



*Addendum to*  
PowerPC 603e™ RISC Microprocessor User's Manual:  
**PowerPC 603e Microprocessor Supplement and  
User's Manual Errata**

This document is provided as a supplement to the *PowerPC 603e RISC Microprocessor User's Manual*. It provides information on a lower-power implementation of the 603e family as well as corrections to the user's manual.

This supplement is divided into two parts:

- Part 1: The PowerPC 603e Microprocessor Supplement provides an overview of the PID7v-603e, with detailed information about features that differ from those of the PID6-603e described in the user's manual.

Note that the 603e is implemented in both a 3.3-volt version (PID 0006 PowerPC 603e microprocessor, abbreviated as PID6-603e) and a 2.5-volt version (PID 0007v PowerPC 603e microprocessor, abbreviated as PID7v-603e). Note that in this document, 603e refers to both the PID6-603e and PID7v-603e implementations, unless otherwise noted.

- Part 2: Errata to *PowerPC 603e RISC Microprocessor User's Manual* contains corrections to the user's manual.

Addendum to 603e User's Manual

The PowerPC name, the PowerPC logotype, PowerPC 601, PowerPC 603, and PowerPC 603e, 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 by Motorola and IBM. Motorola and IBM reserve the right to change or discontinue this product without notice.

© Motorola Inc., 1996. All rights reserved.

Portions hereof © International Business Machines Corporation, 1991–1996. All rights reserved.



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 term '603' is used as an abbreviation for the phrase, 'PowerPC 603™ microprocessor', and the term '603e' is used as an abbreviation for the phrase 'PowerPC 603e microprocessor'. The PowerPC 603e microprocessors are available from IBM as PPC603e and from Motorola as MPC603e.

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>.

## **Part 1: The PowerPC 603e Microprocessor Supplement**

This section describes the details of the 603e, provides a block diagram showing the major functional units, and describes briefly how those units interact. Any differences between the PID6-603e and PID7v-603e implementations are noted.

### **1.1 Overview**

This section describes the features of the 603e, provides a block diagram showing the major functional units, and briefly describes how those units interact.

The 603e is a low-power implementation of the PowerPC microprocessor family of reduced instruction set computing (RISC) microprocessors. The 603e implements the 32-bit portion of the PowerPC architecture, which provides 32-bit effective addresses, integer data types of 8, 16, and 32 bits, and floating-point data types of 32 and 64 bits.

The 603e is a superscalar processor that can issue and retire as many as three instructions per clock. Instructions can execute out of order for increased performance; however, the 603e makes completion appear sequential.

The 603e integrates five execution units—an integer unit (IU), a floating-point unit (FPU), a branch processing unit (BPU), a load/store unit (LSU), and a system register unit (SRU). The ability to execute five instructions in parallel and the use of simple instructions with rapid execution times yield high efficiency and throughput for 603e-based systems. Most integer instructions execute in one clock cycle. The FPU is pipelined so a single-precision multiply-add instruction can be issued and completed every clock cycle.

The 603e provides independent on-chip, 16-Kbyte, four-way set-associative, physically addressed caches for instructions and data and on-chip instruction and data memory management units (MMUs). The MMUs contain 64-entry, two-way set-associative, data and instruction translation lookaside buffers (DTLB and ITLB) that provide support for demand-paged virtual memory address translation and variable-sized block translation. The TLBs and caches use a least recently used (LRU) replacement algorithm. The 603e also

supports block address translation through the use of two independent instruction and data block address translation (IBAT and DBAT) arrays of four entries each. Effective addresses are compared simultaneously with all four entries in the BAT array during block translation. In accordance with the PowerPC architecture, if an effective address hits in both the TLB and BAT array, the BAT translation takes priority.

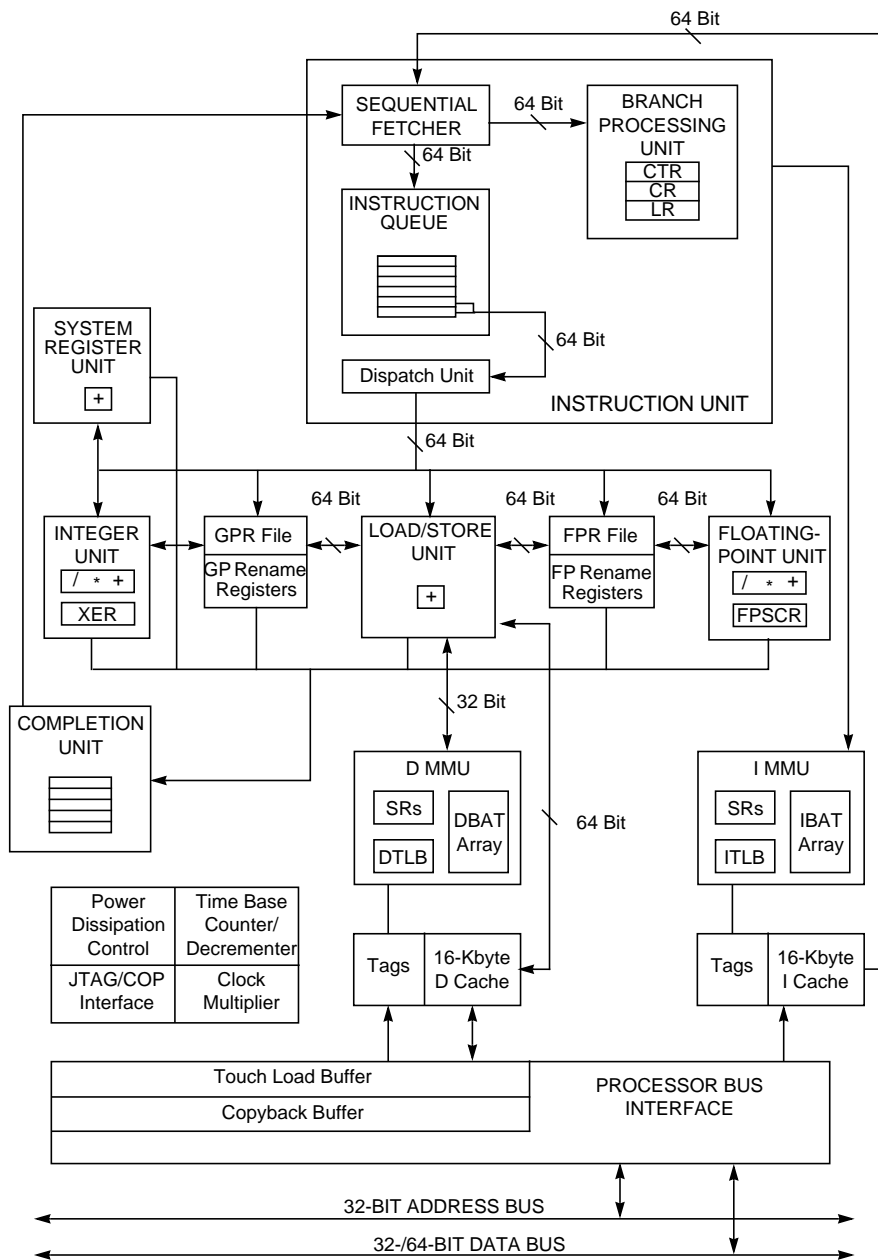
The 603e has a selectable 32- or 64-bit data bus and a 32-bit address bus. The 603e interface protocol allows multiple masters to compete for system resources through a central external arbiter. The 603e provides a three-state coherency protocol that supports the exclusive, modified, and invalid cache states. This protocol is a compatible subset of the MESI (modified/exclusive/shared/invalid) four-state protocol and operates coherently in systems that contain four-state caches. The 603e supports single-beat and burst data transfers for memory accesses, and supports memory-mapped I/O operations.

