SA14-2056-01
(IBM Order Number)

MPC604UMAD/AD
(Motorola Order Number)

9/96
REV 1

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

PowerPC™

Addendum to
PowerPC 604™ RISC Microprocessor User's Manual:

PowerPC 604e™ Microprocessor Supplement and User's Manual Errata

This addendum to the *PowerPC 604 RISC Microprocessor User's Manual* is in two parts:

- Part 1: The PowerPC 604e Microprocessor Supplement provides an overview of the 604e, with detailed information about features that differ from those of the 604 described in the user's manual. This information is presented in the same order as in the user's manual.

  Note that an index is provided for Part 1 of this document.
- Part 2: Errata to *PowerPC 604 RISC Microprocessor User's Manual* contains corrections to the user's manual.

This document is designed to be used in conjunction with the user's manual and *PowerPC Microprocessor Family: The Programming Environments*, referred to as *The Programming Environments Manual*.

In this document, the terms '604', '604e', '603', and '603e' are used as abbreviations for 'PowerPC 604 microprocessor', 'PowerPC 604e microprocessor', 'PowerPC 603™ microprocessor', and 'PowerPC 603e™ microprocessor', respectively. The PowerPC 604e microprocessors are available from IBM as PPC604e and from Motorola as MPC604e.

To locate any published errata or updates for this document, refer to the website at http://www.mot.com/powerpc/ or at http://www.chips.ibm.com/products/ppc.

The PowerPC name, PowerPC logotype, PowerPC 604e, PowerPC 604, PowerPC 603e, and PowerPC 603 are trademarks of International Business Machines Corporation used by Motorola under license from International Business Machines Corporation.

This document contains information on a new product under development. Specifications and information herein are subject to change without notice.

© Motorola Inc. 1996
Portions hereof © International Business Machines Corp. 1991–1996. All rights reserved.

IBM

MOTOROLA

Freescale Semiconductor, Inc.

Part 1: The PowerPC 604e Microprocessor Supplement

This part of the document provides an overview of features of the 604e and provides detailed information about the features that are either not implemented on or are implemented differently from the 604.

1.1 Overview

This section describes features of the 604e, provides a block diagram showing the major functional units, and describes briefly how those units interact.

The 604e is an implementation of the PowerPC™ family of reduced instruction set computer (RISC) microprocessors. The 604e implements the PowerPC architecture as it is specified for 32-bit addressing, which provides 32-bit effective (logical) addresses, integer data types of 8, 16, and 32 bits, and floating-point data types of 32 and 64 bits (single- and double-precision, respectively). For 64-bit PowerPC implementations, the PowerPC architecture provides additional 64-bit integer data types, 64-bit addressing, and related features.

The 604e is a superscalar processor capable of issuing four instructions simultaneously. As many as seven instructions can finish execution in parallel. The 604e has seven execution units that can operate in parallel:

- Floating-point unit (FPU)
- Branch processing unit (BPU)
- Condition register unit (CRU)
- Load/store unit (LSU)
- Three integer units (IUs):
  - Two single-cycle integer units (SCIUs)
  - One multiple-cycle integer unit (MCIU)

This parallel design, combined with the PowerPC architecture's specification of uniform instructions that allows for rapid execution times, yields high efficiency and throughput. The 604e's rename buffers, reservation stations, dynamic branch prediction, and completion unit increase instruction throughput, guarantee in-order completion, and ensure a precise exception model. (Note that the PowerPC architecture specification refers to all exceptions as interrupts.)

The 604e has separate memory management units (MMUs) and separate 32-Kbyte on-chip caches for instructions and data. The 604e implements two 128-entry, two-way set associative translation lookaside buffers (TLBs), one for instructions and one for data, and provides support for demand-paged virtual memory address translation and variable-sized block translation. The TLBs and the cache use least-recently used (LRU) replacement algorithms.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

The 604e has a 64-bit external data bus and a 32-bit address bus. The 604e interface protocol allows multiple masters to compete for system resources through a central external arbiter. Additionally, on-chip snooping logic maintains data cache coherency for multiprocessor applications. The 604e supports single-beat and burst data transfers for memory accesses and memory-mapped I/O accesses.

The 604e uses an advanced, 2.5-V CMOS process technology and is fully compatible with TTL devices.

1.2 PowerPC 604e Microprocessor Features

This section summarizes features of the 604e's implementation of the PowerPC architecture.

Figure 1 provides a block diagram showing features of the 604e. Note that this is a conceptual diagram that shows basic features and does not attempt to show how these features are physically implemented on the chip.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

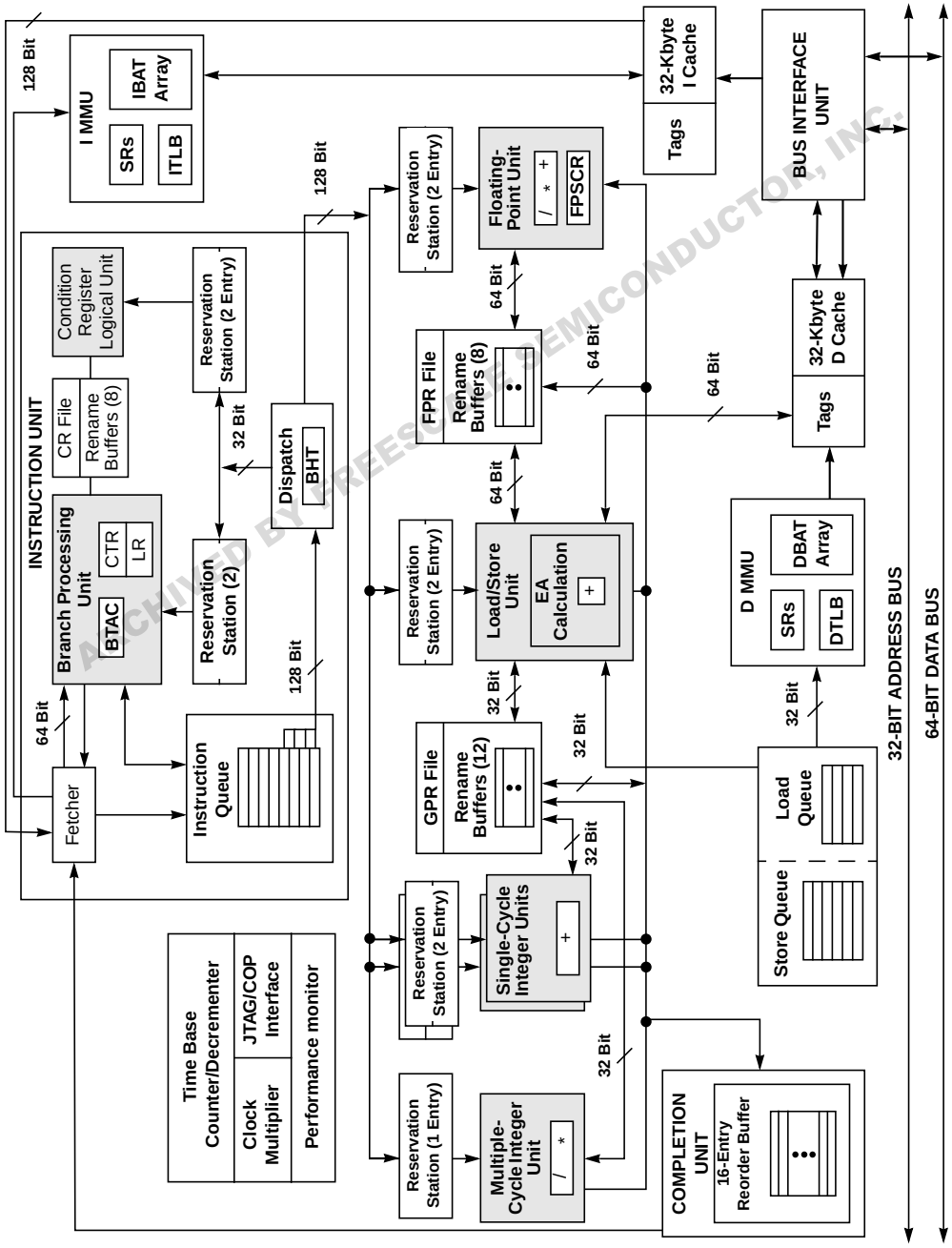


Figure 1. PowerPC 604e Microprocessor Block Diagram

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.
For More Information On This Product,
Go to: www.freescale.com

1.2.1 New Features of the PowerPC 604e Microprocessor

Features of the 604e that are not implemented in the 604 are as follows:

- Additional special-purpose registers
  - Hardware implementation-dependent register 1 (HID1) provides four read-only PLL_CFG bits for indicating the processor/bus clock ratio.
  - Three additional registers to support the performance monitor—MMCR1 is a second control register that includes bits to support the use of two additional counter registers, PMC3 and PMC4.
- Instruction execution
  - Separate execution units for branch and condition register (CR) instructions. The 604e implements a condition register unit (CRU) that executes condition register logical instructions that were executed in the 604's BPU. The CRU makes it possible for branch instructions to execute and resolve before preceding CR logical instructions. The 604e can dispatch one CR logical or branch instruction per cycle, but it can execute both branch and CR logical instructions at the same time.
  - Branch correction in decode stage. Branch correction in the decode stage can now predict branches whose target is taken from the count or link registers if no updates of the count and link register are pending. This saves at least one cycle on branch correction when the Move to Special-Purpose Register (**mtspr**) instruction can be sufficiently separated from the branch that uses the SPR as a target address.
  - Ability to disable the branch target address cache (BTAC)—HID0[30] has been defined to allow the BTAC to be disabled. When HID0[30] is set, the BTAC contents are invalidated and the BTAC behaves as if it were empty. New entries cannot be added until the BTAC is enabled.
- Enhancements to cache implementation
  - 32-Kbyte, physically addressed, split data and instruction caches. Like the 604, both caches are four-way set associative; however, each cache has twice as many sets, logically separated into 128 sets of odd lines and 128 sets of even lines.
  - Data cache line-fill buffer forwarding. In the 604, only the critical double word of a burst operation was made available to the requesting unit at the time it was burst into the line-fill buffer. Subsequent data was unavailable until the cache block was filled. In the 604e, subsequent data is also made available as it arrives in the line-fill buffer.
  - Additional cache copy-back buffers. The 604e implements three copy-back write buffers (increased from one in the 604). Having multiple copy-back buffers provides the ability for certain instructions to take fuller advantage of the pipelined system bus to provide more efficient handling of cache copy-back, block invalidate operations caused by the Data Cache Block Flush (**dcbf**)

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

instruction, and cache block clean operations resulting from the Data Cache Block Store (**dcbst**) instruction.

— Coherency support for instruction fetching. Instruction fetching coherency is controlled by HID0[23]. In the default mode, HID0[23] is 0, $\overline{\text{GBL}}$ is not asserted for instruction accesses, as is the case with the 604. If the bit is set, and instruction translation is enabled (MSR[IR] = 1), the $\overline{\text{GBL}}$ signal is set to reflect the M bit for this page or block. If instruction translation is disabled (MSR[IR] = 0), the $\overline{\text{GBL}}$ signal is asserted for instruction fetches.

- System interface operation
  - — The 604e has the same signal configuration as the 604; however, on the 604e Vdd and AVdd must be connected to 2.5 Vdc and OVdd must be connected to 3.3 Vdc. The 604e uses split voltage planes, and for replacement compatibility, 604/604e designs should provide both 2.5-V and 3.3-V planes and the ability to connect those two planes together and disable the 2.5-V plane for operation with a 604.
  - — Support for additional processor/bus clock ratios (7:2, 5:2, and 4:1). Configuration of the processor/bus clock ratios is displayed through a new 604e-specific register, HID1. Note that although this register is not defined by the PowerPC architecture, it is consistent with implementation-specific registers implemented on some other processors.
  - — To support the changes in the clocking configuration, different precharge timings for the $\overline{\text{ABB}}$, $\overline{\text{DBB}}$, $\overline{\text{ARTRY}}$, and $\overline{\text{SHD}}$ signals are implemented internally by the processor. Selectable precharge timings for $\overline{\text{ARTRY}}$ and $\overline{\text{SHD}}$ can be disabled by setting HID0[7]. Precharge timings are provided in the 604e hardware specifications.
  - — No-$\overline{\text{DRTRY}}$ mode. In addition to the normal and data streaming modes implemented on the 604, a no-$\overline{\text{DRTRY}}$ mode is implemented on the 604e that improves performance on read operations for systems that do not use the $\overline{\text{DRTRY}}$ signal. No-$\overline{\text{DRTRY}}$ mode makes read data available to the processor one bus clock cycle sooner than in normal mode. In no-$\overline{\text{DRTRY}}$ mode, the $\overline{\text{DRTRY}}$ signal is no longer sampled as part of a qualified bus grant.
  - — The VOLTDETGND output signal is implemented only on BGA packages as an indicator of the core voltage.
- Full hardware support for little-endian accesses. Little-endian accesses take alignment exceptions for only the same set of causes as big-endian accesses. Accesses that cross a word boundary require two accesses with the lower-addressed word accessed first.
- Additional events that can be tracked by the performance monitor.

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

For More Information On This Product,
Go to: www.freescale.com

1.2.2 Overview of the PowerPC 604e Microprocessor Features

Major features of the 604e are as follows:

- High-performance, superscalar microprocessor
  - — As many as four instructions can be issued per clock
  - — As many as seven instructions can be executing per clock (including three integer instructions)
  - — Single-clock-cycle execution for most instructions
- Seven independent execution units and two register files
  - — BPU featuring dynamic branch prediction
    - – Two-entry reservation station
    - – Out-of-order execution through two branches
    - – Shares dispatch bus with CRU
    - – 64-entry fully-associative branch target address cache (BTAC). In the 604e, the BTAC can be disabled and invalidated.
    - – 512-entry branch history table (BHT) with two bits per entry for four levels of prediction—not-taken, strongly not-taken, taken, strongly taken
  - — Condition register unit (CRU)
    - – Two-entry reservation station
    - – Shares dispatch bus with BPU
  - — Two single-cycle IUs (SCIUs) and one multiple-cycle IU (MCIU)
    - – Instructions that execute in the SCIU take one cycle to execute; most instructions that execute in the MCIU take multiple cycles to execute.
    - – Each SCIU has a two-entry reservation station to minimize stalls
    - – The MCIU has a single-entry reservation station and provides early exit (three cycles) for 16- x 32-bit and overflow operations.
    - – Thirty-two GPRs for integer operands
  - — Three-stage floating-point unit (FPU)
    - – Fully IEEE 754-1985-compliant FPU for both single- and double-precision operations
    - – Supports non-IEEE mode for time-critical operations
    - – Fully pipelined, single-pass double-precision design
    - – Hardware support for denormalized numbers
    - – Two-entry reservation station to minimize stalls
    - – Thirty-two 64-bit FPRs for single- or double-precision operands

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

- Load/store unit (LSU)
  - Two-entry reservation station to minimize stalls
  - Single-cycle, pipelined cache access
  - Dedicated adder performs EA calculations
  - Performs alignment and precision conversion for floating-point data
  - Performs alignment and sign extension for integer data
  - Four-entry finish load queue (FLQ) provides load miss buffering
  - Six-entry store queue
  - Supports both big- and little-endian modes

- Rename buffers
  - Twelve GPR rename buffers
  - Eight FPR rename buffers
  - Eight condition register (CR) rename buffers
- Completion unit
  - Retires an instruction from the 16-entry reorder buffer when all instructions ahead of it have been completed and the instruction has finished execution.
  - Guarantees sequential programming model (precise exception model)
  - Monitors all dispatched instructions and retires them in order
  - Tracks unresolved branches and flushes executed, dispatched, and fetched instructions if branch is mispredicted
  - Retires as many as four instructions per clock
- Separate on-chip instruction and data caches (Harvard architecture)
  - 32-Kbyte, four-way set-associative instruction and data caches
  - LRU replacement algorithm
  - 32-byte (eight-word) cache block size
  - Physically indexed/physical tags. (Note that the PowerPC architecture refers to physical address space as real address space.)
  - Cache write-back or write-through operation programmable on a per page or per block basis
  - Instruction cache can provide four instructions per clock; data cache can provide two words per clock.
  - Caches can be disabled in software.
  - Caches can be locked.
  - Parity checking performed on both caches

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

  - — Data cache coherency (MESI) maintained in hardware
  - — Secondary data cache support provided
  - — Instruction cache coherency optionally maintained in hardware
  - — Data cache line-fill buffer forwarding. In the 604, only the critical double word of the cache block was made available to the requesting unit at the time it was burst into the line-fill buffer; subsequent data was unavailable until the cache block was filled. In the 604e, subsequent data is also made available as it arrives in the line-fill buffer.
- Separate memory management units (MMUs) for instructions and data
  - — Address translation facilities for 4-Kbyte page size, variable block size, and 256-Mbyte segment size
  - — Both TLBs are 128-entry and two-way set associative
  - — The page table search is performed in hardware
  - — Separate IBATs and DBATs (four each) also defined as SPRs
  - — Separate instruction and data translation lookaside buffers (TLBs)
  - — LRU replacement algorithm
  - — 52-bit virtual address; 32-bit physical address
- Bus interface features include the following:
  - — Selectable processor-to-bus clock frequency ratios (1:1, 3:2, 2:1, 5:2, 3:1, 7:2, and 4:1)
  - — A 64-bit split-transaction external data bus with burst transfers
  - — Support for address pipelining and limited out-of-order bus transactions
  - — Four burst write queues—three for cache copy-back operations and one for snoop push operations
  - — Two single-beat write queues
  - — Additional signals and signal redefinition for direct-store operations
  - — Provides a data streaming mode that allows consecutive burst read data transfers to occur without intervening dead cycles. This mode also disables data retry operations.
  - — No-$\overline{\text{DRTRY}}$ mode eliminates the $\overline{\text{DRTRY}}$ signal from the qualified data bus grant condition. This improves performance on read operations for systems that do not use the $\overline{\text{DRTRY}}$ signal. No-$\overline{\text{DRTRY}}$ mode makes read data available to the processor one bus clock cycle sooner than if normal mode is used.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

- Multiprocessing support features include the following:
  - — Hardware enforced, four-state cache coherency protocol (MESI) for data cache. Bits are provided in the instruction cache to indicate only whether a cache block is valid or invalid.
  - — Separate port into data cache tags for bus snooping
  - — Load/store with reservation instruction pair for atomic memory references, semaphores, and other multiprocessor operations
- Power management
  - — Nap mode supports full shut down and snooping
  - — Operating voltage of 2.5 ± 0.2 V for processor core, 3.3 V for external signals
- Performance monitor can be used to help in debugging system designs and improving software efficiency, especially in multiprocessor systems.
- In-system testability and debugging features through JTAG boundary-scan capability

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

1.3 PowerPC Architecture Implementation

The PowerPC architecture shares the benefits of the POWER architecture optimized for single-chip implementations. The PowerPC architecture design facilitates parallel instruction execution and is scalable to take advantage of future technological gains.

