

31	RS	RA	RB	247	▨
0	6	11	16	21	31

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

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

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

The EA is placed into register RA.

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

Registers Altered

- RA

Invalid Instruction Forms

- Reserved fields
- RA = 0

Architecture Note

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

stbx RS, RA, RB

31	RS	RA	RB	215	
0	6	11	16	21	31

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

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

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

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

Registers Altered

- None

Invalid Instruction Forms

- Reserved fields

Architecture Note

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

sth

Store Halfword

sth RS, D(RA)

44	RS	RA	D	
0	6	11	16	31

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

An effective address (EA) is formed by adding a displacement to a base address. The displacement is obtained by sign-extending the 16-bit D field to 32 bits. The base address is 0 when the RA field is 0 and is the contents of register RA otherwise.

The least significant halfword of register RS is stored into the halfword at the EA in main storage.

Registers Altered

- None

Architecture Note

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

sthbrx
Store Halfword Byte-Reverse Indexed

sthbrx RS, RA, RB

31	RS	RA	RB	918	
0	6	11	16	21	31

```
$EA \leftarrow (RA|0) + (RB)$
$MS(EA, 2) \leftarrow (RS)_{24:31} \parallel (RS)_{16:23}$
```

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.

sthu

Store Halfword with Update

sthu RS, D(RA)

45	RS	RA	D	
0	6	11	16	31

```
EA ← (RA|0) + EXTS(D)
MS(EA, 2) ← $(RS)_{16:31}$
(RA) ← 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

31	RS	RA	RB	439	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
MS(EA, 2) ← $(RS)_{16:31}$
(RA) ← 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

31	RS	RA	RB	407	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
MS(EA, 2) ← $(RS)_{16:31}$
```

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)

47	RS	RA	D	
0	6	11	16	31

```
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

31	RS	RA	NB	725	
0	6	11	16	21	31

```
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

31	RS	RA	RB	661	
0	6	11	16	21	31

```
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.

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)

36	RS	RA	D	
0	6	11	16	31

```
EA ← (RA|0) + EXTS(D)
MS(EA, 4) ← (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

31	RS	RA	RB	662	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
MS(EA, 4) ← $(RS)_{24:31}$ || $(RS)_{16:23}$ || $(RS)_{8:15}$ || $(RS)_{0:7}$
```

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

31	RS	RA	RB	150	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
if RESERVE = 1 then
    MS(EA, 4) ← (RS)
    RESERVE ← 0
    (CR[CR0]) ← $^{2}0 \parallel 1 \parallel XER_{SO}$
else
    (CR[CR0]) ← $^{2}0 \parallel 0 \parallel XER_{SO}$
```

An effective address (EA) is formed by adding an index to a base address. The index is the contents of register RB. The base address is 0 when the RA field is 0, and is the contents of register RA otherwise.

If the reservation bit contains 1 when the instruction is executed, the contents of register RS are stored into the word at the EA and the reservation bit is cleared. If the reservation bit contains 0 when the instruction is executed, no store operation is performed.

CR[CR0] is set as follows:

- $CR[CR0]_{LT,\ GT}$ 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 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

Store Word with Update

stwu RS, D(RA)

37	RS	RA	D
0	6	11	16 … 31

```
EA ← (RA|0) + EXTS(D)
MS(EA, 4) ← (RS)
(RA) ← 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

31	RS	RA	RB	183	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
MS(EA, 4) ← (RS)
(RA) ← 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

31	RS	RA	RB	151	▨
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
MS(EA,4) ← (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

31	RT	RA	RB	OE	40	Rc
0	6	11	16	21	22	31

(RT) ← ¬(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) ← ¬(RB) + (RA) + 1. *Extended mnemonic for* **subf RT,RB,RA**	
sub.		*Extended mnemonic for* **subf. RT,RB,RA**	CR[CR0]
subo		*Extended mnemonic for* **subfo RT,RB,RA**	XER[SO, OV]
subo.		*Extended mnemonic for* **subfo. RT,RB,RA**	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

31	RT	RA	RB	OE	8	Rc
0	6	11	16	21	22	31

```
$(RT) \leftarrow \neg(RA) + (RB) + 1$
if $\neg(RA) + (RB) + 1 \overset{u}{>} 2^{32} - 1$ then
   $XER[CA] \leftarrow 1$
else
   $XER[CA] \leftarrow 0$
```

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]. *Extended mnemonic for* **subfc RT,RB,RA**	
subc.		*Extended mnemonic for* **subfc. RT,RB,RA**	CR[CR0]
subco		*Extended mnemonic for* **subfco RT,RB,RA**	XER[SO, OV]
subco.		*Extended mnemonic for* **subfco. RT,RB,RA**	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

31	RT	RA	RB	OE	136	Rc
0	6	11	16	21	22	31

```
(RT) $\leftarrow$ $\neg$(RA) + (RB) + XER[CA]
if $\neg$(RA) + (RB) + XER[CA] $\overset{u}{>}$ $2^{32} - 1$ then
   XER[CA] $\leftarrow$ 1
else
   XER[CA] $\leftarrow$ 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 RT, RA, IM

8	RT	RA	IM	
0	6	11	16	31

```
$(RT) \leftarrow \neg(RA) + EXTS(IM) + 1$
if $\neg(RA) + EXTS(IM) + 1 \overset{u}{>} 2^{32} - 1$ then
    $XER[CA] \leftarrow 1$
else
    $XER[CA] \leftarrow 0$
```

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

31	RT	RA	▨	OE	232	Rc
0	6	11	16	21	22	31

```
(RT) ← ¬(RA) – 1 + XER[CA]
if ¬(RA) + 0xFFFF FFFF + XER[CA] $\overset{u}{>}$ $2^{32}$ – 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	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

31	RT	RA		OE	200	Rc
0	6	11	16	21	22	31

```
(RT) ← ¬(RA) + XER[CA]
if ¬(RA) + XER[CA] $\overset{u}{>}$ $2^{32} - 1$ 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

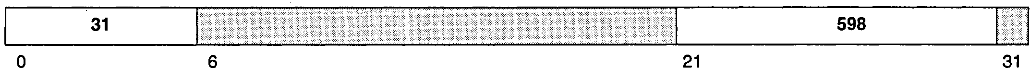
Architecture Note

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

31	▨	370	▨
0	6	21	31

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

31	RT	RA	WS	946	▨
0	6	11	16	21	31

```
if $WS_4$ = 1
    (RT) ← TLBLO[($RA_{26:31}$)]
else
    (RT) ← TLBHI[($RA_{26:31}$)]
    (PID) ← TID from TLB[($RA_{26:31}$)]
```

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) ← TLBHI[(RA)] (PID) ← $TLB[(RA)]_{TID}$ *Extended mnemonic for* **tlbre RT,RA,0**	
tlbrelo	RT, RA	Load TLBLO portion of the selected TLB entry into RT. (RT) ← TLBLO[(RA)] *Extended mnemonic for* **tlbre RT,RA,1**	

tlbsx

TLB Search Indexed

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

31	RT	RA	RB	914	Rc
0	6	11	16	21	31

```
EA ← (RA|0) + (RB)
if Rc = 1
   $CR[CR0]_{LT}$ ← 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 (Rc) 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 **tlbsx.** instruction if there is a possibility that the search may fail.

Registers Altered

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

31		566	
0	6	21	31

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

31	RS	RA	WS	978	▨
0	6	11	16	21	31

```
if $WS_4 = 1$
    TLBLO[($RA_{26:31}$)] ← (RS)
else
    TLBHI[($RA_{26:31}$)] ← (RS)
    TID of TLB[($RA_{26:31}$)] ← ($PID_{24:31}$)
```

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] ← RT[0:21]
    EX,WR ← RS[22:23]
    ZSEL[0:3] ← RS[24:27]
    WIMG ← 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)] ← (RS) TLB[(RA)]$_{TID}$ ← (PID$_{24:31}$) *Extended mnemonic for* **tlbwe RS,RA,0**	
tlbwelo	RS, RA	Write TLBLO portion of the selected TLB entry from RS. TLBLO[(RA)] ← (RS) *Extended mnemonic for* **tlbwe RS,RA,1**	

tw

Trap Word

tw TO, RA, RB

31	TO	RA	RB	4	
0	6	11	16	21	31

```
if ( ((RA) $<$ (RB) $\wedge$ $TO_0$ = 1) $\vee$
     ((RA) $>$ (RB) $\wedge$ $TO_1$ = 1) $\vee$
     ((RA) $=$ (RB) $\wedge$ $TO_2$ = 1) $\vee$
     ((RA) $\overset{u}{<}$ (RB) $\wedge$ $TO_3$ = 1) $\vee$
     ((RA) $\overset{u}{>}$ (RB) $\wedge$ $TO_4$ = 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. *Extended mnemonic for* **tw 31,0,0**	
tweq	RA, RB	Trap if (RA) equal to (RB). *Extended mnemonic for* **tw 4,RA,RB**	
twge	RA, RB	Trap if (RA) greater than or equal to (RB). *Extended mnemonic for* **tw 12,RA,RB**	
twgt	RA, RB	Trap if (RA) greater than (RB). *Extended mnemonic for* **tw 8,RA,RB**	
twle	RA, RB	Trap if (RA) less than or equal to (RB). *Extended mnemonic for* **tw 20,RA,RB**	
twlge	RA, RB	Trap if (RA) logically greater than or equal to (RB). *Extended mnemonic for* **tw 5,RA,RB**	
twlgt	RA, RB	Trap if (RA) logically greater than (RB). *Extended mnemonic for* **tw 1,RA,RB**	

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

Mnemonic	Operands	Function	Other Registers Altered
twlle	RA, RB	Trap if (RA) logically less than or equal to (RB). *Extended mnemonic for* **tw 6,RA,RB**	
twllt	RA, RB	Trap if (RA) logically less than (RB). *Extended mnemonic for* **tw 2,RA,RB**	
twlng	RA, RB	Trap if (RA) logically not greater than (RB). *Extended mnemonic for* **tw 6,RA,RB**	
twlnl	RA, RB	Trap if (RA) logically not less than (RB). *Extended mnemonic for* **tw 5,RA,RB**	
twlt	RA, RB	Trap if (RA) less than (RB). *Extended mnemonic for* **tw 16,RA,RB**	
twne	RA, RB	Trap if (RA) not equal to (RB). *Extended mnemonic for* **tw 24,RA,RB**	
twng	RA, RB	Trap if (RA) not greater than (RB). *Extended mnemonic for* **tw 20,RA,RB**	
twnl	RA, RB	Trap if (RA) not less than (RB). *Extended mnemonic for* **tw 12,RA,RB**	