The 603e is fabricated using an advanced CMOS process technology and is fully compatible with TTL devices.

## 1.2 PowerPC 603e Microprocessor Features

This section summarizes features of the 603e's implementation of the PowerPC architecture.

Figure 1 provides a block diagram of the 603e that illustrates how the execution units—IU, FPU, BPU, LSU, and SRU—operate independently and in parallel. Note that this is a conceptual diagram and does not attempt to show how these features are physically implemented on the chip. For more information on the execution units, refer to *PowerPC 603e RISC Microprocessor Technical Summary* Revision 1.



**Figure 1. PowerPC 603e Microprocessor Block Diagram**

### 1.2.1 New Features of the PowerPC 603e Microprocessor

This section describes those features specific to the PID7v-603e:

- Enhancements to the register set—The PID7v-603e adds two new bits to the HID0 register:
  - The address bus enable (ABE) bit, bit 28, gives the PID7v-603e microprocessor the ability to broadcast **dcbf**, **dcbi**, and **dcbst** onto the 60x bus.
  - The instruction fetch enable M (IFEM) bit, bit 24, allows the PID7v-603e to reflect the value of the M-bit onto the 60x bus during instruction translation.
- Enhancements to cache implementation
  - The instruction cache is blocked only until the critical load completes (hit under reloads allowed).
  - The critical double word is simultaneously written to the cache and forwarded to the requesting unit, thus minimizing stalls due to load delays.
  - Provides for an optional data cache operation broadcast feature (enabled by the HID0[ABE] bit) that allows for correct system management utilizing an external copyback L2 cache.
  - Optional broadcast of cache control instructions **dcbi**, **dcbf**, and **dcbst** through configuration of HID0[ABE] bit.
- Exceptions
  - The PID7v-603e now offers hardware support for misaligned little-endian accesses. Little-endian load/store accesses that are not on a word boundary, with the exception of strings and multiples, generate exceptions under the same circumstances as big-endian accesses.
  - The PID7v-603e removed misalignment support for **eciwx** and **ecowx** graphics instructions. These instructions cause an alignment exception if the access is not on a word boundary.
- Bus clock—New bus multipliers of 4.5x, 5x, 5.5x, and 6x that are selected by the unused encodings of the PLL\_CFG[0–3]. Bus multipliers of 1x and 1.5x are not supported by PID7v-603e.
- Power management—Internal voltage supply changed from 3.3 volts to 2.5 volts. The core logic of the chip now uses a 2.5-volt supply.
- Instruction timing—The integer divide instructions, **divwux** and **divwx**, execute in 20 clock cycles (execution of these instructions in the PID6-603e requires 37 clock cycles).

### 1.2.2 Overview of PowerPC 603e Microprocessor Features

This section describes the major features of the 603e noting where the two implementations (PID6-603e and PID7v-603e) differ:

- High-performance, superscalar microprocessor
  - As many as three instructions issued and retired per clock
  - As many as five instructions in execution per clock
  - Single-cycle execution for most instructions
  - Pipelined FPU for all single-precision and most double-precision operations
- Five independent execution units and two register files
  - BPU featuring static branch prediction
  - A 32-bit IU
  - Fully IEEE 754-compliant FPU for both single- and double-precision operations
  - LSU for data transfer between data cache and GPRs and FPRs
  - SRU that executes condition register (CR), special-purpose register (SPR), and integer add/compare instructions
  - Thirty-two GPRs for integer operands
  - Thirty-two FPRs for single- or double-precision operands
- High instruction and data throughput
  - Zero-cycle branch capability (branch folding)
  - Programmable static branch prediction on unresolved conditional branches
  - Instruction fetch unit capable of fetching two instructions per clock from the instruction cache
  - A six-entry instruction queue that provides lookahead capability
  - Independent pipelines with feed-forwarding that reduces data dependencies in hardware
  - 16-Kbyte data cache—four-way set-associative, physically addressed; LRU replacement algorithm
  - 16-Kbyte instruction cache—four-way set-associative, physically addressed; LRU replacement algorithm
  - Cache write-back or write-through operation programmable on a per page or per block basis
  - BPU that performs CR lookahead operations
  - Address translation facilities for 4-Kbyte page size, variable block size, and 256-Mbyte segment size

- A 64-entry, two-way set-associative ITLB
- A 64-entry, two-way set-associative DTLB
- Four-entry data and instruction BAT arrays providing 128-Kbyte to 256-Mbyte blocks
- Software table search operations and updates supported through fast trap mechanism
- 52-bit virtual address; 32-bit physical address
- Facilities for enhanced system performance
  - A 32- or 64-bit split-transaction external data bus with burst transfers
  - Support for one-level address pipelining and out-of-order bus transactions
  - Hardware support for misaligned little-endian accesses (PID7v-603e)
- Integrated power management
  - Low-power 2.5-volt and 3.3-volt designs
  - Internal processor/bus clock multiplier ratios as follows:
    - 1/1, 1.5/1, 2/1, 2.5/1, 3/1, 3.5/1, and 4/1 (PID6-603e)
    - 2/1, 2.5/1, 3/1, 3.5/1, 4/1, 4.5/1, 5/1, 5.5/1, and 6/1 (PID7v-603e)
  - Three power-saving modes: doze, nap, and sleep
  - Automatic dynamic power reduction when internal functional units are idle
- In-system testability and debugging features through JTAG boundary-scan capability