This section describes the PowerPC architecture in general, and specific details about the implementation of the 604e as a low-power, 32-bit member of the PowerPC processor family. Note that the individual section headings indicate the chapters in the user's manual to which they correspond.

- Section 1.3.1, "Features," describes general features of the 604e with respect to the PowerPC architecture.
- Section 1.3.2, "PowerPC 604e Processor Programming Model (Chapter 2)," describes the aspects of the register and instruction implementation that are specific to the 604e.
- Section 1.3.3, "Cache and Bus Interface Unit Operation (Chapter 3)," describes the 604e-specific cache features.
- Section 1.3.4, "Exceptions (Chapter 4)," indicates that the 604e exception model is identical to that of the 604.
- Section 1.3.5, "Memory Management (Chapter 5)," indicates that the 604e MMU implementation is identical to that of the 604.
- Section 1.3.6, "Instruction Timing (Chapter 6)," describes specific characteristics of the 604e instruction timing model.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

- Section 1.3.7, “Signal Descriptions (Chapter 7),” describes differences in the operation of the signals implemented on the 604e.
- Section 1.3.8, “System Interface Operation (Chapter 8),” describes differences in the 604e bus protocol.
- Section 1.3.9, “Performance Monitor (Chapter 9),” defines additional features and changes in the 604e implementation of the performance monitor.

1.3.1 Features

The 604e is a high-performance, superscalar implementation of the PowerPC architecture. Like other PowerPC processors, it adheres to the PowerPC architecture specifications but also has additional features not defined by the architecture. These features do not affect software compatibility. The PowerPC architecture allows optimizing compilers to schedule instructions to maximize performance through efficient use of the PowerPC instruction set and register model. The multiple, independent execution units in the 604e allow compilers to maximize parallelism and instruction throughput. Compilers that take advantage of the flexibility of the PowerPC architecture can additionally optimize instruction processing of the PowerPC processors.

The following sections summarize the features of the 604e, including both those that are defined by the architecture and those that are unique to the 604e implementation.

The PowerPC architecture consists of the following layers, and adherence to the PowerPC architecture can be measured in terms of which of the following levels of the architecture is implemented:

- PowerPC user instruction set architecture (UISA)—Defines the base user-level instruction set, user-level registers, data types, floating-point exception model, memory models for a uniprocessor environment, and programming model for a uniprocessor environment.
- PowerPC virtual environment architecture (VEA)—Describes the memory model for a multiprocessor environment, defines cache control instructions, and describes other aspects of virtual environments. Implementations that conform to the VEA also adhere to the UISA, but may not necessarily adhere to the OEA.
- PowerPC operating environment architecture (OEA)—Defines the memory management model, supervisor-level registers, synchronization requirements, and the exception model. Implementations that conform to the OEA also adhere to the UISA and the VEA.

For more information, refer to *The Programming Environments Manual*.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

The 604e complies to all three levels of the PowerPC architecture. Note that the PowerPC architecture defines additional instructions for 64-bit data types. These instructions cause an illegal instruction exception on the 604e. PowerPC processors are allowed to have implementation-specific features that fall outside, but do not conflict with, the PowerPC architecture specification. Examples of features that are specific to the 604e include the performance monitor and nap mode.

1.3.2 PowerPC 604e Processor Programming Model (Chapter 2)

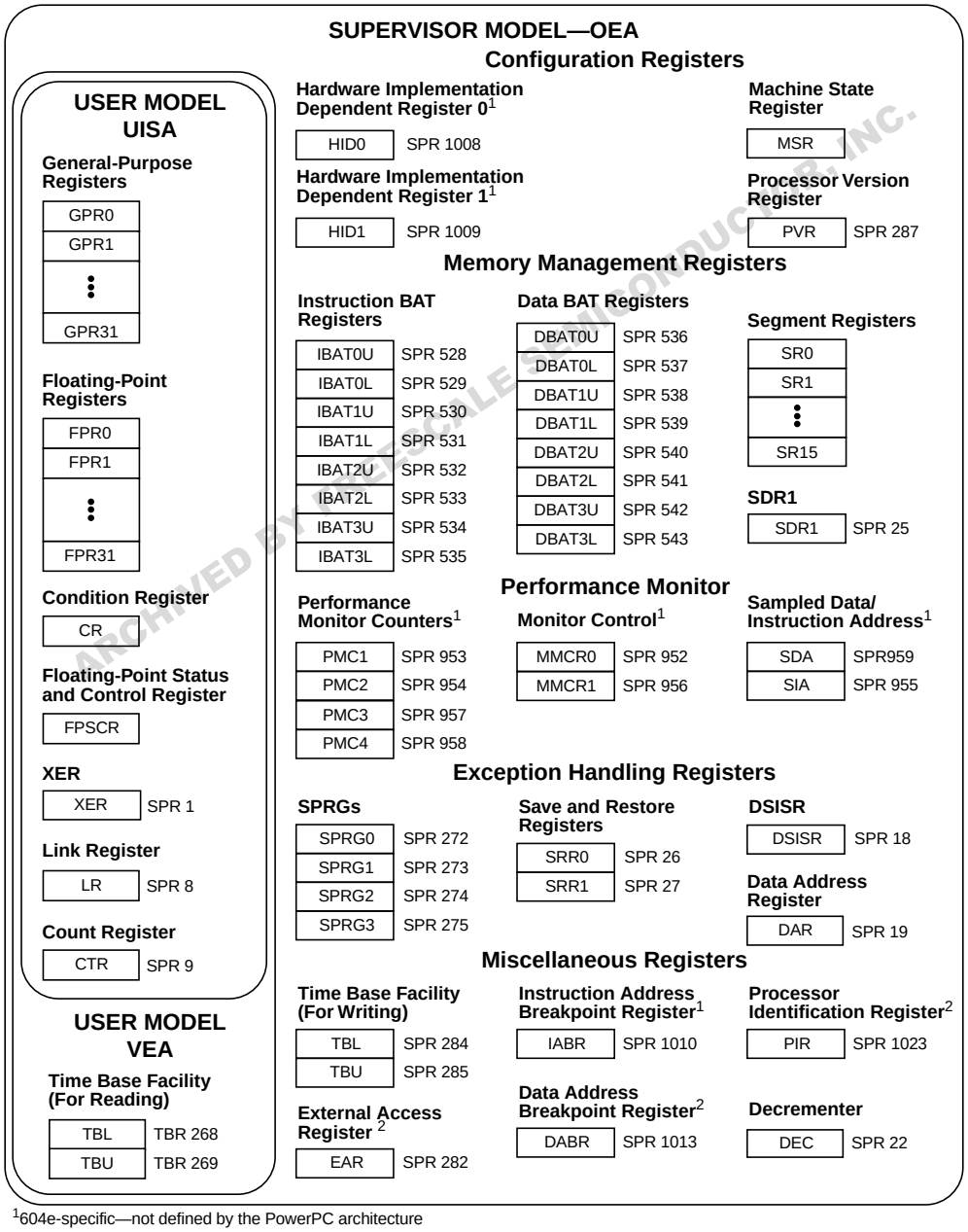
The 604e and 604 implement the register set required by the 32-bit portion of the PowerPC architecture. In addition, the 604e supports all 604-specific registers as well as several 604e-specific registers, as described in this section.

Figure 2 shows the registers implemented in the 604e, indicating those that are defined by the PowerPC architecture and those that are 604e-specific. All registers except the FPRs are 32 bits wide.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com



[1]604e-specific—not defined by the PowerPC architecture

[2] Optional to the PowerPC Architecture

Figure 2. Programming Model—PowerPC 604e Microprocessor Registers

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

For More Information On This Product,
Go to: www.freescale.com

The 604e includes the following registers not defined by the PowerPC architecture that are either not provided in the 604 or incorporate changes from the 604 implementation:

- Hardware implementation-dependent register 1 (HID1)—This register, which is not implemented in the 604, is used to display the PLL configuration. This register is described in Section 1.3.2.2, “HID1—PLL Configuration Register.”
- Performance monitor counter registers (PMC3–PMC4). The counters are used to record the number of times a certain event has occurred. PMC3 and PMC4 are not implemented in the 604. PMC1 and PMC2 are implemented in the 604 and are described in the user’s manual. See Section 1.3.2.4.3, “Performance Monitor Counter Registers (PMC3 and PMC4),” for more information.
- Performance monitor mode control register 0 (MMCR0)—MMCR0 has additional bits not described in the user’s manual. The additional bits are described in Section 1.3.2.4.1, “Changes to Monitor Mode Control Register 0—MMCR0.”
- Performance monitor mode control register 1 (MMCR1)—The performance monitor control registers are used for enabling various performance monitoring interrupt conditions and establishes the function of the counters. MMCR1 is not implemented in the 604. See Section 1.3.2.4.2, “Monitor Mode Control Register 1—MMCR1,” for more information.
- Hardware implementation-dependent register 0 (HID0)—This register is used to control various functions within the 604 and 604e, such as enabling checkstop conditions, and locking, enabling, and invalidating the instruction and data caches. Additional bits defined in the HID0 register disable the BTAC, control whether coherency is maintained for instruction fetches, and disable the default precharge values for the shared ($\overline{\text{SHD}}$) and address retry ($\overline{\text{ARTRY}}$) signals. The 604e defines additional bits not included in the 604 implementations of the HID0 register. These bits are described in Section 1.3.2.1, “PowerPC 604e-Specific Bits in HID0.”

Note that while it is not guaranteed that the implementation of HID registers is consistent among PowerPC processors, other processors may be implemented with similar or identical HID registers.

1.3.2.1 PowerPC 604e-Specific Bits in HID0

The 604e has three additional bits in HID0, described in Table 1. The HID0 bits not shown here are implemented as they are in the 604 and are described in Section 2.1.2.3, “Hardware Implementation-Dependent Register 0,” in the user’s manual.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Table 1. Additional HID0 Bits

Bit	Description
12	Reserved. This bit should always be set to zero.
23	Coherent instruction fetch enable—controls whether instruction fetch bus operations are snooped. 0 In this default state, all instruction fetch address tenures are nonglobal, regardless of the state of the MSR[IR] or the WIMG bits. Therefore, coherency checking on instruction fetches is disabled, as it is on the 604. 1 The 604e presents a value on the $\overline{\text{GBL}}$ signal for instruction fetch address tenures that reflects the state of the M bit if MSR[IR] = 1. If IR = 0 and HID0[23] is set, the $\overline{\text{GBL}}$ signal is asserted for all instruction fetch address tenures. When modifying the instruction cache enable or instruction cache lock bits, software should place an **isync** instruction after the **mtspr**[HID0] instruction to ensure that the subsequent instructions are fetched with the proper cache mode. Note that, like the 604, the 604e never snoops its data cache during its own instruction fetch address tenure, regardless of the state of $\overline{\text{GBL}}$. Therefore, assertion of the $\overline{\text{GBL}}$ signal does not guarantee coherency between the 604e's own instruction cache and data cache. As in the 604, coherency between the instruction and data caches must be maintained by software. Additional information is provided in Section 1.3.3.1, "Instruction Cache."
30	BTAC disable—used to disable use of the 64-entry branch target address cache. 0 The BTAC is enabled and new entries can be added. 1 The BTAC contents are invalidated and the BTAC behaves as if it were empty. New entries cannot be added until the BTAC is enabled. Note that the BTAC can be flushed by disabling and re-enabling the BTAC using two successive **mtspr** instructions.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

When modifying the data cache enable or data cache lock bits, software should place a **sync** instruction both before and after the move to the HID0 register to ensure that the data cache is properly updated by instructions both before and after the move to HID0 instruction.

1.3.2.2 HID1—PLL Configuration Register

The 604e has added HID1, (SPR 1009), shown in Figure 3. HID1 is a supervisor-level register that allows software to read the current PLL_CFG value. The PLL_CFG signal values are read from bits HID1[0–3]. The remaining bits are reserved and are read as zeros. HID1 is a read-only register.

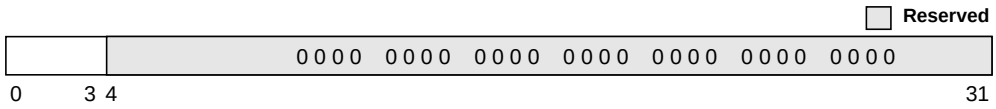


Figure 3. HID1 Clock Configuration Register

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

The bit settings in HID1 are described in Table 2.

Table 2. HID1 Bit Settings

Bits	Description
0–3	PLL configuration bits (0–3)
4–31	Reserved (Read as zero)

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

1.3.2.3 Processor Version Register

The processor version number is 9 for the 604e. The processor revision level starts at 0x0100 and changes for each chip revision. The revision level is updated on all silicon revisions. For more information, see “Processor Version Register (PVR),” in Chapter 2, “PowerPC Register Set,” of *The Programming Environments Manual*.

1.3.2.4 Performance Monitor Registers

The 604e implements an additional monitor mode control register (MMCR1) in addition to the MMCR0 implemented in the 604. Changes to the 604e implementation of MMCR0 are described in Section 1.3.2.4.1, “Changes to Monitor Mode Control Register 0—MMCR0.”

In addition, the 604e also implements two additional performance monitor counter registers (PMC3 and PMC4). These are described in Section 1.3.2.4.3, “Performance Monitor Counter Registers (PMC3 and PMC4).”

1.3.2.4.1 Changes to Monitor Mode Control Register 0—MMCR0

Changes to MMCR0 in the 604e are described in Table 3. These changes primarily reflect the addition of two performance monitor counter registers (PMC3 and PMC4). For information about other bit settings, see Section 2.1.2.4.1, “Monitor Mode Control Register 0,” in the user’s manual.

Table 3. PowerPC 604e-Specific MMCR0 Bit Settings

Bits	Name	Description
6	DISCOUNT	Disable counting of PMC1–PMC4 when a performance monitor interrupt is signalled or the occurrence of an enabled time base transition with ((INTONBITTRANS =1) & (ENINT = 1)). 0 Signalling a performance monitoring interrupt does not affect the counting status of PMC1–PMC4. 1 The signalling of a performance monitoring interrupt prevents the changing of the PMC1 counter. The PMC2–PMC4 counters does not change if PMCTRIGGER = 0. Because, a time base signal could have occurred along with an enabled counter negative condition, software should always reset INTONBITTRANS to zero, if the value in INTONBITTRANS was a one.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Table 3. PowerPC 604e-Specific MMCR0 Bit Settings (Continued)

Bits	Name	Description
10–15	THRESHOLD	Threshold value. All 6 bits are supported by the 604e. The threshold value is multiplied by 4, allowing threshold values from 0 to 252 in increments of 4. The intent of the THRESHOLD support is to be able to characterize L1 data cache misses.
17	PMCINTCONTROL	Enable interrupt signalling due to any PMCn (n>1) counter negative. 0 Disable PMCn (n>1) interrupt signalling due to PMCn (n>1) counter negative. 1 Enable PMCn (n>1) interrupt signalling due to PMCn (n>1) counter negative.
18	PMCTRIGGER	PMCTRIGGER may be used to trigger counting of PMCn (n>1) after PMC1 has become negative or after a performance monitoring interrupt is signalled. 0 Enable PMCn (n>1) counting 1 Disable PMCn (n>1) counting until PMC1 bit 0 is “on” or until a performance monitor interrupt is signalled. PMCTRIGGER may be used to trigger counting of PMCn (n>1) after PMC1 has become negative. This provides a triggering mechanism to allow counting after a certain condition occurs or after enough time has occurred. It can be used to support getting the count associated with a specific event.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

1.3.2.4.2 Monitor Mode Control Register 1—MMCR1

The 604e defines an additional monitor mode control register (MMCR1), which functions as an event selector for the two 604e-specific performance monitor counter registers (PMC3 and PMC4). MMCR1 is SPR 956. The MMCR1 register is shown in Figure 4.