twi TO, RA, IM

3	TO	RA	IM
0	6	11	16 … 31

```
if ( ((RA) $<$ EXTS(IM) $\wedge$ $TO_0$ = 1) $\vee$
     ((RA) $>$ EXTS(IM) $\wedge$ $TO_1$ = 1) $\vee$
     ((RA) $=$ EXTS(IM) $\wedge$ $TO_2$ = 1) $\vee$
     ((RA) $\overset{u}{<}$ EXTS(IM) $\wedge$ $TO_3$ = 1) $\vee$
     ((RA) $\overset{u}{>}$ EXTS(IM) $\wedge$ $TO_4$ = 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 ← $EVPR_{0:15}$ || 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 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 ← $EVPR_{0:15}$ || 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 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). *Extended mnemonic for* **twi 4,RA,IM**	
twgei	RA, IM	Trap if (RA) greater than or equal to EXTS(IM). *Extended mnemonic for* **twi 12,RA,IM**	
twgti	RA, IM	Trap if (RA) greater than EXTS(IM). *Extended mnemonic for* **twi 8,RA,IM**	
twlei	RA, IM	Trap if (RA) less than or equal to EXTS(IM). *Extended mnemonic for* **twi 20,RA,IM**	
twlgei	RA, IM	Trap if (RA) logically greater than or equal to EXTS(IM). *Extended mnemonic for* **twi 5,RA,IM**	
twlgti	RA, IM	Trap if (RA) logically greater than EXTS(IM). *Extended mnemonic for* **twi 1,RA,IM**	
twllei	RA, IM	Trap if (RA) logically less than or equal to EXTS(IM). *Extended mnemonic for* **twi 6,RA,IM**	
twllti	RA, IM	Trap if (RA) logically less than EXTS(IM). *Extended mnemonic for* **twi 2,RA,IM**	
twlngi	RA, IM	Trap if (RA) logically not greater than EXTS(IM). *Extended mnemonic for* **twi 6,RA,IM**	

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). *Extended mnemonic for* **twi 5,RA,IM**	
twlti	RA, IM	Trap if (RA) less than EXTS(IM). *Extended mnemonic for* **twi 16,RA,IM**	
twnei	RA, IM	Trap if (RA) not equal to EXTS(IM). *Extended mnemonic for* **twi 24,RA,IM**	
twngi	RA, IM	Trap if (RA) not greater than EXTS(IM). *Extended mnemonic for* **twi 20,RA,IM**	
twnli	RA, IM	Trap if (RA) not less than EXTS(IM). *Extended mnemonic for* **twi 12,RA,IM**	

wrtee

Write External Enable

wrtee RS

31	RS	▨	131	▨
0	6	11	21	31

MSR[EE] ← $(RS)_{16}$

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

31		E		163	
0	6	16	17	21	31

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

31	**RS**	**RA**	**RB**	**316**	**Rc**
0	6	11	16	21	31

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

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

26	RS	RA	IM
0	6	11	16 31

$(RA) \leftarrow (RS) \oplus (^{16}0 \parallel IM)$

The IM field is extended to 32 bits by concatenating 16 0-bits on the left. The contents of register RS are XORed with the extended IM field; the result is placed into register RA.

Registers Altered

- RA

Architecture Note

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

xoris

XOR Immediate Shifted

xoris RA, RS, IM

27	RS	RA	IM
0	6	11	16 31

$(RA) \leftarrow (RS) \oplus (IM \parallel {}^{16}0)$

The IM field is extended to 32 bits by concatenating 16 0-bits on the right. The contents of register RS are XORed with the extended IM field; the result is placed into register RA.

Registers Altered

- RA

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 **mfspr** and **mtspr** instructions. The column labeled "SPRF" lists the corresponding fields contained in the *machine code* of **mfspr** and **mtspr**. The SPRN field contains the five-bit subfields of the SPRF field, which are *reversed* in the machine code for the **mfspr** and **mtspr** instructions (SPRN $\leftarrow$ $SPRF_{5:9}$ || $SPRF_{0:4}$) 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 Decimal	SPRN Hex	SPRF	Access
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 TBL.

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 $\leftarrow$ $TBRF_{5:9}$ || $TBRF_{0:4}$). 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 **mfdcr** and **mtdcr** instructions.

The **mfdcr** and **mtdcr** instructions are privileged, for all DCR numbers. Therefore, all DCR accesses are privileged. All DCR numbers are reserved, and should be neither read nor written.

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_PMCSRBSE	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.

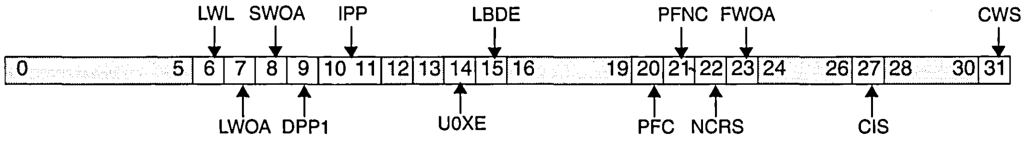


Figure 25-1. Core Configuration Register 0 (CCR0)

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	LDBE	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.

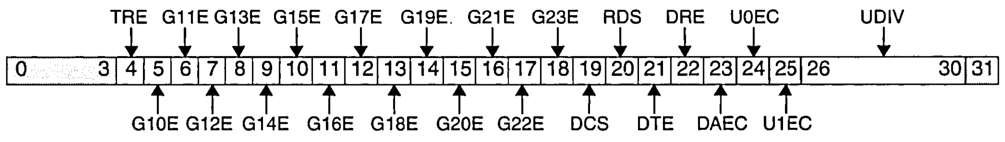


Figure 25-2. Chip Control Register 0 (CPC0_CR0)

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.

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

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 × OPB frequency
31		Reserved	

CPC0_CR1

Chip Control Register 1

DCR 0x0B2

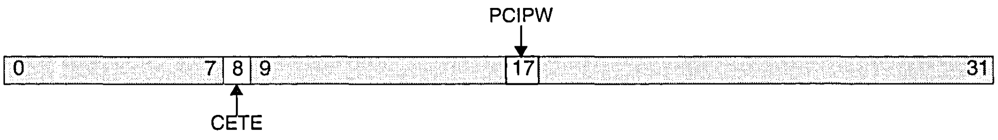


Figure 25-3. Chip Control Register 1 (CPC0_CR1)

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.

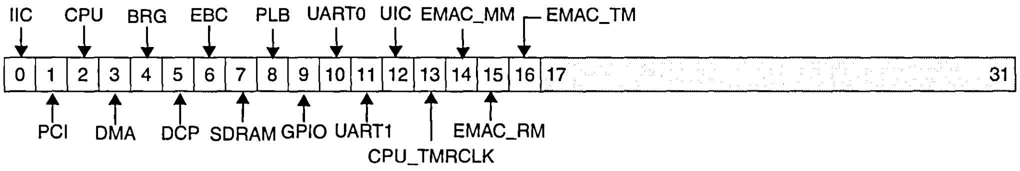


Figure 25-4. CPM Enable Register (CPC0_ER)

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.

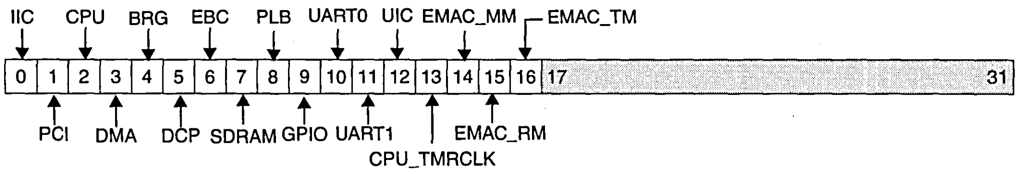


Figure 25-5. CPM Force Register (CPC0_FR)

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	

DCR 0x0B5 Read-Only

See “JTAG ID Register (CPC0_JTAGID)” on page 12-4.

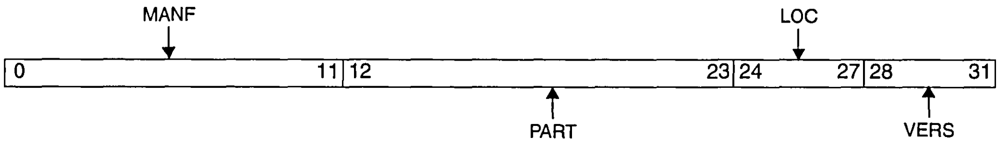


Figure 25-6. JTAG ID Register (CPC0_JTAGID)

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.

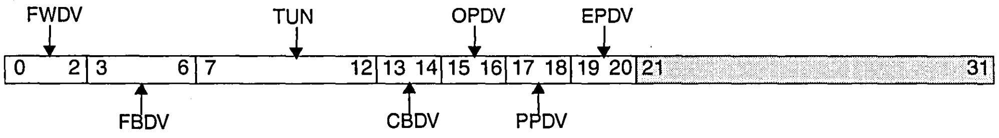


Figure 25-7. PLL Mode Register (CPC0_PLLMR)

0:2	FWDV	Forward Divisor 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.	PLLOUTA Value: 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 **Note:** PLLOUTA is the CPU clock in PPC405GP.
3:6	FBDV	Feedback Divisor 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.	
7:12	TUN	TUNE[5:0] Field	**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.

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.

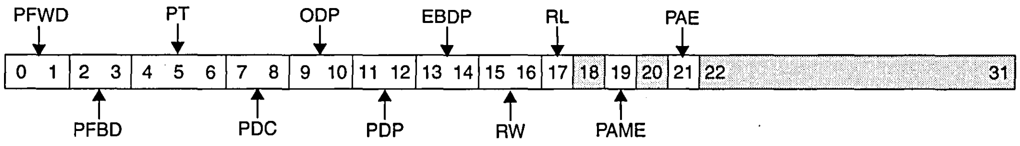


Figure 25-8. Chip Pin Strapping Register (CPC0_PSR)

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	

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.

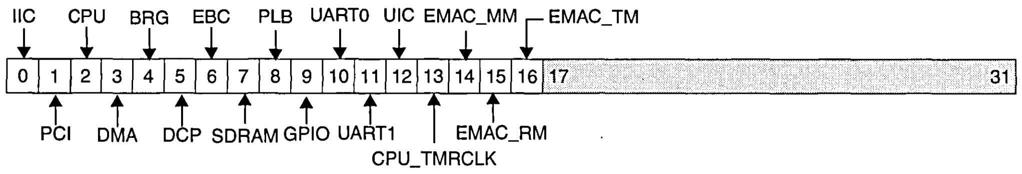


Figure 25-9. CPM Status Register (CPC0_SR)

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.

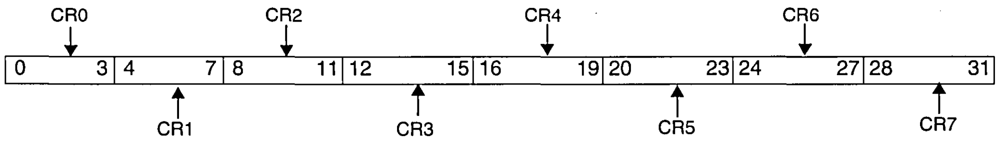


Figure 25-10. Condition Register (CR)

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.

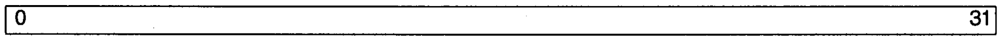


Figure 25-11. Count Register (CTR)

0:31		Count	Used as count for branch conditional with decrement instructions, or as address for branch-to-counter instructions.

SPR 0x3F6–0x3F7

See “Data Address Compare Registers (DAC1–DAC2)” on page 12-14.

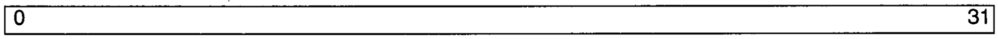


Figure 25-12. Data Address Compare Registers (DAC1–DAC2)

0:31		Data Address Compare (DAC) byte address	DBCR0[D1S] determines which address bits are examined.

DBCR0

Debug Control Register 0

SPR 0x3F2

See "Debug Control Registers" on page 12-9.

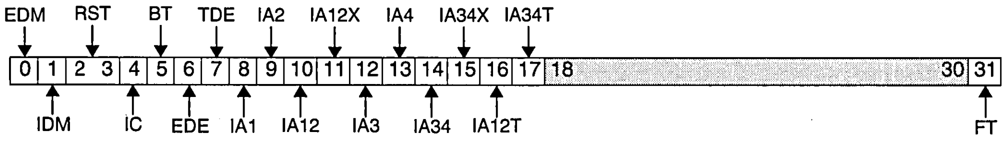


Figure 25-13. Debug Control Register 0 (DBCR0)

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.

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.

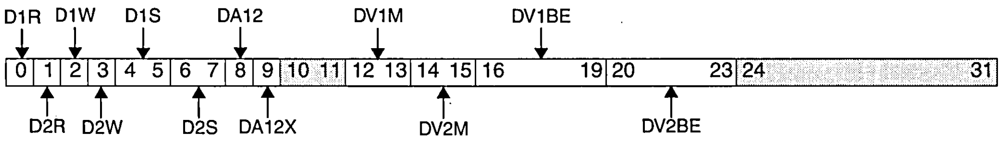


Figure 25-14. Debug Control Register 1 (DBCR1)

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	

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.

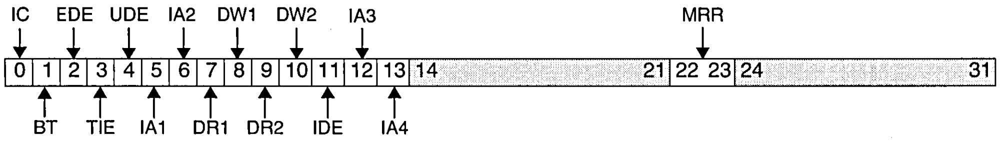


Figure 25-15. Debug Status Register (DBSR)

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

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.

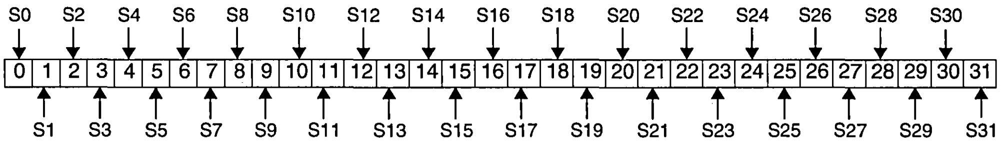


Figure 25-16. Data Cache Cachability Register (DCCR)

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

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.

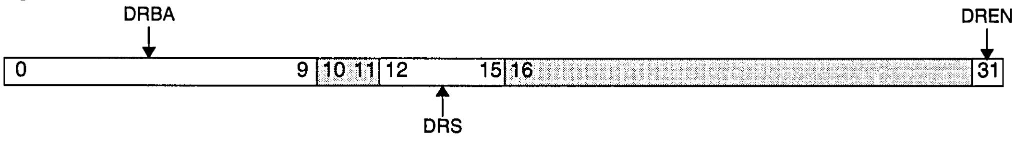


Figure 25-17. Decompression Address Decode Definition Registers (DCP0_ADDR0–DCP0_ADDR1)

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.

DCR Accessed using DCP0_CFGADDR, DCP0_CFGDATA; Offset 0x40

See “Decompression Configuration Register (DCP0_CFG)” on page 14-5.

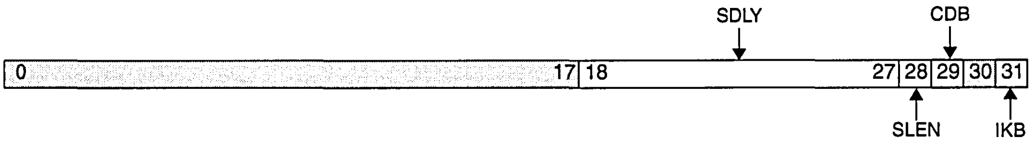


Figure 25-18. Decompression Controller Configuration Register (DCP0_CFG)

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 0 Normal operation 1 Clear decompression buffer	Self-clearing; always reads 0
30		Reserved	
31	IKB	Enable Decrompression 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.

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.

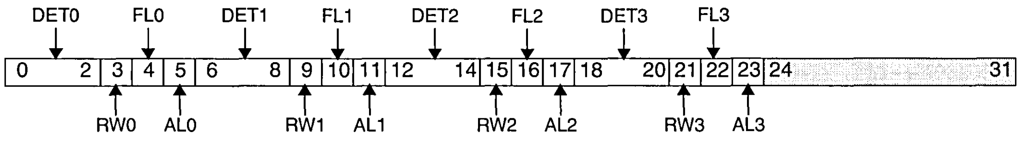


Figure 25-19. Decompression Controller Error Status Register 0 (DCP0_ESR)

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	FL0	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.

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.

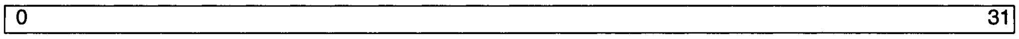


Figure 25-20. Decompression Controller ID Register (DCP0_ID)

0:31		Decompression Controller ID	Read-only, value is 0000504D

DCR Accessed using DCP0_CFGADDR, DCP0_CFGDATA; Offset 0x00–0x03

See “Index Table Origin Registers (DCP0_ITOR0–DCP0_ITOR3)” on page 14-4.

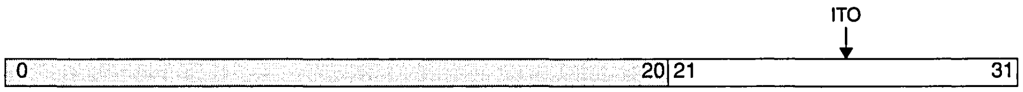


Figure 25-21. Decompression Index Table Origin Registers (DCP0_ITOR0–DCP0_ITOR3)

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.

0	31

Figure 25-22. Decompression Controller Bus Error Address Register (DCP0_MEMBEAR)

0:31		Address of SDRAM or EBC Error

DCR Accessed using DCP0_CFGADDR, DCP0_CFGDATA; Offset 0x50 Read-Only

See “Decompression Controller PLB Error Address Register (DCP0_PLBBEAR)” on page 14-7.

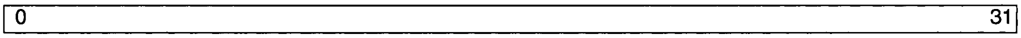


Figure 25-23. Decompression Controller PLB Error Address Register (DCP0_PLBBEAR)

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.

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.

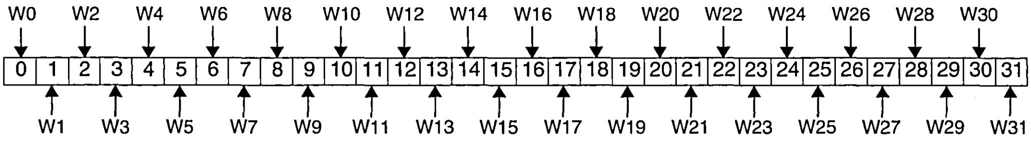


Figure 25-25. Data Cache Write-through Register (DCWR)

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

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)

0:31		Address of Data Error (synchronous)

DCR 0x100, 0x108, 0x110, 0x118

See “DMA Channel Control Registers (DMA0_CR0–DMA0_CR3)” on page 18-8.

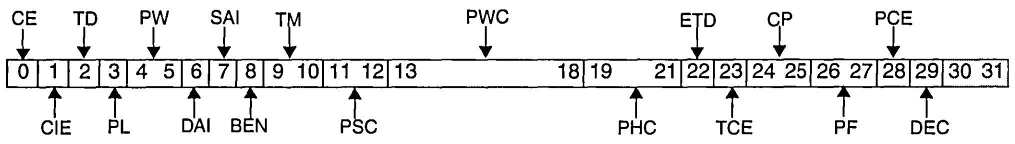


Figure 25-27. DMA Channel Control Registers (DMA0_CR0-DMA0_CR3)

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)	

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 deviced-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).

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.

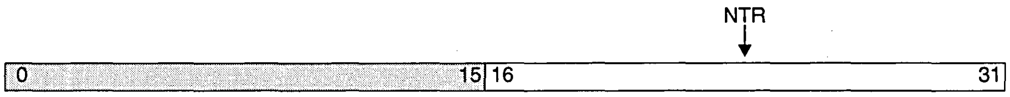


Figure 25-28. DMA Count Registers (DMA0_CT0-DMA0_CT3)

0:15	▨	Reserved
16:31	NTR	Number of transfers remaining

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.

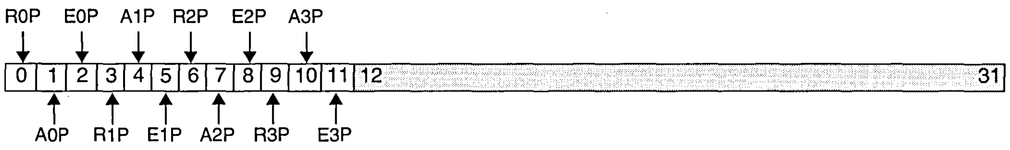


Figure 25-30. DMA Polarity Configuration Register (DMA0_POL)

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

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.

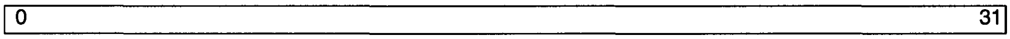


Figure 25-31. DMA Source Address Registers (DMA0_SA0–DMA0_SA3)

0:31		Source address for memory-to-memory and memory-to-peripheral transfers.

DCR 0x104, 0x10C, 0x114, 0x11C

See “DMA Scatter/Gather Descriptor Address Registers (DMA0_SG0–DMA0_SG3)” on page 18-12.

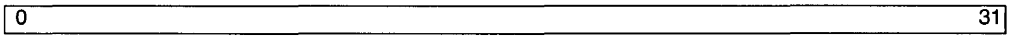


Figure 25-32. DMA Scatter/Gather Descriptor Address Registers (DMA0_SG0–DMA0_SG3)

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.

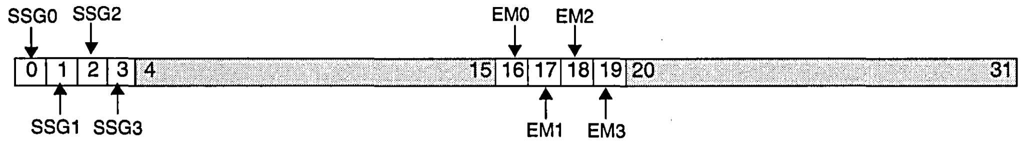


Figure 25-33. DMA Scatter/Gather Command Register (DMA0_SGC)

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.

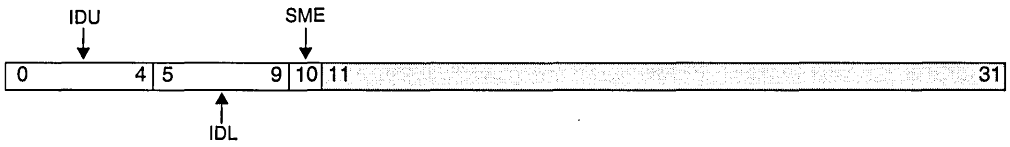


Figure 25-34. DMA Sleep Mode Register (DMA0_SLP)

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.

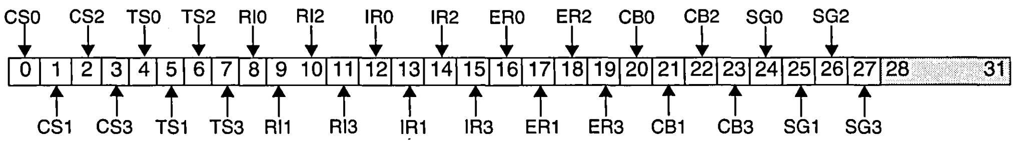


Figure 25-35. DMA Status Register (DMA0_SR)

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	

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.

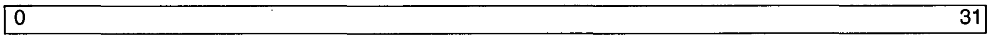


Figure 25-37. Peripheral Bus Error Address Register (EBC0_BEAR)

0:31		Address of Bus Error (asynchronous)

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x21

See "Peripheral Bus Error Status Register 0 (EBC0_BESR0)" on page 16-30.

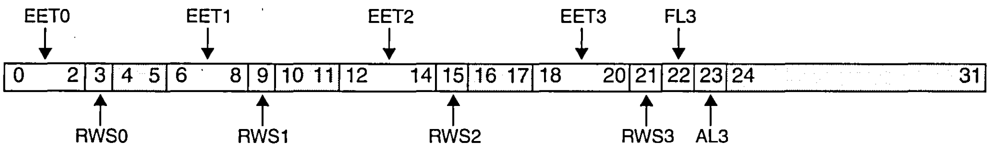


Figure 25-38. Peripheral Bus Error Status Register 0 (EBC0_BESR0)

0:2	EET0	Error type for master 0 000 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 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 Parity error 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error	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 Parity error 010 Reserved 011 Reserved 100 Protection error 101 Reserved 110 External bus input error 111 External bus timeout error	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	

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	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	
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.

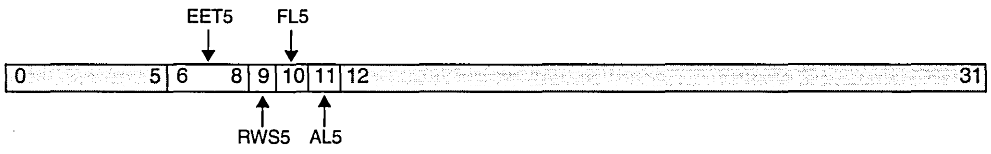


Figure 25-39. Peripheral Bus Error Status Register 1 (EBC0_BESR1)

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	

12:31	▨	Reserved

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x10–0x17

See “Peripheral Bank Access Parameters (EBC0_BnAP)” on page 16-26.

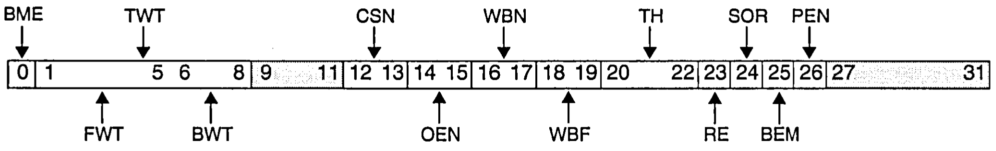


Figure 25-40. Peripheral Bank Access Parameters (EBC0_BnAP)

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 $\overline{\text{PerCSn}}$ low.
14:15	OEN	Output Enable On Timing 0-3 PerClk cycles	Number of cycles from $\overline{\text{PerCSn}}$ low to $\overline{\text{PerOE}}$ low.
16:17	WBN	Write Byte Enable On Timing 0-3 PerClk cycles	If BEM=0, number of cycles from $\overline{\text{PerCSn}}$ low to $\overline{\text{PerWBE0:3}}$ active.
18:19	WBF	Write Byte Enable Off Timing 0-3 PerClk cycles	If BEM=0 and RE=0, number of cycles $\overline{\text{PerWBEn}}$ becomes inactive prior to $\overline{\text{PerCSn}}$ 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 $\overline{\text{PerWBE0:3}}$ are only active for write cycles 1 $\overline{\text{PerWBE0:3}}$ are active for read and write cycles	If BEM=0, $\overline{\text{PerWBE0:3}}$ timing is controlled by WBN and WBF. If BEM=1, $\overline{\text{PerWBE0:3}}$ has the same timing as PerAddr0:31.
26	PEN	Parity Enable 0 Disable parity checking 1 Enable parity checking	The EBC implements odd parity.

27:31	▨	Reserved

DCR Accessed using EBC0_CFGADDR; EBC0_CFGDATA; Offset 0x00–0x07

See “Peripheral Bank Configuration Registers (EBC0_BnCR)” on page 16-25.

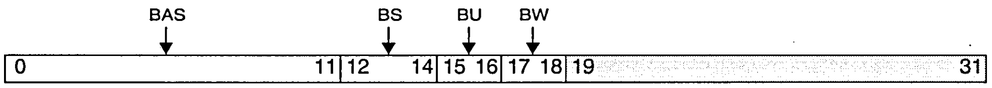


Figure 25-41. Peripheral Bank Configuration Registers (EBC0_BnCR)

0:11	BAS	Base Address Select	Specifies the bank starting address, which must be a multiple of the bank size.
12:14	BS	Bank Size 000 1 MB bank 001 2 MB bank 010 4 MB bank 011 8 MB bank 100 16 MB bank 101 32 MB bank 110 64 MB bank 111 128 MB bank	
15:16	BU	Bank Usage 00 Disabled 01 Read-only 10 Write-only 11 Read/Write	Specifies the type of accesses allowed for the bank. A protect error occurs if a write is attempted to a read-only bank or a read from a write-only bank.
17:18	BW	Bus Width 00 8-bit bus 01 16-bit bus 10 32-bit bus 11 Reserved	
19:31	▨	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.

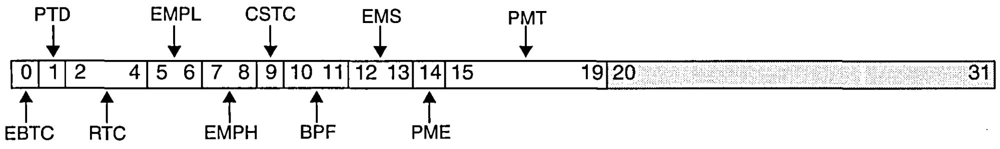


Figure 25-42. EBC Configuration Register (EBC0_CFG)

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 $\overline{\text{PerCS0:7}}$ are high-Z between EBC transfers and when an external master is active (HoldAck=1) 1 $\overline{\text{PerCS0:7}}$ 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.

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.

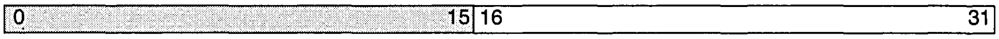


Figure 25-43. Group Address Hash Tables 1–4 (EMAC0_GAHT1–EMAC0_GAHT4)

0:15	▨	Reserved
16:31		Group Address Hash Number

MMIO 0xEF60081C

See “Individual Address High (EMAC0_IAHR)” on page 19-35.

0 ▨ 15	16 31

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.

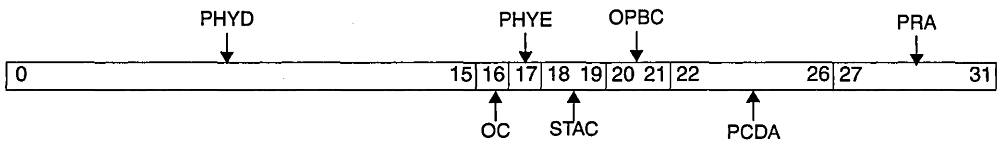


Figure 25-45. STA Control Register (EMAC0_STACR)

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	

MMIO 0xEF600808

See "Transmit Mode Register 0 (EMAC0_TMR0)" on page 19-27.

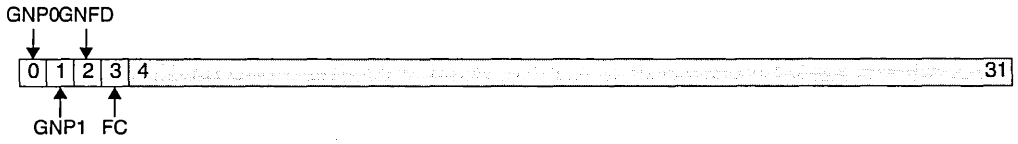


Figure 25-46. Transmit Mode Register 0 (EMAC0_TMR0)

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.

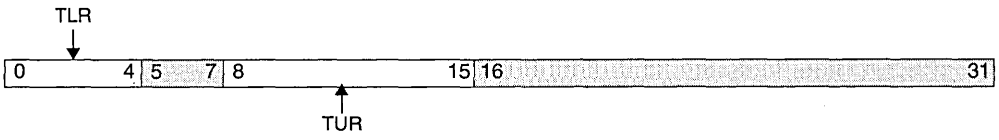


Figure 25-47. Transmit Mode Register 1 (EMAC0_TMR1)

0:4	TLR	Transmit Low Request
5:7		Reserved
8:15	TUR	Transmit Urgent Request
16:31		Reserved

MMIO 0xEF600860

See “Transmit Request Threshold Register (EMAC0_TRTR)” on page 19-40.

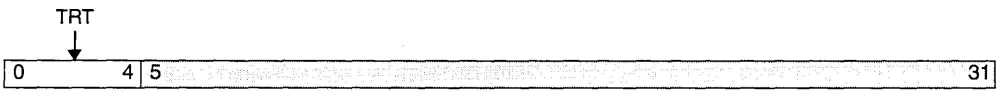


Figure 25-48. Transmit Request Threshold Register (EMAC0_TRTR)

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.

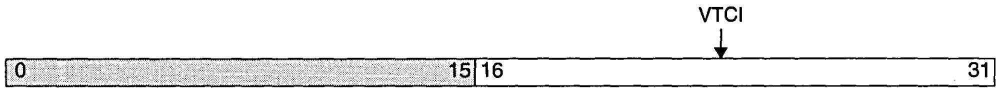


Figure 25-49. VLAN TCI Register (EMAC0_VTCI)

0:15	▨	Reserved
16:31	VTCI	VLAN TCI tag

MMIO 0xEF600824

See "VLAN TPID Register (EMAC0_VTPID)" on page 19-36.

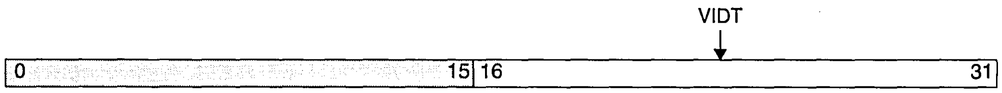


Figure 25-50. VLAN TPID Register (EMAC0_VTPID)

0:15	▨	Reserved
16:31	VIDT	VLAN ID tag

ESR

Exception Syndrome Register

SPR 0x3D4

See “Exception Syndrome Register (ESR)” on page 10-31.

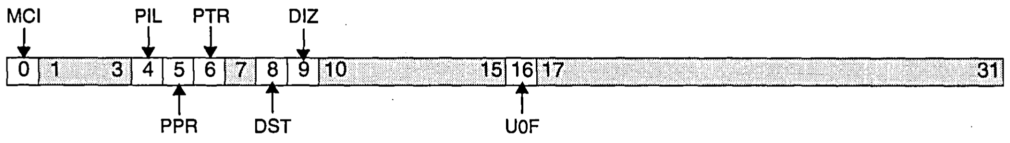


Figure 25-51. Exception Syndrome Register (ESR)

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 **dcbi**, **dcbz**, and **dccci**).
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	U0F	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

SPR 0x3D6

See “Exception Vector Prefix Register (EVPR)” on page 10-31.

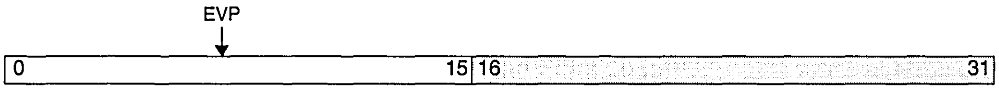


Figure 25-52. Exception Vector Prefix Register (EVPR)

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.

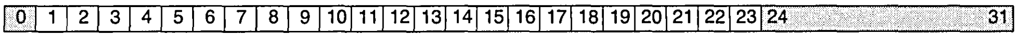


Figure 25-53. GPIO Input Register (GPIO0_IR)

0	▨	Reserved
1:23		GPIO register bits
24:31	▨	Reserved

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

GPIO0_OR

GPIO Output Register

MMIO 0xEF600700

See “GPIO Output Register (GPIO0_OR)” on page 23-5.

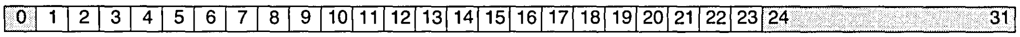


Figure 25-55. GPIO Output Register (GPIO0_OR)

0	▨	Reserved
1:23		GPIO register bits
24:31	▨	Reserved

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.

0	31

Figure 25-57. General Purpose Registers (R0-R31)

0:31		General Purpose Register data

SPR 0x3F4–0x3F5, 0x3B4–0x3B5

See "Instruction Address Compare Registers (IAC1–IAC4)" on page 12-14.

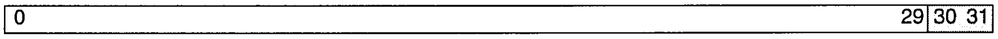


Figure 25-58. Instruction Address Compare Registers (IAC1–IAC4)

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.

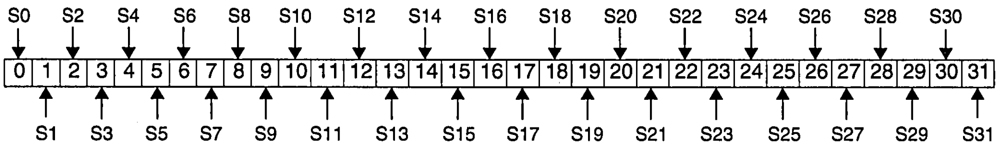


Figure 25-59. Instruction Cache Cachability Register (ICCR)

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

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

MMIO 0xEF60050C

See "IIC0 Clock Divide Register" on page 22-15.

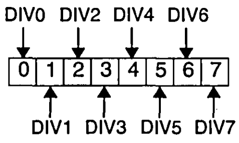


Figure 25-61. IIC0 Clock Divide Register (IIC0_CLKDIV)

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.

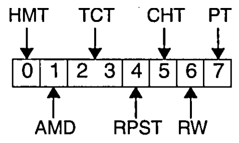


Figure 25-62. IIC0 Control Register (IIC0_CNTL)

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.	

MMIO 0xEF600510

See “IIC0 Direct Control Register” on page 22-20.

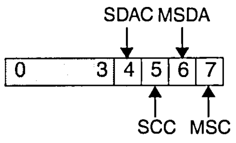


Figure 25-63. IIC0 Direct Control Register (IIC0_DIRECTCNTL)