### 1.3 PowerPC Architecture Implementation

The PowerPC architecture is derived from the IBM POWER (Performance Optimized with Enhanced RISC) architecture. 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 provides specific details about the implementation of the PID7v-603e as a low-power, 32-bit member of the PowerPC processor family. For more information, refer to the *PowerPC 603e RISC Microprocessor User's Manual*. The organization of this section follows:

- Section 1.3.1, “PowerPC 603e Microprocessor Programming Model (Chapter 2),” describes the PowerPC programming model and register set briefly and indicates where the PID7v-603e registers are different.
- Section 1.3.2, “Instruction and Data Cache Operation (Chapter 3),” describes the cache model that is defined generally for PowerPC processors by the virtual environment architecture (VEA). It also provides specific details about the PID7v-603e cache implementation.
- Section 1.3.3, “Exceptions (Chapter 4),” describes the exception model of the PowerPC operating environment architecture (OEA) and the implementation-specific features introduced by the PID7v-603e.
- Section 1.3.4, “Memory Management (Chapter 5),” indicates that the 603e MMU is identical on each of the 603e implementations (the PID6-603e, and both revisions of the PID7v-603e).
- Section 1.3.5, “Instruction Timing (Chapter 6),” provides a general description of the instruction timing provided by the superscalar, parallel execution supported by the PowerPC architecture and the PID7v-603e, in particular.
- Section 1.3.6, “Signal Descriptions (Chapter 7),” discusses the TT0 encodings used by the new HID0[ABE] bit for use in address-only bus transactions.
- Section 1.3.7, “System Interface Operation (Chapter 8),” remains unchanged from the *PowerPC 603e RISC Microprocessor User’s Manual*.
- Section 1.3.8, “Power Management (Chapter 9),” provides a general description of the power management features of the 603e, including specific details of the PID7v-603e.

The 603e is a high-performance, superscalar PowerPC microprocessor. 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 allow compilers to optimize instruction throughput. Compilers that take advantage of the flexibility of the PowerPC architecture can additionally optimize system performance of the PowerPC processors.

The following sections summarize the features of the 603e, including both those that are defined by the architecture and those that are unique to the 603e 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.

The PowerPC architecture allows a wide range of designs for features such as cache and system interface implementations. The 603e implementations support the three levels of the architecture described above. For more information about the PowerPC architecture, refer to *PowerPC Microprocessor Family: The Programming Environments*.

Specific features of the 603e, including the PID6-603e and PID7v-603e, are listed in Section 1.2, “PowerPC 603e Microprocessor Features.”

### **1.3.1 PowerPC 603e Microprocessor Programming Model (Chapter 2)**

The PowerPC architecture defines register-to-register operations for most computational instructions. Source operands for these instructions are accessed from the registers or are provided as immediate values embedded in the instruction opcode. The three-register instruction format allows specification of a target register distinct from the two source operands. Load and store instructions transfer data between registers and memory.

PowerPC processors have two levels of privilege—supervisor mode of operation (typically used by the operating system) and user mode of operation (used by the application software). The programming models incorporate 32 GPRs, 32 FPRs, special-purpose registers (SPRs), and several miscellaneous registers. Each PowerPC microprocessor also has its own unique set of hardware implementation (HID) registers.

Having access to privileged instructions, registers, and other resources allows the operating system to control the application environment (providing virtual memory and protecting operating-system and critical machine resources). Instructions that control the state of the

processor, the address translation mechanism, and supervisor registers can be executed only when the processor is operating in supervisor mode.

Figure 2 shows all the 603e registers available at the user and supervisor level. The numbers to the right of the SPRs indicate the number that is used in the syntax of the instruction operands to access the register.

The following subsections describe the PID7v-603e implementation-specific features as they apply to registers.

### 1.3.1.1 Processor Version Register (PVR)

The processor version number is 6 for the PID6-603e and 7 for the PID7v-603e. The processor revision level starts at 0x0100 and changes for each chip revision. The revision level is updated on all silicon revisions.

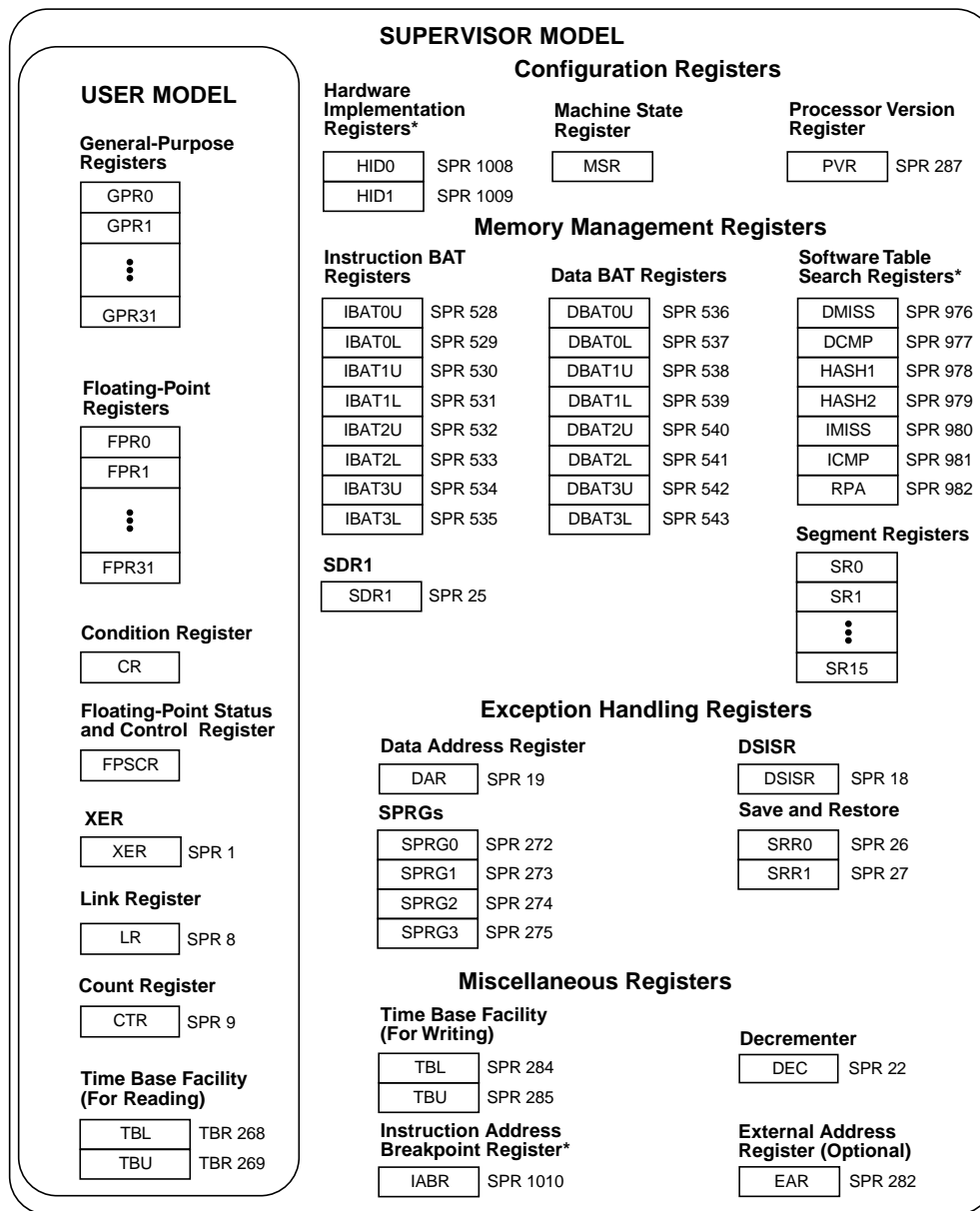
### 1.3.1.2 Hardware Implementation Register 0 (HID0)

