MPR603ETS-01
(IBM Order Number)

MPC603UMAD/AD
(Motorola Order Number)

8/95

PowerPC™

Errata to
PowerPC 603™ RISC Microprocessor User's Manual

This errata describes corrections to the *PowerPC 603 RISC Microprocessor User's Manual.* For convenience, the section number and page number of the errata item in the user's manual are provided.

In this document, the term "603" is used as an abbreviation for the phrase, "PowerPC 603 microprocessor." The PowerPC 603 microprocessors are available from IBM as PPC603 and from Motorola as MPC603.

IBM

MOTOROLA

Section #/Page #	Changes
1.1.4.3, Page 1-7	Replace the term “speculative” with the term “out-of-order” in the third paragraph of this section and in all other instances in the 603 user’s manual.
2.1.4, Page 2-10	Figure 2-5— bit 29 is shown as ‘Reserved’; bit 29 should be the NI bit as described in Table 2-4, “FPSCR Bit Settings.”
2.1.4, Page 2-11	Table 2-4—Delete the text about denormalization for bit 25 (OE).
2.1.4, Page 2-12	Table 2-4—Change the last line of description for bit 29 (NI) as follows: 1 non-IEEE-compliant (nondenormalized numbers)
2.3.1, Page 2-22	Replace the description for bit 0 in Table 2-8 as follows: Reserved, but saved in SRR1 when an exception (other than TLB miss exception) occurs.
2.3.1, Page 2-25	Delete note 2 and associated reference from Table 2-10.
2.3.4, Page 2-28	Replace Figure 2-15 with the following:

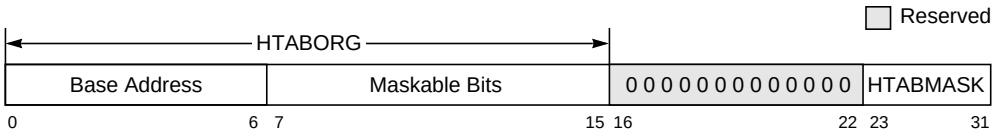

Figure 2-15. SDR1 Register

2.3.4, Page 2-29	Replace the fourth sentence in the third paragraph with the following sentence: The value in HTABMASK must be 0x007 and the value in HTABORG must have its low-order 3 bits (bits 13–15 of SDR1) equal to 0. This means that the page table must begin on a $2^{3 + 10 + 6} = 2^{19} = 512$ Kbytes boundary.
2.4.1, Page 2-37	Replace the entry for HID0[21] with the following:

21	DCFI	Data cache flash invalidate

2.4.3, Page 2-38	Replace Figure 2-27 with the following:

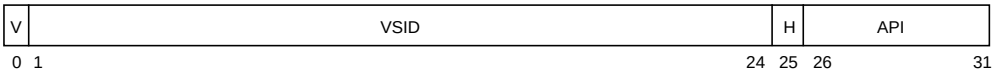

Figure 2-27. DCMP and ICMP Registers

Section #/Page # **Changes**

2.4.5, Page 2-39 Replace the third sentence in the first paragraph with the following:

When the **tlbld** or **tlbli** instruction is executed, the contents of the RPA register and the DMISS or IMISS register are merged and loaded into the selected TLB entry.

2.5.1, Page 2-43 Replace the last sentence on page 2-43 with the following:

Note that any time the IBAT registers are updated, the changes are guaranteed to take effect (including changes of the K*x* bits) only after a context-synchronizing operation has completed.

3.3.1, Page 3-17 Replace Table 3-6 with the following:

Table 3-6. Memory Operands

Operand	Length	Addr[28–31] If Aligned
Byte	8 bits	xxxx
Half word	2 bytes	xxx0
Word	4 bytes	xx00
Double word	8 bytes	x000
Quad word	16 bytes	0000

Note: An “x” in an address bit position indicates that the bit can be 0 or 1 independent of the state of other bits in the address.

4.7.5, Page 4-44 Replace the second sentence in the second paragraph with the following:

Execution of load multiple and store multiple instructions with misaligned operands (that is, the EA is not a multiple of 4) causes the 603 to invoke an alignment exception.

4.7.9.1, Page 4-51 Replace the conversion descriptions with the following:

No Denormalization Required (includes Zero/Infinity/NaN)

```
if frS[1–11] > 896 or frS[1–63] = 0 then
   WORD[0–1] ← frS[0–1]
   WORD[2–31] ← frS[5–34]
```

Denormalization Required

```
if 874 ≤ frS[1–11] ≤ 896 then
   sign ← frS[0]
   exp ← frS[1–11] – 1023
   frac ← 0b1 || frS[12–63]
   Denormalize operand
                Do while exp < –126
                   frac ← 0b0 || frac[0–62]
                   exp ← exp + 1
                End
```

Section #/Page #	Changes
	WORD[0] ← sign WORD[1–8] ← 0x00 WORD[9–31] ← frac[1–23] else WORD ← undefined
4.7.9.1, Page 4-52	Replace the first sentence of the second paragraph with the following sentence: For double-precision floating-point store instructions and for the **stfiwx** instruction, no conversion is required as the data from the FPRs is copied directly into memory.
4.7.9.2, Page 4-53	Table 4-20—Replace the second sentence in the “Operation” column for the **stfiwx** instruction with the following sentence: The contents of **fr**S[32–63] are stored, without conversion, into the word in memory addressed by the EA.
4.8.7, Page 4-65	Replace the last paragraph with the following: PowerPC-compliant assemblers provide the mnemonics and symbols listed in Section G.5, “Simplified Mnemonics for Branch Instructions.”
4.10, Page 4-72	Delete the fourth paragraph.
5.5.5.3, Page 5-14	Replace the last paragraph in Section 5.5.5.3 with the following sentence: Instruction fetching from I/O controller interface segments (segment register [T] = 1) is not permitted.