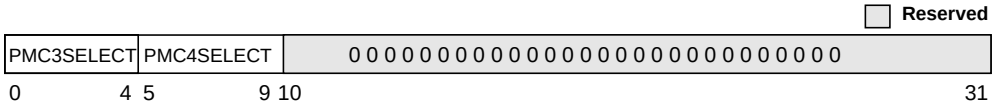


Figure 4. Monitor Mode Control Register 1 (MMCR1)

Bit settings for MMCR1 are shown in Table 4. The corresponding events are described in the Section 1.3.2.4.3, “Performance Monitor Counter Registers (PMC3 and PMC4).”

Table 4. MMCR1 Bit Settings

Bits	Name	Description
0–4	PMC3SELECT	PMC3 event selector
5–9	PMC4SELECT	PMC4 event selector
10–31	—	Reserved

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

1.3.2.4.3 Performance Monitor Counter Registers (PMC3 and PMC4)

The 604e implements two additional performance monitor counter registers, PMC3 and PMC4. PMC3 is SPR 957 and PMC4 is SPR 958. The events are described in Table 5 and Table 6.

Table 5. Selectable Events—PMC3

MMCR1[0–4]	Comments
0 0000	Register counter holds current value.
0 0001	Count every cycle.
0 0010	Indicates the number of instructions being completed every cycle
0 0011	RTCSELECT bit transition. 0 = 47, 1 = 51, 2 = 55, 3 = 63 (bits from the time base lower register).
0 0100	Number of instructions dispatched
0 0101	Number of cycles the LSU stalls due to BIU or cache busy. Counts cycles between when a load or store request is made and a response was expected. For example, when a store is retried, there are four cycles before the same instruction is presented to the cache again. Cycles in between are not counted.
0 0110	Number of cycles the LSU stalls due to a full store queue
0 0111	Number of cycles the LSU stalls due to operands not available in the reservation station
0 1000	Number of instructions written into the load queue. Misaligned loads are split into two transactions with the first part always written into the load queue. If both parts are cache hits, data is returned to the rename registers and the first part is flushed from the load queue. To count the instructions that enter the load queue to stay, the misaligned load hits must be subtracted. See event 8 in Table 6.
0 1001	Number of cycles that completion stalls for a store instruction
0 1010	Number of cycles that completion stalls for an unfinished instruction. This event is a superset of PMC3 event 9 and PMC4 event 10.
0 1011	Number of system calls
0 1100	Number of cycles the BPU stalled as branch waits for its operand
0 1101	Number of fetch corrections made at the dispatch stage. Prioritized behind the execute stage.
0 1110	Number of cycles the dispatch stalls waiting for instructions
0 1111	Number of cycles the dispatch stalls due to unavailability of reorder buffer (ROB) entry. No ROB entry was available for the first nondispatched instruction.
1 0000	Number of cycles the dispatch unit stalls due to no FPR rename buffer available. First nondispatched instruction required a floating-point reorder buffer and none was available.
1 0001	Number of instruction table search operations
1 0010	Number of data table search operations. Completion could result from a page fault or a PTE match.
1 0011	Number of cycles the FPU stalled
1 0100	Number of cycles the SCIU1 stalled
1 0101	Number of times the BIU forwards noncritical data from the line-fill buffer

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Table 5. Selectable Events—PMC3 (Continued)

MMCR1[0–4]	Comments
1 0110	Number of data bus transactions completed with pipelining one deep with no additional bus transactions queued behind it
1 0111	Number of data bus transactions completed with two data bus transactions queued behind
1 1000	Counts pairs of back-to-back burst reads streamed without a dead cycle between them in data streaming mode
1 1001	Counts non-$\overline{\text{ARTRY}}$d processor kill transactions caused by a write-hit-on-shared condition
1 1010	This event counts non-$\overline{\text{ARTRY}}$d write-with-kill address operations that originate from the three castout buffers. These include high-priority write-with-kill transactions caused by a snoop hit on modified data in one of the BIU's three copy-back buffers. When the cache block on a data cache miss is modified, it is queued in one of three copy-back buffers. The miss is serviced before the copy-back buffer is written back to memory as a write-with-kill transaction.
1 1011	Number of cycles when exactly two castout buffers are occupied
1 1100	Number of data cache accesses retried due to occupied castout buffers
1 1101	Number of read transactions from load misses brought into the cache in a shared state
1 1110	CRU Indicates that a CR logical instruction is being finished.

Table 6 lists the selectable events for PMC4.

Table 6. Selectable Events—PMC4

MMCR1[5–9]	Description
0 0000	Register counter holds current value
0 0001	Count every cycle
0 0010	Number of instructions being completed
0 0011	RTCSELECT bit transition. 0 = 47, 1 = 51, 2 = 55, 3 = 63 (bits from the time base lower register).
0 0100	Number of instructions dispatched
0 0101	Number of cycles the LSU stalls due to busy MMU
0 0110	Number of cycles the LSU stalls due to the load queue full
0 0111	Number of cycles the LSU stalls due to address collision
0 1000	Number of misaligned loads that are cache hits for both the first and second accesses. Related to event 8 in PMC3.
0 1001	Number of instructions written into the store queue
0 1010	Number of cycles that completion stalls for a load instruction
0 1011	Number of hits in the BTAC. **Warning**—if decode buffers cannot accept new instructions, the processor refetches the same address multiple times.
0 1100	Number of times the four basic blocks in the completion buffer from which instructions can be retired were used

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Table 6. Selectable Events—PMC4 (Continued)

MMCR1[5–9]	Description
0 1101	Number of fetch corrections made at decode stage
0 1110	Number of cycles the dispatch unit stalls due to no unit available. First nondispatched instruction requires an execution unit that is either full or a previous instruction is being dispatched to that unit.
0 1111	Number of cycles the dispatch unit stalls due to unavailability of GPR rename buffer. First nondispatched instruction requires a GPR reorder buffer and none are available.
1 0000	Number of cycles the dispatch unit stalls due to no CR rename buffer available. First nondispatched instruction requires a CR rename buffer and none is available.
1 0001	Number of cycles the dispatch unit stalls due to CTR/LR interlock. First nondispatched instruction could not dispatch due to CTR/LR/**mtcrf** interlock.
1 0010	Number of cycles spent doing instruction table search operations
1 0011	Number of cycles spent doing data table search operations
1 0100	Number of cycles SCIU0 was stalled
1 0101	Number of cycles MCIU was stalled
1 0110	Number of bus cycles after an internal bus request without a qualified bus grant
1 0111	Number of data bus transactions completed with one data bus transaction queued behind
1 1000	Number of write data transactions that have been reordered before a previous read data transaction using the $\overline{\text{DBWO}}$ feature
1 1001	Number of $\overline{\text{ARTRY}}$d processor address bus transactions
1 1010	Number of high-priority snoop pushes. Snoop transactions, except for write-with-kill, that hit modified data in the data cache cause a high-priority write (snoop push) of that modified cache block to memory.This operation has a transaction type of write-with-kill. This event counts the number of non-$\overline{\text{ARTRY}}$d processor write-with-kill transactions that were caused by a snoop hit on modified data in the data cache. It does not count high-priority write-with-kill transactions caused by snoop hits on modified data in one of the BIU's three copy-back buffers.
1 1011	Number of cycles for which exactly one castout buffer is occupied
1 1100	Number of cycles for which exactly three castout buffers are occupied
1 1101	Number of read transactions from load misses brought into the cache in an exclusive (E) state
1 1110	Number of undispatched instructions beyond branch

1.3.2.5 Support for Misaligned Little-Endian Accesses

The 604e provides hardware support for misaligned little-endian accesses. Little-endian accesses in the 604e take an alignment exception for the same cases that big-endian accesses take alignment exceptions. Any data access that crosses a word boundary requires two accesses regardless of whether the data is in big- or little-endian format. When two accesses are required, the lower addressed word (in the current addressing mode) is accessed first. Consider the memory mapping in Figure 5.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

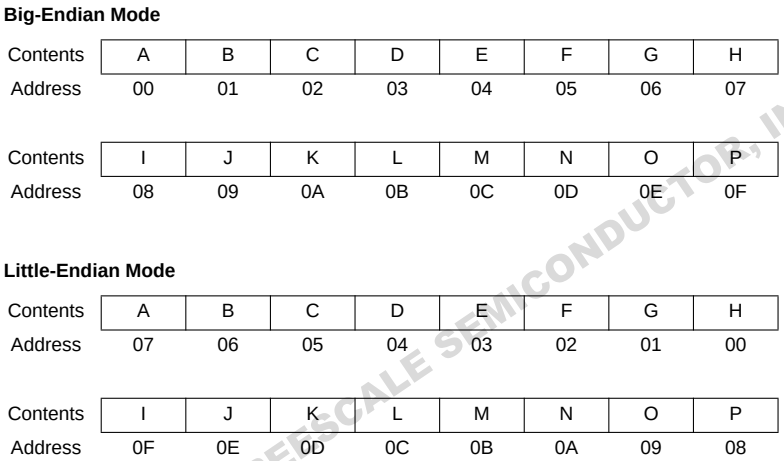


Figure 5. Big-Endian and Little-Endian Memory Mapping

RETRIEVED BY FREESCALE SEMICONDUCTOR, INC.

If two bytes are requested starting at little-endian address 0x3, one byte at big-endian address 0x4 containing data **E** is accessed first followed by one byte at big-endian address 0x3 containing data **D**. For a load halfword, the data written back to the GPR would be **D**, **E**. If four bytes are requested starting at little-endian address 0x6, two bytes at big-endian address 0x0 containing data **A**, **B** are accessed first followed by two bytes at big-endian address 0xE containing data **O**, **P**. For a load word, the data written back to the GPR would be **O**, **P**, **A**, **B**.

Misaligned little-endian accesses to direct-storage segments are boundedly-undefined.

1.3.2.6 Instruction Set

The 604e implements the same set of instructions that are implemented in the 604; that is, the entire PowerPC instruction set (for 32-bit implementations) and most optional PowerPC instructions. For information, see Section 2.3.3, “Instruction Set Overview,” in the user’s manual. The following changes affect information provided in the user’s manual.

- The undefined result of an integer divide overflow differs from that of the 604.
- Changes to the behavior of the **dcbst** and **dcbtst** instructions are described in Section 1.3.3.4, “Changes to dcbt/dcbtst Instruction Behavior.”

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

1.3.3 Cache and Bus Interface Unit Operation (Chapter 3)

The 604e has separate 32-Kbyte data and instruction caches. This is double the size of the 604 caches. The 604e caches are logically organized as a four-way set with 256 sets compared to the 604's 128 sets. The physical address bits that determine the set are 19 through 26 with 19 being the most-significant bit of the index. If bit 19 is zero, the block of data is an even 4-Kbyte page that resides in sets 0–127; otherwise, bit 19 is one and the block of data is an odd 4-Kbyte page that resides in sets 128–255. Because the caches are four-way set-associative, the cache set element (CSE0–CSE1) signals remain unchanged from the 604. Figure 6 shows the organization of the caches.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

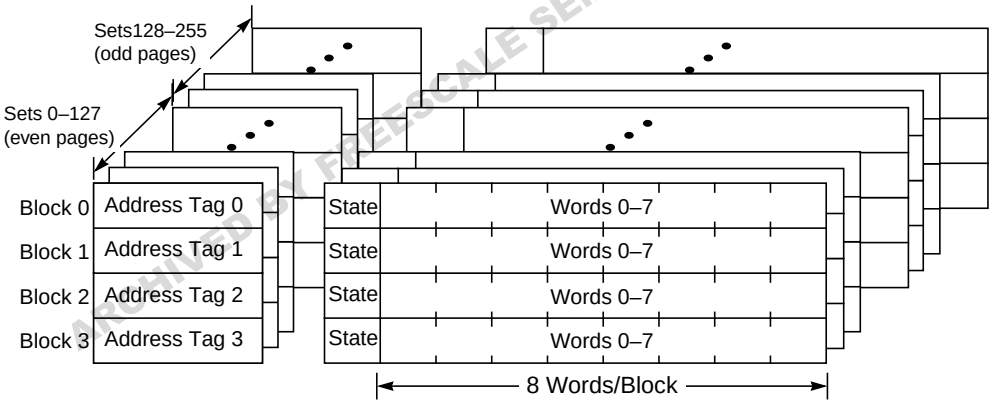


Figure 6. Cache Unit Organization

1.3.3.1 Instruction Cache

The 604e's 32-Kbyte, four-way set-associative instruction cache is physically indexed. Within a single cycle, the instruction cache provides up to four instructions.

The 604e provides coherency checking for instruction fetches. Instruction fetching coherency is controlled by HID0[23]. In the default mode, HID0[23] is 0 and the $\overline{\text{GBL}}$ signal is not asserted for instruction accesses on the bus, as is the case with the 604. If the bit is set and instruction translation is enabled (MSR[IR] = 1), the $\overline{\text{GBL}}$ signal is set to reflect the M bit for this page or block. If HID0[23] is set and instruction translation is disabled (MSR[IR] = 0), the $\overline{\text{GBL}}$ signal is asserted and coherency is maintained in the instruction cache.

The PowerPC architecture defines a special set of instructions for managing the instruction cache. The instruction cache can be invalidated entirely or on a cache-block basis. In addition, the instruction cache can be disabled and invalidated by setting the HID0[16] and HID0[20] bits, respectively. The instruction cache can be locked by setting HID0[18].

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

1.3.3.2 Data Cache

The 604e's data cache is a 32-Kbyte, four-way set-associative cache. It is a physically-indexed, nonblocking, write-back cache with hardware support for reloading on cache misses. Within one cycle, the data cache provides double-word access to the LSU.

The 604e provides additional support for data cache line-fill buffer forwarding. In the 604, only the critical double word of a burst operation was made available to the requesting unit at the time it was burst into the line-fill buffer. Subsequent data was unavailable until the cache block was filled. On the 604e, subsequent data is also made available as it arrives in the line-fill buffer.

The 604e implements three copy-back write buffers (the 604 has one). The additional copy-back buffers allow certain instructions to take further advantage of the pipelined system bus to provide highly efficient handling of cache copy-back operations, block invalidate operations caused by the Data Cache Block Flush (**dcbf**) instruction, and cache block clean operations resulting from the Data Cache Block Store (**dcbst**) instruction.

Like the 604, the data cache tags are dual-ported, so snooping does not affect the internal operation of other transactions on the system interface. If a snoop hit occurs in a modified block, the LSU is blocked internally for one cycle to allow the eight-word block of data to be copied to the write-back buffer, if necessary.

Like the instruction cache, the data cache can be invalidated all at once or on a per cache block basis. The data cache can be disabled and invalidated by setting the HID0[17] and HID0[21] bits, respectively. The data cache can be locked by setting HID0[19].

1.3.3.3 Data Cache Line-Fill Buffer Forwarding

When a load misses the cache, it is placed into the load queue. The critical data comes back first and is unconditionally forwarded to the load/store unit. If a subsequent in-order load to the same cache block hits on valid data in the data line-fill buffer, it is forwarded to the load/store unit from the line-fill buffer. In the 604, a subsequent in-order load to the same cache block is required to wait until the line-fill buffer is completely written into the cache before data is accessed from the cache.

1.3.3.4 Changes to dcbt/dcbtst Instruction Behavior

Both the 604 and the 604e treat the **dcbt** and **dcbtst** instructions as no-ops if any of the following conditions is met:

- The address misses in the TLB and in the BAT.
- The address is directed to a direct-store segment.
- The address is directed to a cache-inhibited page.

The 604e also treats the instructions as no-ops if the data cache lock bit HID0[19] is set.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

1.3.3.5 Snooping Protocol Change for Read-with-Intent-to-Modify Bus Operations

It is now illegal for any snooping device to generate a $\overline{\text{SHD}}$ snoop response without an $\overline{\text{ARTRY}}$ response to a RWITM address tenure. This change is required for the 604 and 604e. This change is also effective for later revisions of the 604. For more information, see the entry for "Section 3.9.6, Page 3-21" in Part 2: "Errata to *PowerPC 604 RISC Microprocessor User's Manual*," of this document.

1.3.3.6 Two Additional Cache Copy-Back Write Buffers

The 604e bus interface unit has six write buffers, four for burst write operations and two for single-beat operations.

- The four burst write buffers can hold a full 32-byte cache block of data for burst write data bus tenures. Of the four burst write buffers, one is a snoop push buffer and the other three are cache copy-back buffers.
  - The snoop push buffer is dedicated for snoop push write operations.
  - The three copy-back buffers are used for cache copy-back operations, block invalidates due to the Data Cache Block Flush (**dcbf**) instruction or block cleans due to the Data Cache Block Store (**dcbst**) instruction.
- Each of the two single-beat write buffers can hold up to 8 bytes of data.

The 604 implements only one copy-back buffer, but is otherwise the same as the 604e implementation.