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	IICSCL Output Control Directly controls the IICSCL output 0 IICSCL is a logic 0 1 IICSCL is a logic 1	
6	MSDA	Monitor IICSDA Used to monitor the IICSDA input 0 IICSDA is a logic 0 1 IICSDA is a logic 1	Read-only
7	MSC	Monitor IICSCL. Used to monitor the IICSCL input. 0 IICSCL is a logic 0 1 IICSCL 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.

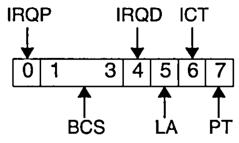


Figure 25-64. IIC0 Extended Status Register (IIC0_EXTSTS)

0	IRQP	IRQ Pending 0 No IRQ is pending. 1 An IRQ is active, another IRQ is on-deck, and another interrupt-generating condition has occurred.	• 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	Bus Control State 000 Unused; if this value is read, a major IIC hardware problem occurred. 001 Slave-selected state; the IIC interface has detected and decoded a slave transfer request on the IIC bus. 010 Slave Transfer state; the IIC interface has detected but has not decoded a slave transfer request on the IIC bus. 011 Master Transfer state; entered after a master transfer request has started on the IIC bus. 100 Free Bus state; the bus is free and no transfer request is pending. 101 Busy Bus state; the bus is busy. 110 Unknown state; value after IIC reset. 111 Unused; if this value is read, a major IIC hardware problem occurred.	Read-only.

4	IRQD	IRQ On-Deck 0 No IRQ is on-deck. 1 An interrupt is active, and another interrupt-generating condition has occurred.	• 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	Lost Arbitration 0 Normal operation. 1 Loss of arbitration has ended the requested master transfer.	• 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	Incomplete Transfer 0 Normal operation. 1 Some of the bytes of the requested master transfer were not transferred.	For an incomplete transfer, read the transfer count, IIC0_XFRCNT, to determine how bytes were transferred.
7	XFRA	Transfer Aborted 0 No transfer is pending, or transfer is in progress. 1 A requested master transfer was aborted by a NACK during the transfer of the address byte, or was aborted because arbitration was lost. Lost arbitration can be caused by the loss of data during the transfer of the second or subsequent data byte.	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.

IIC0_HMADR

IIC0 High Master Address

MMIO 0xEF600505

See "IIC0 High Master Address Register" on page 22-6.

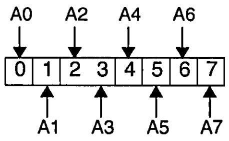


Figure 25-65. IIC0 High Master Address Register (IIC0_HMADR)

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.

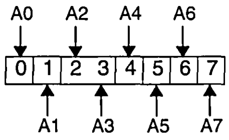


Figure 25-66. IIC0 High Slave Address Register (IIC0_HSADR)

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.

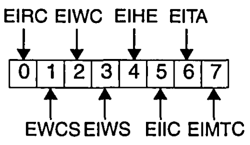


Figure 25-67. IIC0 Interrupt Mask Register (IIC0_INTRMSK)

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:** IIC0XTCNTLSS[SWC] = 1 indicates a Slave Write Compete.
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.

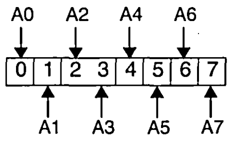


Figure 25-68. IIC0 Low Master Address Register (IIC0_LMADR)

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.

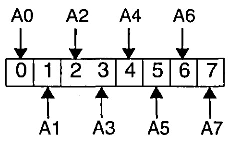


Figure 25-69. IIC0 Low Slave Address Register (IIC0_LSADR)

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

MMIO 0xEF600500

See “IIC0 Master Data Buffer” on page 22-3.

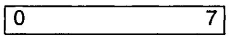


Figure 25-70. IIC0 Master Data Buffer (IIC0_MDBUF)

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.

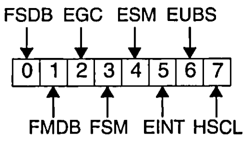


Figure 25-71. IIC0 Mode Control Register (IIC0_MDCNTL)

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 IICSCL 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.

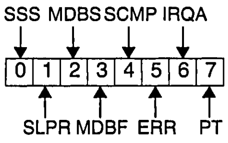


Figure 25-73. IIC0 Status Register (IIC0_STS)

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[SRC, 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.

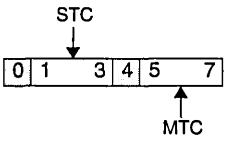


Figure 25-74. IIC0 Transfer Count Register (IIC0_XFRCNT)

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.

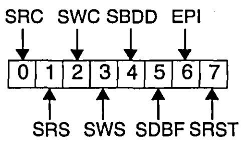


Figure 25-75. IIC0 Extended Control and Slave Status Register (IIC0_XTCNTLSS)

0	SRC	Slave Read Complete 0 Normal operation, or IIC0_MDCNTL[HSCL] = 0, IIC0_SDBUF is empty, and a read operation is in progress. 1 A NACK or Stop condition was received over the IIC bus, or a repeated Start condition ended a read operation.	Check whether the read operation emptied IIC0_SDBUF.
1	SRS	Slave Read Needs Service 0 Normal operation or slave read does not need service. 1 IIC0_SDBUF is empty, and a read operation was requested on the IIC bus. The set condition may also indicate that IIC0_SDBUF is empty due to a slave read and additional data is requested by the master.	1. If IIC0_MDCNTL[HSCL]=0 and IIC0_SDBUF contains no data, the slave will issue a NACK and IIC0_XTCNTLSS[SRS] is set. 2. If ICOMDCNTL[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. 3. If IIC0_MDCNTL[HSCL]=0, and IIC0_SDBUF contains no data, the slave will hold IICSCL low to indicate the slave is busy. IIC0_XTCNTLSS[SRS] is set until the IIC0_SDBUF is filled. Once filled, IICSCL is released, IIC0_XTCNTLSS[SRS] is cleared, and the slave sends the data. 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.
2	SWC	Slave Write Complete 0 Normal operation or slave write in progress. 1 A Stop signal was received during a write operation, or a repeated Start condition ended a write operation.	

3	SWS	Slave Write Needs Service 0 Normal operation or slave write does not need service. 1 IIC0_SDBUF is full during a slave write.	1. If IIC0_MDCNTL[HSCL] = 1 and IIC0_SDBUF is full, the slave will hold IICSCL low to indicate the slave is busy. IIC0_XTCNTLSS[SWS] is set until IIC0_SDBUF is empty. Once empty, IICSCL is released, IIC0_XTCNTLSS[SWS] is cleared, and the slave receives the data. 2. If IIC0_MDCNTL[HSCL] = 0 and IIC0_SDBUF is full, the slave will issue a NACK and IIC0_XTCNTLSS[SWS] is set.
4	SBDD	Slave Data Buffer Has Data 0 IIC0_SDBUF is empty 1 IIC0_SDBUF contains data	Read-only
5	SDBF	Slave Data Buffer Full 0 IIC0_SDBUF is not full 1 IIC0_SDBUF is full	Read-only
6	EPI	Enable Pulsed IRQ on Transfer Aborted 0 The internal IIC interrupt signal to the UIC remains active until the status is cleared, IIC0STS[IRQA] =0. 1 The internal IIC interrupt signal to the PPC405GP UIC is active for one OPB clock cycle.	
7	SRST	Soft Reset 0 Normal operation 1 Soft reset	

LR

Link Register

SPR 0x008

See "Link Register (LR)" on page 3-8.

0	31

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.

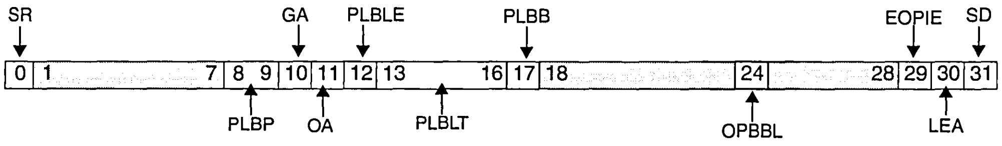


Figure 25-77. MAL Configuration Register (MAL0_CFG)

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 (MAL0_ESR)" on page 20-29.

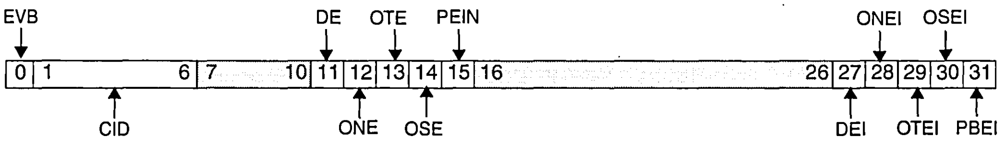


Figure 25-78. MAL Error Status Register (MAL0_ESR)

0	EVB	Error Valid Bit 0 Bit 1:15 are available for latching new error information. 1 Bits 1:15 contain last error. A new error cannot be latched.	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. 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. 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
1:6	CID	Channel ID	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). Bits 2:6 indicates the number of the channel that caused the error. 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.
7:10		Reserved	
11	DE	Descriptor Error 0 No error 1 Non-valid descriptor	Indicates that the error is a non-valid descriptor, which is *not* the first descriptor in a TX packet.
12	ONE	OPB Non-fullword Error 0 No error 1 Non-fullword asserted	Indicates that the error is a non-fullword acknowledge asserted by an OPB slave.
13	OTE	OPB Timeout Error 0 No error 1 OPB timeout	Indicates the error is an OPB timeout.

MAL0_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 *not* 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.

DCR 0x182 Read/Only

See "MAL Interrupt Enable Register (MAL0_IER)" on page 20-31.

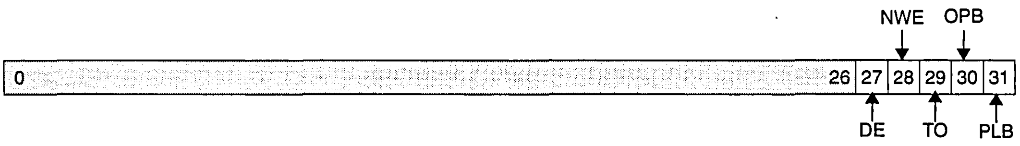


Figure 25-79. MAL Interrupt Enable Register (MAL0_IER)			
0:26	▨	Reserved	
27	DE	Descriptor Error	When set, this bit enables the descriptor error (descriptor not valid) interrupt.
28	NWE	Non_W_Err_Int_Enable	When set, this bit enables OPB non-word transfer error interrupt.
29	TO	Time_Out_Int_Enable	When 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.

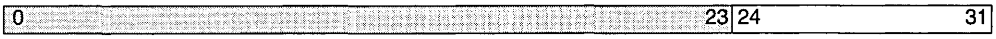


Figure 25-80. RX Channel Buffer Size Register 0 (MAL0_RCBS0)

0:23	▨	Reserved
24:31		Receive Channel Buffer Size

DCR 0x191

See “Channel Active Set and Reset Registers” on page 20-26.

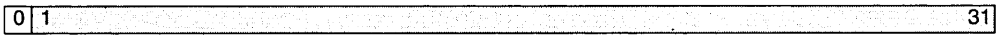


Figure 25-81. RX Channel_Active Reset Register (MAL0_RXCARR)

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.

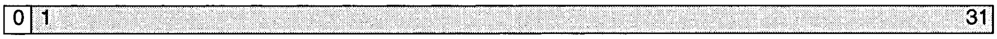


Figure 25-82. RX Channel_Active Set Register (MAL0_RXCASR)

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	

DCR 0x1C0

See “Channel Table Pointer Registers (MAL0_TXCTPxR, MAL0_RXCTP0R)” on page 20-33.

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.

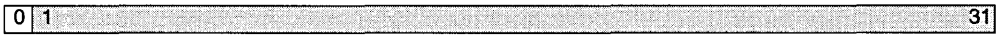


Figure 25-84. RX Descriptor Error Interrupt Register (MAL0_RXDEIR)

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	

DCR 0x192 Read/Clear

See “End of Buffer Interrupt Status Registers” on page 20-28.

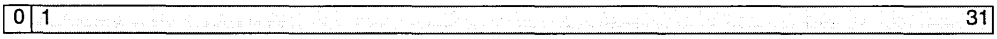


Figure 25-85. RX End of Buffer Interrupt Status Register (MAL0_RXEOBISR)

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.

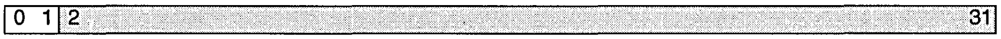


Figure 25-86. TX Channel_Active Reset Register (MAL0_TXCARR)

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	

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).

DCR 0x187 Read/Clear

See “Descriptor Error Interrupt Registers (MAL0_TXDEIR, MAL0_RXDEIR)” on page 20-32.

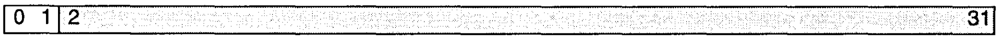


Figure 25-89. TX Descriptor Error Interrupt Register (MAL0_TXDEIR)

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.

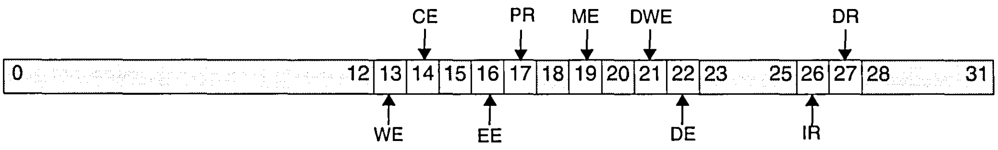


Figure 25-91. Machine State Register (MSR)

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.	

27	DR	Data Relocate 0 Data address translation is disabled. 1 Data address translation is enabled.
28:31	▨	Reserved

DCR 0x01A

See "OCM Data-Side Address Range Compare Register (OCM0_DSARC)" on page 5-6.

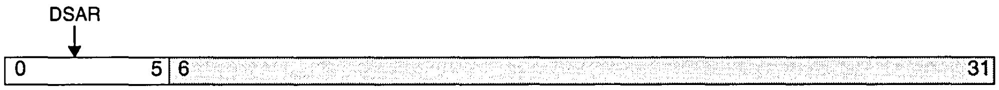


Figure 25-92. OCM Data-Side Address Range Compare Register (OCM0_DSARC)

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.

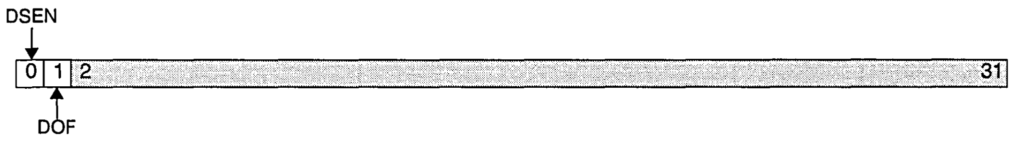


Figure 25-93. OCM Data-Side Control Register (OCM0_DSCNTL)

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

DCR 0x018

See “OCM Instruction-Side Address Range Compare Register (OCM0_ISARC)” on page 5-5.

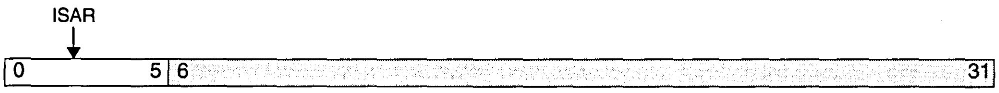


Figure 25-94. OCM Instruction-Side Address Range Compare Register (OCM0_ISARC)

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.

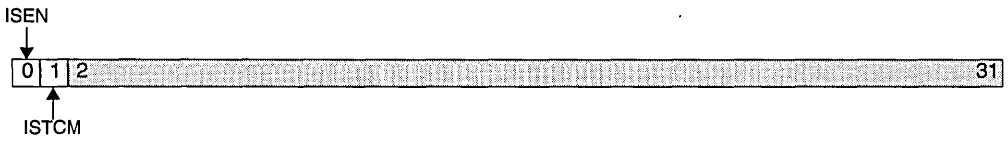


Figure 25-95. OCM Instruction-Side Control Register (OCM0_ISCNTL)

0	ISEN	Instruction-Side OCM Enable 0 Instruction-side OCM accesses are disabled. 1 Instruction-side OCM accesses are enabled.	
1	ISTCM	Instruction-Side Two Cycle Mode 0 Instruction-side OCM accesses are returned in one cycle. 1 Instruction-side OCM accesses are returned in two cycles.	OCM0_ISCNTL[ISTCM], which has a reset value of 1, should be set to OCM0_ISCNTL[ISTCM] = 0 during chip initialization.
2:31	▨	Reserved	

MMIO 0xEF600600

See “OPB Arbiter Control Register (OPBA0_CR)” on page 2-12.

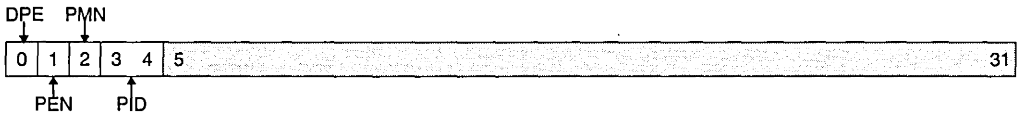


Figure 25-96. OPB Arbiter Control Register (OPBA0_CR)

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.

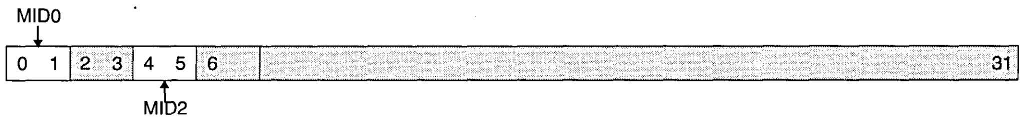


Figure 25-97. OPB Arbiter Priority Register (OPBA0_PR)

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	

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x10 Read-Only

See “Unused PCI Base Address Register Space” on page 17-37.

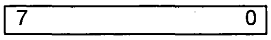


Figure 25-98. PCI Base Address Register (PCIC0_BAR0)

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.

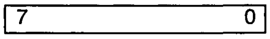


Figure 25-99. PCI Built-in Self Test Control Register (PCIC0_BIST)

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.

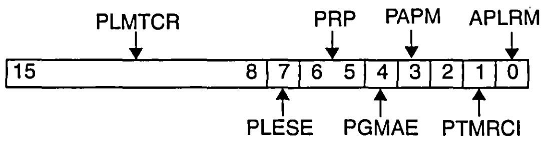


Figure 25-100. Bridge Options 1 Register (PCIC0_BRDGOPT1)

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	PAPM	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 se t 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.

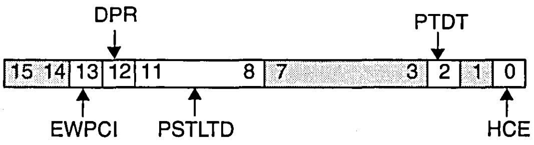


Figure 25-101. Bridge Options 2 Register (PCIC0_BRDGOPT2)

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 $\overline{\text{PCIReset}}$ pin to be asserted.	Software that asserts this bit must leave is asserted long enough to guarantee the PCI pulse width requirements. DPR does not reset PLB bus interface registers or PCI bridge registers. $\overline{\text{PCIReset}}$ 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, PSSLTD equals the maximum number of PCI clocks to disconnect. In asynchronous mode, PSSLTD 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 give the local CPU (PLB master) time to initialize them before the host sees them.

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.

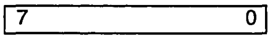


Figure 25-103. PCI Capabilities Pointer (PCIC0_CAP)

7:0		PCI Capabilities Pointer

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x58 Read-Only

See “Capability Identifier (PCIC0_CAPID)” on page 17-49.

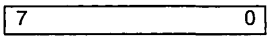


Figure 25-104. Capability Identifier (PCIC0_CAPID)

7:0		PCI Capability Identifier

PCIC0_CFGADDR

PCI Configuration Address Register

0xEEC00000

See “PCI Configuration Address Register (PCIC0_CFGADDR)” on page 17-29.

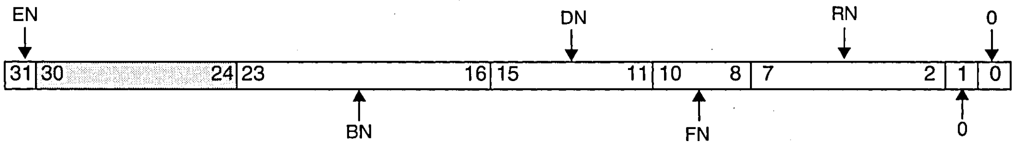


Figure 25-105. PCI Configuration Address Register (PCIC0_CFGADDR)

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	

0xEEC00004

See “PCI Configuration Data Register (PCIC0_CFGDATA)” on page 17-30.

Figure 25-106. PCI Configuration Data Register (PCIC0_CFGDATA)

31:0		Configuration Data

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.

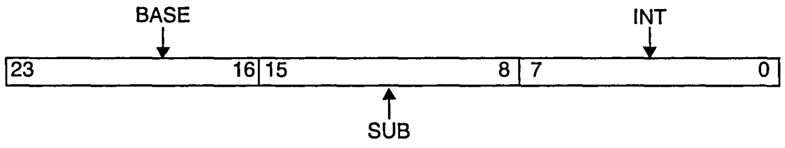


Figure 25-107. PCI Class Register (PCIC0_PCICLS)

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.

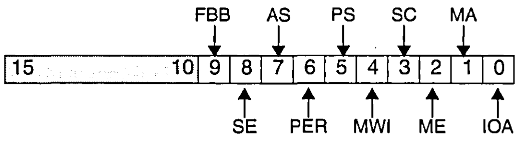


Figure 25-108. PCI Command Register (PCIC0_CMD)

15:10		Reserved	
9	FBB	Fast Back-to-Back Write Enable	Enables PCI masters to perform fast back-to-back transactions. Because he PCI bridge does not perform fast back-to-back transactions; FBB is read-only and returns 0 when read.
8	SE	$\overline{\text{PCISErr}}$ Enable 0 Disabled 1 Enabled	Enables driving $\overline{\text{PCISErr}}$ 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: • 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 $\overline{\text{PCIPErr}}$ (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.

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x5F Read-Only

See “PCI Data Register (PCIC0_DATA)” on page 17-52.

7	0

Figure 25-109. PCI Data (PCIC0_DATA)

7:0		PCI 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.

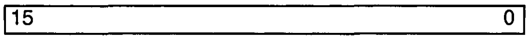


Figure 25-110. PCI Device ID Register (PCIC0_DEVID)

15:0		PCI Device ID

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x48

See "Error Enable Register (PCIC0_ERREN)" on page 17-42.

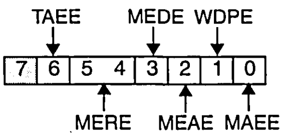


Figure 25-111. Error Enable Register (PCIC0_ERREN)

7		Reserved	
6	TAEE	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 TAEE 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 $\overline{\text{PCISErr}}$ on the PCI bus. 10 Target should target abort the offending read. 11 Indicates the PCI target should drive $\overline{\text{PCISErr}}$ 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 $\overline{\text{PCISErr}}$ Enable 0 Disabled 1 Enabled.	The PCI bridge drives $\overline{\text{PCISErr}}$ 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 Sl_MErr on the PLB bus in response to a master abort. If this bit is disabled, driving of Sl_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.

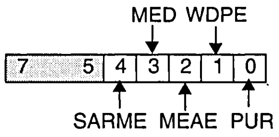


Figure 25-112. Error Status Register (PCIC0_ERRSTS)

7:5		Reserved	
4	SARME	$\overline{\text{PCISErr}}$ Asserted on Received PLB Bus Error	Set when PCI bridge asserts $\overline{\text{PCISErr}}$ 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	$\overline{\text{PCISerr}}$ on Write Data Parity Error	Set when the PCI bridge drives $\overline{\text{PCISErr}}$ in response to a data parity error detected on a PCI write to PLB memory. $\overline{\text{PCIPErr}}$ 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.

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x0E Read-Only

See “PCI Header Type Register (PCIC0_HDTYPE)” on page 17-36.

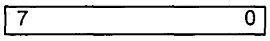


Figure 25-113. PCI Header Type Register (PCIC0_HDTYPE)

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.

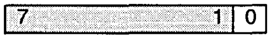


Figure 25-114. PCI Interrupt Control/Status Register

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.

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x3C

See “PCI Interrupt Line Register (PCIC0_INTLN)” on page 17-40.

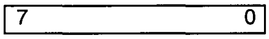


Figure 25-115. PCI Interrupt Line Register (PCIC0_INTLN)

7:0		PCI Interrupt Line

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)

7:0		PCI Interrupt Pin

Accessed using PCICO_CFGADDR, PCICO_CFGDATA; offset 0x0D

See “PCI Latency Timer Register (PCICO_LATTIM)” on page 17-36.

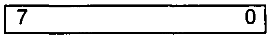


Figure 25-117. PCI Latency Timer Register (PCICO_LATTIM)

7:0		PCI Latency Timer

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.

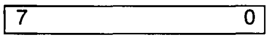


Figure 25-118. PCI Maximum Latency Register (PCIC0_MAXLTNCY)

7:0		PCI Maximum Latency

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.

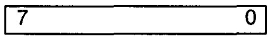


Figure 25-120. Next Item Pointer (PCIC0_NEXTIPTR)

7:0		PCI Next Item Pointer

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.

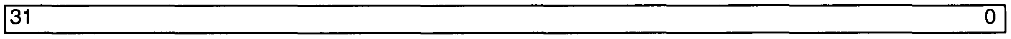


Figure 25-121. PLB Slave Error Address Register (PCIC0_PLBBEAR)

31:0		PLB Slave Error Address

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.

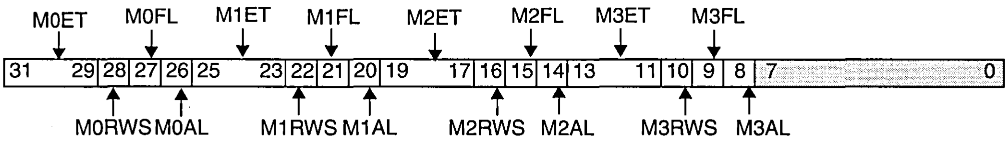


Figure 25-122. PLB Slave Error Syndrome Register 0 (PCIC0_PLBBESR0)

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

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.

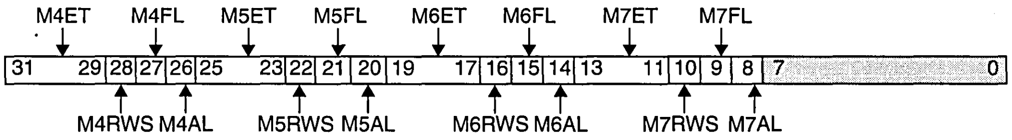


Figure 25-123. PLB Slave Error Syndrome 1 (PCIC0_PLBBESR1)

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	

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.

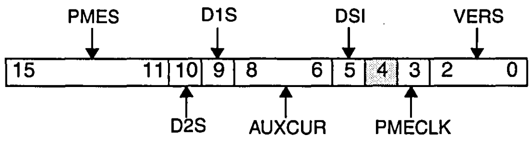


Figure 25-124. Power Management Capabilities Register (PCIC0_PMC)

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 *PCI Power Management Interface Specification.*

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x5D–0x5C