Section #/Page # **Changes**

5.6.2, Page 5-16 Replace Figure 5-4 with the following. Note that transition from the exclusive to the invalid state caused by snoop hit of cacheable read/write operation does not cause snoop push.

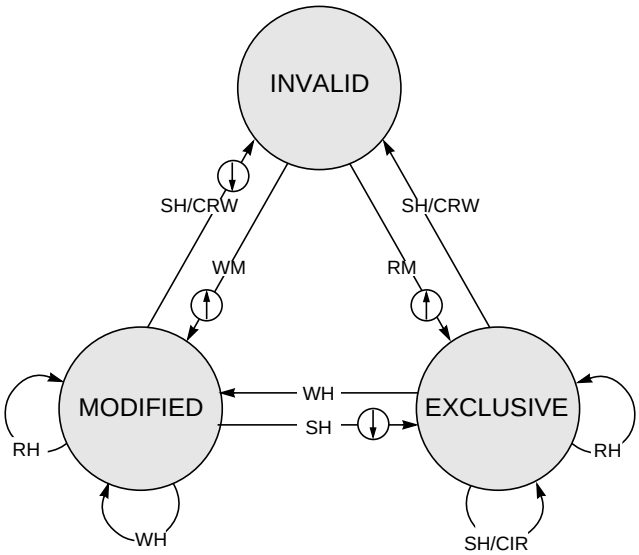

Figure 5-4. MEI Cache Coherency Protocol—State Diagram (WIM = 001)

5.6.3, Page 5-16 Replace the fourth sentence of the first paragraph with the following sentence:

The snoop is not given priority into the tags when the snoop coincides with a tag write (for example, validation after a cache block load).

5.6.4.1, Page 5-18 Replace the first bullet item with the following:

- Load or store to a caching-inhibited page (WIM = 0bX1X) and a cache hit occurs.

  Caching is inhibited for this page (I = 1)—Load or store operations to a caching-inhibited page that hit in the cache cause boundedly undefined results.

Section #/Page # **Changes**

5.6.7, Page 5-19 Replace the third row of Table 5-6 with the following:

Write-with-flush Write-with-flush-atomic	Write-with-flush and write-with-flush-atomic operations occur after the processor issues a store or **stwcx.** instruction, respectively. • If the addressed block is in the exclusive state, the address snoop forces the state of the addressed block to invalid. • If the addressed block is in the modified state, the address snoop causes $\overline{\text{ARTRY}}$ to be asserted and initiates a push of the modified block out of the cache and changes the state of the block to invalid. • The execution of an **stwcx.** instruction cancels the reservation associated with any address.

5.6.8, Page 5-20 Replace all four bullet items in Section 5.6.8 with the following information:

- Snoop hits to a block in the M state (flush or clean)

  This case is a normal snoop hit and will result in $\overline{\text{ARTRY}}$ being asserted if the snooped transaction was a “flush” or “clean” request. If the snooped transaction was a “kill” request, then $\overline{\text{ARTRY}}$ will not be asserted.

- Snoop attempt during the last $\overline{\text{TA}}$ of a cache line fill

  In no-$\overline{\text{DRTRY}}$ mode, during the cycle that the last $\overline{\text{TA}}$ is asserted to the 603 on a cache line fill, the tag is being written to its new state by the 603 and is not accessible. This will result in a collision being signaled by asserting $\overline{\text{ARTRY}}$. With $\overline{\text{DRTRY}}$ enabled, the cache tags are inaccessible to a snoop operation one cycle after the last $\overline{\text{TA}}$.

- Snoop hit after the first $\overline{\text{TA}}$ of a burst load operation

  After the first $\overline{\text{TA}}$ of a burst load operation, the data tags are committed to being written; snoop operations cannot be serviced until the load completes, thereby causing the assertion of $\overline{\text{ARTRY}}$.

- Snoop hits to line in the cast-out buffer

  The 603's cast-out buffer is kept coherent with main memory, and snoop operations that hit in the cast-out buffer will cause the assertion of $\overline{\text{ARTRY}}$.

- Snoop attempt during cycles that **dcbz** instruction or load or store operation is updating the tag

  During the execution of a **dcbz** instruction or during a load or store operation that requires a cache line cast-out, the cache tags will be inaccessible during the first and last cycle of the operation.

Section #/Page # **Changes**

- Snoop attempt during the cycle that a **dcbf** or **dcbst** instruction is updating the tag

  If the EA of a **dcbf** or **dcbst** instruction hits in the cache, the tag will be changed to its new state. During that clock, the tag is not accessible and snoop transactions during that cycle will cause the assertion of $\overline{\text{ARTRY}}$.

5.7.4, Page 5-23 Replace the third paragraph in Section 5.7.4 with the following:

This instruction is treated as a load from the addressed byte with respect to address translation and protection.

5.7.5, Page 5-23 Replace the last paragraph in Section 5.7.5 with the following:

The 603 treats this instruction as a load from the addressed byte with respect to address translation and protection.

6.1, Page 6-4 Table 6-2—Replace the instruction access section (row #2 on page 6-4) of Table 6-2 with the following:

Instruction access	00400	An instruction access exception is caused when an instruction fetch cannot be performed for any of the following reasons: • The effective address cannot be translated. That is, there is a page fault for this portion of the translation, so an instruction access exception must be taken to load the PTE (and possibly the page) into memory. • The fetch access is to an I/O controller interface segment. • The fetch access is to a no-execute segment. • The fetch access is to guarded storage and MSR[IR] = 1. • The fetch access violates memory protection. If the key bits (Ks and Kp) in the segment register and the PP bits in the PTE are set to prohibit read access, instructions cannot be fetched from this location.