Typically, these three copy-back buffers are written to memory in the same order in which they are filled, having the lowest priority access among all the bus interface unit's memory queues. Write operations from these buffers can occur out-of-order under the two following conditions:

- A snoop hit on one or more copy-back buffers causes the copy-back buffers to have the second highest priority among the BIU's memory queues, after only the snoop-push buffer. In this case, the next write from these three copy-back buffers will be from the buffer that contains the newest data corresponding to the snoop hit. If the snoop address hit on multiple copy-back buffers (possibly due to the **dcbst** instruction), the accesses for all matching buffers except the one with the newest data are cancelled.
- Similarly, if execution of the **dcbst** instruction causes multiple copy-back buffers to contain the same address, each buffer that contains this address is cancelled unless it contains the newest data or unless the buffer is the next address transaction to go to the bus.

The three copy-back buffers in the 604e notably improve the performance of multiple **dcbf** and **dcbst** instructions because the address and data tenures of burst writes can be pipelined.

ARCHIVED BY FREESCALE SEMICONDUCTOR INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

1.3.4 Exceptions (Chapter 4)

The 604e implements the same set of exceptions as the 604, as described in the user's manual.

1.3.5 Memory Management (Chapter 5)

The 604e MMU implementation is the same as is used in the 604.

1.3.6 Instruction Timing (Chapter 6)

The 604e instruction timing model has a few changes from the 604, although it is basically the same design. A conceptual model of the 604e hardware design showing the relationships between the various units that affect the instruction timing is shown in Figure 7.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

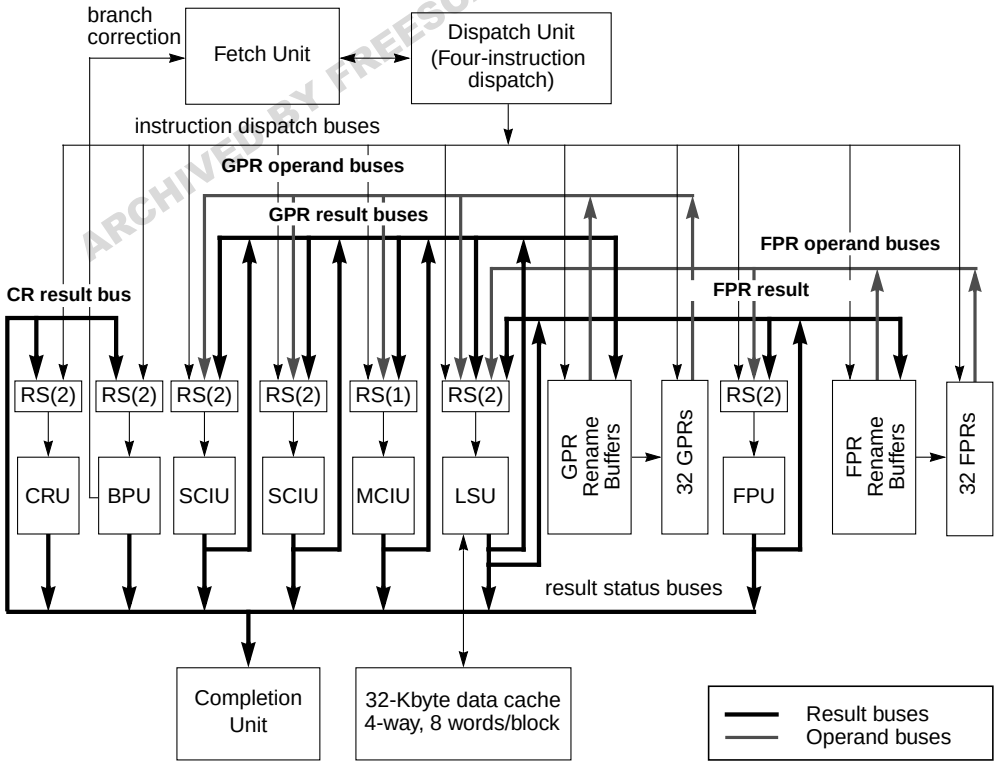


Figure 7. Block Diagram—Internal Data Paths

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

The instruction timing in the 604e incorporates the following changes:

- Addition of a condition register unit (CRU)—The CRU executes all condition register logical and flow control instructions. Because the CRU shares the dispatch bus with the BPU, only one condition register or branch instruction can be issued per clock cycle. In the 604, the CR logical unit operations are handled by the BPU. The addition of the CRU allows branch instructions to potentially execute/resolve before a preceding CR logical instruction. Although one CR logical or branch instruction can be dispatched per clock cycle, both branch and CR logical instructions can execute simultaneously. Branches are still executed in order with respect to other branch instructions. If either the CR logical reservation station or the branch reservation station is full then no instructions can be dispatched to either unit.
- Branch correction in decode stage—Branch correction in the decode stage has been modified to predict branches whose target is taken from the CTR or LR. This correction occurs if no CTR or LR updates are pending. This correction like all other decode stage corrections is done only on the first two instructions of the decode stage. This correction saves at least one cycle on branch correction when the **mtspr** instruction can be separated from the branch that uses the SPR as a target address.
- Instruction fetch when translation is disabled—If translation is disabled (MSR[IR] = 0), the 604e fetches instructions when they hit in the cache or if the previous completed instruction fetch was to the same page as this instruction fetch. Where an instruction access hits in the cache, the 604e continues to fetch any consecutive accesses to that same page.

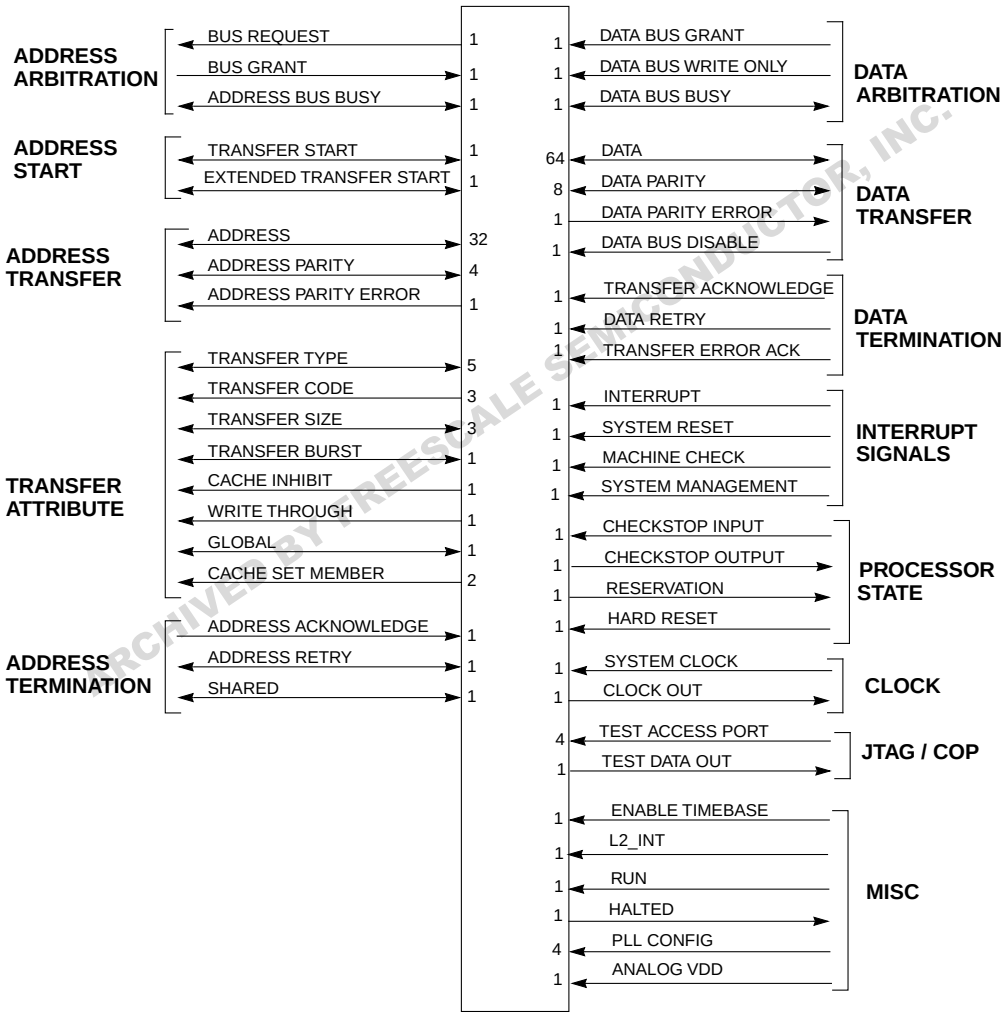
1.3.7 Signal Descriptions (Chapter 7)

The 604e provides a versatile bus interface that allows a wide variety of system design options. The interface includes a 72-bit data bus (64 bits of data and 8 bits of parity), a 36-bit address bus (32 bits of address and 4 bits of parity), and sufficient control signals to allow for a variety of system-level optimizations. The system interface is specific for each PowerPC processor implementation. The 604e system interface is shown in Figure 8.

NOTE

A bar over a signal name indicates that the signal is active low—for example, $\overline{\text{ARTRY}}$ (address retry) and $\overline{\text{TS}}$ (transfer start). Active-low signals are referred to as asserted (active) when they are low and negated when they are high. Signals that are not active-low, such as AP[0–3] (address bus parity signals) and TT[0–4] (transfer type signals) are referred to as asserted when they are high and negated when they are low.

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.
For More Information On This Product,
Go to: www.freescale.com



ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Figure 8. PowerPC 604e Microprocessor Signal Groups

The 604e system interface differs from that of the 604 in the following respects:

- The 604e has the same signal configuration as the 604; however, on the 604e Vdd and AVdd must be connected to 2.5 Vdc and OVdd must be connected to 3.3 Vdc. The 604e uses split voltage planes, and for replacement compatibility, 604/604e designs should provide both 2.5-V and 3.3-V planes and the ability to connect those two planes together and disable the 2.5-V plane for operation with a 604.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

- Addition of no-$\overline{\text{DRTRY}}$ mode. In addition to the normal and data-streaming modes implemented on the 604, a no-$\overline{\text{DRTRY}}$ mode is implemented on the 604e that improves performance on read operations for systems that do not use the $\overline{\text{DRTRY}}$ signal. No-$\overline{\text{DRTRY}}$ mode makes read data available to the processor one bus clock cycle sooner than in normal mode. In no-$\overline{\text{DRTRY}}$ mode, the $\overline{\text{DRTRY}}$ signal is no longer sampled as part of a qualified bus grant.

  This functionality is described more fully in Section 1.3.8, “System Interface Operation (Chapter 8).”

- Power management signals—The 604e implements signals that allow the processor to operate in three different modes—normal, nap, and doze.
  - HALTED signal—The HALTED signal is asserted when the processor is halted internally and no snoop copy-back operations are in progress.
    - In nap mode, the HALTED signal is always asserted.
    - In doze mode, the HALTED signal is asserted unless a snoop-triggered copy-back is pending.
    - In normal mode, the HALTED signal is not asserted.
  - RUN signal—The 604e supports nap mode with a RUN signal similar to the 604. Asserting the RUN signal is equivalent to the doze mode in the 603.

  The operation of power management on the 604e is described in Section 1.3.7.1, “Power Management.”

- Internal clocking changes—The 604e internal clocking scheme is more similar to the 603e than to the 604. The 604e requires a single system clock (SYSCLK) input that sets the frequency of operation for the bus interface. Internally, the 604e uses a phase-locked loop (PLL) circuit to generate a master clock for all of the CPU circuitry (including the bus interface circuitry) which is phase-locked to the SYSCLK input.
- Bus clock ratios—The 604e supports processor-to-bus frequency ratios of 1:1, 3:2, 2:1, 5:2, 3:1, 4:1, and 7:2. Each ratio is limited to the frequency ranges specified in the PLL_CFG encodings shown in Table 7. Support for processor/bus clock ratios 5:2, 7:2, and 4:1 is not supported in the 604.
- To support the changes in the clocking configuration, different precharge timings for the $\overline{\text{ABB}}$, $\overline{\text{DBB}}$, $\overline{\text{ARTRY}}$, and $\overline{\text{SHD}}$ signals are implemented internally by the processor. Selectable precharge timings for $\overline{\text{ARTRY}}$ and $\overline{\text{SHD}}$ can be disabled by setting HID0[7]. Precharge timings are provided in the 604e hardware specifications.
- The 604e’s PLL_CFG settings are compatible with the 603e and the 604, although the supported frequency ranges may differ. Changing the PLL_CFG setting during nap mode is not permitted. Table 7 lists PLL_CFG settings used for specifying processor/bus frequency ratios (*r*) and VCO divider values (*d*). For specific information, see the hardware specifications.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Table 7. PLL Configuration Encodings

PLL_CFG[0–3]		Processor/Bus Frequency Ratio (r)	VCO Divider (*d*)
Bin	Dec		
0000	0	1x	/2
0001	1	1x	/8
0010	2	7x	/2
0011	3	PLL bypass	n/a
0100	4	2x	/2
0101	5	6.5x	/2
0110	6	2.5x	/2
0111	7	4.5x	/2
1000	8	3x	/2
1001	9	5.5x	/2
1010	10	4x	/2
1011	11	5x	/2
1100	12	1.5x	/2
1101	13	6x	/2
1110	14	3.5x	/2
1111	15	Off	n/a

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

The processor/bus frequency ratio (*r*) and the value of the VCO divider (*d*) shown in Table 7 together determine the resulting frequency ranges according to the following formulas:

— SYSCLK frequency range:
  – Min = VCO_{min}/(r*d)
  – Max = VCO_{max}/(r*d)

— Core frequency range:
  – Min = VCO_{min}/d
  – Max = VCO_{max}/d

The actual values supported by a given 604e are provided in the 604e hardware specifications.

- The addition of the VOLTDETGND output signal (BGA package only). The VOLTDETGND signal is an indicator of the core voltage for use with power supplies capable of providing 2.5-V and 3.3-V outputs.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

1.3.7.1 Power Management

The 604e supports three power management modes—normal, doze, and nap.

- In normal mode, all clocks are running and instruction execution is proceeding normally.
- In doze mode, no instructions are being executed, but clocks are still running to allow snooping of the caches. If necessary, the caches perform copybacks of modified data.
- In nap mode, all internal clocks except those necessary to keep the decrementer, timebase, and interrupt logic running are stopped.

A transition state table for the three modes is shown in Figure 9.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

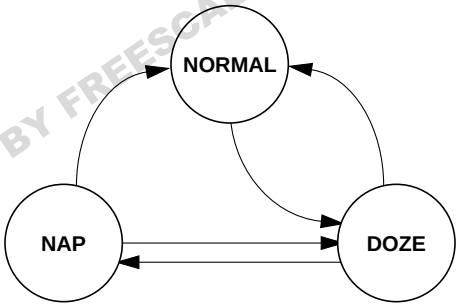


Figure 9. Power Management States

The following sections describe how the processor can go from one mode to the other.

1.3.7.1.1 State Transition from Normal Mode to Doze Mode

As shown in Figure 9, the only state transition allowed from the normal mode is to the doze mode. This transition requires system support. The RUN signal must be asserted by the system for at least 10 bus cycles before the software power management sequence can begin. The RUN signal does not affect the 604e operation in the normal mode, but affects operation during the transition from normal mode to doze mode. The software power management sequence is the following code:

```
sync
mtmsr
isync
branch back to the sync instruction
```

The **mtmsr** instruction should modify only MSR[POW]. All other MSR values such as the external interrupt enable should be set up before the software power management sequence is begun. When **mtmsr** is executed, the processor waits for its internal state to be idle before asserting HALTED, putting the processor in the doze mode. When entering the doze mode,

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

the system must assert RUN for at least 10 bus cycles after HALTED is asserted. When in the doze state, the HALTED signal is deasserted only when a snoop-triggered copy-back is in progress. The system must continually assert RUN whenever HALTED is negated in doze mode due to a snoop copy-back.

1.3.7.1.2 State Transition from Doze Mode to Nap Mode

A processor in doze mode can enter nap mode by doing the following:

1. The system should ensure that the bus is idle and the HALTED signal is asserted for at least 10 bus cycles.
2. The system should negate RUN and continue to prevent bus grants for at least 10 additional bus cycles. At this point, the processor is in the nap mode and bus transactions can be resumed. The processor does not snoop any subsequent bus transactions.

In going from doze to the nap mode, the system must ensure that the 604e not receive any $\overline{\text{TS}}$ (or $\overline{\text{XATS}}$) assertions by negating address bus grants to other bus masters. If the bus is not quiescent throughout the 10 clock transition window, the system may hang.

1.3.7.1.3 State Transition from Nap Mode to Doze Mode

A processor in nap mode can enter doze mode with the following sequence:

1. The system should ensure that the bus is idle for at least 10 bus cycles.
2. The system should assert the RUN signal and continue to prevent bus grants for at least an additional 10 bus cycles. At this point, the processor is in doze mode and all bus transactions can be snooped.

1.3.7.1.4 State Transition from Nap Mode to Normal Mode

Normal execution resumes from the nap mode when an interrupt or reset condition occurs. The transition from nap to normal mode is triggered by hard reset, soft reset, system management interrupt, machine check interrupt (if MSR[ME] = 1), external interrupt (if MSR[EE] = 1), or decrementer interrupt (if MSR[EE] = 1). When this transition occurs, the processor resumes clocking and vectors to the proper exception handler. Note that SRR0 points to an instruction inside the software power management sequence.