Revision 2.0 of the PID7v-603e (designated by PVR level 0x0200) defines additional bits in the hardware implementation register 0 (HID0), a supervisor-level register that provides the means for enabling the 603e's checkstops and features, and allows software to read the configuration of the PLL configuration signals.

The HID0 bits with changed bit assignments are shown in Table 1. The HID0 bits that are not shown are implemented as they are in the *PowerPC 603e RISC Microprocessor User's Manual*, Section 2.1.2.1, "Hardware Implementation Registers (HID0 and HID1)."

**Table 1. Additional/Changed HID0 Bits**

Bit(s)	Description
24	Instruction fetch enable M (IFEM) bit—Enables the M bit on the bus. Used for instruction fetches.
25–26	Reserved
28	Address broadcast enable (ABE)—This configuration bit allows for the broadcast of <b>dcbf</b> , <b>dcbi</b> , and <b>dcbst</b> on the bus. Note that these cache control instruction broadcasts are not snooped by the PID7v-603e. Refer to Section 1.3.2, "Instruction and Data Cache Operation (Chapter 3)," for more information.
29–30	Reserved



Note: These registers are 603e-specific (PID6-603e and PID7v-603e) registers. They may not be supported by other PowerPC processors.

Figure 2. PowerPC 603e Microprocessor Programming Model—Registers

### 1.3.2 Instruction and Data Cache Operation (Chapter 3)

The following subsections describe the general cache characteristics as implemented in the PowerPC architecture, and the 603e implementation, specifically. PID7v-603e specific information is noted where applicable.

#### 1.3.2.1 PowerPC Cache Characteristics

The PowerPC architecture does not define hardware aspects of cache implementations. For example, some PowerPC processors, including the 603e, have separate instruction and data caches (Harvard architecture), while others, such as the PowerPC 601™ microprocessor, implement a unified cache.

PowerPC microprocessors control the following memory access modes on a page or block basis:

- Write-back/write-through mode
- Caching-inhibited mode
- Memory coherency

Note that in the 603e, a cache block is defined as eight words. The VEA defines cache management instructions that provide a means by which the application programmer can affect the cache contents.

#### 1.3.2.2 PowerPC 603e Microprocessor Cache Implementation

The 603e has two 16-Kbyte, four-way set-associative (instruction and data) caches. The caches are physically addressed, and the data cache can operate in either write-back or write-through mode as specified by the PowerPC architecture.

The data cache is configured as 128 sets of four blocks each. Each block consists of 32 bytes, two state bits, and an address tag. The two state bits implement the three-state MEI (modified/exclusive/invalid) protocol. Each block contains eight 32-bit words. Note that the PowerPC architecture defines the term block as the cacheable unit. For the 603e, the block size is equivalent to a cache line. A block diagram of the data cache organization is shown in Figure 3.

The instruction cache also consists of 128 sets of four blocks, and each block consists of 32 bytes, an address tag, and a valid bit. The instruction cache may not be written to except through a block fill operation. In the PID7v-603e, the instruction cache is blocked only until the critical load completes. The PID7v-603e supports instruction fetching from other instruction cache lines following the forwarding of the critical first double word of a cache line load operation. Successive instruction fetches from the cache line being loaded are forwarded, and accesses to other instruction cache lines can proceed during the cache line load operation. The instruction cache is not snooped, and cache coherency must be maintained by software. A fast hardware invalidation capability is provided to support

cache maintenance. The organization of the instruction cache is very similar to the data cache shown in Figure 3.

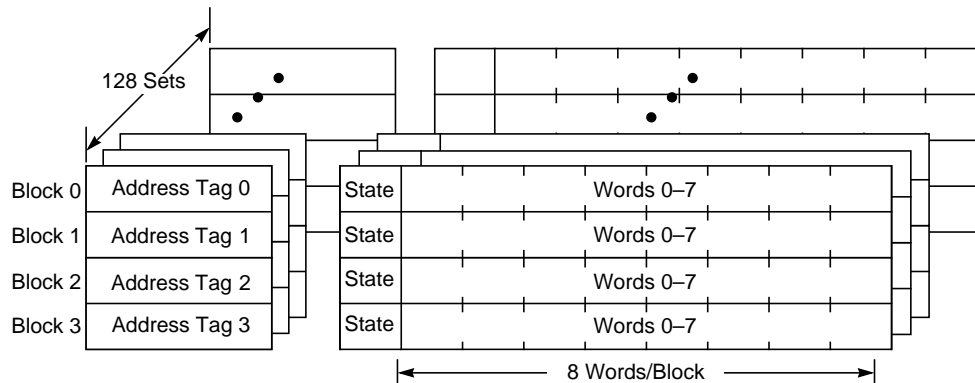
Each cache block contains eight contiguous words from memory that are loaded from an 8-word boundary (that is, bits A[27–31] of the effective addresses are zero); thus, a cache block never crosses a page boundary. Misaligned accesses across a page boundary can incur a performance penalty.

The 603e's cache blocks are loaded in four beats of 64 bits each when the 603e is configured with a 64-bit data bus; when the 603e is configured with a 32-bit bus, cache block loads are performed with eight beats of 32 bits each. The burst load is performed as "critical double-word first." The data cache is blocked to internal accesses until the load completes; the instruction cache allows sequential fetching during a cache block load. In the PID7v-603e, the critical double word is simultaneously written to the cache and forwarded to the requesting unit, thus minimizing stalls due to load delays.