See “Power Management Control/Status Register (PCIC0_PMCSR)” on page 17-50.

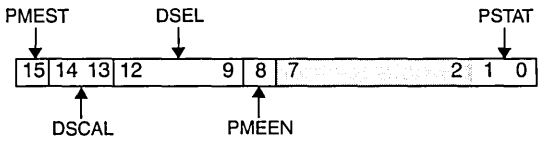


Figure 25-125. Power Management Control/Status Register (PCIC0_PMCSR)

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 fiield may change PCI0_PMSCRR.

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.

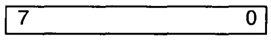


Figure 25-126. PMCSR PCI to PCI Bridge Support Extensions (PCIC0_PMCSRBSE)

7:0		PCI to PCI Bridge Support Extensions

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x64

See "Power Management State Change Request Register (PCIC0_PMSCRR)" on page 17-54.

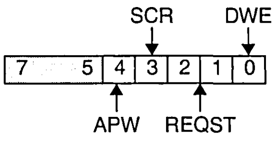


Figure 25-127. Power Management State Change Request Register (PCIC0_PMSCRR)

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.

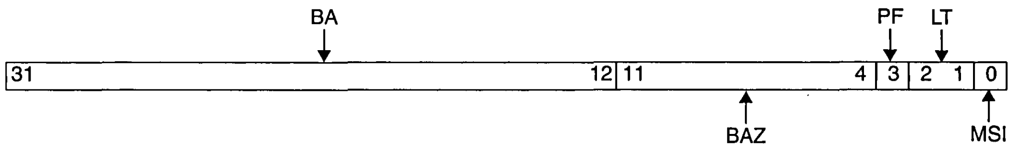


Figure 25-128. PCI PTM 1 BAR Register (PCIC0_PTM1BAR)

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.

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x1B–0x18

See "PCI PTM 2 BAR (PCIC0_PTM2BAR)" on page 17-38.

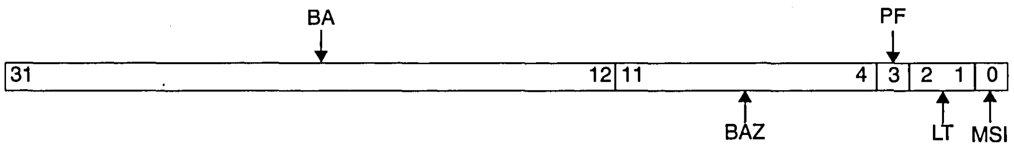


Figure 25-129. PCI PTM 2 BAR Register (PCIC0_PTM2BAR)

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.

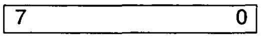


Figure 25-130. PCI Revision ID Register (PCIC0_REVID)

7:0		Revision ID	Revision level of device.

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; Offset 0x2D–0x2C

See “PCI Subsystem ID Register (PCIC0_SBSYSID)” on page 17-39.

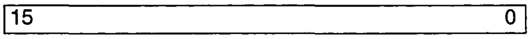


Figure 25-131. PCI Subsystem ID Register (PCIC0_SBSYSID)

15:0		PCI Subsystem ID

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.

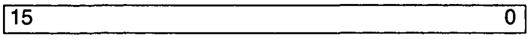


Figure 25-132. PCI Subsystem Vendor ID Register (PCIC0_SBSYSVID)

15:0		PCI Subsystem Vendor ID

Accessed using PCIC0_CFGADDR, PCIC0_CFGDATA; offset 0x07–0x06

See “PCI Status Register (PCIC0_STATUS)” on page 17-33.

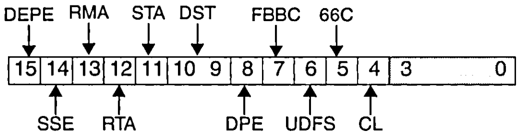


Figure 25-133. PCI Status Register (PCIC0_STATUS)

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: • 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 $\overline{\text{PCISErr}}$ (see “Error Handling” on page 17-55 for causes of $\overline{\text{PCISErr}}$ 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	$\overline{\text{PCIDevSel}}$ Response Timing Read-only.	The PCI bridge asserts $\overline{\text{PCIDevSel}}$ on the second clock after $\overline{\text{PCIFframe}}$ 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: • The PCI bridge detects a data parity error ($\overline{\text{PCIPErr}}$ is asserted) when the PCI bridge is the master on a PCI read cycle, or is the master when it samples $\overline{\text{PCIPErr}}$ 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.

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)

15:0		Vendor ID

PCIL0_PMM0LA

PMM 0 Local Address

MMIO 0xEF400000

See “PMM 0 Local Address Register (PCIL0_PMM0LA)” on page 17-21.

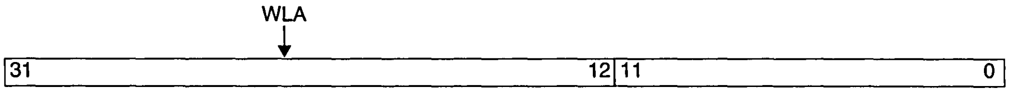


Figure 25-135. PMM 0 Local Address Register (PCIL0_PMM0LA)

31:12	WLA	Writable PLB Local Address	
11:0		PLB Local Address	Always 0

MMIO 0xEF400004

See "PMM 0 Mask/Attribute Register (PCIL0_PMM0MA)" on page 17-22.

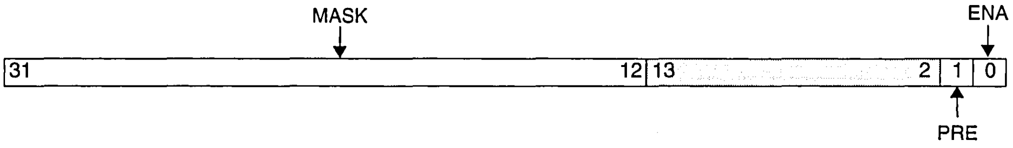


Figure 25-136. PMM 0 Mask/Attribute Register (PCIL0_PMM0MA)

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 PCIL0_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 PCIL0_PMM0LA, PCIL0_PMM0PCIHA, and PCIL0_PMM0PCILA must be initialized before enabling.

PCIL0_PMM0PCIHA

PMM 0 PCI High Address

MMIO 0xEF40000C

See “PMM 0 PCI High Address Register (PCIL0_PMM0PCIHA)” on page 17-23.

31	0

Figure 25-137. PMM 1 PCI High Address Register (PCIL0_PMM1PCIHA)

31:0		PCI High Address

MMIO 0xEF400008

See “PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA)” on page 17-22.

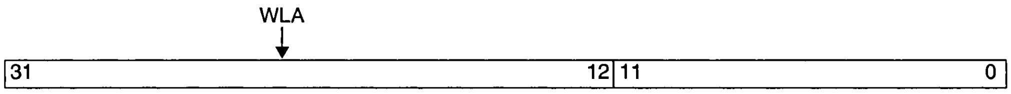


Figure 25-138. PMM 0 PCI Low Address Register (PCIL0_PMM0PCILA)

31:12	WLA	Writable PCI Low Address	
11:0		PCI Low Address	Always 0

PCIL0_PMM1LA

PMM 1 Local Address

MMIO 0xEF400010

See "PMM 1 Local Address Register (PCIL0_PMM1LA)" on page 17-23.

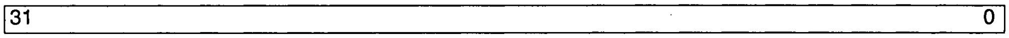


Figure 25-139. PMM 1 Local Address Register (PCIL0_PMM1LA)

31:0		PLB Local Address

MMIO 0xEF400014

See “PMM 1 Mask/Attribute Register (PCIL0_PMM1MA)” on page 17-24.

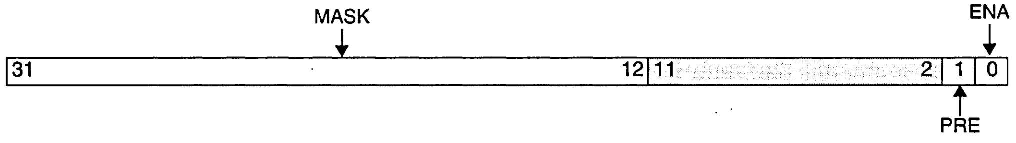


Figure 25-140. PMM 1 Mask/Attribute Register (PCIL0_PMM1MA)

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.

31	0

Figure 25-141. PMM 1 PCI High Address Register (PCIL0_PMM1PCIHA)

31:0		PCI High 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)

31:0		PCI Low Address

PCIL0_PMM2LA

PMM 2 Local Address

MMIO 0xEF400020

See “PMM 2 Local Address Register (PCIL0_PMM2LA)” on page 17-25.

31	0

Figure 25-143. PMM 2 Local Address Register (PCIL0_PMM2LA)

31:0		PLB Local Address

MMIO 0xEF400024

See "PMM 2 Mask/Attribute Register (PCILO_PMM2MA)" on page 17-25.

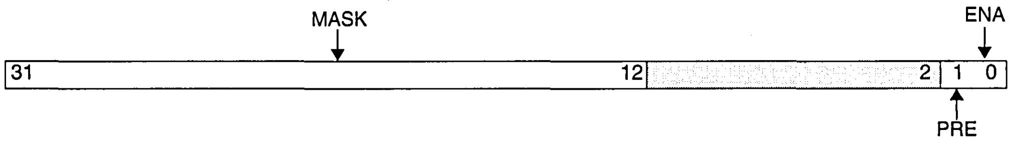


Figure 25-144. PMM 2 Mask/Attribute Register (PCILO_PMM2MA)

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 PCILO_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 PCILO_PMM2LA, PCILO_PMM2PCIHA, and PCILO_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.

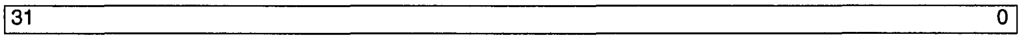


Figure 25-145. PMM 2 PCI High Address Register (PCIL0_PMM2PCIHA)

31:0		PCI High Address

MMIO 0xEF400028

See "PMM 2 PCI Low Address Register (PCIL0_PMM2PCILA)" on page 17-26.

31	0

Figure 25-146. PMM 2 Low Address Register (PCIL0_PMM2PCILA)

31:0		PCI Low Address

PCIL0_PTM1LA

PTM 1 Local Address

MMIO 0xEF400034

See "PTM 1 Local Address Register (PCIL0_PTM1LA)" on page 17-27.

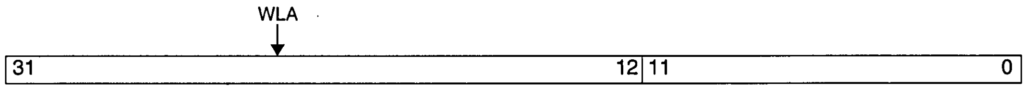


Figure 25-147. PTM 2 Local Address Register (PCIL0_PTM1LA)

31:12	WLA	Writable PTM 1 Local Address	Writable
11:0		PTM 1 Local Address	Always 0

MMIO 0xEF400030

See “PTM 1 Memory Size/Attribute Register (PCIL0_PTM1MS)” on page 17-27.

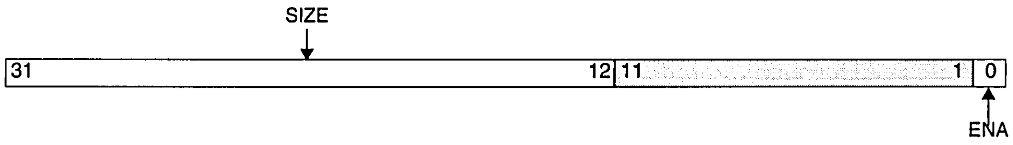


Figure 25-148. PTM 1 Memory Size/Attribute Register (PCIL0_PTM1MS)

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.

31	0

Figure 25-149. PTM 2 Local Address Register (PCIL0_PTM2LA)

31:0		PTM 2 Local Address

MMIO 0xEF400038

See "PTM 2 Memory Size/Attribute Register (PCIL0_PTM2MS)" on page 17-28.

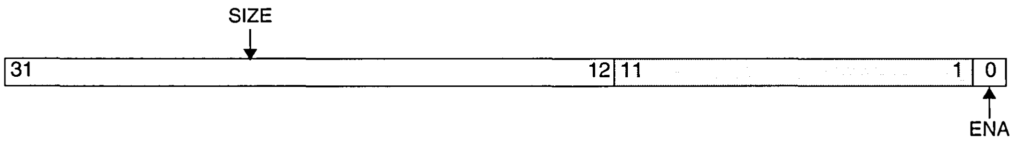


Figure 25-150. PTM 2 Memory Size/Attribute Register (PCIL0_PTM2MS)

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.

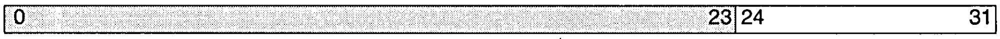


Figure 25-151. Process ID (PID)

0:23	▨	Reserved
24:31		Process ID

SPR 0x3DB

See “Programmable Interval Timer (PIT)” on page 11-4.

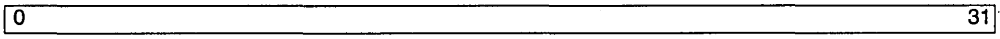


Figure 25-152. Programmable Interval Timer (PIT)

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.

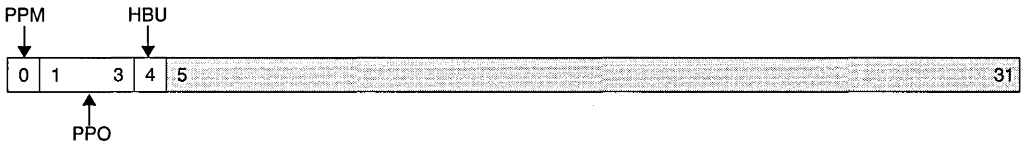


Figure 25-153. PLB Arbiter Control Register (PLB0_ACR)

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

DCR 0x086

See “PLB Error Address Register (PLB0_BEAR)” on page 2-5.

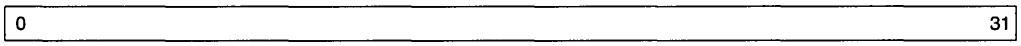


Figure 25-154. PLB Error Address Register (PLB0_BEAR)

0:31		Address of bus timeout error

PLB0_BESR

PLB Error Status Register

DCR 0x084 Read/Clear

See "PLB Error Status Register (PLB0_BESR)" on page 2-6.

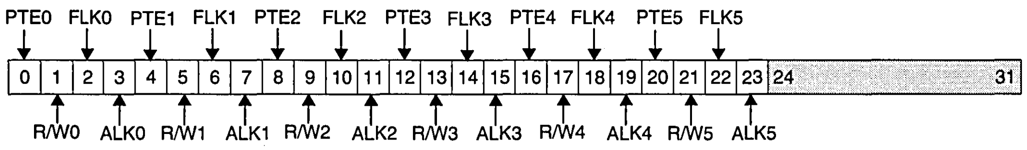


Figure 25-155. PLB Error Status Register (PLB0_BESR)

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 1PLB0_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	

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.

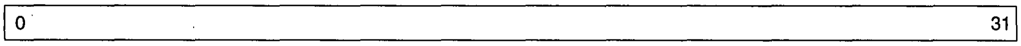


Figure 25-156. Bridge Error Address Register (POB0_BEAR)

0:31	0x0B2	Address of bus error

DCR 0x0A0 Read/Clear

See "Bridge Error Status Registers (POB0_BESR0–POB0_BESR1)" on page 2-8.

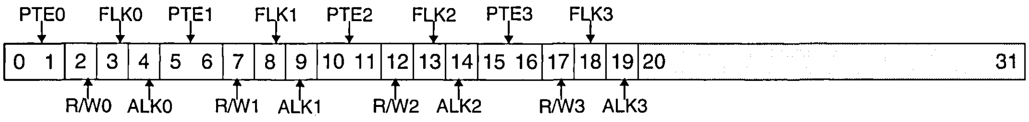


Figure 25-157. Bridge Error Status Register 0 (POB0_BESR0)

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	Master 3 is PCI.
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.

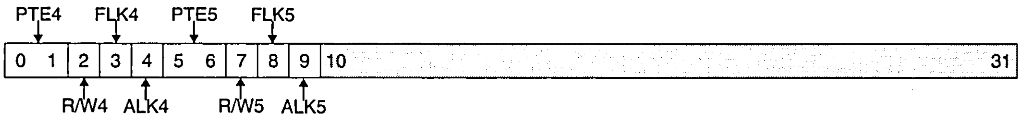


Figure 25-158. Bridge Error Status Register 1 (POB0_BESR1)

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.

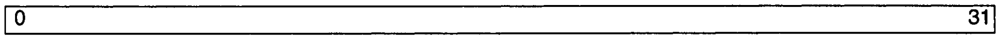


Figure 25-159. Processor Version Register (PVR)

0:31		Assigned PVR value

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x40–0x4C

See "Memory Bank 0–3 Configuration (SDRAM0_B0CR–SDRAM0_B3CR)" on page 15-6.

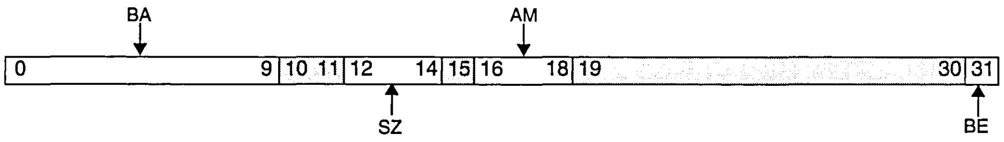


Figure 25-160. Memory Bank 0–3 Configuration Registers (SDRAM0_B0CR–SDRAM0_B3CR)

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.

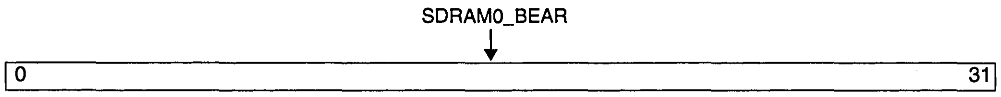


Figure 25-161. Bus Error Address Register (SDRAM0_BEAR)

0:31	SDRAM0_BEAR	Address of ECC Error.

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x00 Read/Clear

See "Bus Error Syndrome Register 0 (SDRAM0_BESR0)" on page 15-17.

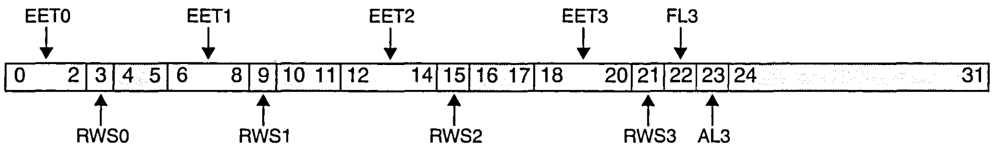


Figure 25-162. Bus Error Syndrome Register 0 (SDRAM0_BESR0)

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

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x08 Read/Clear

See "Bus Error Syndrome Register 1 (SDRAM0_BESR1)" on page 15-18.

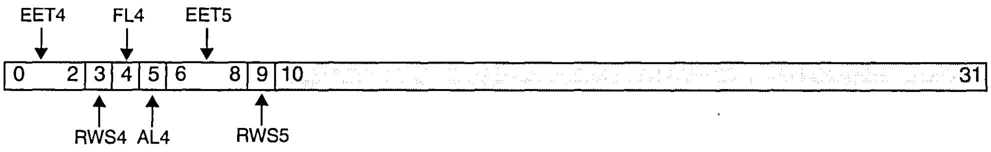


Figure 25-163. Bus Error Status Register 1 (SDRAM0_BESR1)

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.

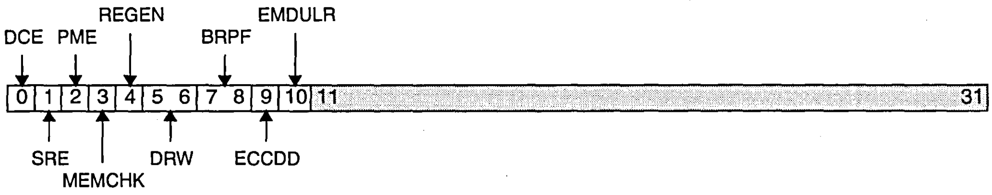


Figure 25-164. Memory Controller Configuration (SDRAM0_CFG)

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.	

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.

DCR 0x011

See “Accessing SDRAM Registers” on page 15-2.

This register is used to indirectly access SDRAM controller DCRs.

SDRAM0_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.

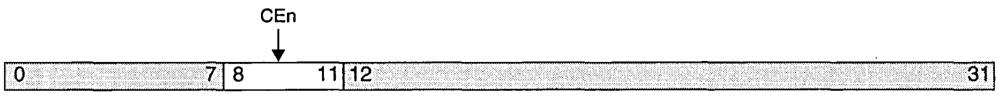


Figure 25-165. ECC Configuration Register (SDRAM0_ECCCFG)

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 ($\overline{\text{BankSeln}}$). When cleared, the ECC logic ignores the check bits and passes read data along unmodified.
12:31	▨	Reserved	

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x98 Read-Only

See "ECC Error Status Register (SDRAM0_ECCESR)" on page 15-16.

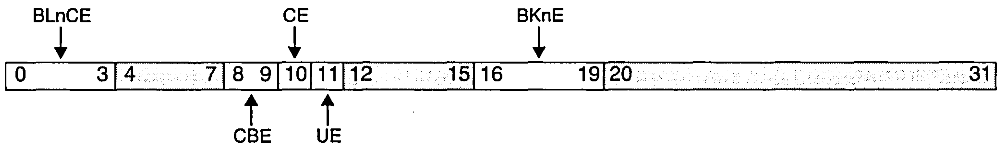


Figure 25-166. ECC Error Status Register (SDRAM0_ECCESR)

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.

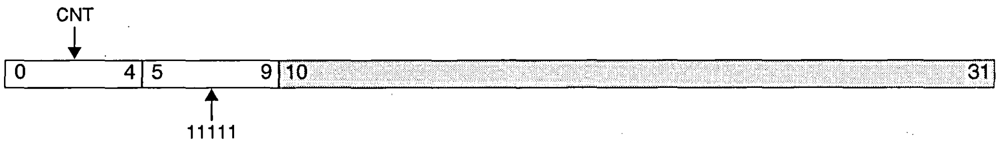


Figure 25-167. Power Management Idle Timer (SDRAM0_PMIT)

0:4	CNT	Count 0-31
5:9		Always 0b11111
10:31	▨	Reserved

DCR Accessed using SDRAM0_CFGADDR; SDRAM0_CFGDATA; Offset 0x30

See “Refresh Timer Register (SDRAM0_RTR)” on page 15-13.

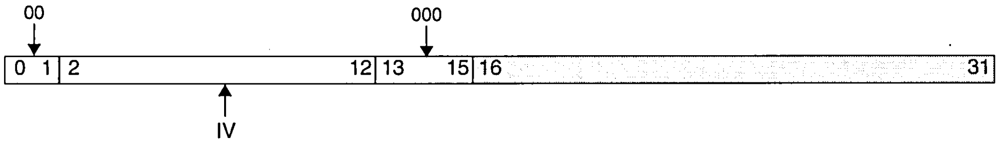


Figure 25-168. Refresh Timing Register (SDRAM0_RTR)

0:1		Always 0b00	
2:12	IV	Interval Programmable between 0b00000000000 and 0b11111111111 (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.

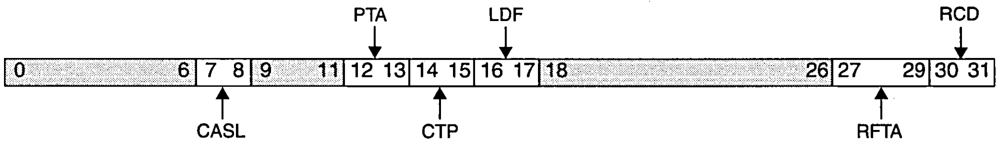


Figure 25-169. SDRAM Timing Register (SDRAM0_TR)

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

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.

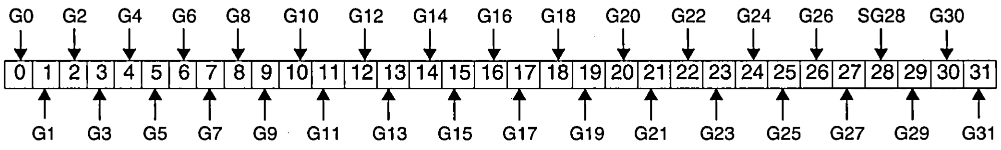


Figure 25-170. Storage Guarded Register (SGR)

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

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.

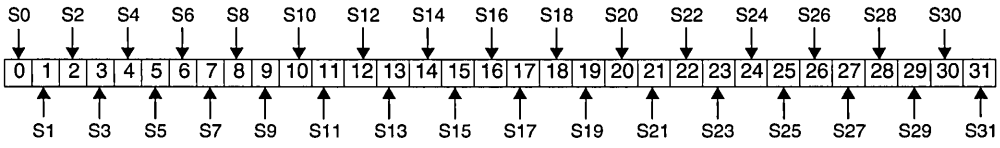


Figure 25-171. Storage Little-Endian Register (SLER)

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

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.

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.

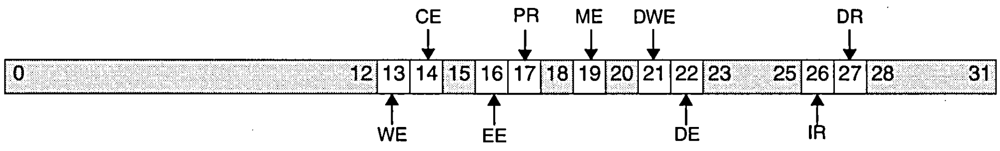


Figure 25-174. Save/Restore Register 1 (SRR1)

0:31	SRR1 receives a copy of the MSR when an interrupt is taken; the MSR is restored from SRR1 when **rfi** executes.

SPR 0x3DE

See “Save/Restore Registers 2 and 3 (SRR2–SRR3)” on page 10-30.

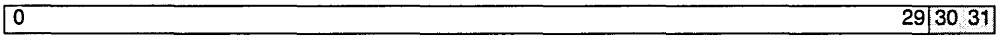


Figure 25-175. Save/Restore Register 2 (SRR2)

0:29		SRR2 receives an instruction address when a critical interrupt is taken; the Program Counter is restored from SRR2 when **rfci** executes.
30:31	▨	Reserved

SRR3

Save/Restore Register 3

SPR 0x3DF

See “Save/Restore Registers 2 and 3 (SRR2–SRR3)” on page 10-30.

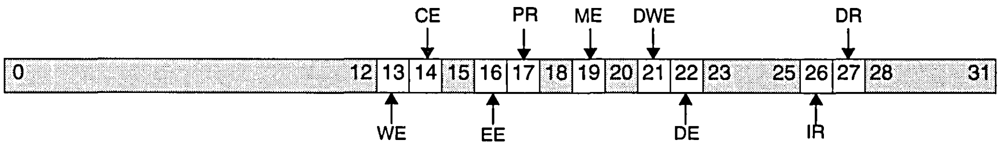


Figure 25-176. Save/Restore Register 3 (SRR3)

0:31	SRR3 receives a copy of the MSR when a critical interrupt is taken; the MSR is restored from SRR3 when **rfci** executes.

SPR 0x3BC

See “Real-mode Storage Attribute Control” on page 6-17.

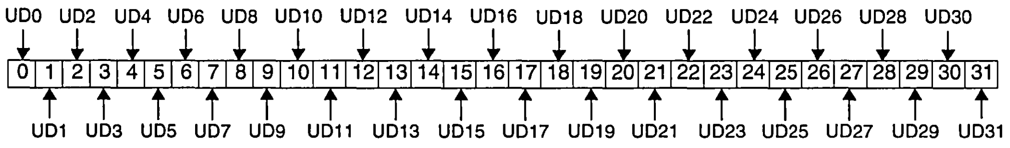


Figure 25-177. Storage User-defined 0 Register (SU0R)

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

SU0R (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

TBR 0x10C (Read-only); SPR 0x11C (Privileged write-only)

See “Time Base” on page 11-2.

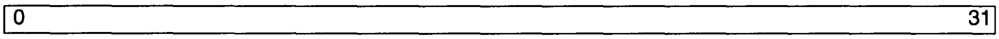


Figure 25-178. Time Base Lower (TBL)

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.

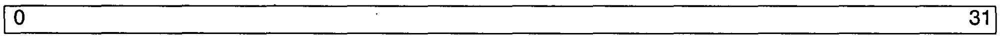


Figure 25-179. Time Base Upper (TBU)

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.

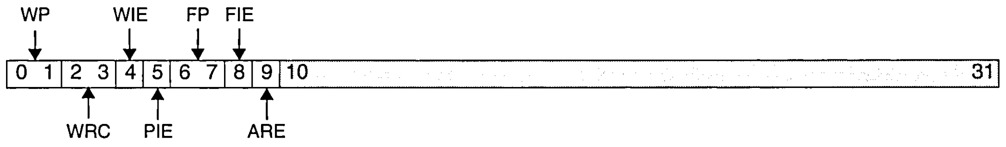


Figure 25-180. Timer Control Register (TCR)

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.

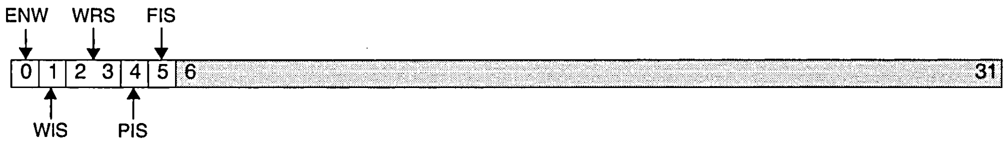


Figure 25-181. Timer Status Register (TSR)

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.

8 15

Figure 25-182. UART Baud-Rate Divisor Latch (LSB) Registers (UARTx_DLL)

8:15		Data bits

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

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.

0	7

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.

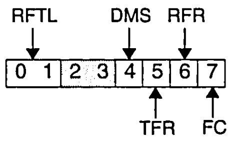


Figure 25-184. UART FIFO Control Registers (UARTx_FCR)

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.

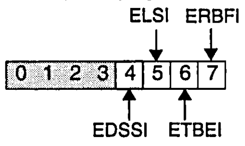


Figure 25-185. UART Interrupt Enable Registers (UARTx_IER)

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.

MMIO 0xEF600302, 0xEF600402 Read-Only

See “Interrupt Identification Registers (UARTx_IIR)” on page 21-6.

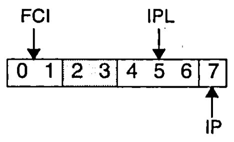


Figure 25-186. UART Interrupt Identification Registers (UARTx_IIR)

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.

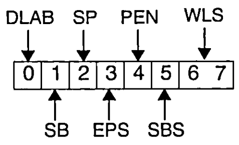


Figure 25-187. UART Line Control Registers (UARTx_LCR)

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.

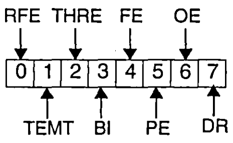


Figure 25-188. UART Line Status Registers (UARTx_LSR)

0	RFE	Receiver FIFO Error Indicator 0 In FIFO mode, reset to 0 when the processor reads the UARTx_LSR, provided there are no subsequent errors in the FIFO. 1 There are one or more instances of parity error, framing error or break indication in the FIFO.	Always 0 in 16450 mode.
1	TEMT	Transmitter Empty Indicator 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. 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.	
2	THRE	Transmitter Holding Register Empty Indicator 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. 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.	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	Break Interrupt Indicator. 0 Reset to 0 whenever processor reads Line Status Register (LSR). 1 Set to 1 whenever the received data input is held at the spacing level (0) for longer than a full word transmission time.	The full word transmission time is the time required for the start bit, data bits (can be 5–8 bits), parity and stop bits. In FIFO mode, this error is revealed to the processor when the character this error is associated with is at the top of the FIFO. Only one 0 character is loaded into the receiver FIFO when a break occurs. After the next valid start bit is received and has gone into the marking state, the next character transfer is enabled. Error causes a Receiver Line Status Interrupt.

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

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

MMIO 0xEF600304, 0xEF600404

See "Modem Control Registers (UARTx_MCR)" on page 21-10.

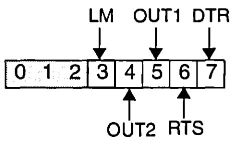


Figure 25-189. UART Modem Control Registers (UARTx_MCR)

0:2		Reserved	Always 0.
3	LM	Loopback Mode 0 Disabled 1 Enabled	Provides a local loopback feature for diagnostic testing of the UART. The following occurs: 1. SOUT is set to the marking state (logic 1) SIN is disconnected. 2. The output of the transmitter shift register feeds the input of the receiver shift register. 3. The four modem control inputs $\overline{\text{DSR}}$, $\overline{\text{CTS}}$, $\overline{\text{RI}}$, and $\overline{\text{DCD}}$ are disconnected. 4. The four modem control outputs $\overline{\text{DTR}}$, $\overline{\text{RTS}}$, $\overline{\text{OUT1}}$, and $\overline{\text{OUT2}}$ are set to a logic 1 (their inactive state). 5. The four modem control outputs are connected internally to the four modem control inputs. Transmitted data is immediately received to verify the UART transmit and receive data paths. 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.
4	OUT2	User Output 2 0 $\overline{\text{OUT2}}$ inactive (1) 1 $\overline{\text{OUT2}}$ active (0)	Auxiliary user designated output.
5	OUT1	User Output 1 0 $\overline{\text{OUT1}}$ inactive (1) 1 $\overline{\text{OUT1}}$ active (0)	Auxiliary user designated output.
6	RTS	Request To Send 0 $\overline{\text{RTS}}$ inactive (1) 1 $\overline{\text{RTS}}$ active (0)	
7	DTR	Data Terminal Ready 0 $\overline{\text{DTR}}$ inactive (1) 1 $\overline{\text{DTR}}$ active (0)	

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

UARTx_MSR

UART Modem Status Registers

MMIO 0xEF600306, 0xEF600406

See "Modem Status Registers (UARTx_MSR)" on page 21-13.

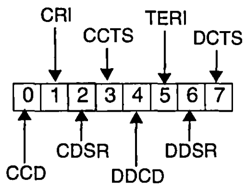


Figure 25-190. UART Modem Status Registers (UARTx_MSR)

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.

MMIO 0xEF600300, 0xEF600400 Read-Only
See “Receiver Buffer Registers (UARTx_RBR)” on page 21-5.

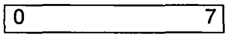


Figure 25-191. UART Receiver Buffer Registers (UARTx_RBR)

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.

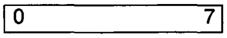


Figure 25-192. Scratchpad Registers (UARTx_SCR)

0:7		Data bits

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

MMIO 0xEF600300, 0xEF600400 Write-Only

See “Transmitter Holding Registers (UARTx_THR)” on page 21-5.

0	7

Figure 25-193. UART Transmitter Holding Registers (UARTx_THR)

0:7		Data bit

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

UIC0_CR

UIC Critical Register

DCR 0x0C3

See “UIC Critical Register (UIC0_CR)” on page 10-8.

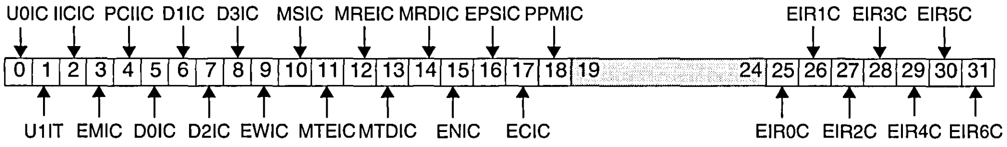


Figure 25-194. UIC Critical Register (UIC0_CR)

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	IICIC	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.

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.

DCR 0x0C2

See "UIC Enable Register (UIC0_ER)" on page 10-6.

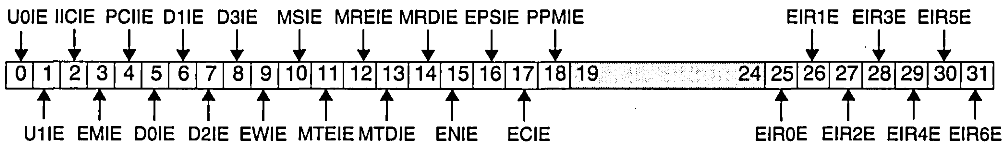


Figure 25-195. UIC Enable Register (UIC0_ER)

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 100 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.

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.

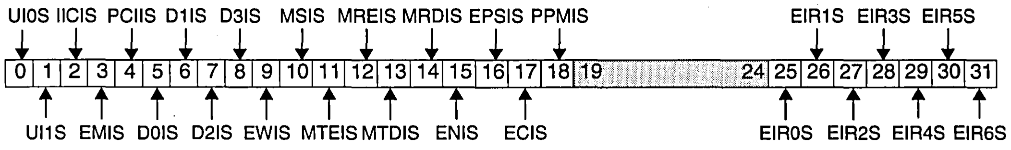