To exit power management, the exception handler should return to code outside this loop.

To re-enter power management, the system must ensure that the above mode transition rules are followed.

1.3.7.1.5 State Transition from Doze Mode to Normal Mode

The transition from doze to normal mode can be triggered by the same conditions as the nap to normal mode transition. This transition can also be triggered by a snoop detecting a parity error and causing a machine check exception. Other than the additional trigger condition, this transition is identical to the nap-to-normal mode transition.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

1.3.7.2 Addition of the VOLTDETGND Signal (BGA Package Only)

The VOLTDETGND output signal, which is implemented only on BGA packages, is an indicator of the core voltage. On the 604e, which has a 2.5-V core, VOLTDETGND is tied to ground internally to indicate to a power supply that a low-power processor is present. This signal connects to a control signal on a power supply capable of providing 2.5-V and 3.3-V outputs. Refer to the hardware specifications for more information about VOLTDETGND.

1.3.8 System Interface Operation (Chapter 8)

The 604e supports the three following bus modes:

- Normal mode. Default mode, as implemented by the 604.
- Data streaming mode (referred to as no-$\overline{\text{DRTRY}}$/data streaming mode in the user's manual and referred to elsewhere as fast-L2 mode). For information about the 604e implementation of data streaming mode, see Section 1.3.8.3, "Data Bus Arbitration in Data Streaming Mode."
- No-$\overline{\text{DRTRY}}$ mode that improves performance for data read operations. In no-$\overline{\text{DRTRY}}$ mode the data retry function is not available, and all read data is used by the processor one bus cycle earlier than in normal mode. (Not implemented on the 604.)

  Note that this mode is identical to the no-$\overline{\text{DRTRY}}$ mode in the 603 except for the manner in which it is entered during hard reset. Data streaming is not allowed in no-$\overline{\text{DRTRY}}$ mode—there always must be at least one dead cycle between data tenures.

1.3.8.1 No-$\overline{\text{DRTRY}}$ Mode

No-$\overline{\text{DRTRY}}$ mode disables the data retry function provided through the $\overline{\text{DRTRY}}$ signal. In normal mode, the memory system can cancel a data read operation by the master on the bus cycle after $\overline{\text{TA}}$ was asserted. This functionality requires the load data to be held an additional cycle to validate the data, and if necessary to assert $\overline{\text{DRTRY}}$ to cancel the operation. Disabling data retry eliminates the need for this cycle and allows data to be forwarded during load operations one bus cycle sooner—immediately when the assertion of $\overline{\text{TA}}$ is recognized. In no-$\overline{\text{DRTRY}}$ mode, the system must ensure that there are no attempts at late cancellation, which may cause improper operation by the 604e. The system must also ensure that a snooping device asserts $\overline{\text{ARTRY}}$ no later than the first assertion of $\overline{\text{TA}}$ to the 604e, but not on the cycle after the first assertion of $\overline{\text{TA}}$.

To enter no-$\overline{\text{DRTRY}}$ mode, the system must assert $\overline{\text{DRTRY}}$ coincidentally with $\overline{\text{HRESET}}$. This can be done by tying $\overline{\text{DRTRY}}$ asserted in hardware. $\overline{\text{DRTRY}}$ must remain asserted.

In no-$\overline{\text{DRTRY}}$ mode, data bus arbitration is unchanged except that $\overline{\text{DRTRY}}$ is no longer used to determine a qualified $\overline{\text{DBG}}$. A qualified $\overline{\text{DBG}}$ in no-$\overline{\text{DRTRY}}$ mode is simply the assertion of $\overline{\text{DBG}}$ and the negation of $\overline{\text{DBB}}$ (plus possibly additional qualifications due to

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.
For More Information On This Product,
Go to: www.freescale.com

$\overline{\text{ARTRY}}$ identical to those qualifications in normal and data streaming bus modes).

The system must define the beginning of the window in which the snoop response is valid and ensure that no data is transferred before the same cycle as the beginning of that window in no-$\overline{\text{DRTRY}}$ mode. For example, if the system defines a snoop response window that begins the second cycle after $\overline{\text{TS}}$, the earliest $\overline{\text{TA}}$ can be asserted to the 604e is the second cycle after $\overline{\text{TS}}$.

This no-$\overline{\text{DRTRY}}$ mode timing constraint on the earliest allowable assertion of $\overline{\text{TA}}$ with respect to $\overline{\text{ARTRY}}$ is identical to that constraint in data streaming mode.

To upgrade a 604-based system to the 604e and use no-$\overline{\text{DRTRY}}$ mode, the following considerations should be observed:

- The system uses the 604 in normal bus mode, described earlier in this section.
- The $\overline{\text{DRTRY}}$ must be tied negated and never used.
- The system must never assert $\overline{\text{TA}}$ before the first cycle of the system's snoop response window.

This system would then see a performance improvement due to the shorter effective latency seen by the 604e on read operations. This reduction in latency is equal to one bus cycle (three processor cycles in 3:1 bus mode).

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

1.3.8.2 PowerPC 604e Processor Configuration during $\overline{\text{HRESET}}$

The 604e has three modes that are configurable during a hard reset. Table 8 describes how the 604e is configured during hard reset. Normal mode and data-streaming mode $\overline{\text{HRESET}}$ configurations are identical to those on the 604.

Table 8. PowerPC 604e Processor Modes Configurable during Assertion of $\overline{\text{HRESET}}$

604e Mode	Input Signal	Timing Requirements	Notes
Normal	$\overline{\text{DRTRY}}$	Must be negated throughout the duration of the $\overline{\text{HRESET}}$ assertion. After $\overline{\text{HRESET}}$ negation, $\overline{\text{DRTRY}}$ can be used normally.	—
Data streaming	$\overline{\text{DRTRY}}$	Must be asserted and negated with $\overline{\text{HRESET}}$ and remain negated during normal operation.	Can be done by tying $\overline{\text{DRTRY}}$ to $\overline{\text{HRESET}}$
No-$\overline{\text{DRTRY}}$	$\overline{\text{DRTRY}}$	Must be asserted with $\overline{\text{HRESET}}$ and remain asserted during normal operation.	Can be done by statically tying $\overline{\text{DRTRY}}$ asserted.

1.3.8.3 Data Bus Arbitration in Data Streaming Mode

When the 604 operates in data streaming mode, $\overline{\text{DBG}}$ must be asserted for exactly one cycle per data bus tenure, in the cycle before the data tenure is to begin. The system cannot either assert $\overline{\text{DBG}}$ earlier than one cycle before the data tenure is to begin, park $\overline{\text{DBG}}$, or assert it for multiple consecutive cycles.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

In data streaming mode, the 604e is compatible with the 604's assertion requirements for $\overline{\text{DBG}}$, but less restrictive regarding successive data tenures mastered by the 604e. For the 604e, $\overline{\text{DBG}}$ must be asserted no earlier than the cycle before the 604e's data tenure is to begin only when another master currently controls the data bus (that is, when $\overline{\text{DBB}}$ would normally be asserted for a data tenure). If no other masters currently control the data bus (are asserting $\overline{\text{DBB}}$), the 604e allows the system to park $\overline{\text{DBG}}$ on the 604e. $\overline{\text{DBB}}$ remains an output-only signal in data streaming mode (that is, $\overline{\text{DBB}}$ does not participate in determining a qualified data bus grant), requiring the system to use $\overline{\text{DBG}}$ to ensure that different masters don't collide on data tenures.

Like the 604, the 604e requires a dead cycle between successive data tenures for which it is master, except for back-to-back burst read operations that can be streamed without a dead cycle. For back-to-back data tenures that cannot be streamed, the 604e does not accept an early data bus grant for the second tenure and negates its $\overline{\text{DBB}}$ output signal for one cycle between the first and second data tenure. The system must not attempt to stream consecutive $\overline{\text{TA}}$ assertions from the first to second data tenure in this case. Instead, a minimum of one dead cycle must be placed between the $\overline{\text{DBB}}$s of two tenures if the two tenures are not both burst reads.

1.3.9 Performance Monitor (Chapter 9)

The 604e incorporates a performance monitor facility that system designers can use to help bring up, debug, and optimize software performance, especially in multiprocessing systems. The performance monitor is a software-accessible mechanism that provides detailed information concerning the dispatch, execution, completion, and memory access of PowerPC instructions.

A performance monitor control register (MMCR0 or MMCR1) can be used to specify the conditions for which a performance monitoring interrupt is taken. For example, one such condition is associated with one of the counter registers (PMC1–PMC4) incrementing until the most-significant bit indicates a negative value. Additionally, the sampled instruction address and sampled data address registers (SIA and SDA) are used to hold addresses for instruction and data related to the performance monitoring interrupt.

The performance monitor has two additional counter registers and one additional control registers. The control register is MMCR1 (SPR 956). The counters, PMC3 and PMC4, are SPR 957 and SPR 958, respectively. MMCR0 has also been changed slightly from the original 604 definition. These registers are described in Section 1.3.2.4, "Performance Monitor Registers."

When the 604e vectors to the performance monitor interrupt exception handler, it automatically clears any pending performance monitor interrupts. Note that unlike the 604, the 604e does not require MMCR0[ENINT] to be cleared (and possibly reset) before external interrupts can be re-enabled.

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.
For More Information On This Product,
Go to: www.freescale.com

INDEX

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

INDEX

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Part 2: Errata to *PowerPC 604 RISC Microprocessor User's Manual*

This errata describes corrections to the *PowerPC 604 RISC Microprocessor User's Manual* and changes to the 604 that require the information in the user's manual to be updated. For convenience, the section number and page number of the errata item in the user's manual are provided.

Unless otherwise stated, these corrections apply to the 604 and the 604e.

2.1.1, Page 2-6, Add the following to the second-level bullet with the heading "Block address translation (BAT) registers":

> The 604 implements the G bit in the IBAT registers; however, attempting to execute code from an IBAT area with G = 1 causes an ISI exception. This complies with the revision of the architecture described in *PowerPC Microprocessor Family: The Programming Environments.*

2.1.2.1, Page 2-9 The first sentence in the paragraph after Table 2-2 should read as follows:

> The instruction that triggers the instruction address breakpoint exception is not executed before the exception handler is invoked.

2.1.2.2, Page 2-10 The SPR number for the PIR should be shown as 1023 rather than 1013.

2.1.2.3, Page 2-11 Table 2-3, last row. The note at the end of the description for HID0[29] is incorrect. The BHT is not initialized at power-on reset.

2.1.2.4.2, Page 2-14 Replace Table 2-5 and Table 2-6 with the following:

Table 2-5. Selectable Events—PMC1

MMCR0[0–4]	Description
000 0000	Nothing. Register counter holds current value.
000 0001	Processor cycles 0b1. Count every cycle.
000 0010	Number of instructions completed every cycle
000 0011	RTCSELECT bit transition. 0 = 47, 1 = 51, 2 = 55, 3 = 63 (bits from the time base lower register).
000 0100	Number of instructions dispatched
000 0101	Instruction cache misses
000 0110	Data TLB misses (in order)
000 0111	Branch misprediction correction from execute stage

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page # **Changes**

Table 2-5. Selectable Events—PMC1 (Continued)

MMCR0[0–4]	Description
000 1000	Number of reservations requested. The **lwarx** instruction is ready for execution in the LSU.
000 1001	Number of data cache load misses exceeding the threshold value with lateral L2 cache intervention
000 1010	Number of data cache store misses exceeding the threshold value with lateral L2 cache intervention
000 1011	Number of **mtspr** instructions dispatched
000 1100	Number of **sync** instructions completed
000 1101	Number of **eieio** instructions completed
000 1110	Number of integer instructions completed every cycle (no loads or stores)
000 1111	Number of floating-point instructions completed every cycle (no loads or stores)
001 0000	LSU produced result.
001 0001	SCIU1 produced result for an add, subtract, compare, rotate, shift, or logical instruction.
001 0010	FPU produced result.
001 0011	Number of instructions dispatched to the LSU
001 0100	Number of instructions dispatched to the SCIU1
001 0101	Number of instructions dispatched to the FPU
001 0110	Valid snoop requests received from outside the 604. Does not distinguish hits or misses.
001 0111	Number of data cache load misses exceeding the threshold value without lateral L2 intervention
001 1000	Number of data cache store misses exceeding the threshold value without lateral L2 intervention
001 1001	Number of cycles the branch unit is idle
001 1010	Number of cycles MCIU0 is idle
001 1011	Number of cycles the LSU is idle. No new instructions are executing; however, active loads or stores may be in the queues.
001 1100	Number of times the L2_INT is asserted (regardless of $\overline{TA}$ state)
001 1101	Number of unaligned loads
001 1110	Number of entries in the load queue each cycle (maximum of five). Although the load queue has four entries, a load miss latch may hold a load waiting for data from memory.
001 1111	Number of instruction breakpoint hits

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page # **Changes**

Table 2-6. Selectable Events—PMC2

MMCR0[26–31]	Description
00 0000	Register counter holds current value.
00 0001	Processor cycles 0b1. Count every cycle.
00 0010	Number of instructions completed. Legal values are 000, 001, 010, 011, 100.
00 0011	RTCSELECT bit transition. 0 = 47, 1 = 51, 2 = 55, 3 = 63 (bits from the time base lower register).
00 0100	Number of instructions dispatched (0 to 4 instructions per cycle)
00 0101	Number of cycles a load miss takes
00 0110	Data cache misses (in order)
00 0111	Number of instruction TLB misses
00 1000	Number of branches completed. Indicates the number of branch instructions being completed every cycle (00 = none, 10 = one, 11 = two, 01 is an illegal value).
00 1001	Number of reservations successfully obtained (**stwcx.** operation completed successfully)
00 1010	Number of **mfspr** instructions dispatched (in order)
00 1011	Number of **icbi** instructions. It may not hit in the cache.
00 1100	Number of pipeline “flushing” instructions (**sc**, **isync**, **mtspr** (XER), **mcrxr**, floating-point operation with divide by 0 or invalid operand and MSR[FE0, FE1] = 00, branch with MSR[BE] = 1, load string indexed with XER = 0, and SO bit getting set)
00 1101	BPU produced result.
00 1110	SCIU0 produced result (of an add, subtract, compare, rotate, shift, or logical instruction).
00 1111	MCIU produced result (of a multiply/divide or SPR instruction).
01 0000	Number of instructions dispatched to the branch unit.
01 0001	Number of instructions dispatched to the SCIU0.
01 0010	Number of loads completed. These include all cache operations and **tlbie**, **tlbsync**, **sync**, **eieio**, and **icbi** instructions.
01 0011	Number of instructions dispatched to the MCIU
01 0100	Number of snoop hits occurred
01 0101	Number of cycles during which the MSR[EE] bit is cleared
01 0110	Number of cycles the MCIU is idle
01 0111	Number of cycles SCIU1 is idle
01 1000	Number of cycles the FPU is idle
01 1001	Number of cycles the L2_INT signal is active (regardless of $\overline{\text{TA}}$ state)
01 1010	Number of times four instructions were dispatched
01 1011	Number of times three instructions were dispatched
01 1100	Number of times two instructions were dispatched

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page # **Changes**

Table 2-6. Selectable Events—PMC2 (Continued)

MMCR0[26–31]	Description
01 1101	Number of times one instruction was dispatched
01 1110	Number of unaligned stores
01 1111	Number of entries in the store queue each cycle (maximum of six)

2.1.3, Page 2-16 Add Section 2.1.3, “Reset Settings,” as follows:

Table 2-6a shows the state of the registers after a hard reset and before the first instruction is fetched from address 0xFFF0_0100 (the system reset exception vector).