6.3, Page 6-12 Replace the description for bit 0 in Table 6-7 with the following:

Reserved, but saved in SRR1 when an exception (other than TLB miss exception) occurs.

6.5.1.2, Page 6-18 Replace Table 6-11 with the following:

Table 6-11. Settings Caused by Hard Reset

Register	Setting	Register	Setting
GPRs	Unknown	SDR1	00000000
FPRs	Unknown	SRR0	00000000
FPSCR	00000000	SRR1	00000000
CR	All 0s	SPRGs	00000000
SRs	Unknown	EAR	00000000
MSR	00000040	PVR	0003000*n*
XER	00000000	BATs	Unknown

Section #/Page # **Changes**

Register	Setting	Register	Setting
TBU	00000000	HID0	00000000
TBL	00000000	DMISS and IMISS	All 0s
LR	00000000	DCMP and ICMP	All 0s
CTR	00000000	RPA	All 0s
DSISR	00000000	IABR	All 0s
DAR	00000000	TLBs	Unknown
DEC	FFFFFFFF	Cache	All cache blocks invalidated
Tag directory	All 0s. (However, LRU bits are initialized so each side of the cache has a unique LRU value.)		

6.5.3, Page 6-21 Change the last sentence of the third to last paragraph to the following:

In this case, the data address register (DAR) always points to a byte address in the first word of the offending page.

6.5.4, Page 6-23 Add the two following bullet items to the first paragraph:

- An attempt is made to fetch an instruction from no-execute storage.
- An attempt is made to fetch an instruction from guarded storage when MSR[IR] = 1.

Table 6-14—Replace the SRR1 section (row #2) of Table 6-14 with the following:

SRR1	0–2 Cleared 3 Set if the fetch access was to an I/O controller interface segment (SR[T] = 1) or to a no-execution segment, or if the fetch access was to guarded storage when MSR[IR] = 1; otherwise cleared. 4 Set if the exception is due to a protection violation; otherwise cleared. 5–15 Cleared 16–31 Loaded from bits 16–31 of the MSR Note that on other PowerPC processors, bit 1 may be set if the translation of an attempted access is not found in the primary hash table entry group (HTEG), or in the rehashed secondary HTEG, or in the range of an IBAT register. Also on other PowerPC processors, bit 10 may be set if a page table search fails to find a translation for the effective address.

6.5.6, Page 6-25 Delete the 4th bullet item from the list of alignment conditions.

6.5.6.1, Page 6-27 Delete the following information from Table 6-17:

0	0	Direct translation access

Section #/Page #	Changes
6.5.6.1.1, Page 6-27	Delete Section 6.5.6.1.1.
6.5.6.1.2, Page 6-27	Replace the second bullet item with the following: **lwarx** or **stwcx.** instructions that map into an I/O controller interface segment, or cross a segment boundary cause a data access exception.
6.5.6.1.3, Page 6-27	The following changes are applicable for Section 6.5.6.13: • Replace the first sentence with the following: A page-address translation access occurs when MSR[DR] is set, SR[T] is cleared and there is not a match in the BAT. • Delete the first list item under the first bullet item. • Replace the second bullet item with the following: The **dcbz** instruction causes an alignment exception if the access is to a page or block with the W (write-through) or I (cache-inhibit) bit set in either the TLB or BAT, respectively.
7.2.3, Page 7-9	For clarity, reword the last two sentences of the fourth paragraph (paragraph starting, “For memory accesses.....”) to the following: However, if the page address translation misses in an on-chip TLB, the MMU causes a search of the page tables in memory (using the virtual address information and a hashing function) to locate the required physical address. When this occurs, the 603 vectors to exception handlers that search the page tables with software.
7.2.4, Page 7-11	For completeness, edit the first sentence as follows: In addition to the translation of effective addresses to physical addresses, the MMUs provide access protection of supervisor areas from user access and can designate areas of memory as read-only, as well as no-execute, or guarded.
7.4.1, Page 7-23	The last sentence of the last paragraph in this section should begin as follows: Thus, the valid bits of the BAT array entries must be explicitly cleared by the system software....
7.4.2, Page 7-23	Replace the second sentence in the second paragraph as follows: Also, a matching BAT array entry always takes precedence over any segment descriptor translation, independent of the setting of the SR[T] bit, and the segment descriptor information is completely ignored.
7.5.3, Page 7-38	Replace the second bullet on the page with the following: • For TLB misses, when a table search operation is in progress to locate a PTE, the R and C bits are updated (set, if required) to reflect the status of the page based on this access.

Section #/Page # **Changes**

7.5.3.3, Page 7-41 Replace the last two rows in Table 7-15 with the following:

13	**dcbt**, **dcbtst**, **dcbst**, or **dcbf** instruction	maybe	yes	no	no
13	**icbi**	maybe	no	no	no
14	**dcbi** instruction	maybe[1]	yes	maybe[1]	yes

[1] If C is set, R is guaranteed to also be set.
[2] This includes the case in which the instruction was speculatively fetched and R was not set (does not apply for 603).

7.5.5, Page 7-47 Replace the second sentence in the last paragraph with the following:

Thus TLB entries must be explicitly cleared by the system software (with the **tlbie** instruction) before the valid entries are loaded and address translation is enabled.

7.6.3.1, Page 7-68 Replace the last sentence in the third bullet as follows:

The software can change this value, effectively overriding the replacement algorithm.