Figure 25-196. UIC Masked Status Register (UIC0_MSR)

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.

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.

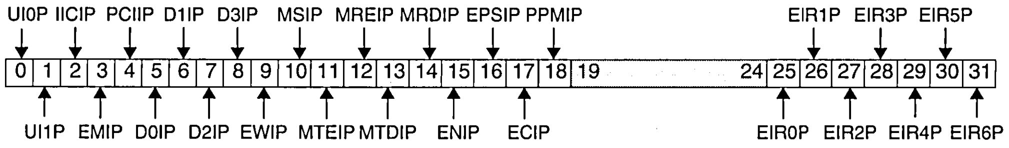


Figure 25-197. UIC Polarity Register (UIC0_PR)

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.

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.	

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.

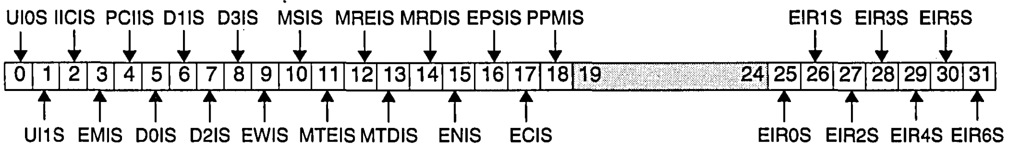


Figure 25-198. UIC Status Register (UIC0_SR)

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.

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.

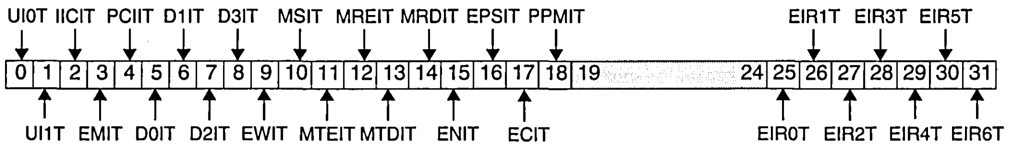


Figure 25-199. UIC Trigger Register (UIC0_TR)

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.

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	EIR0T	External IRQ 0 Trigger 0 An external IRQ 0 interrupt is level-sensitive. 1 An external IRQ 0 interrupt is edge-sensitive.	

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.

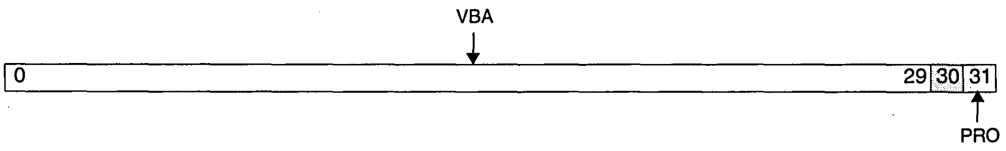


Figure 25-200. UIC Vector Configuration Register (UIC0_VCR)

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.

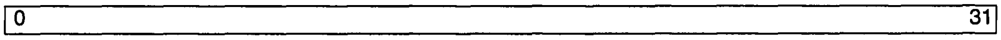


Figure 25-201. UIC Vector Register (UIC0_VR)

0:31		Interrupt Vector

USPRG0

User Special Purpose Register General 0

SPR 0x100 (User R/W)

See “Special Purpose Register General (SPRG0–SPRG7)” on page 3-11.

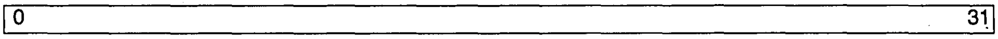


Figure 25-202. User SPR General 0 (USPRG0)

0:31		General data	Software value; no hardware usage.

SPR 0x001

See “Fixed Point Exception Register (XER)” on page 3-8.

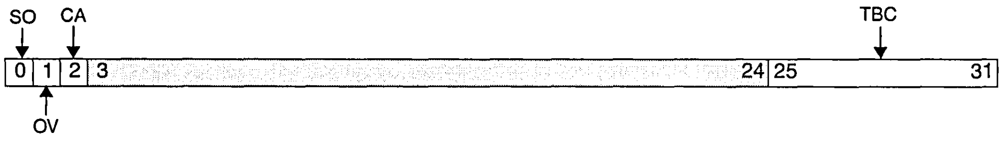


Figure 25-203. Fixed Point Exception Register (XER)

0	SO	Summary Overflow 0 No overflow has occurred. 1 Overflow has occurred.	Can be *set* by **mtspr** or by using “o” form instructions; can be *reset* by **mtspr** or by **mcrxr**.
1	OV	Overflow 0 No overflow has occurred. 0 Overflow has occurred.	Can be *set* by **mtspr** or by using “o” form instructions; can be *reset* by **mtspr**, by **mcrxr**, or “o” form instructions.
2	CA	Carry 0 Carry has not occurred. 1 Carry has occurred.	Can be *set* by **mtspr** or arithmetic instructions that update the CA field; can be *reset* 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.

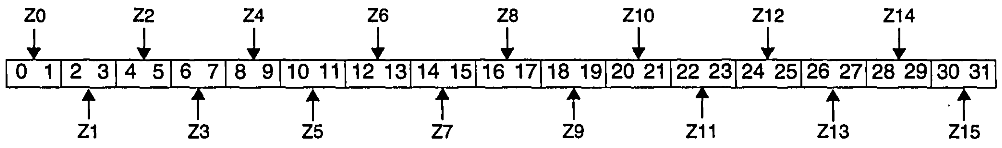


Figure 25-204. Zone Protection Register (ZPR)

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
$\overline{\text{BankSel0:3}}$	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
DQMCB	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
$\overline{\text{Halt}}$	System	26-9
HoldAck	External Master Peripheral	26-8
HoldPri	External Master Peripheral	26-8
HoldReq	External Master Peripheral	26-8
IICSCL	Internal Peripheral	26-9
IICSDA	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
PCIAD0:31	PCI	26-5
PCIC0:3[$\overline{\text{BE0:3}}$]	PCI	26-5
PCIClk	PCI	26-5
$\overline{\text{PCIDevSel}}$	PCI	26-5
PCIIDSel	PCI	26-5
$\overline{\text{PCIFrame}}$	PCI	26-5
$\overline{\text{PCIGnt0}}$[$\overline{\text{Req}}$]	PCI	26-6
$\overline{\text{PCIGnt1:5}}$	PCI	26-6
$\overline{\text{PCIINT}}$[$\overline{\text{PerWE}}$]	PCI	26-6
$\overline{\text{PCIIRDY}}$	PCI	26-5
PCIParity	PCI	26-5
PCIPErr	PCI	26-5
$\overline{\text{PCIReq0}}$[$\overline{\text{Gnt}}$]	PCI	26-6
$\overline{\text{PCIReq1:5}}$	PCI	26-6
$\overline{\text{PCIReset}}$	PCI	26-5
$\overline{\text{PCISErr}}$	PCI	26-5
$\overline{\text{PCIStop}}$	PCI	26-5

Table 26-1. Alphabetical Signal List

Signal Name	Interface	Page
$\overline{\text{PCITRDY}}$	PCI	26-5
PerAddr0:31	External Slave Peripheral	26-7
$\overline{\text{PerBLast}}$	External Slave Peripheral	26-8
PerClk	External Master Peripheral	26-8
$\overline{\text{PerCS0}}$	External Slave Peripheral	26-7
$\overline{\text{PerCS1:7}}$[GPIO10:16]	External Slave Peripheral	26-3
PerData0:31	External Slave Peripheral	26-7
PerErr	External Master Peripheral	26-8
$\overline{\text{PerOE}}$	External Slave Peripheral	26-7
PerPar0:3	External Slave Peripheral	26-7
PerReady	External Slave Peripheral	26-8
PerR/$\overline{\text{W}}$	External Slave Peripheral	26-8
$\overline{\text{PerWBE0:3}}$	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
$\overline{\text{RAS}}$	SDRAM	26-7
RcvrInh	System	26-10
SysClk	System	26-9
SysErr	System	26-9
$\overline{\text{SysReset}}$	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
$\overline{\text{UART0\_CTS}}$	Internal Peripheral	26-9
$\overline{\text{UART0\_DCD}}$	Internal Peripheral	26-9
$\overline{\text{UART0\_DSR}}$	Internal Peripheral	26-9
$\overline{\text{UART0\_DTR}}$	Internal Peripheral	26-9
$\overline{\text{UART0\_RI}}$	Internal Peripheral	26-9
$\overline{\text{UART0\_RTS}}$	Internal Peripheral	26-9
UART0_Rx	Internal Peripheral	26-8
UART0_Tx	Internal Peripheral	26-8
$\overline{\text{UART1\_DSR}}[\overline{\text{UART1\_CTS}}]$	Internal Peripheral	26-9
$\overline{\text{UART1 \_RTS}}[\overline{\text{UART1\_DTR}}]$	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
$\overline{\text{PCIINT}}$[$\overline{\text{PerWE}}$]	O	PCI interrupt or Peripheral write enable. Logical AND of the four $\overline{\text{PerWBE0:3}}$ write byte enables
$\overline{\text{PCIReq0}}$[$\overline{\text{Gnt}}$]	I	$\overline{\text{PCIReq0}}$ when internal arbiter is used or $\overline{\text{Gnt}}$ when external arbiter is used.
$\overline{\text{PCIReq1:5}}$	I	$\overline{\text{PCIReq1:5}}$ input when internal arbiter is used
$\overline{\text{PCIGnt0}}$[$\overline{\text{Req}}$]	O	$\overline{\text{PCIGnt0}}$ when internal arbiter is used or $\overline{\text{Req}}$ when external arbiter is used.
$\overline{\text{PCIGnt1:5}}$	O	$\overline{\text{PCIGnt1:5}}$ 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. EMCTxEnl 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.
PHYCol	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
$\overline{\text{RAS}}$	O	Row Address Strobe
$\overline{\text{CAS}}$	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
$\overline{\text{BankSel0:3}}$	O	Select up to four external SDRAM banks
$\overline{\text{WE}}$	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
$\overline{\text{PerWBE0:3}}$	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
$\overline{\text{PerCS0}}$	O	Peripheral chip select 0
$\overline{\text{PerCS1:7}}$[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.
$\overline{\text{PerOE}}$	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
$\overline{\text{PerBLast}}$	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
$\overline{\text{ExtReset}}$	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
$\overline{\text{ExtReq}}$	I	Used by an external master to indicate it is prepared to transfer data
$\overline{\text{ExtAck}}$	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
$\overline{\text{UART0\_DCD}}$	I	UART0 Data Carrier Detect
$\overline{\text{UART0\_DSR}}$	I	UART0 Data Set Ready
$\overline{\text{UART0\_CTS}}$	I	UART0 Clear To Send
$\overline{\text{UART0\_DTR}}$	O	UART0 Data Terminal Ready
$\overline{\text{UART0\_RTS}}$	O	UART0 Request To Send
$\overline{\text{UART0\_RI}}$	I	UART0 Ring Indicator
UART1_Rx	I	UART1 Serial In data.
UART1_Tx	O	UART1 Serial Out data
$\overline{\text{UART1\_DSR}}$[$\overline{\text{UART1\_CTS}}$]	I	UART1 Data Set Ready or UART1 Clear To Send. To access this function, software must toggle a DCR bit.
$\overline{\text{UART1\_RTS}}$[$\overline{\text{UART1\_DTR}}$]	O	UART1 Request To Send or UART1 Data Terminal Ready. To access this function, software must toggle a DCR bit.
IICSCL	I/O	IIC Serial Clock
IICSDA	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
$\overline{\text{TRST}}$	I	JTAG reset
System Interface		
SysClk	I	Main system clock input
$\overline{\text{SysReset}}$	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.
$\overline{\text{Halt}}$	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.
$\overline{\mathrm{RcvrInh}}$	I	Receiver Inhibit. Reserved for manufacturing test.
$\overline{\mathrm{DrvrInh1}}$ $\overline{\mathrm{DrvrInh2}}$	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]	
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 (RA|0). Place result in RT.		24-9
addic	RT, RA, IM	Add EXTS(IM) to (RA|0). Place result in RT. Place carry-out in XER[CA].		24-6
addic.	RT, RA, IM	Add EXTS(IM) to (RA|0). Place result in RT. Place carry-out in XER[CA].	CR[CR0]	24-11
addis	RT, RA, IM	Add (IM || $^{16}0$) to (RA|0). 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 ¬(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 ← $(\text{target} - \text{CIA})_{6:29}$
		NIA ← CIA + EXTS(LI || $^{2}0$)
				24-19

ba		Branch unconditional absolute.
		LI ← $\text{target}_{6:29}$
		NIA ← EXTS(LI || $^{2}0$)		

bl		Branch unconditional relative.
		LI ← $(\text{target} - \text{CIA})_{6:29}$
		NIA ← CIA + EXTS(LI || $^{2}0$)
			(LR) ← CIA + 4.	

bla		Branch unconditional absolute.
		LI ← $\text{target}_{6:29}$
		NIA ← EXTS(LI || $^{2}0$)
			(LR) ← CIA + 4.	

bc	BO, BI, target	Branch conditional relative.
		BD ← $(\text{target} - \text{CIA})_{16:29}$
		NIA ← CIA + EXTS(BD || $^{2}0$)
			CTR if $BO_2 = 0$.	24-20

bca		Branch conditional absolute.
		BD ← $\text{target}_{16:29}$
		NIA ← EXTS(BD || $^{2}0$)
			CTR if $BO_2 = 0$.	

bcl		Branch conditional relative.
		BD ← $(\text{target} - \text{CIA})_{16:29}$
		NIA ← CIA + EXTS(BD || $^{2}0$)
			CTR if $BO_2 = 0$.
			(LR) ← CIA + 4.	

bcla		Branch conditional absolute.
		BD ← $\text{target}_{16:29}$
		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 ← $\text{CTR}_{0:29}$ || $^{2}0$.	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 ← $\text{LR}_{0:29}$ || $^{2}0$.	CTR if $BO_2 = 0$.	24-30
bclrl			CTR if $BO_2 = 0$. (LR) ← CIA + 4.	
bctr		Branch unconditionally to address in CTR. *Extended mnemonic for* **bcctr 20,0**		24-26
bctrl		*Extended mnemonic for* **bcctrl 20,0**	(LR) ← CIA + 4.	
bdnz	target	Decrement CTR. Branch if CTR ≠ 0. *Extended mnemonic for* **bc 16,0,target**		24-20
bdnza		*Extended mnemonic for* **bca 16,0,target**		
bdnzl		*Extended mnemonic for* **bcl 16,0,target**	(LR) ← CIA + 4.	
bdnzla		*Extended mnemonic for* **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 ≠ 0 to address in LR. *Extended mnemonic for* **bclr 16,0**		24-30
bdnzlrl		*Extended mnemonic for* **bclrl 16,0**	(LR) ← CIA + 4.	
bdnzf	cr_bit, target	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 0. *Extended mnemonic for* **bc 0,cr_bit,target**		24-20
bdnzfa		*Extended mnemonic for* **bca 0,cr_bit,target**		
bdnzfl		*Extended mnemonic for* **bcl 0,cr_bit,target**	(LR) ← CIA + 4.	
bdnzfla		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 0,cr_bit**		24-30
bdnzflrl		*Extended mnemonic for* **bclrl 0,cr_bit**	(LR) ← CIA + 4.	
bdnzt	cr_bit, target	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 1. *Extended mnemonic for* **bc 8,cr_bit,target**		24-20
bdnzta		*Extended mnemonic for* **bca 8,cr_bit,target**		
bdnztl		*Extended mnemonic for* **bcl 8,cr_bit,target**	(LR) ← CIA + 4.	
bdnztla		*Extended mnemonic for* **bcla 8,cr_bit,target**	(LR) ← CIA + 4.	
bdnztlr	cr_bit	Decrement CTR. Branch if CTR ≠ 0 AND CR_{cr_bit} = 1 to address in LR. *Extended mnemonic for* **bclr 8,cr_bit**		24-30
bdnztlrl		*Extended mnemonic for* **bclrl 8,cr_bit**	(LR) ← CIA + 4.	
bdz	target	Decrement CTR. Branch if CTR = 0. *Extended mnemonic for* **bc 18,0,target**		24-20
bdza		*Extended mnemonic for* **bca 18,0,target**		
bdzl		*Extended mnemonic for* **bcl 18,0,target**	(LR) ← CIA + 4.	
bdzla		*Extended mnemonic for* **bcla 18,0,target**	(LR) ← 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. *Extended mnemonic for* **bclr 18,0**		24-30
bdzlrl		*Extended mnemonic for* **bclrl 18,0**	(LR) ← CIA + 4.	
bdzf	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 0. *Extended mnemonic for* **bc 2,cr_bit,target**		24-20
bdzfa		*Extended mnemonic for* **bca 2,cr_bit,target**		
bdzfl		*Extended mnemonic for* **bcl 2,cr_bit,target**	(LR) ← CIA + 4.	
bdzfla		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 2,cr_bit**		24-30
bdzflrl		*Extended mnemonic for* **bclrl 2,cr_bit**	(LR) ← CIA + 4.	
bdzt	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 1. *Extended mnemonic for* **bc 10,cr_bit,target**		24-20
bdzta		*Extended mnemonic for* **bca 10,cr_bit,target**		
bdztl		*Extended mnemonic for* **bcl 10,cr_bit,target**	(LR) ← CIA + 4.	
bdztla		*Extended mnemonic for* **bcla 10,cr_bit,target**	(LR) ← CIA + 4.	
bdztlr	cr_bit	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 1to address in LR. *Extended mnemonic for* **bclr 10,cr_bit**		24-30
bdztlrl		*Extended mnemonic for* **bclrl 10,cr_bit**	(LR) ← CIA + 4.	
beq	[cr_field], target	Branch if equal. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+2,target**		24-20
beqa		*Extended mnemonic for* **bca 12,4∗cr_field+2,target**		
beql		*Extended mnemonic for* **bcl 12,4∗cr_field+2,target**	(LR) ← CIA + 4.	
beqla		*Extended mnemonic for* **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 CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4*cr_field+2**		24-26
beqctrl		*Extended mnemonic for* **bcctrl 12,4*cr_field+2**	(LR) ← CIA + 4.	
beqlr	[cr_field]	Branch if equal to address in LR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bclr 12,4*cr_field+2**		24-30
beqlrl		*Extended mnemonic for* **bclrl 12,4*cr_field+2**	(LR) ← CIA + 4.	
bf	cr_bit, target	Branch if $CR_{cr\_bit} = 0$. *Extended mnemonic for* **bc 4,cr_bit,target**		24-20
bfa		*Extended mnemonic for* **bca 4,cr_bit,target**		
bfl		*Extended mnemonic for* **bcl 4,cr_bit,target**	(LR) ← CIA + 4.	
bfla		*Extended mnemonic for* **bcla 4,cr_bit,target**	(LR) ← CIA + 4.	
bfctr	cr_bit	Branch if $CR_{cr\_bit} = 0$ to address in CTR. *Extended mnemonic for* **bcctr 4,cr_bit**		24-26
bfctrl		*Extended mnemonic for* **bcctrl 4,cr_bit**	(LR) ← CIA + 4.	
bflr	cr_bit	Branch if $CR_{cr\_bit} = 0$ to address in LR. *Extended mnemonic for* **bclr 4,cr_bit**		24-30
bflrl		*Extended mnemonic for* **bclrl 4,cr_bit**	(LR) ← CIA + 4.	
bge	[cr_field], target	Branch if greater than or equal. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4*cr_field+0,target**		24-20
bgea		*Extended mnemonic for* **bca 4,4*cr_field+0,target**		
bgel		*Extended mnemonic for* **bcl 4,4*cr_field+0,target**	(LR) ← CIA + 4.	
bgela		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 4,4*cr_field+0**		24-26
bgectrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 4,4∗cr_field+0**		24-30
bgelrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+0**	(LR) ← CIA + 4.	
bgt	[cr_field], target	Branch if greater than. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+1,target**		24-20
bgta		*Extended mnemonic for* **bca 12,4∗cr_field+1,target**		
bgtl		*Extended mnemonic for* **bcl 12,4∗cr_field+1,target**	(LR) ← CIA + 4.	
bgtla		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 12,4∗cr_field+1**		24-26
bgtctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 12,4∗cr_field+1**		24-30
bgtlrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bc 4,4∗cr_field+1,target**		24-20
blea		*Extended mnemonic for* **bca 4,4∗cr_field+1,target**		
blel		*Extended mnemonic for* **bcl 4,4∗cr_field+1,target**	(LR) ← CIA + 4.	
blela		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 4,4∗cr_field+1**		24-26
blectrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 4,4∗cr_field+1**		24-30
blelrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+1**	(LR) ← CIA + 4.	
blr		Branch unconditionally to address in LR. *Extended mnemonic for* **bclr 20,0**		24-30
blrl		*Extended mnemonic for* **bclrl 20,0**	(LR) ← CIA + 4.	
blt	[cr_field], target	Branch if less than. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+0,target**		24-20
blta		*Extended mnemonic for* **bca 12,4∗cr_field+0,target**		
bltl		*Extended mnemonic for* **bcl 12,4∗cr_field+0,target**	(LR) ← CIA + 4.	
bltla		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 12,4∗cr_field+0**		24-26
bltctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 12,4∗cr_field+0**		24-30
bltlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+0**	(LR) ← CIA + 4.	
bne	[cr_field], target	Branch if not equal. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+2,target**		24-20
bnea		*Extended mnemonic for* **bca 4,4∗cr_field+2,target**		
bnel		*Extended mnemonic for* **bcl 4,4∗cr_field+2,target**	(LR) ← CIA + 4.	
bnela		*Extended mnemonic for* **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
bnectr	[cr_field]	Branch if not equal to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+2**		24-26
bnectrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 4,4∗cr_field+2**		24-30
bnelrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bc 4,4∗cr_field+1,target**		24-20
bnga		*Extended mnemonic for* **bca 4,4∗cr_field+1,target**		
bngl		*Extended mnemonic for* **bcl 4,4∗cr_field+1,target**	(LR) ← CIA + 4.	
bngla		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 4,4∗cr_field+1**		24-26
bngctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 4,4∗cr_field+1**		24-30
bnglrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bc 4,4∗cr_field+0,target**		24-20
bnla		*Extended mnemonic for* **bca 4,4∗cr_field+0,target**		
bnll		*Extended mnemonic for* **bcl 4,4∗cr_field+0,target**	(LR) ← CIA + 4.	
bnlla		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 4,4∗cr_field+0**		24-26
bnlctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 4,4∗cr_field+0**		24-30
bnllrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bc 4,4∗cr_field+3,target**		24-20
bnsa		*Extended mnemonic for* **bca 4,4∗cr_field+3,target**		
bnsl		*Extended mnemonic for* **bcl 4,4∗cr_field+3,target**	(LR) ← CIA + 4.	
bnsla		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 4,4∗cr_field+3**		24-26
bnsctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 4,4∗cr_field+3**		24-30
bnslrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+3**	(LR) ← CIA + 4.	
bnu	[cr_field], target	Branch if not unordered. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+3,target**		24-20
bnua		*Extended mnemonic for* **bca 4,4∗cr_field+3,target**		
bnul		*Extended mnemonic for* **bcl 4,4∗cr_field+3,target**	(LR) ← CIA + 4.	
bnula		*Extended mnemonic* for **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. *Extended mnemonic for* **bcctr 4,4∗cr_field+3**		24-26
bnuctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 4,4∗cr_field+3**		24-30
bnulrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+3**	(LR) ← CIA + 4.	
bso	[cr_field], target	Branch if summary overflow. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+3,target**		24-20
bsoa		*Extended mnemonic for* **bca 12,4∗cr_field+3,target**		
bsol		*Extended mnemonic for* **bcl 12,4∗cr_field+3,target**	(LR) ← CIA + 4.	
bsola		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 12,4∗cr_field+3**		24-26
bsoctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 12,4∗cr_field+3**		24-30
bsolrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+3**	(LR) ← CIA + 4.	
bt	cr_bit, target	Branch if $CR_{cr\_bit} = 1$. *Extended mnemonic for* **bc 12,cr_bit,target**		24-20
bta		*Extended mnemonic for* **bca 12,cr_bit,target**		
btl		*Extended mnemonic for* **bcl 12,cr_bit,target**	(LR) ← CIA + 4.	
btla		*Extended mnemonic for* **bcla 12,cr_bit,target**	(LR) ← CIA + 4.	