Table 2–6a. Settings after Hard Reset (Used at Power-On)

Register	Setting	Register	Setting
BATs	Undefined	LR	Undefined
Caches*	Undefined and disabled	MSR	0x00000040 (only IP set)
CR	Undefined	PIR	Undefined
CTR	Undefined	PVR	ROM value
DABR	Breakpoint is disabled. Address is undefined.	Reservation address	Undefined
DAR	Undefined	Reservation flag	Cleared
DEC	Undefined	SDR1	Undefined
DSISR	Undefined	SPRG0–SPGR3	Undefined
EAR	E is cleared; RID is undefined.	SR	Undefined
FPR	Undefined	SRR0	Undefined
FPSCR	Set to 0	SRR1	Undefined
GPR	Undefined	Time base	Undefined
HID0	0x00000000	TLB	Undefined
IABR	Breakpoint is disabled. Address is undefined.	XER	Undefined

* The processor automatically begins operations by issuing an instruction fetch. Because caching is inhibited at start-up, this generates a single-beat load operation on the bus.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page #	Changes
2.3.4.7, Page 2-47	Table 2-35—Add the following note: An attempt to perform an atomic memory access (**lwarx** or **stwcx.**) to a location in write-through-required mode causes a DSI exception and DSISR[5] is set.
2.3.5.3.1, Page 2-51	Table 2-38. Add the following to the description of the **dcbst** instruction: A **dcbst** instruction followed by a store operation may appear out of order on the bus so that systems that have L2 caches that check for cache paradox conditions may detect a cache paradox. When a 604 executes a **dcbst** instruction to a cache block in shared state followed by a store instruction to the same cache block, the **dcbst** instruction causes a clean transaction on the bus if the 604's L1 cache block is not in modified data state. The store operation should cause a kill operation on the bus because it should hit on shared data in the L1 cache. However, the 604 may send out the kill operation before the clean operation. An L2 controller that performs paradox checking could be confused by this kill/clean sequence to the same cache block. The kill operation (with TC0–TC2 = 000) implies that the 604 is obtaining exclusive rights and will modify the line. The following clean operation implies that the 604 does not have the block modified. This may confuse the L2 controller. To avoid this, put a **sync** instruction after the **dcbst** instruction or don't check for this paradox.
3.4.1, Page 3-9	Add the following note to Table 3-2: Table 3-2 states that when the 604 issues a kill operation (that does not receive an $\overline{\text{ARTRY}}$ snoop response) the associated 604's cache block state changes from shared to modified. But if an **lwarx** instruction is followed by an **stwcx.** instruction to a different address, the 604 may broadcast a kill operation without marking the cache block in the on-chip cache modified. In designing an L2 cache controller for the 604, it should not be assumed that a kill operation issued by the 604 results in the 604 gaining modified ownership. The 604e does not broadcast the kill operation without marking the cache block as modified.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page #	Changes
3.4.2, Page 3-8	Add the following section:

3.4.2 General Comments on 604 Snooping

When a 604 is not the bus master, it monitors all bus traffic and performs cache and memory queue snooping as appropriate. The snooping is triggered by the receipt of a qualified snoop request, as indicated by the simultaneous assertion of the transfer start ($\overline{TS}$) and the global ($\overline{GBL}$) bus signals. The only exception to this qualified snoop request is for four address-only transactions; the 604 also snoops its own TLB invalidate, TLBSYNC, SYNC, and ICBI transactions regardless of the global ($\overline{GBL}$) bit setting.

The 604 drives two snoop status signals, $\overline{ARTRY}$ and $\overline{SHD}$, in response to qualified snoop requests. These signals provide information about the state of the addressed block with respect to 604 for the current bus operation. These signals are described in more detail in this document. The following additional comments apply:

- Any bus transaction that does not have the $\overline{GBL}$ signal asserted can be ignored by all bus snoopers. All such transactions, except the self-snooping transactions, are ignored by the 604.
- Several bus transactions (write with flush, read, and read with intent to modify) are defined twice, once with the TT0 reset and once with it set (for atomic operations). These operations behave in exactly the same manner with respect to bus snooping.
- The receiving processor may assert $\overline{ARTRY}$ in response to any bus transaction as a result of internal conflicts that prevent the appropriate snooping.
- The receiving processor may clear its reservation due to snoop address hit with several bus transactions (write-with-flush, read-with-intent-to-modify, write-with-kill, and kill). The reservation is clear even if the 604 $\overline{ARTRY}$s the particular bus transaction.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page # **Changes**

3.6.4, Page 3-14 The following should replace Figure 3-6 and Figure 8-15 in Chapter 8.

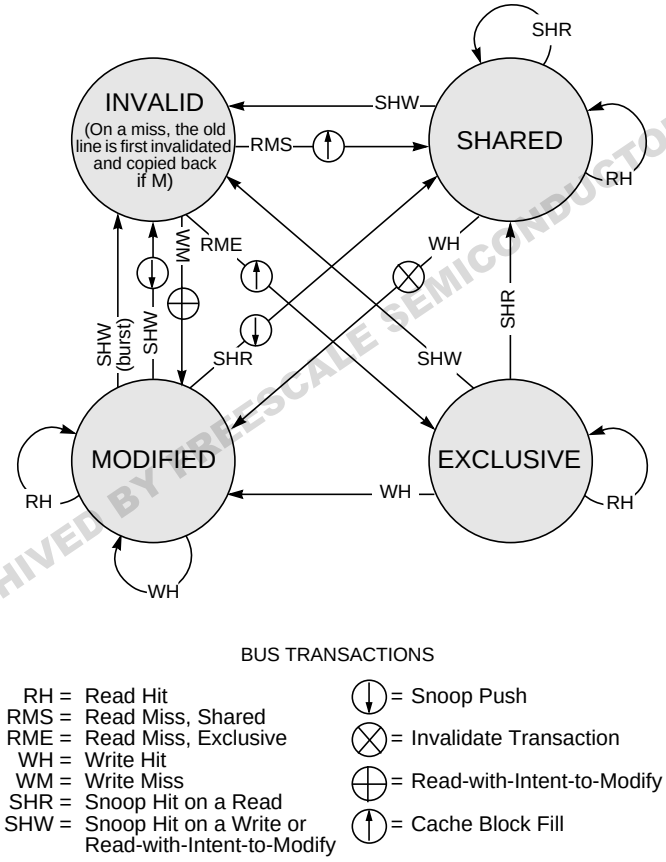


Figure 9-1. MESI Cache Coherency Protocol—State Diagram (WIM = 001)

3.9.4, Page 3-19 Add the following paragraphs:

In a multiprocessor system, a processor can execute an **lwarx** instruction and another processor can broadcast a flush bus operation to the target address of the **lwarx**, invalidating the cache block without canceling the reservation. Therefore, the first processor may broadcast a reservation set (TT = 0x01, address only) tenure without having a valid copy of the reservation address in its data cache.

After a data cache hit for an **lwarx** instruction, the only condition that can cancel the corresponding **lwarx** reservation set transaction is another snoop, which clears the reservation before the transaction wins arbitration to the address bus.

Freescale Semiconductor, Inc.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page #	Changes
	If the processor detects that a snoop flush operation to the reservation address has invalidated the cache for the reservation address between the time at which the **lwarx** hit the cache and the time the **lwarx** reservation set broadcast won arbitration to the address bus, the processor always retries the **lwarx** at the cache even though it still performs the reservation set address tenure. In this case, the retried **lwarx** instruction misses in the cache and causes a read-atomic transaction on the bus. Externally this would be seen as the following: snoop: flush (address A) processor: **lwarx** reservation set operation (address A) processor: read atomic (address A) To avoid this paradox, paradox checking mechanisms should allow an **lwarx** reservation set operation to be broadcast when the processor can have a valid reservation but does not have a valid copy of the **lwarx** target in its data cache.
3.9.6, Page 3-20	Table 3-4, row 4. Add the following to the description of "Kill block": A kill block hit on a cache block marked modified causes a cache block push operation, and then the block is invalidated. Note that if a kill operation hits on a write queue entry, it does not cause that entry to be purged. Instead the kill operation is $\overline{\text{ARTRY}}$d and the entry is pushed to memory.
	Table 3-4, row 5. Add the following to the description of "write-with-kill": A global write-with-kill operation on the bus can cause a loss of memory coherency and make it appear that a program has not executed serially. Note that the 604 (or 604e) never issues a global write-with-kill operation. If data is stored at a memory location and a subsequent store to that address writes different data into the L1 cache, it is possible for the 604 to $\overline{\text{ARTRY}}$ a snooped write-with-kill operation to an address in the same cache block and simultaneously invalidate the L1 cache line for address A. If the 604 attempts to load data from address A, it will miss in the L1 cache and the 604 will arbitrate for the bus. If the 604 wins arbitration over the $\overline{\text{ARTRY}}$d write-with-kill operation, the load operation retrieves the original data before the data for the write-with-kill is written to memory. Since the older data is returned instead of the newer data, it appears that the program is not executed sequentially.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page #	Changes
	A similar scenario occurs when data is in the 604's copy-back buffer, and other data is in the L1 cache. In this scenario, the write-with-kill is $\overline{\text{ARTRY}}$d, the data in the copy-back buffer is pushed to memory and the data in the cache is killed. The subsequent load retrieves from memory the data that had been in the copy-back buffer. The probability of encountering either of these scenarios is increased by performing a **dcbst** to the address before storing the newer data. To avoid this scenario, do not write software that attempts to read from a location that may still be in the L1 cache, and is the target address for a write-with-kill access (for example a DMA operation). This may be done by flushing the block from the cache before the DMA operation is initiated, or by using a software lock to indicate when the DMA operation is complete and the location is safe for reading. Alternatively, use write-with-flush instead of write-with-kill.
3.9.6, Page 3-21	Table 3-4, row 1. Add the following to the entry for "Read-with-intent-to-modify (RWITM) RWITM Atomic": It is now illegal for any snooping device to generate a $\overline{\text{SHD}}$ snoop response without an $\overline{\text{ARTRY}}$ response to an RWITM address tenure. This change is also required for the 604. If the processor sees this illegal snoop response to its RWITM address tenure, it will not respond correctly to snoops to that address until that data is fully loaded into the data cache from the line-fill buffer. For a snoop-read/RWNITC to that address that hits on the line-fill buffer, the processor asserts $\overline{\text{SHD}}$ instead of $\overline{\text{ARTRY}}$. In this case, the processor updates the data cache to be modified and the reading device has a copy marked S (shared). Store operations to the cache block could be lost at this point. For all invalidating snoop operations to that address, the processor asserts no response instead of asserting $\overline{\text{ARTRY}}$. In this case, the processor updates the data cache to be modified while another device could also have a modified copy. The processor's stores to this cache block or another processor's stores to this cache block could be lost.
3.9.6, Page 3-21	Table 3-4, row 7. The first sentence in the "Response" column for "Read-with-no-intent-to-cache (RWNITC)" should read as follows: An RWNITC operation is issued by a bus-attached device as TT0–TT4 = 0b01011.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page # **Changes**

3.9.6, Page 3-22 Add the following to Table 3-4:

Transaction	Response
ICBI	An ICBI transaction is issued by a processor that executes an **icbi** instruction. All copies of the addressed block in bus-attached instruction caches are invalidated. In this transaction, a 604 could assert $\overline{\text{ARTRY}}$ in response to its own transaction.

3.10, Page 3-23 The introduction to Table 3-6 should include the following:

Table 3-6 provides information about general cache conditions and does not take into account all possible interactions and conditions. In particular, Table 3-6 does not address many of the conditions that might be encountered in an in-line L2 cache implementation.

3.10, Page 3-25, 26 Table 3-6. Entries where **lwarx** is the action that causes "**lwarx** reservation set" bus operation, should also say the following:

It is possible for a snoop invalidate operation that invalidates both the cache block and the reservation to preempt the operation and cause the 604 to generate a "read atomic" operation instead. It is also possible that between the time that the **lwarx** instruction hits in the cache and the **lwarx** reservation set is broadcast that a flush snoop operation can remove the cache block from the cache without canceling the reservation. In this case, the **lwarx** broadcast still occurs even through the cache block is not in the data cache.

4.1, Page 4-4 Table 4-2. The entry for floating-point unavailable exception should read as follows:

The floating-point unavailable exception is implemented as defined in the PowerPC architecture.

4.5.1, Page 4-13 The following should be added to the end of this section:

Asserting $\overline{\text{SRESET}}$ causes the 604 to perform a system reset exception. $\overline{\text{SRESET}}$ is an edge-sensitive signal that may be asserted and deasserted asynchronously, provided the minimum pulse width specified in the *PowerPC 604 RISC Microprocessor Hardware Specifications* is met. This exception modifies the MSR, SRR0, and SRR1, as described in *The Programming Environments Manual*. Unlike hard reset, soft reset does not directly affect the states of output signals. Attempts to use $\overline{\text{SRESET}}$ during a hard reset sequence or while the JTAG logic is non-idle cause unpredictable results. Processing interrupted by a $\overline{\text{SRESET}}$ can be restarted.

A hard reset is initiated by asserting $\overline{\text{HRESET}}$. Hard reset is used primarily for power-on reset (POR), but can also be used to restart a running processor. The $\overline{\text{HRESET}}$ signal should be asserted during power up and must remain asserted for a period that allows the PLL

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page #	Changes
	to achieve lock and the internal logic to be reset. This period is specified in the *PowerPC 604 RISC Microprocessor Hardware Specifications*. The 604 internal state after the hard reset interval is defined in Table 2-6a. If $\overline{\text{HRESET}}$ is asserted for less than this amount of time, the results are not predictable. If $\overline{\text{HRESET}}$ is asserted during normal operation, all operations cease and the machine state is lost.
4.5.2.1, Page 4-15	Table 4-8. The entries for SRR1 bits 10 and 11 should be swapped, as follows: 10. Set when an instruction cache parity error is detected, otherwise zero 11. Set when a data cache parity error is detected, otherwise zero The entry for SRR1[30] should read as follows: SRR1[30] is zero for $\overline{\text{APE}}$, $\overline{\text{DPE}}$, instruction or data cache parity error, or $\overline{\text{TEA}}$. For $\overline{\text{MCP}}$ or other conditions, SRR1[30] is set to value of MSR[30]. If $\overline{\text{MCP}}$ and $\overline{\text{TEA}}$ are asserted simultaneously, SRR1[30] is zero and the exception is not recoverable.
4.5.6, Page 4-17	In bullets 3 and 4, parenthetical comments that currently say "(and double-precision values aligned on a double-word boundary)" should read "(and double-precision values not aligned on a double-word boundary)."
5.1.3, Page 5-10	Replace the last paragraph in this section with the following: Real addressing mode translation occurs when address translation is disabled; in this case the physical address generated is identical to the effective address. Instruction and data address translation is enabled with the MSR[IR] and MSR[DR] bits, respectively. Thus when the processor generates an access, and the corresponding address translation enable bit in MSR (MSR[IR] for instruction accesses and MSR[DR] for data accesses) is cleared, the resulting physical address is identical to the effective address and all other translation mechanisms are ignored.
5.1.4, Page 5-11	Replace the first sentence after Table 5-2 with the following: The operating system programs whether instructions can be fetched from an area of memory by appropriately using the no-execute option provided in the segment register.
5.1.6.3, Page 5-16	Change this heading to 5.1.6.2.1.
5.1.6.4, Page 5-16	Change this heading to 5.1.6.2.2.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page #	Changes
5.1.7, Page 5-17	Table 5-3, row 3. The reference in the description column for “Page Protection Violation” should refer to “Page Memory Protection” instead of “Block Memory Protection.” Add a note to the entry in the “Exception” column that DSISR[6] is also set for store operations.
5.2, Page 5-20	Add a note at the end of the section that the PowerPC architecture states the following: For data accesses performed in real addressing mode (MSR[DR] = 0), the WIMG bits are assumed to be 0b0011 (the data is write-back, caching is enabled, memory coherency is enforced, and memory is guarded). For instruction accesses performed in real addressing mode (MSR[IR] = 0), the WIMG bits are assumed to be 0b0001 (the data is write-back, caching is enabled, memory coherency is not enforced, and memory is guarded).
5.4.5, Page 5-29	Step 2. In the second sentence, change “PTE reads should occur...” to “PTE reads occur...” Step 5. In the second sentence, delete the word ‘typically’.
5.4.6, Page 5-33	Delete from the second full paragraph everything from the third sentence (beginning with “In the examples below...”) to the end of the paragraph. Delete the third full paragraph (beginning with “On single-processor systems, ...”).
6.4.4, Page 6-24	Add the following note to the last paragraph on the page: Note that clearing HID0[29] disables the use of the branch history table.
6.4.4.1.2, Page 6-27	Replace the description of the timing example for Figure 6-10 with the following: 0. In cycle 1, instructions 0 and 1 are in decode stage, but instructions 2–5 cannot be fetched because of a miss in the BTAC. 1. In cycle 2, instructions 0 and 1 are dispatched and instructions 2–5 are located and fetched. 2. In cycle 3, instructions 0 and 1 are in the execute stage and instructions 2–5 are in the decode stage, and the instruction timing proceeds as normal.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page #	Changes
7.1, Page 7-3	The DRVMOD input signals shown in the Processor Configuration group are not described. These signals must be pulled up to VDD for the 604/604e to operate in accordance with the hardware specifications.
7.2.1.1, Page 7-4	Bus Request ($\overline{BR}$)—Output Replace the last sentence in the "Timing Comments—Negation" with the following: It is also negated for at least one cycle after the assertion of $\overline{ARTRY}$, unless that processor was responsible for the assertion of $\overline{ARTRY}$ due to the need to perform a cache block push for that snoop operation.
7.2.1.3.1, Page 7-5	Address Bus Busy ($\overline{ABB}$)—Output Add the following comment to "Timing Comments—High Impedance": Occurs during fractional portion of the bus cycle in which $\overline{ABB}$ is negated. $\overline{ABB}$ is guaranteed by design to be high impedance by the end of the cycle in which it is negated.
7.2.2.1.1, Page 7-6	Transfer Start ($\overline{TS}$)—Output Replace the description of "State Meaning—Negated" with the following: Negated—Has no special meaning. However, $\overline{TS}$ is negated during an entire direct-store address tenure. Replace the description for "Timing Comments—High Impedance" with the following: High Impedance—Occurs one bus clock cycle after the negation of $\overline{TS}$. For the 604, the $\overline{TS}$ negation is only one bus cycle long, regardless of the $\overline{TS}$-to-$\overline{AACK}$ delay.
7.2.2.1.2, Page 7-6	Transfer Start ($\overline{TS}$)—Input Replace the description of "Timing Comments—Assertion" with the following: Assertion—May occur at any time outside of the cycles that define the window of an address tenure. This window is marked by either the interval that includes the cycle of a previous $\overline{TS}$ assertion through the cycle after $\overline{AACK}$.
7.2.2.2.1, Page 7-6, 7	Extended Address Transfer Start ($\overline{XATS}$)—Output Replace the description of "State Meaning—Negated" with the following: Negated—Has no special meaning; however, $\overline{XATS}$ remains negated during an entire memory address tenure.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page # **Changes**

Replace the description of “Timing Comments—High Impedance” with the following:

High Impedance—Occurs one bus clock cycle after the negation of $\overline{\text{XATS}}$. For the 604, the $\overline{\text{XATS}}$ negation is only one bus-cycle long, regardless of the $\overline{\text{XATS}}$-to-$\overline{\text{AACK}}$ delay.