7.6.3.1.1, Page 7-68 Replace the fourth sentence in the section with the following sentence:

The contents are used by the processor when calculating the values of HASH1 and HASH2, and by the **tlbld** and **tlbli** instructions when loading a new TLB entry.

7.6.3.1.2, Page 7-69 Replace the last sentence in the first paragraph with the following:

Upon execution of a **tlbld** or **tlbli** instruction, the contents of the DCMP or ICMP register is loaded into the first word of the selected TLB entry.

7.6.3.1.4, Page 7-70 Replace the last sentence in the first paragraph with the following sentence:

Upon execution of a **tlbld** or **tlbli** instruction, the contents of the DCMP or ICMP register are loaded into the first word of the selected TLB entry.

7.6.3.2.1, Page 7-72 The left branch of the first decision bubble in Figure 7-34 should have the condition of “temp ← compare_value.”

7.6.3.2.2, Page 7-75 There are seven errors in the code sequences shown in this section (seven lines of code that need to be replaced with corrections). The sections of code that contain the errors are shown beginning with the previous label, and the corrected instruction is shown in bold.

Section #/Page # Changes

Corrections #1 and #2 on page 7-77:

```
im1:    lwzu    r1, 8(r2)     # get next pte
        cmp     c0, r1, r3    # see if found pte
        bdneq   im1           # dec count br if cmp ne and if count not zero
        bne     instrSecHash# if not found set up second hash or exit
        l       r1, +4(r2)    # load tlb entry lower-word
        andi.   r3, r1, 8     # check G-bit
        bne     doIAEp        # if guarded, take an IAE
        mtctr   r0            # restore counter
        mfspr   r0, iMiss     # get the miss address for the tlbli
        mfspr   r3, srr1      # get the saved cr0 bits
        mtcrf   0x80, r3      # restore CR0
        mtspr   rpa, r1       # set the pte
        ori     r1, r1, 0x100# set reference bit
        srw     r1, r1, 8     # get byte 7 of pte
        tlbli   r0            # load the itlb
        stb     r1, +6(r2)    # update page table
        rfi                   # return to executing program
```

Correction #3 on page 7-78:

```
doIAE:
        mfspr   r3, srr1      # get srr1
        andi.   r2, r3, 0xffff# clean srr1
        addis   r2, r2, 0x4000# or in srr1<1> = 1 to flag pte not found
        mtctr   r0            # restore counter
iae1    mtspr   srr1,r2       # set srr1
        mfmsr   r0            # get msr
```

Corrections #4 and #5 on page 7-79:

```
dm1:    lwzu    r1, 8(r2)     # get next pte
        cmp     c0, r1, r3    # see if found pte
        bdneq   dm1           # dec count br if cmp ne and if count not zero
        bne     dataSecHash# if not found set up second hash or exit
        l       r1, +4(r2)    # load tlb entry lower-word
        mtctr   r0            # restore counter
        mfspr   r0, dMiss     # get the miss address for the tlbld
        mfspr   r3, srr1      # get the saved cr0 bits
        mtcrf   0x80, r3      # restore CR0
        mtspr   rpa, r1       # set the pte
        ori     r1, r1, 0x100# set reference bit
        srw     r1, r1, 8     # get byte 7 of pte
        tlbld   r0            # load the dtlb
        stb     r1, +6(r2)    # update page table
        rfi                   # return to executing program
```

Section #/Page # **Changes**

Correction #6 on page 7-80:

```
ceq1:   lwzu    r1, 8(r2)     # get next pte
        cmp     c0, r1, r3    # see if found pte
        bdneq   ceq1          # dec count br if cmp ne and if count not zero
        bne     cEq0SecHash# if not found set up second hash or exit
        l       r1, +4(r2)    # load tlb entry lower-word
        andi.   r3,r1,0x80    # check the C-bit
```

Correction #7 on page 7-81:

```
dae2:   mtspr   dar, r1       # put in dar
        mfmsr   r0            # get msr
        xoris   r0, r0, 0x2   # flip the msr<tgpr> bit
        mtcrf   0x80, r3      # restore CR0
        mtmsr   r0            # flip back to the native gprs
        b       vec300        # branch to data access exception
```

8.6, Page 8-19 Delete the second paragraph in Section 8.6.

9.2.3.3, Page 9-9 Add the following sentence after the third sentence in the first paragraph:

The $\overline{\text{APE}}$ signal will not be asserted if address parity checking is disabled (HID0[EBA] cleared to 0).

9.2.4.1.2, Page 9-11 Change the first sentence on page 9-11 to read as follows:

Table 9-1 describes the transfer encodings for a 603 bus master.

9.2.9.3, Page 9-25 Replace the first sentence in “State Meaning” with the following:

Asserted—The 603 initiates a machine check interrupt operation if MSR[ME] and HID0[EMCP] are set; if MSR[ME] is cleared and HID0[EMCP] is set, the 603 must terminate operation by internally gating off all clocks, and releasing all outputs (except $\overline{\text{CHECKSTOP}}$) to the high impedance state.

9.2.13, Page 9-32 Add the following new section at the end of Chapter 9:

9.2.13 Power and Ground Signals

The 603 provides the following connections for power and ground:

- VDD and OVDD—The VDD and OVDD signals provide the connection for the supply voltage. On the 603, there is no electrical distinction between the VDD and the OVDD signals.
- AVDD—The AVDD power signal provides power to the clock generation phase-lock loop. See the *PowerPC 603 RISC Microprocessor Hardware Specifications* for information about how to use this signal.

Section #/Page # **Changes**

- GND and OGND—The GND and OGND signals provide the connection for grounding the 603. On the 603, there is no electrical distinction between the GND and OGND signals.