btctr	cr_bit	Branch if $CR_{cr\_bit} = 1$ to address in CTR. *Extended mnemonic for* **bcctr 12,cr_bit**		24-26
btctrl		*Extended mnemonic for* **bcctrl 12,cr_bit**	(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. *Extended mnemonic for* **bclr 12,cr_bit**		24-30
btlrl		*Extended mnemonic for* **bclrl 12,cr_bit**	(LR) ← CIA + 4.	
bun	[cr_field], target	Branch if unordered. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+3,target**		24-20.
buna		*Extended mnemonic for* **bca 12,4∗cr_field+3,target**		
bunl		*Extended mnemonic for* **bcl 12,4∗cr_field+3,target**	(LR) ← CIA + 4.	
bunla		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 12,4∗cr_field+3**		24-26
bunctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 12,4∗cr_field+3**		24-30
bunlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+3**	(LR) ← CIA + 4.	
clrlwi	RA, RS, n	Clear left immediate. $(n < 32)$ $(RA)_{0:n-1} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,0,n,31**		24-147
clrlwi.		*Extended mnemonic for* **rlwinm. RA,RS,0,n,31**	CR[CR0]	
clrlslwi	RA, RS, b, n	Clear left and shift left immediate. $(n \le b < 32)$ $(RA)_{b-n:31-n} \leftarrow (RS)_{b:31}$ $(RA)_{32-n:31} \leftarrow {}^{n}0$ $(RA)_{0:b-n-1} \leftarrow {}^{b-n}0$ *Extended mnemonic for* **rlwinm RA,RS,n,b−n,31−n**		24-147
clrlslwi.		*Extended mnemonic for* **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. ($n < 32$) $(RA)_{32-n:31} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,0,0,31−n**		24-147
clrrwi.		*Extended mnemonic for* **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
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 ($^{16}0 \Vert$ IM), unsigned. Results in CR[CRn], where n = BF.		24-37
cmplw	[BF,] RA, RB	Compare Logical Word. Use CR0 if BF is omitted. *Extended mnemonic for* **cmpl BF,0,RA,RB**		24-36
cmplwi	[BF,] RA, IM	Compare Logical Word Immediate. Use CR0 if BF is omitted. *Extended mnemonic for* **cmpli BF,0,RA,IM**		24-37
cmpw	[BF,] RA, RB	Compare Word. Use CR0 if BF is omitted. *Extended mnemonic for* **cmp BF,0,RA,RB**		24-34
cmpwi	[BF,] RA, IM	Compare Word Immediate. Use CR0 if BF is omitted. *Extended mnemonic for* **cmpi 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. *Extended mnemonic for* **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. *Extended mnemonic for* **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. *Extended mnemonic for* **crnor bx,by,by**		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. *Extended mnemonic for* **creqv bx,bx,bx**		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 (RA|0) + (RB).		24-47
dcbf	RA, RB	Flush (store, then invalidate) the data cache block which contains the effective address (RA|0) + (RB).		24-49
dcbi	RA, RB	Invalidate the data cache block which contains the effective address (RA|0) + (RB).		24-50
dcbst	RA, RB	Store the data cache block which contains the effective address (RA|0) + (RB).		24-51
dcbt	RA, RB	Load the data cache block which contains the effective address (RA|0) + (RB).		24-52
dcbtst	RA,RB	Load the data cache block which contains the effective address (RA|0) + (RB).		24-53
dcbz	RA, RB	Zero the data cache block which contains the effective address (RA|0) + (RB).		24-54
dccci	RA, RB	Invalidate the data cache congruence class associated with the effective address (RA|0) + (RB).		24-56
dcread	RT, RA, RB	Read either tag or data information from the data cache congruence class associated with the effective address (RA|0) + (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) ← ¬((RS) ⊕ (RB))		24-62
eqv.			CR[CR0]	
extlwi	RA, RS, n, b	Extract and left justify immediate. (n > 0) $(RA)_{0:n-1} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{n:31} \leftarrow {}^{32-n}0$ *Extended mnemonic for* **rlwinm RA,RS,b,0,n−1**		24-147
extlwi.		*Extended mnemonic for* **rlwinm. RA,RS,b,0,n−1**	CR[CR0]	
extrwi	RA, RS, n, b	Extract and right justify immediate. (n > 0) $(RA)_{32-n:31} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{0:31-n} \leftarrow {}^{32-n}0$ *Extended mnemonic for* **rlwinm RA,RS,b+n,32−n,31**		24-147
extrwi.		*Extended mnemonic for* **rlwinm. RA,RS,b+n,32−n,31**	CR[CR0]	
extsb	RA, RS	Extend the sign of byte $(RS)_{24:31}$. Place the result in RA.		24-63
extsb.			CR[CR0]	
extsh	RA, RS	Extend the sign of halfword $(RS)_{16:31}$. Place the result in RA.		24-64
extsh.			CR[CR0]	
icbi	RA, RB	Invalidate the instruction cache block which contains the effective address (RA|0) + (RB).		24-65
icbt	RA, RB	Load the instruction cache block which contains the effective address (RA|0) + (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 (RA|0) + (RB). Place the results in ICDBDR.		24-68
inslwi	RA, RS, n, b	Insert from left immediate. (n > 0) $(RA)_{b:b+n-1} \leftarrow (RS)_{0:n-1}$ *Extended mnemonic for* **rlwimi RA,RS,32−b,b,b+n−1**		24-146
inslwi.		*Extended mnemonic for* **rlwimi. RA,RS,32−b,b,b+n−1**	CR[CR0]	
insrwi	RA, RS, n, b	Insert from right immediate. (n > 0) $(RA)_{b:b+n-1} \leftarrow (RS)_{32-n:31}$ *Extended mnemonic for* **rlwimi RA,RS,32−b−n,b,b+n−1**		24-146
insrwi.		*Extended mnemonic for* **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) *Extended mnemonic for* **addi RT,RA,D**		24-9
lbz	RT, D(RA)	Load byte from EA = (RA|0) + EXTS(D) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1).		24-71
lbzu	RT, D(RA)	Load byte from EA = (RA|0) + 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 = (RA|0) + (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 = (RA|0) + (RB) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1).		24-74
lha	RT, D(RA)	Load halfword from EA = (RA|0) + EXTS(D) and sign extend, (RT) ← EXTS(MS(EA,2)).		24-75
lhau	RT, D(RA)	Load halfword from EA = (RA|0) + 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 = (RA|0) + (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 = (RA|0) + (RB) and sign extend, (RT) ← EXTS(MS(EA,2)).		24-78
lhbrx	RT, RA, RB	Load halfword from EA = (RA|0) + (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 = (RA|0) + 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 = (RA|0) + EXTS(D) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2). Update the base address, (RA) ← EA.		24-81
lhzux	RT, RA, RB	Load halfword from EA = (RA|0) + (RB) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2). Update the base address, (RA) ← EA.		24-82
lhzx	RT, RA, RB	Load halfword from EA = (RA|0) + (RB) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2).		24-83
li	RT, IM	Load immediate. (RT) ← EXTS(IM) *Extended mnemonic for* **addi RT,0,value**		24-9
lis	RT, IM	Load immediate shifted. (RT) ← (IM || $^{16}0$) *Extended mnemonic for* **addis RT,0,value**		24-12
lmw	RT, D(RA)	Load multiple words starting from EA = (RA|0) + EXTS(D). Place into consecutive registers RT through GPR(31). RA is not altered unless RA = GPR(31).		24-84
lswi	RT, RA, NB	Load consecutive bytes from EA=(RA|0). Number of bytes n=32 if NB=0, else n=NB. Stack bytes into words in CEIL(n/4) consecutive registers starting with RT, to R_{FINAL} ← ((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=(RA|0)+(RB). Number of bytes n=XER[TBC]. Stack bytes into words in CEIL(n/4) consecutive registers starting with RT, to R_{FINAL} ← ((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 = (RA|0) + (RB) and place in RT, (RT) ← MS(EA,4). Set the Reservation bit.		24-89
lwbrx	RT, RA, RB	Load word from EA = (RA|0) + (RB) then reverse byte order, (RT) ← 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 = (RA|0) + EXTS(D) and place in RT, (RT) ← MS(EA,4).		24-91
lwzu	RT, D(RA)	Load word from EA = (RA|0) + 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 = (RA|0) + (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 = (RA|0) + (RB) and place in RT, (RT) ← MS(EA,4).		24-94
macchw	RT, RA, RB	prod$_{0:31}$ ← (RA)$_{16:31}$ x (RB)$_{0:15}$ signed temp$_{0:32}$ ← 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$_{0:31}$ ← (RA)$_{16:31}$ x (RB)$_{0:15}$ signed temp$_{0:32}$ ← prod$_{0:31}$ + (RT) if ((prod$_0$ = RT$_0$) ∧ (RT$_0$ ≠ temp$_1$)) then (RT) ← (RT$_0$ || ^{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$_{0:31}$ ← (RA)$_{16:31}$ x (RB)$_{0:15}$ unsigned temp$_{0:32}$ ← prod$_{0:31}$ + (RT) (RT) ← (temp$_{1:32}$ ∨ ^{32}temp$_0$)		24-97
macchwsu.			CR[CR0]	
macchwsuo			XER[SO, OV]	
macchwsuo.			CR[CR0] XER[SO, OV]	
macchwu	RT, RA, RB	prod$_{0:31}$ ← (RA)$_{16:31}$ x (RB)$_{0:15}$ unsigned temp$_{0:32}$ ← 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$_{0:15}$ ← (RA)$_{16:31}$ x (RB)$_{0:15}$ signed temp$_{0:32}$ ← 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$_{0:31}$ ← (RA)$_{0:15}$ x (RB)$_{0:15}$ signed temp$_{0:32}$ ← prod$_{0:31}$ + (RT) if ((prod$_0$ = RT$_0$) ∧ (RT$_0$ ≠ temp$_1$)) then (RT) ← (RT$_0$ || ^{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]	
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]	
mcrf	BF, BFA	Move CR field, (CR[CRn]) ← (CR[CRm]) where m ← BFA and n ← BF.		24-107
mcrxr	BF	Move XER[0:3] into field CRn, where n←BF. CR[CRn] ← (XER[SO, OV, CA]). (XER[SO, OV, CA]) ← $^{3}0$.		24-108
mfcr	RT	Move from CR to RT, (RT) ← (CR).		24-109
mfdcr	RT, DCRN	Move from DCR to RT, (RT) ← (DCR(DCRN)).		24-110
mfmsr	RT	Move from MSR to RT, (RT) ← (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** **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. *Extended mnemonic for* **mfspr RT,SPRN** 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) *Extended mnemonic for* **mftb RT,TBL**		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) $\leftarrow$ (TBU) *Extended mnemonic for* **mftb RT,TBU**		24-114
mr	RT, RS	Move register. (RT) $\leftarrow$ (RS) *Extended mnemonic for* **or RT,RS,RS**		24-140
mr.		*Extended mnemonic for* **or. RT,RS,RS**	CR[CR0]	
mtcr	RS	Move to Condition Register. *Extended mnemonic for* **mtcrf 0xFF,RS**		24-116
mtcrf	FXM, RS	Move some or all of the contents of RS into CR as specified by FXM field, mask $\leftarrow$ ${}^4(FXM_0) \parallel {}^4(FXM_1) \parallel \ldots \parallel$ ${}^4(FXM_6) \parallel {}^4(FXM_7)$. (CR)$\leftarrow$((RS) $\wedge$ mask) $\vee$ (CR) $\wedge$ $\neg$mask).		24-116
mtdcr	DCRN, RS	Move to DCR from RS, (DCR(DCRN)) $\leftarrow$ (RS).		24-117
mtmsr	RS	Move to MSR from RS, (MSR) $\leftarrow$ (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** **mticcr** **mticdbdr** **mtlr** **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. *Extended mnemonic for* **mtspr SPRN,RS** 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	$(RT)_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ 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]	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
mulhhw	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{0:15}$ x $(RB)_{0:15}$ signed		24-123
mulhhw.			CR[CR0]	
mulhhwu	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{0:15}$ x $(RB)_{0:15}$ unsigned		24-124
mulhhwu.			CR[CR0]	
mullhw	RT, RA, RB	$(RT)_{0:31} \leftarrow (RA)_{16:31}$ x $(RB)_{16:31}$ signed		24-127
mullhw.			CR[CR0]	
mullhwu	RT, RA, RB	$(RT)_{16:31} \leftarrow (RA)_{16:31}$ x $(RB)_{16:31}$ unsigned		24-128
mullhwu.			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). $(RT) \leftarrow prod_{0:31}$.		24-125
mulhw.			CR[CR0]	
mulhwu	RT, RA, RB	Multiply (RA) and (RB), unsigned. Place high-order result in RT. $prod_{0:63} \leftarrow (RA) \times (RB)$ (unsigned). $(RT) \leftarrow prod_{0:31}$.		24-126
mulhwu.			CR[CR0]	
mulli	RT, RA, IM	Multiply (RA) and IM, signed. Place low-order result in RT. $prod_{0:47} \leftarrow (RA) \times IM$ (signed) $(RT) \leftarrow prod_{16:47}$		24-129
mullw	RT, RA, RB	Multiply (RA) and (RB), signed. Place low-order result in RT. $prod_{0:63} \leftarrow (RA) \times (RB)$ (signed). $(RT) \leftarrow prod_{32:63}$.		24-130
mullw.			CR[CR0]	
mullwo			XER[SO, OV]	
mullwo.			CR[CR0] XER[SO, OV]	
nand	RA, RS, RB	NAND (RS) with (RB). Place result in RA.		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	$nprod_{0:31} \leftarrow -((RA)_{16:31}$ x $(RB)_{0:15})$ 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	$nprod_{0:31} \leftarrow -((RA)_{16:31}$ x $(RB)_{0:15})$ 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	$\text{nprod}_{0:31} \leftarrow -((\text{RA})_{0:15} \times (\text{RB})_{0:15})$ signed $\text{temp}_{0:32} \leftarrow \text{nprod}_{0:31} + (\text{RT})$ $(\text{RT}) \leftarrow \text{temp}_{1:32}$		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. *Extended mnemonic for* **ori 0,0,0**		24-142
nor	RA, RS, RB	NOR (RS) with (RB). Place result in RA.		24-139
nor.			CR[CR0]	
not	RA, RS	Complement register. $(\text{RA}) \leftarrow \neg(\text{RS})$ *Extended mnemonic for* **nor RA,RS,RS**		24-139
not.		*Extended mnemonic for* **nor. RA,RS,RS**	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 ¬(RB). Place result in RA.		24-141
orc.			CR[CR0]	
ori	RA, RS, IM	OR (RS) with $({}^{16}0 \parallel \text{IM})$. Place result in RA.		24-142
oris	RA, RS, IM	OR (RS) with $(\text{IM} \parallel {}^{16}0)$. Place result in RA.		24-143
rfci		Return from critical interrupt $(\text{PC}) \leftarrow (\text{SRR2})$. $(\text{MSR}) \leftarrow (\text{SRR3})$.		24-144
rfi		Return from interrupt. $(\text{PC}) \leftarrow (\text{SRR0})$. $(\text{MSR}) \leftarrow (\text{SRR1})$.		24-145
rlwimi	RA, RS, SH, MB, ME	Rotate left word immediate, then insert according to mask. $r \leftarrow \text{ROTL}((\text{RS}), \text{SH})$ $m \leftarrow \text{MASK}(\text{MB}, \text{ME})$ $(\text{RA}) \leftarrow (r \wedge m) \vee ((\text{RA}) \wedge \neg m)$		24-146
rlwimi.			CR[CR0]	
rlwinm	RA, RS, SH, MB, ME	Rotate left word immediate, then AND with mask. $r \leftarrow \text{ROTL}((\text{RS}), \text{SH})$ $m \leftarrow \text{MASK}(\text{MB}, \text{ME})$ $(\text{RA}) \leftarrow (r \wedge m)$		24-147
rlwinm.			CR[CR0]	
rlwnm	RA, RS, RB, MB, ME	Rotate left word, then AND with mask. $r \leftarrow \text{ROTL}((\text{RS}), (\text{RB})_{27:31})$ $m \leftarrow \text{MASK}(\text{MB}, \text{ME})$ $(\text{RA}) \leftarrow (r \wedge m)$		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) ← ROTL((RS), $(RB)_{27:31}$) *Extended mnemonic for* **rlwnm RA,RS,RB,0,31**		24-150
rotlw.		*Extended mnemonic for* **rlwnm. RA,RS,RB,0,31**	CR[CR0]	
rotlwi	RA, RS, n	Rotate left immediate. (RA) ← ROTL((RS), n) *Extended mnemonic for* **rlwinm RA,RS,n,0,31**		24-147
rotlwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,0,31**	CR[CR0]	
rotrwi	RA, RS, n	Rotate right immediate. (RA) ← ROTL((RS), 32–n) *Extended mnemonic for* **rlwinm RA,RS,32–n,0,31**		24-147
rotrwi.		*Extended mnemonic for* **rlwinm. RA,RS,32–n,0,31**	CR[CR0]	
sc		System call exception is generated. (SRR1) ← (MSR) (SRR0) ← (PC) PC ← $EVPR_{0:15}$ || x'0C00' (MSR[WE, PR, EE, PE, DR, IR]) ← 0		24-151

slw	RA, RS, RB	Shift left (RS) by $(RB)_{27:31}$.
		n ← $(RB)_{27:31}$.
		r ← ROTL((RS), n).
		if $(RB)_{26}$ = 0 then m ← MASK(0, 31 – n)
		else m ← $^{32}0$.
		(RA) ← r $\wedge$ m.
				24-152

slw.			CR[CR0]	
slwi	RA, RS, n	Shift left immediate. (n < 32) $(RA)_{0:31-n}$ ← $(RS)_{n:31}$ $(RA)_{32-n:31}$ ← $^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,n,0,31–n**		24-147
slwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,0,31–n**	CR[CR0]	

sraw	RA, RS, RB	Shift right algebraic (RS) by $(RB)_{27:31}$.
		n ← $(RB)_{27:31}$.
		r ← ROTL((RS), 32 – n).
		if $(RB)_{26}$ = 0 then m ← MASK(n, 31)
		else m ← $^{32}0$.
		s ← $(RS)_0$.
		(RA) ← (r $\wedge$ m) $\vee$ (^{32}s $\wedge$ $\neg$m).
		XER[CA] ← s $\wedge$ ((r $\wedge$ $\neg$m) $\neq$ 0).
				24-153

sraw.			CR[CR0]	

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
srawi	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
srawi.			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) $(RA)_{n:31} \leftarrow (RS)_{0:31-n}$ $(RA)_{0:n-1} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,32–n,n,31**		24-147
srwi.		*Extended mnemonic for* **rlwinm. RA,RS,32–n,n,31**	CR[CR0]	
stb	RS, D(RA)	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + EXTS(D).		24-156
stbu	RS, D(RA)	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + EXTS(D). Update the base address, $(RA) \leftarrow$ EA.		24-157
stbux	RS, RA, RB	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + (RB). Update the base address, $(RA) \leftarrow$ EA.		24-158
stbx	RS, RA, RB	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + (RB).		24-159
sth	RS, D(RA)	Store halfword $(RS)_{16:31}$ in memory at EA = (RA|0) + EXTS(D).		24-160
sthbrx	RS, RA, RB	Store halfword $(RS)_{16:31}$ byte-reversed in memory at EA = (RA|0) + (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 = (RA|0) + EXTS(D). Update the base address, $(RA) \leftarrow$ EA.		24-162
sthux	RS, RA, RB	Store halfword $(RS)_{16:31}$ in memory at EA = (RA|0) + (RB). Update the base address, $(RA) \leftarrow$ EA.		24-163
sthx	RS, RA, RB	Store halfword $(RS)_{16:31}$ in memory at EA = (RA|0) + (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 = (RA|0) + EXTS(D).		24-165
stswi	RS, RA, NB	Store consecutive bytes in memory starting at EA=(RA|0). Number of bytes n=32 if NB=0, else n=NB. Bytes are unstacked from CEIL(n/4) consecutive registers starting with RS. GPR(0) is consecutive to GPR(31).		24-166
stswx	RS, RA, RB	Store consecutive bytes in memory starting at EA=(RA|0)+(RB). Number of bytes n=XER[TBC]. Bytes are unstacked from CEIL(n/4) consecutive registers starting with RS. GPR(0) is consecutive to GPR(31).		24-167
stw	RS, D(RA)	Store word (RS) in memory at EA = (RA|0) + EXTS(D).		24-169
stwbrx	RS, RA, RB	Store word (RS) byte-reversed in memory at EA = (RA|0) + (RB). MS(EA, 4) $\leftarrow$ $(RS)_{24:31}$ || $(RS)_{16:23}$ || $(RS)_{8:15}$ || $(RS)_{0:7}$		24-170
stwcx.	RS, RA, RB	Store word (RS) in memory at EA = (RA|0) + (RB) only if reservation bit is set. if RESERVE = 1 then MS(EA, 4) $\leftarrow$ (RS) RESERVE $\leftarrow$ 0 (CR[CR0]) $\leftarrow$ $^{2}0$ || 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 = (RA|0) + EXTS(D). Update the base address, (RA) $\leftarrow$ EA.		24-173
stwux	RS, RA, RB	Store word (RS) in memory at EA = (RA|0) + (RB). Update the base address, (RA) $\leftarrow$ EA.		24-174
stwx	RS, RA, RB	Store word (RS) in memory at EA = (RA|0) + (RB).		24-175
sub	RT, RA, RB	Subtract (RB) from (RA). (RT) $\leftarrow$ $\neg$(RB) + (RA) + 1. *Extended mnemonic for* **subf RT,RB,RA**		24-176
sub.		*Extended mnemonic for* **subf. RT,RB,RA**	CR[CR0]	
subo		*Extended mnemonic for* **subfo RT,RB,RA**	XER[SO, OV]	
subo.		*Extended mnemonic for* **subfo. RT,RB,RA**	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) ← ¬(RB) + (RA) + 1. Place carry-out in XER[CA]. *Extended mnemonic for* **subfc RT,RB,RA**		24-177
subc.		*Extended mnemonic for* **subfc. RT,RB,RA**	CR[CR0]	
subco		*Extended mnemonic for* **subfco RT,RB,RA**	XER[SO, OV]	
subco.		*Extended mnemonic for* **subfco. RT,RB,RA**	CR[CR0] XER[SO, OV]	
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]	
subi	RT, RA, IM	Subtract EXTS(IM) from (RA|0). Place result in RT. *Extended mnemonic for* **addi RT,RA,–IM**		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]. *Extended mnemonic for* **addic RT,RA,–IM**		24-10
subic.	RT, RA, IM	Subtract EXTS(IM) from (RA). Place result in RT. Place carry-out in XER[CA]. *Extended mnemonic for* **addic. RT,RA,–IM**	CR[CR0]	24-11