7.2.2.2.2, Page 7-7 Extended Address Transfer Start ($\overline{\text{XATS}}$)—Input

Replace the description of “Timing Comments—Assertion” with the following:

Assertion—May occur at any time outside of the cycles that define the window of an address tenure. This window is marked by either the interval that includes the cycle of a previous $\overline{\text{XATS}}$ assertion through the cycle after $\overline{\text{AACK}}$ or by the cycles in which $\overline{\text{ABB}}$ is asserted for a previous address tenure, whichever is greater.

7.2.3.3, Page 7-9 Address Parity Error ($\overline{\text{APE}}$)—Output

Replace the first sentence in “State Meaning—Asserted” with the following:

Asserted—Indicates incorrect address bus parity has been detected by the processor on a snoop of a transaction type that the processor recognizes and can respond to.

7.2.4.1.2, Page 7-10 Transfer Type (TT0–TT4)—Input

Replace Table 7-1 with the following:

Table 7-1. Transfer Encoding for PowerPC 604 Processor Bus Master

TT0–TT4	604 Bus Master Transaction	Transaction	Transaction Source
00000	Clean block	Address only	Cache operation
00100	Flush block	Address only	Cache operation
01000	SYNC	Address only	Cache operation
01100	Kill block	Address only	Store hit/shared or cache operation
10000	Ordered I/O operation	Address only	**eieio** (The 604 does not snoop **eieio** transactions.)
10100	External control word write	Single-beat write	**ecowx** (The 604 does not snoop **ecowx** transactions.)
11000	TLB invalidate	Address only	**tlbie**
11100	External control word read	Single-beat read	**eciwx** (The 604 does not snoop **eciwx** transactions.)
00001	**lwarx** reservation set	Address only	**lwarx** operation that hit in the cache at the time of its execution. The cache block may have been flushed between execution of the **lwarx** and broadcast of the reservation set operation. Note that the 604 does not snoop **lwarx** reservation set operations.
00101	Reserved	Address only	N/A

Freescale Semiconductor, Inc.
ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.
For More Information On This Product,
Go to: www.freescale.com

Section #/Page # **Changes**

Table 7-1. Transfer Encoding for PowerPC 604 Processor Bus Master (Continued)

TT0–TT4	604 Bus Master Transaction	Transaction	Transaction Source
01001	TLBSYNC	Address only	**tlbsync**
01101	ICBI	Address only	N/A
1xx01	Reserved	—	N/A (The 604 does not snoop.)
00010	Write with flush	Single-beat write or burst	Caching-inhibited or write-through store
00110	Write with kill	Single-beat write or burst	Cast-out, snoop copy-back, **dcbf**, or **dcbst** instruction that hit on modified data.
01010	Read	Single-beat read or burst	Cacheable load miss—cacheable instruction miss, cache-inhibited load, cache-inhibited instruction fetch.
01110	Read with intent to modify	Burst	Store miss
10010	Write with flush atomic	Single-beat write	**stwcx.**
10110	Reserved	N/A	N/A
11010	Read atomic	Single-beat read or burst	**lwarx**
11110	Read with intent to modify atomic	Burst	**stwcx.** miss with valid reservation
00011	Reserved	—	N/A (The 604 does not snoop.)
00111	Reserved	—	N/A (The 604 does not snoop.)
01011	Read with no intent to cache	Single-beat read or burst	N/A
01111	Reserved	—	N/A (The 604 does not snoop.)
1xx11	Reserved	—	N/A (The 604 does not snoop.)

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page # **Changes**

7.2.4.4, Page 7-13 Replace Table 7-3 with the following:

Table 7-3. Transfer Code Signal Encoding for the PowerPC 604 Processor

Transfer Type	$\overline{WT}$[1]	TC0-TC2	$\overline{BR}$ Asserted[2, 3]	From Copyback Buffer	$\overline{TS}$ after $\overline{ARTRY}$d Snoop[4]	Final MESI State[5]	Comments
Write with kill	1	100	Never	Always	Don't care	I	Cache copy-back
	0	xx0	No	Yes	Yes	M, E, S or I	Could be cache copy-back, block clean (**dcbst**), or block flush (**dcbf**) To distinguish between these operations, this transaction must be $\overline{ARTRY}$d. This transaction eventually returns (before anything but another snoop push directly from the data cache) indicating another $\overline{WT}$/TC code combination.
		100	No	Yes	No	I	Block flush (**dcbf**)
		000	No	Yes	No	M, E, or I	Block clean (**dcbst**) The **dcbst** instruction changes the data cache state to E when the modified line is placed in the copy-back buffer queue. Before the low-priority copy-back buffer entry successfully completes its address tenure, the data cache line state can be changed to M by a subsequent store to that line; it can be changed to I by either a subsequent **dcbi** instruction or by a cache-miss.
		010	Yes	No	Don't care	S or I	Snoop push[6] directly from data cache (read or read-atomic) The read or read-atomic snoop changes the data cache state to S when the modified line is placed in the snoop push buffer queue. Before the snoop push buffer successfully completes its address tenure, the data cache line state can be changed to I by either a subsequent **dcbi** instruction or cache-miss.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page # Changes

Table 7-3. Transfer Code Signal Encoding for the PowerPC 604 Processor (Continued)

Transfer Type	$\overline{WT}$[1]	TC0-TC2	$\overline{BR}$ Asserted [2, 3]	From Copyback Buffer	$\overline{TS}$ after $\overline{ARTRY}$d Snoop [4]	Final MESI State[5]	Comments
Write with kill	0	010	Yes	Yes	Don't care	S or I	Snoop push[6] from copy-back buffer (read or read-atomic) In this case, the processor keeps a shared copy in the data cache if this copy-back buffer contained a block clean (**dcbst**) transaction. If the copy-back buffer contained a block flush (**dcbf**) or a cache copy-back transaction, the processor has no valid copy of this line in its data cache after this transaction completes successfully. To determine whether the processor has kept a shared copy or has invalidated this line, this transaction must be $\overline{ARTRY}$d. If this transaction originated from the copy-back buffers and no new snoops are given to the processor, the transaction immediately comes back as the next $\overline{TS}$ and indicates a DCBF, DCBST, or copy-back $\overline{WT}$/TC code. If the transaction comes back as a snoop push read, it came from the data cache.
		100	Yes	No	Don't care	I	Snoop push[6] directly from data cache (RWITM, RWITM-atomic, flush, write with flush, write with flush-atomic, or kill)
		100	Yes	Yes	Don't care	I	Snoop push[6] from copy-back buffers (RWITM, RWITM-atomic, flush, write with flush atomic, write with flush, write with kill, or kill)
		000	Yes	No	Don't care	M, E, or I	Snoop push[6] from data cache (clean or RWNITC). The clean or RWNITC snoop changes the data cache state to E when the modified line is placed in the snoop push buffer queue. Before the snoop push buffer successfully completes its address tenure, the data cache line state can be changed to M by a subsequent store to that line, or it can be changed to I by either a subsequent DCBI instruction or cache miss.
		000	Yes	Yes	Don't care	M, E, or I (if **dcbst** in buffer) I (if cache copy-back or **dcbf** in buffer)	Snoop push[6] from copy-back buffers (clean or RWNITC) If this snoop hit on a block flush (**dcbf**) or a cache copy-back in the copy-back buffers, the processor does not have a valid copy of this address after this transaction completes successfully. If this snoop hit on a block store (**dcbst**) in the copy-back buffers, the processor can keep an exclusive copy of the cache block.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page # Changes

Table 7-3. Transfer Code Signal Encoding for the PowerPC 604 Processor (Continued)

Transfer Type	$\overline{WT}$[1]	TC0-TC2	$\overline{BR}$ Asserted [2, 3]	From Copyback Buffer	$\overline{TS}$ after $\overline{ARTRY}$d Snoop [4]	Final MESI State[5]	Comments
Kill block	x	100	Never	No	Don't care	I	Kill block deallocate (**dcbi**)
	1	000				M	Kill block & allocate no castout required (**dcbz**)
	1	001					Kill block & allocate castout required (**dcbz**)
	1	000					Kill block; write to block marked S
Read	W[8]	0x0	Never	No	Don't care	E or S	Data read no castout required The cache state is S if $\overline{SHD}$ was asserted to the processor for a read or read-atomic transaction. If $\overline{SHD}$ was not asserted or if the transaction was an RWITM or RWITM-atomic transaction, the cache state is E.
	W	0x1				E or S	Data read castout required The cache state is S if $\overline{SHD}$ was asserted to the processor for a read or read-atomic transaction. If $\overline{SHD}$ was not asserted, or if the transaction was an RWITM or RWITM-atomic transaction, the cache state is E.
	W	1x0				Valid in instruction cache	Instruction read
ICBI	x	100	Never	No	Don't care	Invalid in instruction cache	Kill block deallocate (**icbi**[9])

[1] The value shown in the $\overline{WT}$ column reflects the actual logic value seen on the signal (active low).
[2] The window of opportunity for the assertion of $\overline{BR}$ is defined as the second cycle after $\overline{AACK}$ if $\overline{ARTRY}$ were asserted the cycle after $\overline{AACK}$.
[3] The full condition for this column is "The $\overline{BR}$ corresponding to this transaction was asserted in the window of opportunity for the last snoop to this address."
[4] The full condition for this column is "This transaction is the first $\overline{TS}$ asserted by this processor after one or more $\overline{ARTRY}$d snoop transactions and the address of this transaction matches the address of at least one of those $\overline{ARTRY}$d snoop transactions."
[5] This column reflects the final MESI state in the processor of the line referenced by this transaction after the transaction completes successfully without $\overline{ARTRY}$.
[6] This snoop push is guaranteed to push the most recently modified data in the processor. No more snoop operations are required to ensure that this snoop has been fully processed by the processor.
[7] READ in this case encompasses all of read or RWITM, normal or atomic.
[8] W = write-through bit from translation. $\overline{WT}$ is active-high and is the inverse of the setting of the W bit.
[9] **icbi** is distinguished from kill block by assertion of TT4.

7.2.5.1, Page 7-16 Address Acknowledge ($\overline{AACK}$)—Input
Replace the description of "State Meaning—Asserted" with the following:

Asserted—Indicates that the address phase of a transaction is complete. The address bus will go to a high-impedance state on the next bus clock cycle. The processor samples $\overline{ARTRY}$ on the bus

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.
Freescale Semiconductor, Inc.
For More Information On This Product,
Go to: www.freescale.com

Section #/Page # Changes

clock cycle following the assertion of $\overline{\text{AACK}}$. The 604 also supports sampling of $\overline{\text{ARTRY}}$ as early as the second cycle after $\overline{\text{TS}}$.

7.2.5.2.1, Page 7-16 Address Retry ($\overline{\text{ARTRY}}$)—Output

Replace the last sentence in "State Meaning—Asserted" with the following:

If the processor needs to update memory as a result of the snoop that caused the retry, the processor asserts $\overline{\text{BR}}$ in the window of opportunity for that snoop. The window of opportunity is defined as the second cycle after $\overline{\text{AACK}}$ if $\overline{\text{ARTRY}}$ was asserted the cycle after $\overline{\text{AACK}}$.

Replace "Timing Comments—Assertion" with the following:

Assertion—Asserted the second bus cycle after the assertion of $\overline{\text{TS}}$ if a retry is required. Thus, when a retry is required, there is only one empty cycle between the assertion of $\overline{\text{TS}}$ and the assertion of $\overline{\text{ARTRY}}$.

Add the following to "Timing Comments—Negation":

$\overline{\text{ARTRY}}$ becomes high impedance for at least one half bus cycle, then is driven high for approximately one bus cycle. $\overline{\text{ARTRY}}$ is then guaranteed by design to become high impedance at latest by the start of third cycle after $\overline{\text{AACK}}$.

7.2.5.2.2, Page 7-17 Address Retry ($\overline{\text{ARTRY}}$)—Input

Remove the last sentence in "State Meaning—Asserted."

7.2.5.3.1, Page 7-17 Shared ($\overline{\text{SHD}}$)—Output

Replace the description of "State Meaning—Asserted" with the following:

Asserted—If $\overline{\text{ARTRY}}$ is not asserted, indicates that after this transaction completes successfully, the master will keep a valid shared copy of the address or that a reservation exists on this address. If $\overline{\text{SHD}}$ is asserted with $\overline{\text{ARTRY}}$ for a given snooping master, this indicates that the snoop scored a hit on modified data that will be pushed from that master as its next address transaction.

Replace the description of "State Meaning—Negated/High Impedance" with the following:

Negated/High Impedance—Indicates that after this address transaction completes successfully, the processor will not have a valid copy of the snooped address.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page # **Changes**

7.2.5.3.2, Page 7-18 Shared ($\overline{\text{SHD}}$)—Input

Replace the description of “State Meaning—Negated” with the following:

Negated—If $\overline{\text{ARTRY}}$ is not asserted, indicates that for a self-generated read or read-atomic transaction, the master can allocate the incoming cache block as exclusive-unmodified.

7.2.6.1, Page 7-18 Data Bus Grant ($\overline{\text{DBG}}$)—Input

Add the following to the description of “State Meaning—Asserted”:

The master achieves the position of master of the data bus (that is, has achieved a qualified data bus grant) when the following conditions are met:

The data bus is not bus busy ($\overline{\text{DBB}}$ is negated). (This condition does not apply to the 604 or 604e in fast-L2 mode.)

$\overline{\text{DRTRY}}$ is negated. (This condition does not apply to the 604 in fast-L2 mode or the 604e in fast-L2 or no-$\overline{\text{DRTRY}}$ mode.)

$\overline{\text{ARTRY}}$ is negated if $\overline{\text{ARTRY}}$ applies to the associated address tenure.

Replace the description of “Timing Comments—Assertion” with the following:

Assertion—May occur any time to indicate that the processor or other master is free to assume the position of master of the data bus. The earliest it is sampled by the processor is the same cycle $\overline{\text{TS}}$ or $\overline{\text{XATS}}$ is asserted.

Note that the 604 timing requirements for $\overline{\text{DBG}}$ in fast-L2 mode are different. For the 604 in fast-L2 mode, $\overline{\text{DBG}}$ must be asserted for exactly one cycle per data bus tenure, the cycle before the data tenure is to begin. The system is not allowed to assert $\overline{\text{DBG}}$ earlier than one cycle before the data tenure is to commence, nor to park $\overline{\text{DBG}}$, nor to assert it for multiple consecutive cycles. $\overline{\text{DBB}}$ does not participate in determining a qualified data bus grant. Therefore, the system is required to assert $\overline{\text{DBG}}$ in a manner such that different masters don't collide on data tenures. Also, the system must assert $\overline{\text{DBG}}$ in a manner such that 604 data tenures are complete before providing another $\overline{\text{DBG}}$. If a $\overline{\text{DBG}}$ is given early to the 604 in fast-L2 mode, the processor drops the current data tenure prematurely in the next cycle and begins the subsequent data tenure if a subsequent data tenure is pending.

The 604e has less restrictive timing requirements in fast-L2 mode. For the 604e in fast-L2 mode, $\overline{\text{DBG}}$ must be asserted no earlier than the cycle before 604e's data tenure is to commence only when

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page #	Changes
	another master currently owns the data bus (that is, when $\overline{\text{DBB}}$ would normally be asserted for a data tenure). If no other masters currently own the data bus (asserting $\overline{\text{DBB}}$), the 604e allows the system to park $\overline{\text{DBG}}$ on 604e. $\overline{\text{DBB}}$ is still an output-only signal in fast-L2 Mode (that is, $\overline{\text{DBB}}$ does not participate in determining qualified data bus grant), requiring the system to use $\overline{\text{DBG}}$ to ensure that different masters don't collide on data tenures. If the system attempts to stream any back-to-back data tenures by asserting $\overline{\text{DBG}}$ with the final $\overline{\text{TA}}$ of the first data tenure, the processor will accept the $\overline{\text{DBG}}$ as a qualified data bus grant only if the current data tenure is a burst read and the next data tenure is a burst read. The 604e will not allow the system to stream any two other types of data tenures.
7.2.6.2, Page 7-19	Data Bus Write Only ($\overline{\text{DBWO}}$)—Input Replace the description of “Timing Comments—Negation” with the following: Negation—May occur any time after a qualified data bus grant and before the next qualified data bus grant.
7.2.6.3.1, Page 7-19	Data Bus Busy ($\overline{\text{DBB}}$)—Output Replace the description of “State Meaning—Negated” with the following: Negated—Indicates that the 604 is not using the data bus, unless the data tenure is being extended by the assertion of $\overline{\text{DRTRY}}$. Note that for the 604e in no-$\overline{\text{DRTRY}}$ mode, $\overline{\text{DRTRY}}$ is tied asserted and is ignored. Replace the description of “Timing Comments—Negation” with the following: Negation—Occurs for a fractional bus clock cycle following the assertion of the final $\overline{\text{TA}}$.
7.2.7.4, Page 7-22	Data Bus Disable ($\overline{\text{DBDIS}}$)—Input Replace the first sentence in the description of “State Meaning—Asserted” with the following: Indicates for a write transaction that the processor must release the data bus (DH[0–31] and DL[0-31]) and the data bus parity (DP[0–7]) to high impedance during the following cycle.