10.1.2, Page 10-4 Replace fourth paragraph with the following:

Note that the Synchronize (**sync**) instruction can be used to enforce strong ordering.

10.3.2.5, Page 10-19 Replace Table 10-7 with the following information. The corrected items are identified in bold.

Table 10-7. Misaligned 32-Bit Data Bus Transfer (Four-Byte Examples)

Transfer Size (Four Bytes)	TSIZ(0–2)	A29–A31	Data Bus Byte Lanes							
			0	1	2	3	4	5	6	7
Aligned	1 0 0	0 0 0	A	A	A	A	x	x	x	x
Misaligned—first access	0 1 1	0 0 1	—	A	A	A	x	x	x	x
second access	0 0 1	1 0 0	A	—	—	—	x	x	x	x
Misaligned—first access	0 1 0	0 1 0	—	—	A	**A**	x	x	x	x
second access	**0 1 0**	1 0 0	A	A	—	—	x	x	x	x
Misaligned—first access	0 0 1	0 1 1	—	—	—	A	x	x	x	x
second access	0 1 1	1 0 0	A	A	A	—	x	x	x	x
Aligned	1 0 0	1 0 0	**A**	**A**	**A**	**A**	x	x	x	x
Misaligned—first access	0 1 1	1 0 1	—	A	A	A	x	x	x	x
second access	0 0 1	0 0 0	A	—	—	—	x	x	x	x
Misaligned—first access	0 1 0	1 1 0	—	—	A	A	x	x	x	x
second access	0 1 0	0 0 0	A	A	—	—	x	x	x	x
Misaligned—first access	0 0 1	1 1 1	—	—	—	A	x	x	x	x
second access	0 1 1	0 0 0	A	A	A	—	x	x	x	x

A: Byte lane used
—: Byte lane not used
x: Byte lane not used in 32-bit bus mode

Section #/Page # **Changes**

10.3.3, Page 10-21 Add the following sentence to the last paragraph in the section:

Note that a nonclocked bus arbiter may detect the assertion of address bus request by the bus master that asserted $\overline{\text{ARTRY}}$, and return a qualified bus grant one cycle earlier than shown in Figure 10-7. Replace Figure 10-7 with the following:

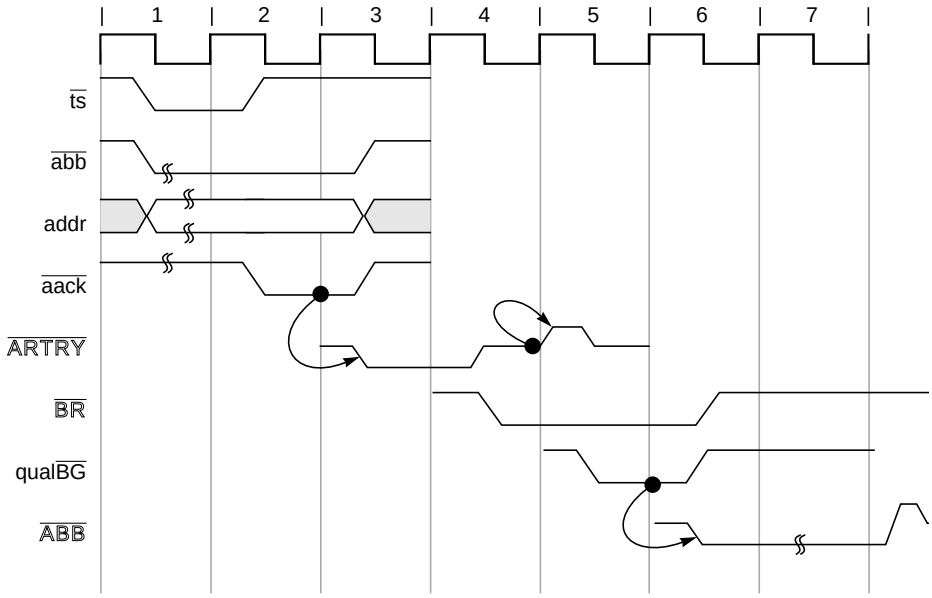

Figure 10-7. Snooped Address Cycle with $\overline{\text{ARTRY}}$

10.4.3, Page 10-24 Replace first sentence of the second paragraph with the following:

The 603 transfers data in either single- or four-beat burst transfers when configured with a 64-bit data bus; when configured with a 32-bit data bus, the 603 performs one-, two-, and eight-beat data transfers.

Section #/Page # **Changes**

10.4.4.1, Page 10-26 Replace Figure 10-9 with the following; $\overline{\text{DRTRY}}$ signal is deasserted in clock cycle 2 as shown in the following figure.

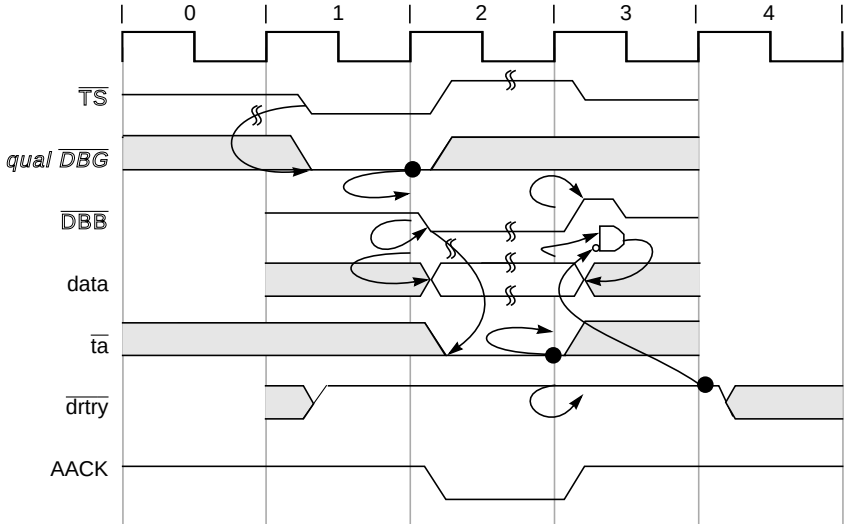

Figure 10-9. Normal Single-Beat Read Termination

10.4.4.1, Page 10-28 Replace Figure 10-12 with the following; $\overline{\text{TA}}$ assertion is extended through bus clock cycles 4 and 5 as shown in the following figure.