subis	RT, RA, IM	Subtract (IM $\|\|$ $^{16}0$) from (RA|0). Place result in RT. *Extended mnemonic for* **addis RT,RA,–IM**		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)]$_{TID}$ *Extended mnemonic for* **tlbre RT,RA,0**		24-184
tlbrelo	RT, RA	Load TLBLO of the selected TLB entry into RT. (RT) ← TLBLO[(RA)] *Extended mnemonic for* **tlbre RT,RA,1**		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 = (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.	$CR[CR0]_{LT,GT,SO}$	
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
tlbwehi	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) $TLB[(RA)]_{TID}$ ← $(PID)_{24:31}$ *Extended mnemonic for* **tlbwe RS,RA,0**		24-188
tlbwelo	RS, RA	Write TLBLO of the selected TLB entry from RS. TLBLO[(RA)] ← (RS) *Extended mnemonic for* **tlbwe RS,RA,1**		24-188

Table A-1. PPC405GP Instruction Syntax Summary (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
trap		Trap unconditionally. *Extended mnemonic for* **tw 31,0,0**		24-190
tweq	RA, RB	Trap if (RA) equal to (RB). *Extended mnemonic for* **tw 4,RA,RB**		
twge		Trap if (RA) greater than or equal to (RB). *Extended mnemonic for* **tw 12,RA,RB**		
twgt		Trap if (RA) greater than (RB). *Extended mnemonic for* **tw 8,RA,RB**		
twle		Trap if (RA) less than or equal to (RB). *Extended mnemonic for* **tw 20,RA,RB**		
twlge		Trap if (RA) logically greater than or equal to (RB). *Extended mnemonic for* **tw 5,RA,RB**		
twlgt		Trap if (RA) logically greater than (RB). *Extended mnemonic for* **tw 1,RA,RB**		
twlle		Trap if (RA) logically less than or equal to (RB). *Extended mnemonic for* **tw 6,RA,RB**		
twllt		Trap if (RA) logically less than (RB). *Extended mnemonic for* **tw 2,RA,RB**		
twlng		Trap if (RA) logically not greater than (RB). *Extended mnemonic for* **tw 6,RA,RB**		
twlnl		Trap if (RA) logically not less than (RB). *Extended mnemonic for* **tw 5,RA,RB**		
twlt		Trap if (RA) less than (RB). *Extended mnemonic for* **tw 16,RA,RB**		
twne		Trap if (RA) not equal to (RB). *Extended mnemonic for* **tw 24,RA,RB**		
twng		Trap if (RA) not greater than (RB). *Extended mnemonic for* *tw 20,RA,RB*		
twnl		Trap if (RA) not less than (RB). *Extended mnemonic for* **tw 12,RA,RB**		
tw	TO, RA, RB	Trap exception is generated if, comparing (RA) with (RB), any condition specified by TO is true.		24-190

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). *Extended mnemonic for* **wi 4,RA,IM**		24-193
twgei		Trap if (RA) greater than or equal to EXTS(IM). *Extended mnemonic for* **twi 12,RA,IM**		
twgti		Trap if (RA) greater than EXTS(IM). *Extended mnemonic for* **twi 8,RA,IM**		
twlei		Trap if (RA) less than or equal to EXTS(IM). *Extended mnemonic for* **twi 20,RA,IM**		
twlgei		Trap if (RA) logically greater than or equal to EXTS(IM). *Extended mnemonic for* **wi 5,RA,IM**		
twlgti		Trap if (RA) logically greater than EXTS(IM). *Extended mnemonic for* **twi 1,RA,IM**		
twllei		Trap if (RA) logically less than or equal to EXTS(IM). *Extended mnemonic for* **twi 6,RA,IM**		
twllti		Trap if (RA) logically less than EXTS(IM). *Extended mnemonic for* **twi 2,RA,IM**		
twlngi		Trap if (RA) logically not greater than EXTS(IM). *Extended mnemonic for* **twi 6,RA,IM**		
twlnli		Trap if (RA) logically not less than EXTS(IM). *Extended mnemonic for* **twi 5,RA,IM**		
twlti		Trap if (RA) less than EXTS(IM). *Extended mnemonic for* **twi 16,RA,IM**		
twnei		Trap if (RA) not equal to EXTS(IM). *Extended mnemonic for* **twi 24,RA,IM**		
twngi		Trap if (RA) not greater than EXTS(IM). *Extended mnemonic for* **twi 20,RA,IM**		
twnli		Trap if (RA) not less than EXTS(IM). *Extended mnemonic for* **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 OPCD 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	**mulhhwu**	RT, RA, RB	24-124
			mulhhwu.		
4	12 (524)	XO	**machhwu**	RT, RA, RB	24-102
			machhwu.		
			machhwuo		
			machhwuo.		
4	40	X	**mulhhw**	RT, RA, RB	24-123
			mulhhw.		
4	44 (556)	XO	**machhw**	RT, RA, RB	24-99
			machhw.		
			machhwo		
			machhwo.		
4	46 (558)	XO	**nmachhw**	RT, RA, RB	24-135
			nmachhw.		
			nmachhwo		
			nmachhwo		

Table A-2. PPC405GP Instructions by Opcode (continued)

Primary Opcode	Secondary Opcode	Form	Mnemonic	Operands	Page
4	76 (588)	XO	**machhwsu**	RT, RA, RB	24-101
			machhwsu.		
			machhwsuo		
			machhwsuo.		
4	108 (620)	XO	**machhws**	RT, RA, RB	24-100
			machhws.		
			machhwso		
			machhwso.		
4	110 (622)	XO	**nmachhws**	RT, RA, RB	24-136
			nmachhws.		
			nmachhwso		
			nmachhwso.		
4	136	X	**mulchwu**	RT, RA, RB	24-122
			mulchwu.		
4	140 (652)	XO	**macchwu**	RT, RA, RB	24-98
			macchwu.		
			macchwuo		
			machhwuo.		
4	168	X	**mulchw**	RT, RA, RB	24-121
			mulchw.		
4	172 (684)	XO	**macchw**	RT, RA, RB	24-95
			macchw.		
			macchwo		
			macchwo.		
4	174 (686)	XO	**nmacchw**	RT, RA, RB	24-133
			nmacchw.		
			nmacchwo		
			nmacchwo.		
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	**mullhwu**	RT, RA, RB	24-128
			mullhwu.		
4	396 (908)	XO	**maclhwu**	RT, RA, RB	24-106
			maclhwu.		
			maclhwuo		
			maclhwuo.		
4	424	X	**mullhw**	RT, RA, RB	24-126
			mullhw.		
4	428 (940)	XO	**maclhw**	RT, RA, RB	24-103
			maclhw.		
			maclhwo		
			maclhwo.		
4	430 (942)	XO	**nmaclhw**	RT, RA, RB	24-137
			nmaclhw.		
			nmaclhwo		
			nmaclhwo.		
4	492 (972)	XO	**maclhws**	RT, RA, RB	24-104
			maclhws.		
			maclhwso		
			maclhwso.		
4	460 (1004)	XO	**maclhwsu**	RT, RA, RB	24-105
			maclhwsu.		
			maclhwsuo		
			maclhwsuo.		
4	494 (1006)	XO	**nmaclhws**	RT, RA, RB	24-138
			nmaclhws.		
			nmaclhwso		
			nmaclhwso.		
7		D	**mulli**	RT, RA, IM	24-129
8		D	**subfic**	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	**rfci**		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	**rlwimi**	RA, RS, SH, MB, ME	24-146
			rlwimi.		
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	**andi.**	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	**mfcr**	RT	24-109
31	20	X	**lwarx**	RT, RA, RB	24-89
31	23	X	**lwzx**	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	**wrteei**	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	**dcbtst**	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	**dcba**	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	**eieio**		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 OPCD 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

OPCD	LI	
0	6	31

Figure A-1. I Instruction Format

A.3.2.2 B-Form

OPCD	BO	BI	BD	AA	LK
0	6	11	16	30	31

Figure A-2. B Instruction Format

A.3.2.3 SC-Form

OPCD	///	///	///	1	/
0	6	11	16	30	31

Figure A-3. SC Instruction Format

A.3.2.4 D-Form

OPCD	RT			RA	D	
OPCD	RS			RA	SI	
OPCD	RS			RA	D	
OPCD	RS			RA	UI	
OPCD	BF	/	L	RA	SI	
OPCD	BF	/	L	RA	UI	
OPCD	TO			RA	SI	
0	6			11	16	31

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 X0-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) (RA|0) + (RB).		24-56
dcread	RT, RA, RB	Read either tag or data information from the data cache congruence class associated with the EA (RA|0) + (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 (RA|0) + (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	$prod_{0:31} \leftarrow (RA)_{16:31}$ x $(RB)_{0:15}$ signed $temp_{0:32} \leftarrow prod_{0:31} + (RT)$ $(RT) \leftarrow temp_{1:32}$		24-95
macchw.			CR[CR0]	
macchwo			XER[SO, OV]	
macchwo.			CR[CR0] XER[SO, OV]	
macchws	RT, RA, RB	$prod_{0:31} \leftarrow (RA)_{16:31}$ x $(RB)_{0:15}$ 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	$prod_{0:31} \leftarrow (RA)_{16:31}$ x $(RB)_{0:15}$ 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	$prod_{0:31} \leftarrow (RA)_{16:31}$ x $(RB)_{0:15}$ 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	$prod_{0:15} \leftarrow (RA)_{16:31}$ x $(RB)_{0:15}$ 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	$prod_{0:31} \leftarrow (RA)_{0:15}$ x $(RB)_{0:15}$ 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	$prod_{0:31} \leftarrow (RA)_{0:15}$ x $(RB)_{0:15}$ 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 $(RT) \leftarrow (RT_0 \Vert {}^{31}(\neg RT_0))$ 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 $(RT) \leftarrow (RT_0 \parallel {}^{31}(\neg RT_0))$ 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 $(RT) \leftarrow (RT_0 \parallel {}^{31}(\neg RT_0))$ else $(RT) \leftarrow temp_{1:32}$		24-136
nmachhws.			CR[CR0]	
nmachhwso			XER[SO, OV]	
nmachhwso.			CR[CR0] XER[SO, OV]	
nmaclhw	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
nmaclhw.			CR[CR0]	
nmaclhwo			XER[SO, OV]	
nmaclhwo.			CR[CR0] XER[SO, OV]	
nmaclhws	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 $(RT) \leftarrow (RT_0 \parallel {}^{31}(\neg RT_0))$ else $(RT) \leftarrow temp_{1:32}$		24-138
nmaclhws.			CR[CR0]	
nmaclhwso			XER[SO, OV]	
nmaclhwso.			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
dcba	RA, RB	Speculatively establish the data cache block which contains the EA (RA|0) + (RB).		24-47
dcbf	RA, RB	Flush (store, then invalidate) the data cache block which contains the EA (RA|0) + (RB).		24-49
dcbi	RA, RB	Invalidate the data cache block which contains the EA (RA|0) + (RB).		24-50
dcbst	RA, RB	Store the data cache block which contains the EA (RA|0) + (RB).		24-51
dcbt	RA, RB	Load the data cache block which contains the EA (RA|0) + (RB).		24-52
dcbtst	RA,RB	Load the data cache block which contains the EA (RA|0) + (RB).		24-53
dcbz	RA, RB	Zero the data cache block which contains the EA (RA|0) + (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 (RA|0) + (RB).		24-65
icbt	RA, RB	Load the instruction cache block which contains the EA (RA|0) + (RB).		24-66
isync		Synchronize execution context by flushing the prefetch queue.		24-70
mfdcr	RT, DCRN	Move from DCR to RT, (RT) $\leftarrow$ (DCR(DCRN)).		24-110
mftb	RT	Move the contents of a Time Base Register (TBR) into RT, TBRN $\leftarrow$ $TBRF_{5:9}$ || $TBRF_{0:4}$ (RT) $\leftarrow$ (TBR(TBRN))		24-114
mfmsr	RT	Move from MSR to RT, (RT) $\leftarrow$ (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
rfci		Return from critical interrupt (PC) ← (SRR2). (MSR) ← (SRR3).		24-144
rfi		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 = (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.	$CR[CR0]_{LT,GT,SO}$	
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 (RA|0) + (RB).		24-50
dccci	RA, RB	Invalidate the data cache congruence class associated with the EA (RA|0) + (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 (RA|0) + (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 (RA|0) + (RB). Place the results in ICDBDR.		24-68
mfdcr	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
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
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) ← $\text{TLB[(RA)]}_{\text{TID}}$ 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 $BO_4 = 0$ unless a specific reason exists otherwise. In the BO field values specified in the following table, $BO_4 = 0$ 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. *Extended mnemonic for* **bcctr 20,0**		24-26
bctrl		*Extended mnemonic for* **bcctrl 20,0**	(LR) ← CIA + 4	
bdnz	target	Decrement CTR. Branch if CTR ≠ 0. *Extended mnemonic for* **bc 16,0,target**		24-20
bdnza		*Extended mnemonic for* **bca 16,0,target**		
bdnzl		*Extended mnemonic for* **bcl 16,0,target**	(LR) ← CIA + 4.	
bdnzla		*Extended mnemonic for* **bcla 16,0,target**	(LR) ← CIA + 4.	
bdnzlr		Decrement CTR. Branch, if CTR ≠ 0,to address in LR. *Extended mnemonic for* **bclr 16,0**		24-30
bdnzlrl		*Extended mnemonic* for **bclrl 16,0**	(LR) ← CIA + 4.	
bdnzf	cr_bit, target	Decrement CTR. Branch if CTR ≠ 0 AND $CR_{cr\_bit} = 0$. *Extended mnemonic for* **bc 0,cr_bit,target**		24-20
bdnzfa		*Extended mnemonic for* **bca 0,cr_bit,target**		
bdnzfl		*Extended mnemonic for* **bcl 0,cr_bit,target**	(LR) ← CIA + 4.	
bdnzfla		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 0,cr_bit**		24-30
bdnzflrl		*Extended mnemonic for* **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 ≠ 0 AND $CR_{cr\_bit} = 1$. *Extended mnemonic for* **bc 8,cr_bit,target**		24-20
bdnzta		*Extended mnemonic for* **bca 8,cr_bit,target**		
bdnztl		*Extended mnemonic for* **bcl 8,cr_bit,target**	(LR) ← CIA + 4.	
bdnztla		*Extended mnemonic for* **bcla 8,cr_bit,target**	(LR) ← CIA + 4.	
bdnztlr	cr_bit	Decrement CTR. Branch, if CTR ≠ 0 AND $CR_{cr\_bit} = 1$, to address in LR. *Extended mnemonic for* **bclr 8,cr_bit**		24-30
bdnztlrl		*Extended mnemonic for* **bclrl 8,cr_bit**	(LR) ← CIA + 4.	
bdz	target	Decrement CTR. Branch if CTR = 0. *Extended mnemonic for* **bc 18,0,target**		24-20
bdza		*Extended mnemonic* for **bca 18,0,target**		
bdzl		*Extended mnemonic for* **bcl 18,0,target**	(LR) ← CIA + 4.	
bdzla		*Extended mnemonic for* **bcla 18,0,target**	(LR) ← CIA + 4.	
bdzlr		Decrement CTR. Branch, if CTR = 0, to address in LR. *Extended mnemonic for* **bclr 18,0**		24-30
bdzlrl		*Extended mnemonic for* **bclrl 18,0**	(LR) ← CIA + 4.	
bdzf	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND $CR_{cr\_bit} = 0$. *Extended mnemonic for* **bc 2,cr_bit,target**		24-20
bdzfa		*Extended mnemonic for* **bca 2,cr_bit,target**		
bdzfl		*Extended mnemonic for* **bcl 2,cr_bit,target**	(LR) ← CIA + 4.	
bdzfla		*Extended mnemonic for* **bcla 2,cr_bit,target**	(LR) ← 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. *Extended mnemonic for* **bclr 2,cr_bit**		24-30
bdzflrl		*Extended mnemonic for* **bclrl 2,cr_bit**	(LR) ← CIA + 4.	
bdzt	cr_bit, target	Decrement CTR. Branch if CTR = 0 AND CR_{cr_bit} = 1. *Extended mnemonic for* **bc 10,cr_bit,target**		24-20
bdzta		*Extended mnemonic for* **bca 10,cr_bit,target**		
bdztl		*Extended mnemonic for* **bcl 10,cr_bit,target**	(LR) ← CIA + 4.	
bdztla		*Extended mnemonic for* **bcla 10,cr_bit,target**	(LR) ← CIA + 4.	
bdztlr	cr_bit	Decrement CTR. Branch, if CTR = 0 AND CR_{cr_bit} = 1, to address in LR. *Extended mnemonic for* **bclr 10,cr_bit**		24-30
bdztlrl		*Extended mnemonic for* **bclrl 10,cr_bit**	(LR) ← CIA + 4.	
beq	[cr_field,] target	Branch if equal. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+2,target**		24-20
beqa		*Extended mnemonic for* **bca 12,4∗cr_field+2,target**		
beql		*Extended mnemonic for* **bcl 12,4∗cr_field+2,target**	(LR) ← CIA + 4.	
beqla		*Extended mnemonic for* **bcla 12,4∗cr_field+2,target**	(LR) ← CIA + 4.	
beqctr	[cr_field]	Branch, if equal, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 12,4∗cr_field+2**		24-26
beqctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 12,4∗cr_field+2**		24-30
beqlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+2**	(LR) ← CIA + 4.	
bf	cr_bit, target	Branch if CR_{cr_bit} = 0. *Extended mnemonic for* **bc 4,cr_bit,target**		24-20
bfa		*Extended mnemonic for* **bca 4,cr_bit,target**		
bfl		*Extended mnemonic for* **bcl 4,cr_bit,target**	(LR) ← CIA + 4.	
bfla		*Extended mnemonic for* **bcla 4,cr_bit,target**	(LR) ← CIA + 4.	
bfctr	cr_bit	Branch, if CR_{cr_bit} = 0, to address in CTR. *Extended mnemonic for* **bcctr 4,cr_bit**		24-26
bfctrl		*Extended mnemonic for* **bcctrl 4,cr_bit**	(LR) ← CIA + 4.	
bflr	cr_bit	Branch, if CR_{cr_bit} = 0, to address in LR. *Extended mnemonic for* **bclr 4,cr_bit**		24-30
bflrl		*Extended mnemonic for* **bclrl 4,cr_bit**	(LR) ← CIA + 4.	
bge	[cr_field,] target	Branch if greater than or equal. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+0,target**		24-20
bgea		*Extended mnemonic for* **bca 4,4∗cr_field+0,target**		
bgel		*Extended mnemonic for* **bcl 4,4∗cr_field+0,target**	(LR) ← CIA + 4.	
bgela		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 4,4∗cr_field+0**		24-26
bgectrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 4,4∗cr_field+0**		24-30
bgelrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+0**	(LR) ← CIA + 4.	
bgt	[cr_field,] target	Branch if greater than. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+1,target**		24-20
bgta		*Extended mnemonic for* **bca 12,4∗cr_field+1,target**		
bgtl		*Extended mnemonic for* **bcl 12,4∗cr_field+1,target**	(LR) ← CIA + 4.	
bgtla		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 12,4∗cr_field+1**		24-26
bgtctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 12,4∗cr_field+1**		24-30
bgtlrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bc 4,4∗cr_field+1,target**		24-20
blea		*Extended mnemonic for* **bca 4,4∗cr_field+1,target**		
blel		*Extended mnemonic for* **bcl 4,4∗cr_field+1,target**	(LR) ← CIA + 4.	
blela		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 4,4∗cr_field+1**		24-26
blectrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 4,4∗cr_field+1**		24-30
blelrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+1**	(LR) ← CIA + 4.	
blr		Branch, unconditionally, to address in LR. *Extended mnemonic for* **bclr 20,0**		24-30
blrl		*Extended mnemonic for* **bclrl 20,0**	(LR) ← CIA + 4.	
blt	[cr_field,] target	Branch if less than. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 12,4∗cr_field+0,target**		24-20
blta		*Extended mnemonic for* **bca 12,4∗cr_field+0,target**		
bltl		*Extended mnemonic for* **bcl 12,4∗cr_field+0,target**	(LR) ← CIA + 4.	
bltla		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 12,4∗cr_field+0**		24-26
bltctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 12,4∗cr_field+0**		24-30
bltlrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bc 4,4*cr_field+2,target**		24-20
bnea		*Extended mnemonic for* **bca 4,4*cr_field+2,target**		
bnel		*Extended mnemonic for* **bcl 4,4*cr_field+2,target**	(LR) ← CIA + 4.	
bnela		*Extended mnemonic for* **bcla 4,4*cr_field+2,target**	(LR) ← CIA + 4.	
bnectr	[cr_field]	Branch, if not equal, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4*cr_field+2**		24-26
bnectrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 4,4*cr_field+2**		24-30
bnelrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bc 4,4*cr_field+1,target**		24-20
bnga		*Extended mnemonic for* **bca 4,4*cr_field+1,target**		
bngl		*Extended mnemonic for* **bcl 4,4*cr_field+1,target**	(LR) ← CIA + 4.	
bngla		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 4,4*cr_field+1**		24-26
bngctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 4,4∗cr_field+1**		24-30
bnglrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bc 4,4∗cr_field+0,target**		24-20
bnla		*Extended mnemonic for* **bca 4,4∗cr_field+0,target**		
bnll		*Extended mnemonic for* **bcl 4,4∗cr_field+0,target**	(LR) ← CIA + 4.	
bnlla		*Extended mnemonic for* **bcla 4,4∗cr_field+0,target**	(LR) ← CIA + 4.	
bnlctr	[cr_field]	Branch, if not less than, to address in CTR. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bcctr 4,4∗cr_field+0**		24-26
bnlctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 4,4∗cr_field+0**		24-30
bnllrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bc 4,4∗cr_field+3,target**		24-20
bnsa		*Extended mnemonic for* **bca 4,4∗cr_field+3,target**		
bnsl		*Extended mnemonic for* **bcl 4,4∗cr_field+3,target**	(LR) ← CIA + 4.	
bnsla		*Extended mnemonic for* **bcla 4,4∗cr_field+3,target**	(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. *Extended mnemonic for* **bcctr 4,4∗cr_field+3**		24-26
bnsctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 4,4∗cr_field+3**		24-30
bnslrl		*Extended mnemonic for* **bclrl 4,4∗cr_field+3**	(LR) ← CIA + 4.	