To ensure coherency among caches in a multiprocessor (or multiple caching-device) implementation, the 603e implements the MEI protocol. These three states, modified, exclusive, and invalid, indicate the state of the cache block as follows:

- **Modified**—The cache block is modified with respect to system memory; that is, data for this address is valid only in the cache and not in system memory.
- **Exclusive**—This cache block holds valid data that is identical to the data at this address in system memory. No other cache has this data.
- **Invalid**—This cache block does not hold valid data.

Cache coherency is enforced by on-chip bus snooping logic. Since the 603e's data cache tags are single ported, a simultaneous load or store and snoop access represent a resource contention. The snoop access is given first access to the tags. The load or store then occurs on the clock following the snoop.



**Figure 3. Data Cache Organization**

### 1.3.2.3 Cache Control Instructions

The PowerPC architecture defines instructions for controlling both the instruction and data caches when they exist. The 603e interprets the cache control instructions (**icbi**, **dcbi**, **dcbt**, **dcbz**, **dcbst**) as if they pertain only to the 603e's caches.

The **dcbz** instruction causes an address-only broadcast on the bus if the contents of the block are from a page marked global through the WIMG bits. If the HID0[ABE] bit is set on a PID7v-603e processor, the execution of the **dcbf**, **dcbi**, and **dcbst** instructions will also cause an address-only broadcast. The **dcbz** instruction is also the only cache operation that is snooped by the 603e. The cache instructions are intended primarily for the management of the on-chip cache, and do not perform address-only broadcasts for the maintenance of other caches in the system. The ability of the PID7v-603e to optionally perform address-only broadcasts when executing the **dcbi**, **dcbf**, and the **dcbst** instructions allows the coherency management of an external copyback L2 cache.

Note that a data access exception is generated if the effective address of a **dcbi**, **dcbst**, **dcbf**, or **dcbz** instruction cannot be translated due to the lack of a TLB entry. Note that exceptions are referred to as interrupts in the architecture specification.

#### 1.3.2.3.1 Data Cache Block Invalidate (dcbi) Instruction

If the block containing the byte addressed by the EA is in the data cache, the cache block is invalidated regardless whether the block is in the exclusive or modified state. If HID0[ABE] is set on a PID7v-603e when a **dcbi** instruction is executed, the PID7v-603e will perform an address-only bus transaction. The **dcbi** instruction can only be executed when the 603e is in the supervisor state.

#### 1.3.2.3.2 Data Cache Block Touch (dcbt) Instruction

This instruction provides a method for improving performance through the use of software-initiated prefetch hints. The 603e performs the fetch for the cases when the address hits in the TLB or the BAT registers, and when it is a permitted load access from the addressed page. The operation is treated similarly to a byte load operation with respect to coherency.

If the address translation does not hit in the TLB or BAT mechanism, or if it does not have load access permission, the instruction is treated as a no-op.

If the cache is locked or disabled, or if the access is to a page that is marked as guarded, the **dcbt** instruction is treated as a no-op.

If the access is directed to a write-through or caching-inhibited page, the instruction is treated as a no-op.

The **dcbt** instruction never affects the referenced or changed bits in the hashed page table.

A successful **dcbt** instruction affects the state of the TLB and cache LRU bits as defined by the LRU algorithm.

The touch load buffer will be marked invalid if the contents of the touch buffer have been moved to the cache, if any data cache management instruction has been executed, if a **dcbz** instruction is executed that matches the address of the cache block in the touch buffer, or if another **dcbt** instruction is executed.

#### 1.3.2.3.3 Data Cache Block Touch for Store (dcbtst) Instruction

The **dcbtst** instruction, like the data cache block touch instruction (**dcbt**), allows software to prefetch a cache block in anticipation of a store operation (read with intent to modify).

#### 1.3.2.3.4 Data Cache Block Clear to Zero (dcbz) Instruction

If the block containing the byte addressed by the EA is in the data cache, all bytes are cleared.

If the block containing the byte addressed by the EA is not in the data cache and the corresponding page is caching-allowed, the block is established in the data cache without fetching the block from main memory, and all bytes of the block are cleared. If the contents of the cache block are from a page marked global through the WIM bits, an address-only bus transaction is run.

If the page containing the byte addressed by the EA is caching-inhibited or write-through, then the system alignment exception handler is invoked.

The **dcbz** instruction is treated as a store to the addressed byte with respect to address translation and protection.

#### 1.3.2.3.5 Data Cache Block Store (dcbst) Instruction

If the block containing the byte addressed by the EA is in coherency-required mode, and a block containing the byte addressed by the EA is in the data cache of any processor and has been modified, the writing of it to main memory is initiated. On a PID7v-603e, if the cache block is unmodified, HID0[ABE] is set, and if the contents of the cache block are from a page marked global through the WIM bits, an address-only bus transaction is run.

The function of this instruction is independent of the write-through and caching-inhibited/caching-allowed modes of the block containing the byte addressed by the EA.

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

#### 1.3.2.3.6 Data Cache Block Flush (dcbf) Instruction

The action taken depends on the memory mode associated with the target, and on the state of the cache block. The list below describes the action taken for the various cases. The actions described are executed regardless of whether the page containing the addressed byte is in caching-inhibited or caching-allowed mode. The following actions occur in both coherency-required mode (WIM = 0bXX1) and coherency-not-required mode (WIM = 0bXX0).

The **dcbf** instruction causes the following cache activity:

- Unmodified block—Invalidates the block in the processor's cache. If the processor is a PID7v-603e with HID0[ABE] set, an address-only bus transaction is performed.
- Modified block—Copies the block to memory and invalidates data cache block.
- Absent block—Does nothing.

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

### 1.3.3 Exceptions (Chapter 4)

This section describes the PowerPC exception model and the 603e implementation, specifically. PID7v-603e specific information is noted where applicable.

#### 1.3.3.1 PowerPC Exception Model

The PowerPC exception mechanism allows the processor to change to supervisor state as a result of external signals, errors, or unusual conditions arising in the execution of instructions, and differ from the arithmetic exceptions defined by the IEEE for floating-point operations. When exceptions occur, information about the state of the processor is saved to certain registers and the processor begins execution at an address (exception vector) predetermined for each exception. Processing of exceptions occurs in supervisor mode.



Although multiple exception conditions can map to a single exception vector, a more specific condition may be determined by examining a register associated with the exception—for example, the DSISR and the FPSCR. Additionally, some exception conditions can be explicitly enabled or disabled by software.