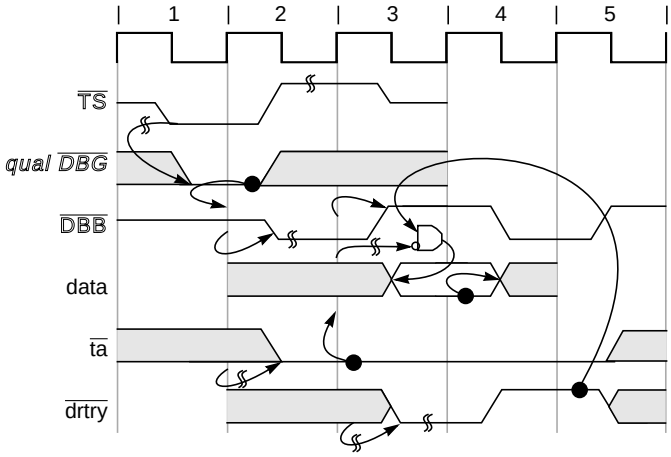

Figure 10-12. Termination with $\overline{\text{DRTRY}}$

Section #/Page #	Changes
10.4.4.2, Page 10-29	Replace Figure 10-13 with the following—the 603 user's manual shows the $\overline{TA}$ signal in Figure 10-13 as don't care in cycle 7; it should be asserted as shown in the following figure.

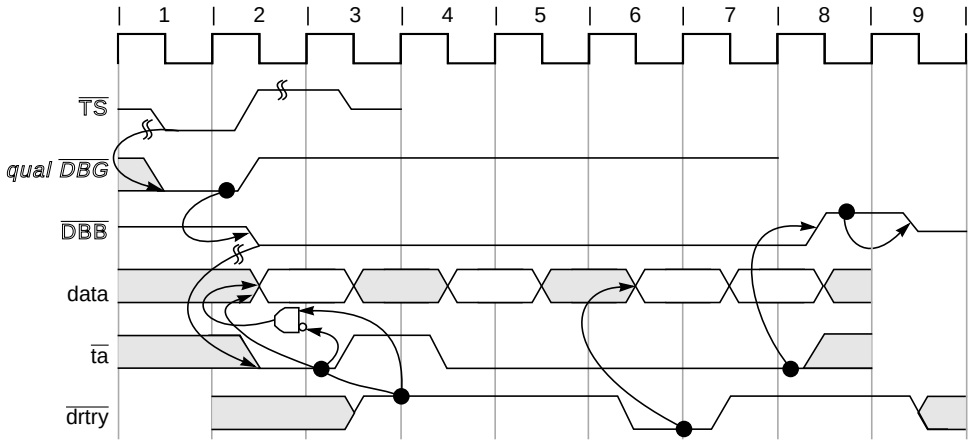

Figure 10-13. Read Burst with $\overline{TA}$ Wait States and $\overline{DRTRY}$

Section #/Page # **Changes**

10.5, Page 10-34 Replace Figure 10-17 with the following figure (the $\overline{TA}$ signal is asserted in clock cycle 12):