ARCHIVED BY FREESCALE SEMICONDUCTOR INC

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page #	Changes
7.2.8.3, Page 7-24	Transfer Error Acknowledge ($\overline{\text{TEA}}$)—Input Add the following note to "State Meaning—Asserted": Note that If $\overline{\text{TEA}}$ is asserted during a direct-store transaction, the machine check or checkstop action of the $\overline{\text{TEA}}$ is delayed and the following direct-store transactions continue until all data transfers from the direct-store segment complete. The bus agent that asserts $\overline{\text{TEA}}$ must assert $\overline{\text{TEA}}$ for every direct-store data tenure including the last one. The processor takes a machine check or a checkstop no sooner than the last direct-store data tenure has been terminated by the assertion of $\overline{\text{TEA}}$. The load or store reply is not necessary after the last data tenure has received a $\overline{\text{TEA}}$ assertion.
7.2.9.3, Page 7-26	Machine Check Interrupt ($\overline{\text{MCP}}$)—Input In the first sentence under "State Meaning—Asserted," both instances of MSR[EE] should be replaced by MSR[ME].
7.2.10.2, Page 7-28	Reservation ($\overline{\text{RSRV}}$)—Output Replace the "Timing Comments—Assertion/Negation" with the following: Assertion—Occurs synchronously one bus clock cycle after the execution of an **lwarx** instruction that sets the internal reservation condition. On 604 and 604e, the $\overline{\text{RSRV}}$ signal is asserted as late as the fourth cycle after $\overline{\text{AACK}}$ for a read-atomic operation if the **lwarx** instruction requires a read-atomic operation. Negation—Occurs synchronously one bus clock cycle after the execution of an **stwcx.** instruction that clears the reservation or as late as the second bus cycle after a $\overline{\text{TS}}$ for a snoop that clears the reservation.
7.2.13, Page 7-32	The information regarding nap, doze, and sleep modes provided in Section 1.3.7.1, "Power Management," on page 30 of this document should be added as Section 7.2.13.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page # **Changes**

8.3.2.4, Page 8-16 Replace Table 8-5 with the following:

Table 8-5. Misaligned Data Transfers (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	—	—	—	—
Misaligned—first access	0 1 1	0 0 1		A	A	A	—	—	—	—
second access	0 0 1	1 0 0	—	—	—	—	A	—	—	—
Misaligned—first access	0 1 0	0 1 0	—	—	A	A	—	—	—	—
second access	0 1 0	1 0 0	—	—	—	—	A	A	—	—
Misaligned—first access	0 0 1	0 1 1	—	—	—	A	—	—	—	—
second access	0 1 1	1 0 0	—	—	—	—	A	A	A	—
Aligned	1 0 0	1 0 0	—	—	—	—	A	A	A	A
Misaligned—first access	0 1 1	1 0 1	—	—	—	—	—	A	A	A
second access	0 0 1	0 0 0	A	—	—	—	—	—	—	—
Misaligned—first access	0 1 0	1 1 0	—	—	—	—	—	—	A	A
second access	0 1 0	0 0 0	A	A	—	—	—	—	—	—
Misaligned—first access	0 0 1	1 1 1	—	—	—	—	—	—	—	A
second access	0 1 1	0 0 0	A	A	A	—	—	—	—	—

A: Byte lane used
—: Byte lane not used

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page # **Changes**

8.3.2.4, Page 8-16 Add the following table after Table 8-5:
Table 8-5b shows the signal configuration for three-word accesses:

Table 8-5b. Misaligned Data Transfer—Three-Byte Examples

Transfer Size	TSIZ0	TSIZ1	TSIZ2	A29–A31	Data Bus Byte Lane(s)							
					0	1	2	3	4	5	6	7
Three Bytes	0	1	1	0 0 0	A	A	A	—	—	—	—	—
	0	1	1	0 0 1	—	A	A	A	—	—	—	—
	0	1	1	0 1 0	—	—	A	A	A	—	—	—
	0	1	1	0 1 1	—	—	—	A	A	A	—	—
	0	1	1	1 0 0	—	—	—	—	A	A	A	—
	0	1	1	1 0 1	—	—	—	—	—	A	A	A
First transfer—two bytes	0	1	0	1 1 0	—	—	—	—	—	—	A	A
Second transfer—one byte	0	0	1	0 0 0	A	—	—	—	—	—	—	—
First transfer—one byte	0	0	1	1 1 1	—	—	—	—	—	—	---	A
Second transfer—two bytes	0	1	0	0 0 0	A	A	—	—	—	—	—	—

8.4.5, Page 8-31 See entry for Section 3.6.4.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page # **Changes**

8.5, Page 8-36 Replace Figure 8-20 with the following:

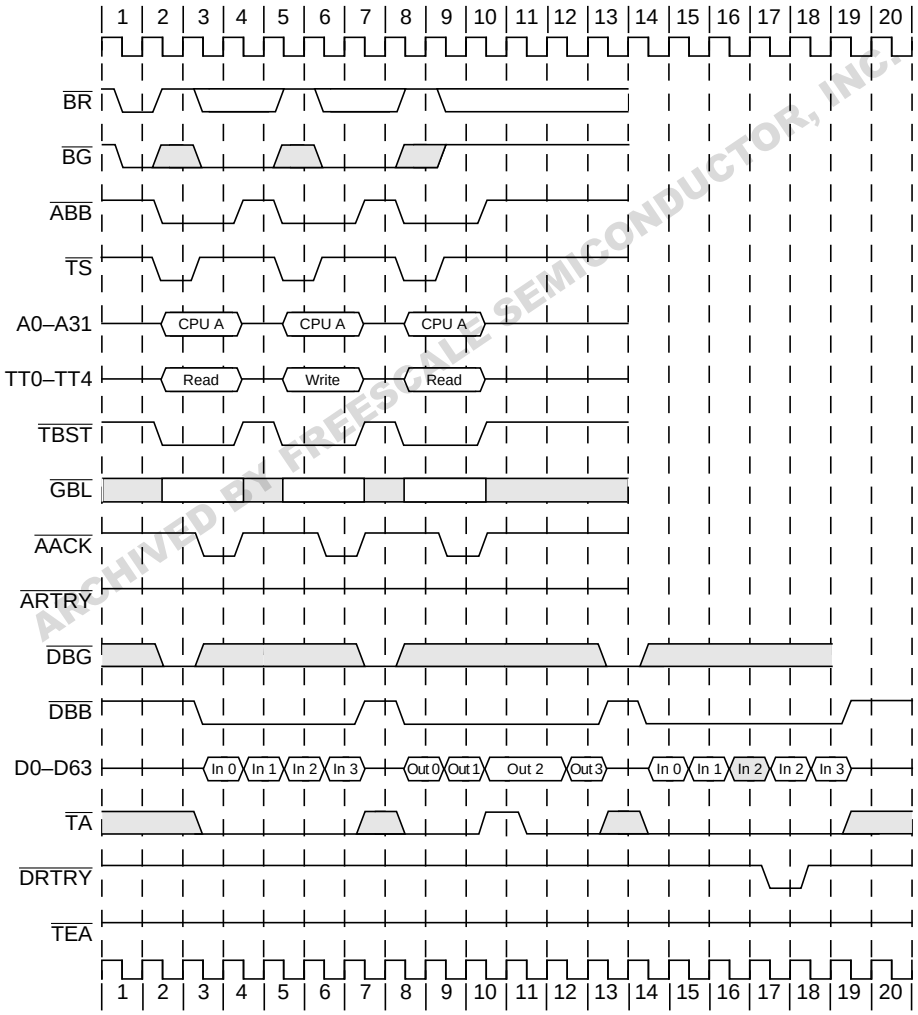


Figure 8-20. Burst Transfers with Data Delay Controls

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page # **Changes**

8.5, Page 8-37 Replace Figure 8-21 with the following:

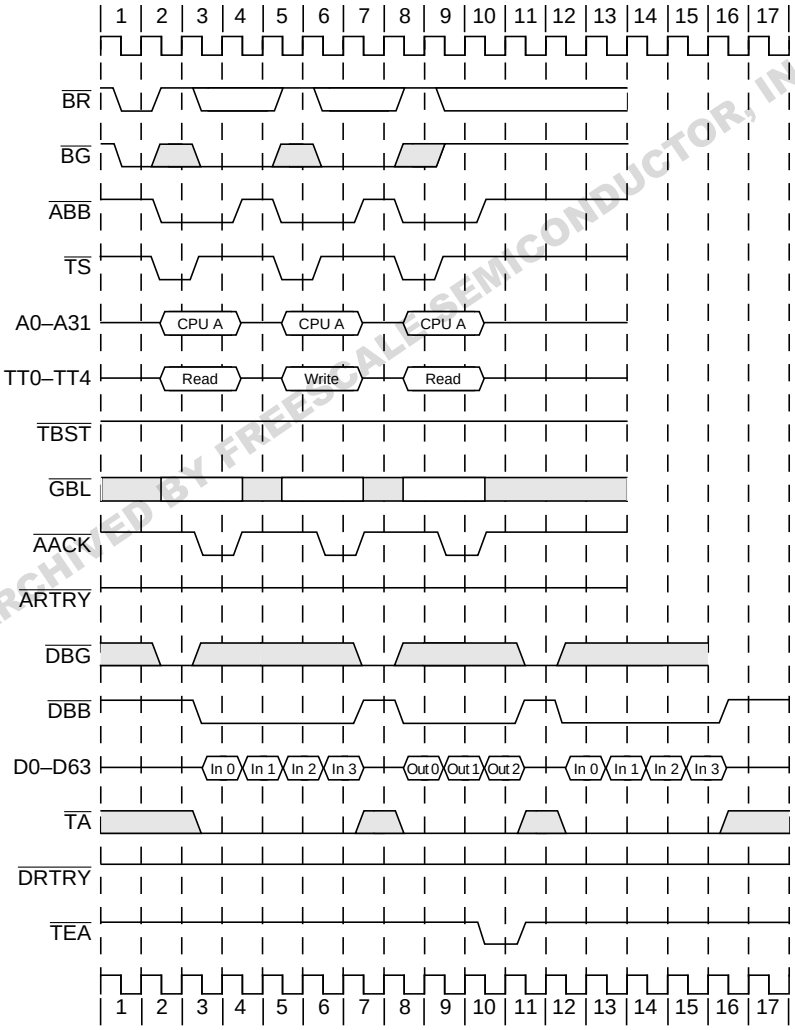


Figure 8-21. Use of Transfer Error Acknowledge ($\overline{\text{TEA}}$)

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Section #/Page #	Changes
8.7.1, Page 8-49	Add the following paragraph as paragraph 2 on this page: In fast-L2 mode, an external device must never assert $\overline{\text{ARTRY}}$ after the cycle of the first $\overline{\text{TA}}$ assertion. Thus, if $\overline{\text{ARTRY}}$ is always asserted by an external device, at latest, the second cycle after $\overline{\text{TS}}$, $\overline{\text{TA}}$ can be asserted by the system as early as the second cycle after $\overline{\text{TS}}$ (with the first cycle of $\overline{\text{ARTRY}}$). Add the following paragraph as last paragraph in this section: It is assumed that systems using fast-L2/data-streaming mode would be running the 604 bus interface at its upper frequency limits for which the cycle time is very short and the partial precharge of $\overline{\text{ABB}}$ and $\overline{\text{DBB}}$ might make it difficult to guarantee that the precharge is successful enough that other devices would see a valid precharge value at the end of the precharge cycle. This timing problem can be solved by not connecting or using $\overline{\text{ABB}}$/$\overline{\text{DBB}}$ in the system design since this design can be done fairly easily.
8.8.1, Page 8-50	The reference to MSR[EE] should say MSR[ME].
9.1, Page 9-2	(604 only) Add the following sentence to the fifth paragraph. Software must always reset ENINT after taking a performance monitor interrupt. This is not the case for the 604e.
9.1.1.1.2, Page 9-5	The entry for 00 0110 should include **dcbz** misses. The entry for 00 1011 should read as follows: Number of **icbi** instructions executed by the processor plus the number of **icbi** instructions snooped from the bus. The **icbi** instruction may not hit in the cache.
9.1.2.2.3, Page 9-11	Change the first sentence of the first bullet to read “Not all load and store operations are monitored when a threshold event is selected in PMC1.” Add the following bullet: If L2_INT is not connected to any source (negated or to an L2 controller) the results obtained from the threshold events 9, 10, 23, and 24 of PMC1 are undefined.
A.1, Page A-8 A.2, Page A-12 A.3, Page A-26 A.4, Page A-33 A.5, Page A-44	Tables A-1, A-2, A-29, A-36, and A-46. The **tlbie** instruction is incorrectly listed as not implemented on the 604. However, the 604 does not implement the **tlbia** instruction.

ARCHIVED BY FREESCALE SEMICONDUCTOR, INC.

Freescale Semiconductor, Inc.

For More Information On This Product,
Go to: www.freescale.com

Freescale Semiconductor, Inc.

MOTOROLA INTELLECTUAL PROPERTY OF FREESCALE SEMICONDUCTOR, INC.

Information in this document is provided solely to enable system and software implementers to use PowerPC microprocessors. There are no express or implied copyright or patent licenses granted hereunder by Motorola or IBM to design, modify the design of, or fabricate circuits based on the information in this document.

The PowerPC 604 and 604e microprocessors embody the intellectual property of Motorola and of IBM. However, neither Motorola nor IBM assumes any responsibility or liability as to any aspects of the performance, operation, or other attributes of the microprocessor as marketed by the other party or by any third party. Neither Motorola nor IBM is to be considered an agent or representative of the other and neither has assumed, created, or granted hereby any right or authority to the other, or to any third party, to assume or create any express or implied obligations on its behalf. Information such as errata sheets and data sheets, as well as sales terms and conditions such as prices, schedules, and support, for the product may vary as between parties selling the product. Accordingly, customers wishing to learn more information about the products as marketed by a given party should contact that party.

Both Motorola and IBM reserve the right to modify this manual and/or any of the products as described herein without further notice. **NOTHING IN THIS MANUAL, NOR IN ANY OF THE ERRATA SHEETS, DATA SHEETS, AND OTHER SUPPORTING DOCUMENTATION, SHALL BE INTERPRETED AS THE CONVEYANCE BY MOTOROLA OR IBM OF AN EXPRESS WARRANTY OF ANY KIND OR IMPLIED WARRANTY, REPRESENTATION, OR GUARANTEE REGARDING THE MERCHANTABILITY OR FITNESS OF THE PRODUCTS FOR ANY PARTICULAR PURPOSE**. Neither Motorola nor IBM assumes any liability or obligation for damages of any kind arising out of the application or use of these materials. Any warranty or other obligations as to the products described herein shall be undertaken solely by the marketing party to the customer, under a separate sale agreement between the marketing party and the customer. In the absence of such an agreement, no liability is assumed by Motorola, IBM, or the marketing party for any damages, actual or otherwise.

"Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals," must be validated for each customer application by customer's technical experts. Neither Motorola nor IBM convey any license under their respective intellectual property rights nor the rights of others. Neither Motorola nor IBM makes any claim, warranty, or representation, express or implied, that the products described in this manual are designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the product could create a situation where personal injury or death may occur. Should customer purchase or use the products for any such unintended or unauthorized application, customer shall indemnify and hold Motorola and IBM and their respective officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola or IBM was negligent regarding the design or manufacture of the part.

Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

IBM, the IBM logo and IBM Microelectronics are trademarks of International Business Machines Corporation.
The PowerPC name, PowerPC logotype, PowerPC 604e, PowerPC 604, PowerPC 603e, and PowerPC 603 are trademarks of International Business Machines Corporation used by Motorola under license from International Business Machines Corporation. International Business Machines Corporation is an Equal Opportunity/Affirmative Action Employer.

International Business Machines Corporation:
IBM Microelectronics Division, 1580 Route 52, Bldg. 504, Hopewell Junction, NY 12533-6531; Tel. (800) PowerPC
World Wide Web Address: http://www.chips.ibm.com/products/ppc
http://www.ibm.com

Motorola Literature Distribution Centers:
USA/EUROPE: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036; Tel.: 1-800-441-2447
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

MFAX: RMFAX0@email.sps.mot.com; TOUCHTONE (602) 244-6609
INTERNET: http://Design-NET.com

Technical Information: Motorola Inc. SPS Customer Support Center; (800) 521-6274.
Document Comments: FAX (512) 891-2638, Attn: RISC Applications Engineering.
World Wide Web Address: http://www.mot.com/powerpc/

IBM. MOTOROLA