The PowerPC architecture requires that exceptions be handled in program order; therefore, although a particular implementation may recognize exception conditions out of order, they are presented strictly in order. When an instruction-caused exception is recognized, any unexecuted instructions that appear earlier in the instruction stream, including any that have not yet entered the execute stage, are required to complete before the exception is taken. Any exceptions caused by those instructions are handled first. Likewise, exceptions that are asynchronous and precise are recognized when they occur, but are not handled until the instruction currently in the completion stage successfully completes execution or generates an exception, and the completed store queue is emptied.

Unless a catastrophic condition causes a system reset or machine check exception, only one exception is handled at a time. If, for example, a single instruction encounters multiple exception conditions, those conditions are handled sequentially. After the exception handler handles an exception, the instruction execution continues until the next exception condition is encountered. However, in many cases there is no attempt to re-execute the instruction. This method of recognizing and handling exception conditions sequentially guarantees that exceptions are recoverable.

Exception handlers should save the information stored in SRR0 and SRR1 early to prevent the program state from being lost due to a system reset or machine check exception or to an instruction-caused exception in the exception handler, and before enabling external interrupts.

The PowerPC architecture supports four types of exceptions:

- Synchronous, precise—These are caused by instructions. All instruction-caused exceptions are handled precisely; that is, the machine state at the time the exception occurs is known and can be completely restored. This means that (excluding the trap and system call exceptions) the address of the faulting instruction is provided to the exception handler and that neither the faulting instruction nor subsequent instructions in the code stream will complete execution before the exception is taken. Once the exception is processed, execution resumes at the address of the faulting instruction (or at an alternate address provided by the exception handler). When an exception is taken due to a trap or system call instruction, execution resumes at an address provided by the handler.

- Synchronous, imprecise—The PowerPC architecture defines two imprecise floating-point exception modes: recoverable and nonrecoverable. Even though the 603e provides a means to enable the imprecise modes, it implements these modes identically to the precise mode (that is, all enabled floating-point enabled exceptions are always precise on the 603e).
- Asynchronous, maskable—The external, system management interrupt (SMI), and decrementer interrupts are maskable asynchronous exceptions. When these exceptions occur, their handling is postponed until the next instruction, and any exceptions associated with that instruction, completes execution. If there are no instructions in the execution units, the exception is taken immediately upon determination of the correct restart address (for loading SRR0).
- Asynchronous, nonmaskable—There are two nonmaskable asynchronous exceptions: system reset and the machine check exception. These exceptions may not be recoverable, or may provide a limited degree of recoverability. All exceptions report recoverability through the MSR[RI] bit.

### 1.3.3.2 PowerPC 603e Microprocessor Exception Model

As specified by the PowerPC architecture, all 603e exceptions can be described as either precise or imprecise and either synchronous or asynchronous. Asynchronous exceptions (some of which are maskable) are caused by events external to the processor's execution; synchronous exceptions, which are all handled precisely by the 603e, are caused by instructions. The 603e exception classes are shown in Table 2.

**Table 2. PowerPC 603e Exception Classifications**

Synchronous/Asynchronous	Precise/Imprecise	Exception Type
Asynchronous, nonmaskable	Imprecise	Machine check System reset
Asynchronous, maskable	Precise	External interrupt Decrementer System management interrupt
Synchronous	Precise	Instruction-caused exceptions

Although exceptions have other characteristics as well, such as whether they are maskable or nonmaskable, the distinctions shown in Table 2 define categories of exceptions that the 603e handles uniquely. Note that Table 2 includes no synchronous imprecise instructions. While the PowerPC architecture supports imprecise handling of floating-point exceptions, the 603e implements these exception modes as precise exceptions.

The 603e's exceptions, and conditions that cause them, are listed in Table 3.

**Table 3. Exceptions and Conditions**

Exception Type	Vector Offset (hex)	Causing Conditions
Reserved	00000	—
System reset	00100	A system reset is caused by the assertion of either $\overline{\text{SRESET}}$ or $\overline{\text{HRESET}}$ .
Machine check	00200	A machine check is caused by the assertion of the $\overline{\text{TEA}}$ signal during a data bus transaction, assertion of $\overline{\text{MCP}}$ , or an address or data parity error.
DSI	00300	The cause of a DSI exception can be determined by the bit settings in the DSISR, listed as follows: 1 Set if the translation of an attempted access is not found in the primary hash table entry group (HTEG), or in the rehashed secondary HTEG, or in the range of a DBAT register; otherwise cleared. 4 Set if a memory access is not permitted by the page or DBAT protection mechanism; otherwise cleared. 5 Set by an <b>eciwx</b> or <b>ecowx</b> instruction if the access is to an address that is marked as write-through, or execution of a load/store instruction that accesses a direct-store segment. 6 Set for a store operation and cleared for a load operation. 11 Set if <b>eciwx</b> or <b>ecowx</b> is used and EAR[E] is cleared.
ISI	00400	An ISI exception is caused when an instruction fetch cannot be performed for any of the following reasons: <ul style="list-style-type: none"> <li>The effective (logical) address cannot be translated. That is, there is a page fault for this portion of the translation, so an ISI exception must be taken to load the PTE (and possibly the page) into memory.</li> <li>The fetch access is to a direct-store segment (indicated by SRR1[3] set).</li> <li>The fetch access violates memory protection (indicated by SRR1[4] set). If the key bits (Ks and Kp) in the segment register and the PP bits in the PTE are set to prohibit read access, instructions cannot be fetched from this location.</li> </ul>
External interrupt	00500	An external interrupt is caused when MSR[EE] = 1 and the $\overline{\text{INT}}$ signal is asserted.
Alignment	00600	An alignment exception is caused when the 603e cannot perform a memory access for any of the following reasons: <ul style="list-style-type: none"> <li>The operand of a floating-point load or store instruction is not word-aligned.</li> <li>The operand of <b>lmw</b>, <b>stmw</b>, <b>lwarx</b>, and <b>stwcx</b> instructions are not aligned.</li> <li>The operand of a single-register load or store operation is not aligned, and the 603e is in little-endian mode (PID6-603e only).</li> <li>The execution of a floating-point load or store instruction to a direct-store segment.</li> <li>The operand of a load, store, load multiple, store multiple, load string, or store string instruction crosses a segment boundary into a direct-store segment, or crosses a protection boundary.</li> <li>Execution of a misaligned <b>eciwx</b> or <b>ecowx</b> instruction (PID7v-603e only).</li> <li>The instruction is <b>lmw</b>, <b>stmw</b>, <b>lswi</b>, <b>lswx</b>, <b>stswi</b>, <b>stswx</b> and the 603e is in little-endian mode.</li> <li>The operand of <b>dcbz</b> is in memory that is write-through-required or caching-inhibited.</li> </ul>