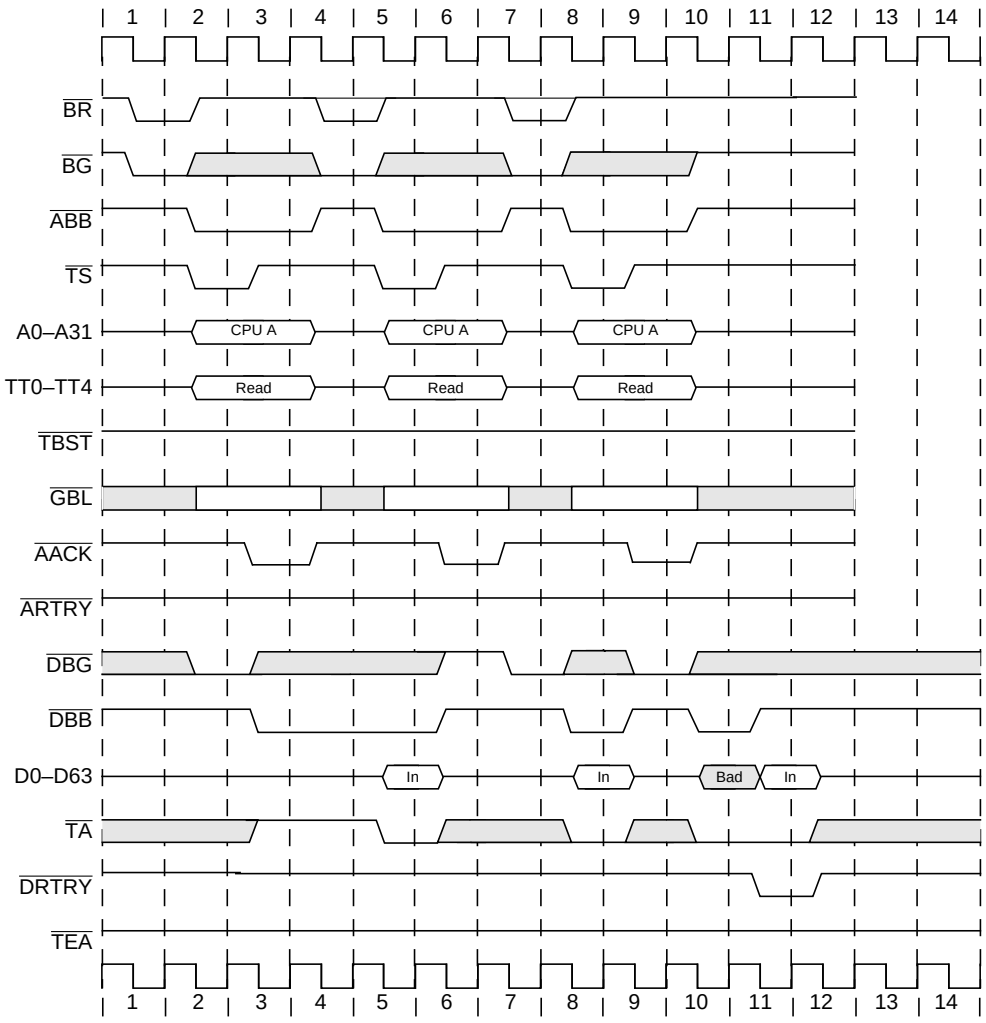

Figure 10-17. Single-Beat Reads Showing Data-Delay Controls

10.7.1, Page 10-48 Add the following sentence at the end of the second paragraph:

All cache-inhibited instruction fetches are performed as word (single-beat) operations.

10.7.1, Page 10-50 Delete the last paragraph in Section 10.7.1.

Section #/Page #	Changes

10.7.1, Page 10-50 Replace Figure 10-28 with the following figure; $\overline{\text{DBB}}$ signal assertion is extended by one bus clock cycle.

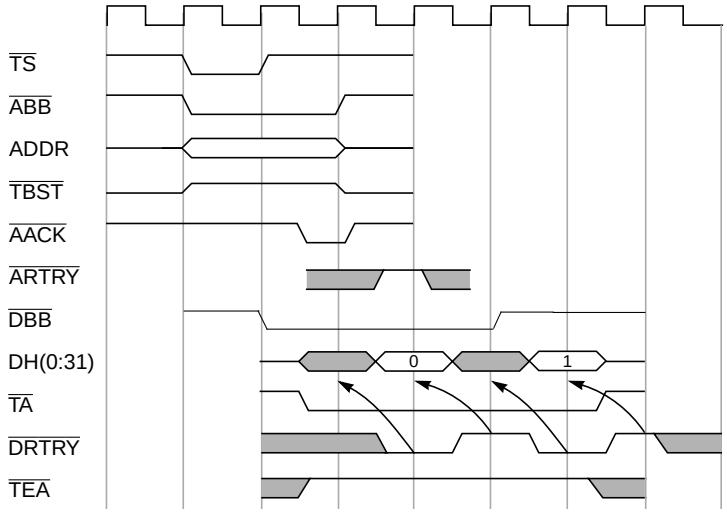

Figure 10-28. 32-Bit Data Bus Transfer (Two-Beat Burst with $\overline{\text{DRTRY}}$)

10.7.2, Page 10-51 Delete last two sentences in Section 10.7.2.

10.7.3, Page 10-51 Replace the last paragraph in Section 10.7.3 with the following paragraph:

The 603 selects either full-pinout or reduced-pinout mode at startup by sampling the state of the $\overline{\text{QACK}}$ signal at the negation of $\overline{\text{HRESET}}$. If the $\overline{\text{QACK}}$ signal is asserted at the negation of $\overline{\text{HRESET}}$, full-pinout mode is selected by the 603. If $\overline{\text{QACK}}$ is negated at the negation of $\overline{\text{HRESET}}$, reduced-pinout mode is selected.

10.8.1, Page 10-52 Replace Section 10.8.1 with the following:

The external interrupt input signals ($\overline{\text{INT}}$, $\overline{\text{SMI}}$ and $\overline{\text{MCP}}$) of the 603 eventually force the processor to take the external interrupt vector, or the system management interrupt vector if the MSR[EE] is set, or the machine check interrupt if the MSR[ME] bit and the HID0[EMCP] bit are set.

10.10.1, Page 10-54 Table 10-12—Delete the pin numbers under the column heading, “Package Pin” from Table 10-12.

Section #/Page #	Changes
Chapter 11	Note that all load and store instructions are executed by the load/store unit, and not by the integer and floating point units. Add the following warning against the use of **lmw**, **lswx**, **lswi**, **stmw**, **stswi**, and **stswx** instructions.
lmw, lswx, lswi	In some implementations, this instruction is likely to have greater latency and take longer to execute, perhaps much longer, than a sequence of indvidual load instructions that produce the same results.
stmw, stswi, stswx	In some implementations, this instruction is likely to have greater latency and take longer to execute, perhaps much longer, than a sequence of individual store instructions that produce the same results.
andi., Page 11-19	Replace the pseudocode description of instruction operation with the following: rA ← (rS) & ((16)0 || UIMM) Replace the first paragraph under the pseudocode description with the following: The contents of rS are ANDed with 0x0000 || UIMM and the result is placed into rA.
dcbz, Page 11-43	Replace the third bullet item with the following: If the page containing the byte addressed by the EA is in caching-inhibited mode, write-through mode, or if the data cache is disabled, the alignment exception handler is invoked.
eieio, Page 11-48	Replace the first four paragraphs with the following: The **eieio** instruction, as described by the PowerPC architecture, provides an ordering function for the effects of load and store instructions executed by a processor. Executing an **eieio** instruction ensures that all applicable memory accesses previously initiated by the processor are complete with respect to main memory before any memory accesses subsequently initiated by the processor access main memory. The 603 treats the **eieio** instruction as a no-op.
mfspr, Page 11-120	Delete the second paragraph and the three associated bullet items. Replace the third and fourth paragraphs with the following: SPR[0] = 1 if and only if reading the register is supervisor-level. Execution of this instruction specifying a defined and supervisor-level register when MSR[PR] = 1 will result in a supervisor-level instruction exception.