bnu	[cr_field,] target	Branch if not unordered. Use CR0 if cr_field is omitted. *Extended mnemonic for* **bc 4,4∗cr_field+3,target**		24-20
bnua		*Extended mnemonic for* **bca 4,4∗cr_field+3,target**		
bnul		*Extended mnemonic for* **bcl 4,4∗cr_field+3,target**	(LR) ← CIA + 4.	
bnula		*Extended mnemonic for* **bcla 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. *Extended mnemonic for* **bcctr 4,4∗cr_field+3**		24-26
bnuctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 4,4∗cr_field+3**		24-30
bnulrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bc 12,4∗cr_field+3,target**		24-20
bsoa		*Extended mnemonic for* **bca 12,4∗cr_field+3,target**		
bsol		*Extended mnemonic for* **bcl 12,4∗cr_field+3,target**	(LR) ← CIA + 4.	
bsola		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 12,4∗cr_field+3**		24-26
bsoctrl		*Extended mnemonic for* **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. *Extended mnemonic for* **bclr 12,4∗cr_field+3**		24-30
bsolrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+3**	(LR) ← CIA + 4.	
bt	cr_bit, target	Branch if CR_{cr_bit} = 1. *Extended mnemonic for* **bc 12,cr_bit,target**		24-20
bta		*Extended mnemonic for* **bca 12,cr_bit,target**		
btl		*Extended mnemonic for* **bcl 12,cr_bit,target**	(LR) ← CIA + 4.	
btla		*Extended mnemonic for* **bcla 12,cr_bit,target**	(LR) ← CIA + 4.	
btctr	cr_bit	Branch if CR_{cr_bit} = 1, to address in CTR. *Extended mnemonic for* **bcctr 12,cr_bit**		24-26
btctrl		*Extended mnemonic for* **bcctrl 12,cr_bit**	(LR) ← CIA + 4.	
btlr	cr_bit	Branch, if CR_{cr_bit} = 1, to address in LR. *Extended mnemonic for* **bclr 12,cr_bit**		24-30
btlrl		*Extended mnemonic for* **bclrl 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. *Extended mnemonic for* **bc 12,4∗cr_field+3,target**		24-20
buna		*Extended mnemonic for* **bca 12,4∗cr_field+3,target**		
bunl		*Extended mnemonic for* **bcl 12,4∗cr_field+3,target**	(LR) ← CIA + 4.	
bunla		*Extended mnemonic for* **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. *Extended mnemonic for* **bcctr 12,4∗cr_field+3**		24-26
bunctrl		*Extended mnemonic* for **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. *Extended mnemonic for* **bclr 12,4∗cr_field+3**		24-30
bunlrl		*Extended mnemonic for* **bclrl 12,4∗cr_field+3**	(LR) ← CIA + 4.	
clrlwi	RA, RS, n	Clear left immediate. $(n < 32)$ $(RA)_{0:n-1} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,0,n,31**		24-147
clrlwi.		*Extended mnemonic for* **rlwinm. RA,RS,0,n,31**	CR[CR0]	
clrlslwi	RA, RS, b, n	Clear left and shift left immediate. $(n \le b < 32)$ $(RA)_{b-n:31-n} \leftarrow (RS)_{b:31}$ $(RA)_{32-n:31} \leftarrow {}^{n}0$ $(RA)_{0:b-n-1} \leftarrow {}^{b-n}0$ *Extended mnemonic for* **rlwinm RA,RS,n,b−n,31−n**		24-147
clrlslwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,b−n,31−n**	CR[CR0]	
clrrwi	RA, RS, n	Clear right immediate. $(n < 32)$ $(RA)_{32-n:31} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,0,0,31−n**		24-147
clrrwi.		*Extended mnemonic for* **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. *Extended mnemonic for* **cmpl BF,0,RA,RB**		24-36
cmplwi	[BF,] RA, IM	Compare Logical Word Immediate. Use CR0 if BF is omitted. *Extended mnemonic for* **cmpli BF,0,RA,IM**		24-37
cmpw	[BF,] RA, RB	Compare Word. Use CR0 if BF is omitted. *Extended mnemonic for* **cmp BF,0,RA,RB**		24-34
cmpwi	[BF,] RA, IM	Compare Word Immediate. Use CR0 if BF is omitted. *Extended mnemonic for* **cmpi BF,0,RA,IM**		24-35
crclr	bx	Condition register clear. *Extended mnemonic for* **crxor bx,bx,bx**		24-46
crmove	bx, by	Condition register move. *Extended mnemonic for* **cror bx,by,by**		24-44
crnot	bx, by	Condition register not. *Extended mnemonic for* **crnor bx,by,by**		24-43
crset	bx	Condition register set. *Extended mnemonic for* **creqv bx,bx,bx**		24-41
extlwi	RA, RS, n, b	Extract and left justify immediate. (n > 0) $(RA)_{0:n-1} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{n:31} \leftarrow {}^{32-n}0$ *Extended mnemonic for* **rlwinm RA,RS,b,0,n–1**		24-147
extlwi.		*Extended mnemonic for* **rlwinm. RA,RS,b,0,n–1**	CR[CR0]	
extrwi	RA, RS, n, b	Extract and right justify immediate. (n > 0) $(RA)_{32-n:31} \leftarrow (RS)_{b:b+n-1}$ $(RA)_{0:31-n} \leftarrow {}^{32-n}0$ *Extended mnemonic for* **rlwinm RA,RS,b+n,32–n,31**		24-147
extrwi.		*Extended mnemonic for* **rlwinm. RA,RS,b+n,32–n,31**	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. $(n > 0)$ $(RA)_{b:b+n-1} \leftarrow (RS)_{0:n-1}$ *Extended mnemonic for* **rlwimi RA,RS,32–b,b,b+n–1**		24-146
inslwi.		*Extended mnemonic for* **rlwimi. RA,RS,32–b,b,b+n–1**	CR[CR0]	
insrwi	RA, RS, n, b	Insert from right immediate. $(n > 0)$ $(RA)_{b:b+n-1} \leftarrow (RS)_{32-n:31}$ *Extended mnemonic for* **rlwimi RA,RS,32–b–n,b,b+n–1**		24-146
insrwi.		*Extended mnemonic for* **rlwimi. 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) ← (RA) + EXTS(D)* *Extended mnemonic* for **addi RT,RA,D**		24-9
li	RT, IM	Load immediate. (RT) ← EXTS(IM) *Extended mnemonic for* **addi RT,0,value**		24-9
lis	RT, IM	Load immediate shifted. $(RT) \leftarrow (IM \parallel {}^{16}0)$ *Extended mnemonic for* **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** **mficdbdr** **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. *Extended mnemonic for* **mfspr RT,SPRN** 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) *Extended mnemonic for* **mftb RT,TBL**		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) *Extended mnemonic for* **mftb RT,TBU**		24-114
mr	RT, RS	Move register. (RT) ← (RS) *Extended mnemonic for* **or RT,RS,RS**		24-140
mr.		*Extended mnemonic for* **or. RT,RS,RS**	CR[CR0]	
mtcr	RS	Move to Condition Register. *Extended mnemonic for* **mtcrf 0xFF,RS**		24-116

Table B-5. Extended Mnemonics for PPC405GP (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** **mticcr** **mticdbdr** **mtlr** **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. *Extended mnemonic* for **mtspr SPRN,RS** 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. *Extended mnemonic for* **ori 0,0,0**		24-142

Table B-5. Extended Mnemonics for PPC405GP (continued)

Mnemonic	Operands	Function	Other Registers Changed	Page
not	RA, RS	Complement register. (RA) ← ¬(RS) *Extended mnemonic for* **nor RA,RS,RS**		24-139
not.		*Extended mnemonic for* **nor. RA,RS,RS**	CR[CR0]	
rotlw	RA, RS, RB	Rotate left. (RA) ← ROTL((RS), $(RB)_{27:31}$) *Extended mnemonic for* **rlwnm RA,RS,RB,0,31**		24-150
rotlw.		*Extended mnemonic for* **rlwnm. RA,RS,RB,0,31**	CR[CR0]	
rotlwi	RA, RS, n	Rotate left immediate. (RA) ← ROTL((RS), n) *Extended mnemonic for* **rlwinm RA,RS,n,0,31**		24-147
rotlwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,0,31**	CR[CR0]	
rotrwi	RA, RS, n	Rotate right immediate. (RA) ← ROTL((RS), 32–n) *Extended mnemonic for* **rlwinm RA,RS,32–n,0,31**		24-147
rotrwi.		*Extended mnemonic for* **rlwinm. RA,RS,32–n,0,31**	CR[CR0]	
slwi	RA, RS, n	Shift left immediate. (n < 32) $(RA)_{0:31-n} \leftarrow (RS)_{n:31}$ $(RA)_{32-n:31} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,n,0,31–n**		24-147
slwi.		*Extended mnemonic for* **rlwinm. RA,RS,n,0,31–n**	CR[CR0]	
srwi	RA, RS, n	Shift right immediate. (n < 32) $(RA)_{n:31} \leftarrow (RS)_{0:31-n}$ $(RA)_{0:n-1} \leftarrow {}^{n}0$ *Extended mnemonic for* **rlwinm RA,RS,32–n,n,31**		24-147
srwi.		*Extended mnemonic for* **rlwinm. RA,RS,32–n,n,31**	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) ← ¬(RB) + (RA) + 1. *Extended mnemonic for* **subf RT,RB,RA**		24-176
sub.		*Extended mnemonic for* **subf. RT,RB,RA**	CR[CR0]	
subo		*Extended mnemonic for* **subfo RT,RB,RA**	XER[SO, OV]	
subo.		*Extended mnemonic for* **subfo. RT,RB,RA**	CR[CR0] XER[SO, OV]	
subc	RT, RA, RB	Subtract (RB) from (RA). (RT) ← ¬(RB) + (RA) + 1. Place carry-out in XER[CA]. *Extended mnemonic for* **subfc RT,RB,RA**		24-177
subc.		*Extended mnemonic for* **subfc. RT,RB,RA**	CR[CR0]	
subco		*Extended mnemonic for* **subfco RT,RB,RA**	XER[SO, OV]	
subco.		*Extended mnemonic for* **subfco. RT,RB,RA**	CR[CR0] XER[SO, OV]	
subi	RT, RA, IM	Subtract EXTS(IM) from (RA|0). Place result in RT. *Extended mnemonic for* **addi RT,RA,–IM**		24-9
subic	RT, RA, IM	Subtract EXTS(IM) from (RA). Place result in RT. Place carry-out in XER[CA]. *Extended mnemonic for* **addic RT,RA,–IM**		24-10
subic.	RT, RA, IM	Subtract EXTS(IM) from (RA). Place result in RT. Place carry-out in XER[CA]. *Extended mnemonic for* **addic. RT,RA,–IM**	CR[CR0]	24-11
subis	RT, RA, IM	Subtract (IM || $^{16}0$) from (RA|0). Place result in RT. *Extended mnemonic for* **addis RT,RA,–IM**		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). *Extended mnemonic* for twi 4,RA,IM		
twgei		Trap if (RA) greater than or equal to EXTS(IM). *Extended mnemonic for* **twi 12,RA,IM**		
twgti		Trap if (RA) greater than EXTS(IM). *Extended mnemonic for* **twi 8,RA,IM**		
twlei		Trap if (RA) less than or equal to EXTS(IM). *Extended mnemonic for* **twi 20,RA,IM**		
twlgei		Trap if (RA) logically greater than or equal to EXTS(IM). *Extended mnemonic for* **twi 5,RA,IM**		
twlgti		Trap if (RA) logically greater than EXTS(IM). *Extended mnemonic for* **twi 1,RA,IM**		
twllei		Trap if (RA) logically less than or equal to EXTS(IM). *Extended mnemonic for* **twi 6,RA,IM**		
twllti		Trap if (RA) logically less than EXTS(IM). *Extended mnemonic for* **twi 2,RA,IM**		
twlngi		Trap if (RA) logically not greater than EXTS(IM). *Extended mnemonic for* **twi 6,RA,IM**		
twlnli		Trap if (RA) logically not less than EXTS(IM). *Extended mnemonic for* **twi 5,RA,IM**		
twlti		Trap if (RA) less than EXTS(IM). *Extended mnemonic for* **twi 16,RA,IM**		
twnei		Trap if (RA) not equal to EXTS(IM). *Extended mnemonic for* **twi 24,RA,IM**		
twngi		Trap if (RA) not greater than EXTS(IM). *Extended mnemonic for* **twi 20,RA,IM**		
twnli		Trap if (RA) not less than EXTS(IM). *Extended mnemonic for* **twi 12,RA,IM**		

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 = (RA|0) + EXTS(D) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1).		24-71
lbzu	RT, D(RA)	Load byte from EA = (RA|0) + 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 = (RA|0) + (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 = (RA|0) + (RB) and pad left with zeroes, (RT) ← $^{24}0$ || MS(EA,1).		24-74
lha	RT, D(RA)	Load halfword from EA = (RA|0) + EXTS(D) and sign extend, (RT) ← EXTS(MS(EA,2)).		24-75
lhau	RT, D(RA)	Load halfword from EA = (RA|0) + 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 = (RA|0) + (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 = (RA|0) + (RB) and sign extend, (RT) ← EXTS(MS(EA,2)).		24-78
lhbrx	RT, RA, RB	Load halfword from EA = (RA|0) + (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 = (RA|0) + EXTS(D) and pad left with zeroes, (RT) ← $^{16}0$ || 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 = (RA|0) + EXTS(D) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2). Update the base address, (RA) ← EA.		24-81
lhzux	RT, RA, RB	Load halfword from EA = (RA|0) + (RB) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2). Update the base address, (RA) ← EA.		24-82
lhzx	RT, RA, RB	Load halfword from EA = (RA|0) + (RB) and pad left with zeroes, (RT) ← $^{16}0$ || MS(EA,2).		24-83
lmw	RT, D(RA)	Load multiple words starting from EA = (RA|0) + EXTS(D). Place into consecutive registers, RT through GPR(31). RA is not altered unless RA = GPR(31).		24-84
lswi	RT, RA, NB	Load consecutive bytes from EA = (RA|0). Number of bytes *n* = 32 if NB = 0, else *n* = NB. Stack bytes into words in CEIL(n/4) consecutive registers starting with RT, to R_{FINAL} ← ((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=(RA|0)+(RB). Number of bytes *n* = XER[TBC]. Stack bytes into words in CEIL(*n*/4) consecutive registers starting with RT, to R_{FINAL} ← ((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 = (RA|0) + (RB)and place in RT, (RT) ← MS(EA,4). Set the Reservation bit.		24-89
lwbrx	RT, RA, RB	Load word from EA = (RA|0) + (RB) then reverse byte order, (RT) ← 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 = (RA|0) + EXTS(D) and place in RT, (RT) ← 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 = (RA|0) + 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 = (RA|0) + (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 = (RA|0) + (RB) and place in RT, (RT) ← MS(EA,4).		24-94
stb	RS, D(RA)	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + EXTS(D).		24-156
stbu	RS, D(RA)	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + EXTS(D). Update the base address, (RA) ← EA.		24-157
stbux	RS, RA, RB	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + (RB). Update the base address, (RA) ← EA.		24-158
stbx	RS, RA, RB	Store byte $(RS)_{24:31}$ in memory at EA = (RA|0) + (RB).		24-159
sth	RS, D(RA)	Store halfword $(RS)_{16:31}$ in memory at EA = (RA|0) + EXTS(D).		24-160
sthbrx	RS, RA, RB	Store halfword $(RS)_{16:31}$ byte-reversed in memory at EA = (RA|0) + (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 = (RA|0) + EXTS(D). Update the base address, (RA) ← EA.		24-162
sthux	RS, RA, RB	Store halfword $(RS)_{16:31}$ in memory at EA = (RA|0) + (RB). Update the base address, (RA) ← EA.		24-163
sthx	RS, RA, RB	Store halfword $(RS)_{16:31}$ in memory at EA = (RA|0) + (RB).		24-164
stmw	RS, D(RA)	Store consecutive words from RS through GPR(31) in memory starting at EA = (RA|0) + 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=(RA|0). Number of bytes *n* = 32 if NB = 0, else *n* = NB. Bytes are unstacked from CEIL(*n*/4) consecutive registers starting with RS. GPR(0) is consecutive to GPR(31).		24-166
stswx	RS, RA, RB	Store consecutive bytes in memory starting at EA=(RA|0)+(RB). Number of bytes *n* = XER[TBC]. Bytes are unstacked from CEIL(*n*/4) consecutive registers starting with RS. GPR(0) is consecutive to GPR(31).		24-167
stw	RS, D(RA)	Store word (RS) in memory at EA = (RA|0) + EXTS(D).		24-169
stwbrx	RS, RA, RB	Store word (RS) byte-reversed in memory at EA = (RA|0) + (RB). MS(EA, 4) ← $(RS)_{24:31}$ || $(RS)_{16:23}$ || $(RS)_{8:15}$ || $(RS)_{0:7}$		24-170

stwcx.	RS, RA, RB	Store word (RS) in memory at EA = (RA|0) + (RB) only if the reservation bit is set.
		if RESERVE = 1 then
		  MS(EA, 4) ← (RS)
		  RESERVE ← 0
		  (CR[CR0]) ← $^{2}0$ || 1 || XER_{so}
		else
		  (CR[CR0]) ← $^{2}0$ || 0 || $XER_{so.}$
				24-171

stwu	RS, D(RA)	Store word (RS) in memory at EA = (RA|0) + EXTS(D). Update the base address, (RA) ← EA.		24-173
stwux	RS, RA, RB	Store word (RS) in memory at EA = (RA|0) + (RB). Update the base address, (RA) ← EA.		24-174
stwx	RS, RA, RB	Store word (RS) in memory at EA = (RA|0) + (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 (RA|0). Place result in RT.		24-9
addic	RT, RA, IM	Add EXTS(IM) to (RA|0). Place result in RT. Place carry-out in XER[CA].		24-10
addic.	RT, RA, IM	Add EXTS(IM) to (RA|0). Place result in RT. Place carry-out in XER[CA].	CR[CR0]	24-11
addis	RT, RA, IM	Add (IM || $^{16}0$) to (RA|0). 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 ¬(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
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) ← ¬((RS) ⊕ (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
mullw	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
mullw.			CR[CR0]	
mullwo			XER[SO, OV]	
mullwo.			CR[CR0] XER[SO, OV]	
nand	RA, RS, RB	NAND (RS) with (RB). Place result in RA.		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 $(^{16}0 \parallel IM)$. Place result in RA.		24-142
oris	RA, RS, IM	OR (RS) with $(IM \parallel {}^{16}0)$. 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 ← (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
bclrl			CTR if BO_2 = 0. (LR) ← 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.
		r ← ROTL((RS), SH)
		m ← MASK(MB, ME)
		(RA) ← (r ∧ m) ∨ ((RA) ∧ ¬m)
				24-146

rlwimi.			CR[CR0]	

rlwinm	RA, RS, SH, MB, ME	Rotate left word immediate, then AND with mask.
		r ← ROTL((RS), SH)
		m ← MASK(MB, ME)
		(RA) ← (r ∧ m)
				24-147

rlwinm.			CR[CR0]	

rlwnm	RA, RS, RB, MB, ME	Rotate left word, then AND with mask.
		r ← ROTL((RS), $(RB)_{27:31}$)
		m ← MASK(MB, ME)
		(RA) ← (r ∧ m)
				24-150

rlwnm.			CR[CR0]	

slw	RA, RS, RB	Shift left (RS) by $(RB)_{27:31}$.
		n ← $(RB)_{27:31}$.
		r ← ROTL((RS), n).
		if $(RB)_{26}$ = 0 then m ← MASK(0, 31 – n)
		else m ← $^{32}0$.
		(RA) ← r ∧ m.
				24-152

slw.			CR[CR0]	

sraw	RA, RS, RB	Shift right algebraic (RS) by $(RB)_{27:31}$.
		n ← $(RB)_{27:31}$.
		r ← ROTL((RS), 32 – *n*).
		if $(RB)_{26}$ = 0 then m ← MASK(n, 31)
		else m ← $^{32}0$.
		s ← $(RS)_0$.
		(RA) ← (r ∧ m) ∨ (^{32}s ∧ ¬m).
		XER[CA] ← s ∧ ((r ∧ ¬m) ≠ 0).
				24-153

sraw.			CR[CR0]	

srawi	RA, RS, SH	Shift right algebraic (RS) by SH.
		n ← SH.
		r ← ROTL((RS), 32 – n).
		m ← MASK(n, 31).
		s ← $(RS)_0$.
		(RA) ← (r ∧ m) ∨ (^{32}s ∧ ¬m).
		XER[CA] ← s ∧ ((r ∧ ¬m)≠0).
				24-154

srawi.			CR[CR0]	

srw	RA, RS, RB	Shift right (RS) by $(RB)_{27:31}$.
		n ← $(RB)_{27:31}$.
		r ← ROTL((RS), 32 – n).
		if $(RB)_{26}$ = 0 then m ← MASK(n, 31)
		else m ← $^{32}0$.
		(RA) ← r ∧ m.
				24-155

srw.			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 (RA|0) + (RB).		24-47
dcbf	RA, RB	Flush (store, then invalidate) the data cache block which contains the EA (RA|0) + (RB).		24-49
dcbi	RA, RB	Invalidate the data cache block which contains the EA (RA|0) + (RB).		24-50
dcbst	RA, RB	Store the data cache block which contains the EA (RA|0) + (RB).		24-51
dcbt	RA, RB	Load the data cache block which contains the EA (RA|0) + (RB).		24-52
dcbtst	RA,RB	Load the data cache block which contains the EA (RA|0) + (RB).		24-53
dcbz	RA, RB	Zero the data cache block which contains the EA (RA|0) + (RB).		24-54
dccci	RA, RB	Invalidate the data cache congruence class associated with the EA (RA|0) + (RB).		24-56
dcread	RT, RA, RB	Read either tag or data information from the data cache congruence class associated with the EA (RA|0) + (RB). Place the results in RT.		24-57
icbi	RA, RB	Invalidate the instruction cache block which contains the EA (RA|0) + (RB).		24-65
icbt	RA, RB	Load the instruction cache block which contains the EA (RA|0) + (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 (RA|0) + (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 CRn, where n←BF. CR[CRn] ← (XER[SO, OV, CA]). (XER[SO, OV, CA]) ← $^{3}0$.		24-108
mfcr	RT	Move from CR to RT, (RT) ← (CR).		24-109
mfdcr	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 || Var 31) 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 **mtcrf** 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 cyle 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
mulli[.], **mullw**[o][.], **mulhw**[.], **mulhwu**[.]	2	3
Halfword × Word		
mulli[.], **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

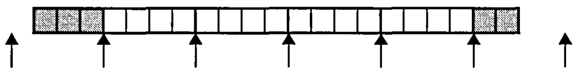
The PPC405GP requires an extra cycle to execute scalar loads and stores having unaligned big or little endian data (except for **lwarx** and **stwcx.**, which require word-aligned operands). If the target data is not operand aligned, and the sum of the least two significant bits of the effective address (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

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 ppcsupp@us.ibm.com

IBM®

PowerPC

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