**Table 3. Exceptions and Conditions (Continued)**

Exception Type	Vector Offset (hex)	Causing Conditions
Program	00700	<p>A program exception is caused by one of the following exception conditions, which correspond to bit settings in SRR1 and arise during execution of an instruction:</p> <ul style="list-style-type: none"> <li>Floating-point enabled exception—A floating-point enabled exception condition is generated when the following condition is met: (MSR[FE0]   MSR[FE1]) &amp; FPSCR[FEX] is 1. FPSCR[FEX] is set by the execution of a floating-point instruction that causes an enabled exception or by the execution of one of the “move to FPSCR” instructions that results in both an exception condition bit and its corresponding enable bit being set in the FPSCR.</li> <li>Illegal instruction—An illegal instruction program exception is generated when execution of an instruction is attempted with an illegal opcode or illegal combination of opcode and extended opcode fields (including PowerPC instructions not implemented in the 603e), or when execution of an optional instruction not provided in the 603e is attempted (these do not include those optional instructions that are treated as no-ops).</li> <li>Privileged instruction—A privileged instruction type program exception is generated when the execution of a privileged instruction is attempted and the MSR register user privilege bit, MSR[PR], is set. In the 603e, this exception is generated for <b>mtspr</b> or <b>mfspr</b> with an invalid SPR field if SPR[0] = 1 and MSR[PR] = 1. This may not be true for all PowerPC processors.</li> <li>Trap—A trap type program exception is generated when any of the conditions specified in a trap instruction is met.</li> </ul>
Floating-point unavailable	00800	A floating-point unavailable exception is caused by an attempt to execute a floating-point instruction (including floating-point load, store, and move instructions) when the floating-point available bit is disabled (MSR[FP] = 0).
Decrementer	00900	The decrementer exception occurs when the most significant bit of the decrementer (DEC) register transitions from 0 to 1. Must also be enabled with the MSR[EE] bit.
Reserved	00A00–00BFF	—
System call	00C00	A system call exception occurs when a System Call ( <b>sc</b> ) instruction is executed.
Trace	00D00	A trace exception is taken when MSR[SE] = 1 or when the currently completing instruction is a branch and MSR[BE] = 1.
Reserved	00E00	The 603e does not generate an exception to this vector. Other PowerPC processors may use this vector for floating-point assist exceptions.
Reserved	00E10–00FFF	—
Instruction translation miss	01000	An instruction translation miss exception is caused when an effective address for an instruction fetch cannot be translated by the ITLB.
Data load translation miss	01100	A data load translation miss exception is caused when an effective address for a data load operation cannot be translated by the DTLB.

**Table 3. Exceptions and Conditions (Continued)**

Exception Type	Vector Offset (hex)	Causing Conditions
Data store translation miss	01200	A data store translation miss exception is caused when an effective address for a data store operation cannot be translated by the DTLB, or where a DTLB hit occurs, and the change bit in the PTE must be set due to a data store operation.
Instruction address breakpoint	01300	An instruction address breakpoint exception occurs when the address (bits 0–29) in the IABR matches the next instruction to complete in the completion unit, and the IABR enable bit (bit 30) is set.
System management interrupt	01400	A system management interrupt is caused when MSR[EE] = 1 and the $\overline{\text{SMI}}$ input signal is asserted.
Reserved	01500–02FFF	—

### 1.3.4 Memory Management (Chapter 5)

Both the PID6-603e and PID7v-603e snoop and perform address-only broadcasts when a **dcbz** instruction is executed, and WIM is set to 001 for the page where the cache block is located. The PID7v-603e also performs address-only broadcasts for the **dcbf**, **dcbi**, and **dcbst** instructions when HID0[ABE] is set.

Table 4 provides an overview of the bus operations initiated by cache control instructions. The cache control, TLB management, and synchronization instructions supported by the PID7v-603e may affect or be affected by the operation of the bus.

Note that Table 4 assumes that the WIM bits are set to 001; that is, since the cache is operating in write-back mode, caching is permitted and coherency is enforced.

**Table 4. Bus Operations Caused by Cache Control Instructions (WIM = 001)**

Operation	Cache State	Next Cache State	Bus Operations	Comment
<b>sync</b>	Don't care	No change	None	Waits for memory queues to complete bus activity
<b>icbi</b>	Don't care	I	None	—
<b>dcbi</b>	Don't care	I	None (Kill block*)	—
<b>dcbf</b>	I, E	I	None (Flush block*)	—
<b>dcbf</b>	M	I	Write with kill	Block is pushed
<b>dcbst</b>	I, E	No change	None (Clean block*)	—
<b>dcbst</b>	M	E	Write	Block is pushed
<b>dcbz</b>	I	M	Write with kill	—
<b>dcbz</b>	E, M	M	Kill block	Writes over modified data
<b>dcbt</b>	I	No change	Read	Fetches cache block is stored in touch load queue
<b>dcbt</b>	E, M	No change	None	—
<b>dcbtst</b>	I	No change	Read-with-intent-to-modify	Fetches cache block is stored in touch load queue
<b>dcbtst</b>	E, M	No change	None	—

Note: Bus operation performed when HIO[ABE] is set.

### 1.3.5 Instruction Timing (Chapter 6)

The 603e is a pipelined superscalar processor. A pipelined processor is one in which the processing of an instruction is reduced into discrete stages. Because the processing of an instruction is broken into a series of stages, an instruction does not require the entire resources of an execution unit. For example, after an instruction completes the decode stage, it can pass on to the next stage, while the subsequent instruction can advance into the decode stage. This improves the throughput of the instruction flow. For example, it may take three cycles for a floating-point instruction to complete, but if there are no stalls in the floating-point pipeline, a series of floating-point instructions can have a throughput of one instruction per cycle.

The instruction pipeline in the 603e has four major pipeline stages, described as follows:

- The fetch pipeline stage primarily involves retrieving instructions from the memory system and determining the location of the next instruction fetch. Additionally, the BPU decodes branches during the fetch stage and folds out branch instructions before the dispatch stage if possible.