Section #/Page #	Changes
	If MSR[PR] = 1, the only effect of executing an instruction with an SPR number that is not shown in Table 11-6 and has SPR[0] = 1 is to cause a supervisor-level instruction type program exception. In cases where MSR[PR] = 0 or SPR[0] = 0, if the SPR field contains any value that is not shown in Table 11-6, the destination register specified by **r**D will be cleared to 0.
mtspr, Page 11-132	Delete the second paragraph and the three associated bullet items. Replace the fifth paragraph with the following paragraph: If MSR[PR] = 1, the only effect of executing an instruction with an SPR number that is not shown in Table 11-8 and has SPR[0] = 1 is to cause a supervisor-level instruction type program exception. In cases where MSR[PR] = 0 or SPR[0] = 0, if the SPR field contains any value that is not shown in Table 11-8, the **mtspr** instruction executes as a no-op.
sc, Page 11-152	Replace the second sentence of the third paragraph with the following: Bits 0, 5–9, and 16–31 of the MSR are placed into the corresponding bits of SRR1, and bits 0–15 of SRR1 are cleared to 0.
srw*x*, Page 11-156	Edit the first sentence below the pseudocode description to read as follows: If **r**B[26] = 0, the contents of **r**S are shifted right the number of bits specified by **r**B[27–31].
stfiwx, Page 11-165	Replace the second paragraph below the pseudocode description to the following: The contents of **fr**S[32–63] are stored, without conversion, into the word in memory addressed by the EA.
tlbie, Page 11-191:	The end of the first paragraph should read as follows: To invalidate all entries within both TLBs, 32 **tlbie** instructions must be executed, incrementing this field by one each time. Block address translation for EA, if any, is ignored.
tlbsync, Page 11-194:	The references to the $\overline{\text{TLBISYNC}}$ signal in the second paragraph do not show this signal as active low. Both the references should be shown as $\overline{\text{TLBISYNC}}$.
E.3, Page E-9	Edit the last sentence in Section E.3 to read as follows: Note that care must be taken in using **fsel** if IEEE compatibility is required or if the values being tested can be NaNs or infinities; see Section E.3.4, “Notes.”

Section #/Page #	Changes

E.5, Page E-12 Replace the conversion descriptions with the following:

No Denormalization Required (includes Zero/Infinity/NaN)

```
if frS[1–11] > 896 or frS[1–63] = 0 then
  WORD[0–1] ← frS[0–1]
  WORD[2–31] ← frS[5–34]
```

Denormalization Required

```
if 874 ≤ frS[1–11] ≤ 896 then
  sign ← frS[0]
  exp ← frS[1–11] – 1023
  frac ← 0b1 || frS[12–63]
  Denormalize operand
        Do while exp < –126
          frac ← 0b0 || frac[0–62]
          exp ← exp + 1
        End
  WORD[0] ← sign
  WORD[1–8] ← 0x00
  WORD[9–31] ← frac[1–23]
else WORD ← undefined
```

Appendix H Replace the contents of Appendix H with the following:

The most recent revision of boundary-scan descriptor language (BSDL) information required to perform testing as described by the IEEE 1149.1 specification is available via FTP and World-Wide Web. The URL for FTP access is ftp://freeware.aus.sps.mot.com/pub/bsdl, and the URL for WWW access is http://freeware.aus.sps.mot.com.

Motorola Literature Distribution Centers:
USA/EUROPE: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036; Tel.: 1-800-441-2447
MFAX: RMFAX0@email.sps.mot.com; TOUCHTONE (602) 244-6609
INTERNET: http://Design-NET.com

JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, Toshikatsu Otsuki,
6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan; Tel.: 03-3521-8315

HONG KONG: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,
51 Ting Kok Road, Tai Po, N.T., Hong Kong; Tel.: 852-26629298

Technical Information: Motorola Inc. Semiconductor Products Sector Technical Responsiveness Center; (800) 521-6274.
Document Comments: FAX (512) 891-2638, Attn: RISC Applications Engineering.

IBM Microelectronics Division:
USA: IBM Microelectronics Division, Mail Stop A25/862-1, PowerPC Marketing, 1000 River Street, Essex Junction, VT 05452-4299;
Tel.: (800) PowerPC [(800) 769-3772]; FAX (800) POWERfax [(800) 769-3732].
EUROPE: IBM Microelectronics Division, PowerPC Marketing, Dept. 1045, 224 Boulevard J.F. Kennedy,
91105 Corbeil-Essonnes CEDEX, France; Tel. (33) 1-60-88 5167; FAX (33) 1-60-88 4920.
JAPAN: IBM Microelectronics Division, PowerPC Marketing, Dept., R0260, 800 Ichimiyake, Yasu-cho, Yasu-gun, Shinga-ken,
Japan 520-23; Tel. (81) 775-87-4745; FAX (81) 775-87-4735.

IBM MOTOROLA