- The dispatch pipeline stage is responsible for decoding the instructions supplied by the instruction fetch stage, and determining which of the instructions are eligible to be dispatched in the current cycle. In addition, the source operands of the instructions are read from the appropriate register file and dispatched with the instruction to the execute pipeline stage. At the end of the dispatch pipeline stage, the dispatched instructions and their operands are latched by the appropriate execution unit.
- During the execute pipeline stage each execution unit that has an executable instruction executes the selected instruction (perhaps over multiple cycles), writes the instruction's result into the appropriate rename register, and notifies the completion stage that the instruction has finished execution. In the case of an internal exception, the execution unit reports the exception to the completion/writeback pipeline stage and discontinues instruction execution until the exception is handled. The exception is not signaled until that instruction is the next to be completed. Execution of most floating-point instructions is pipelined within the FPU allowing up to three instructions to be executing in the FPU concurrently. The pipeline stages for the floating-point unit are multiply, add, and round-convert. Execution of most load/store instructions is also pipelined. The load/store unit has two pipeline stages. The first stage is for effective address calculation and MMU translation and the second stage is for accessing the data in the cache.
- The complete/writeback pipeline stage maintains the correct architectural machine state and transfers the contents of the rename registers to the GPRs and FPRs as instructions are retired. If the completion logic detects an instruction causing an exception, all following instructions are cancelled, their execution results in rename registers are discarded, and instructions are fetched from the correct instruction stream.

A superscalar processor is one that issues multiple independent instructions into multiple pipelines allowing instructions to execute in parallel. The 603e has five independent execution units, one each for integer instructions, floating-point instructions, branch instructions, load/store instructions, and system register instructions. The IU and the FPU each have dedicated register files for maintaining operands (GPRs and FPRs, respectively), allowing integer calculations and floating-point calculations to occur simultaneously without interference. Integer division performance of the PID7v-603e has been improved, with the **divwux** and **divwx** instructions executing in 20 clock cycles, instead of the 37 cycles required in the PID6-603e.

The 603e provides support for single-cycle store and it provides an adder/comparator in the system register unit that allows the dispatch and execution of multiple integer add and compare instructions on each cycle. Refer to the *PowerPC 603e RISC Microprocessor User's Manual* for more information.

Because the PowerPC architecture can be applied to such a wide variety of implementations, instruction timing among various PowerPC processors varies accordingly.

### 1.3.6 Signal Descriptions (Chapter 7)

The 603e provides transfer type signals (TT0–TT4) that characterize bus transfers. When HID0[ABE] is set, the PID7v-603e will perform address-only bus transactions with the encodings shown in Table 5.

**Table 5. Transfer Encoding for the PID7v-603e Processor Bus Master**

TT0	TT1	TT2	TT3	TT4	PID7v-603e Transaction	Transaction	Transaction Source
0	0	0	0	0	Clean Block	Address only	dcbst
0	0	1	0	0	Flush block	Address only	dcbf
0	1	1	0	0	Kill block	Address only	dcbz, dcbi

The 603e provides a CLK\_OUT signal for test purposes that allows the monitoring of the processor and bus clock frequencies. The frequency of the CLK\_OUT signal is determined by the configuration of the HID0[SBCLK] and HID0[ECLK] bits, as shown in Table 6. Note that the PID7v-603e's CLK\_OUT signal will be driven at the processor frequency during the assertion of  $\overline{\text{HRESET}}$ ; when the  $\overline{\text{HRESET}}$  signal is deasserted the CLK\_OUT signal enters the default high-impedance state.

**Table 6. CLK\_OUT Signal Configuration**

HID0[SBCLK]	HID0[ECLK]	CLK_OUT Output State
0	0	High-impedance
0	1	Processor clock frequency
1	0	1/2 bus clock frequency
1	1	Bus clock frequency

### 1.3.7 System Interface Operation (Chapter 8)

The PID7v-603e system interface operates in the same manner as the PID6-603e.

### 1.3.8 Power Management (Chapter 9)

The 603e provides four power modes selectable by setting the appropriate control bits in the machine state register (MSR) and hardware implementation register 0 (HID0) registers. The four power modes are as follows:

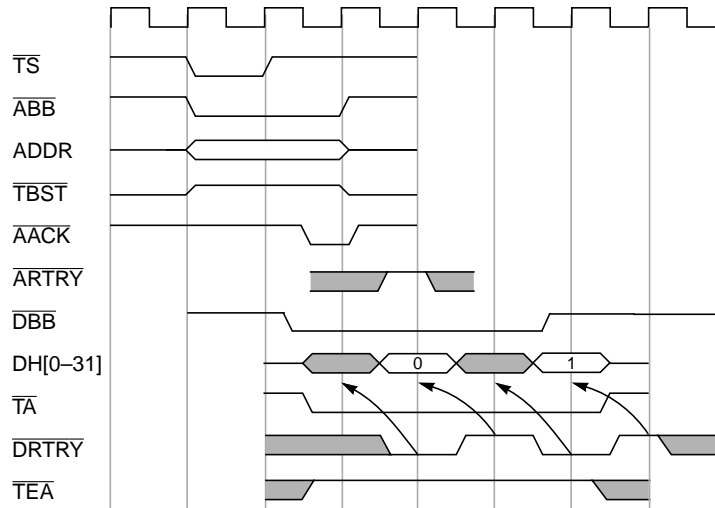


- **Full-power**—This is the default power state of the 603e. The 603e is fully powered and the internal functional units are operating at the full processor clock speed. If the dynamic power management mode is enabled, functional units that are idle will automatically enter a low-power state without affecting performance, software execution, or external hardware.
- **Doze**—All the functional units of the 603e are disabled except for the time base/decrementer registers and the bus snooping logic. When the processor is in doze mode, an external asynchronous interrupt, a system management interrupt, a decrementer exception, a hard or soft reset, or machine check brings the 603e into the full-power state. The 603e in doze mode maintains the PLL in a fully powered state and locked to the system external clock input (SYSCLK) so a transition to the full-power state takes only a few processor clock cycles.
- **Nap**—The nap mode further reduces power consumption by disabling bus snooping, leaving only the time base register and the PLL in a powered state. The 603e returns to the full-power state upon receipt of an external asynchronous interrupt, a system management interrupt, a decrementer exception, a hard or soft reset, or a machine check input ( $\overline{\text{MCP}}$ ) signal. A return to full-power state from a nap state takes only a few processor clock cycles.
- **Sleep**—Sleep mode reduces power consumption to a minimum by disabling all internal functional units, after which external system logic may disable the PLL and SYSCLK. Returning the 603e to the full-power state requires the enabling of the PLL and SYSCLK, followed by the assertion of an external asynchronous interrupt, a system management interrupt, a hard or soft reset, or a machine check input ( $\overline{\text{MCP}}$ ) signal after the time required to relock the PLL.

## Part 2: Errata to *PowerPC 603e RISC Microprocessor User's Manual*

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

Section #/Page #	Changes
2.1.1, Page 2-5	Replace the second implementation note with the following:  <b>Implementation Note</b> —The processor version number for the 603e is 0x0006, and the first mask revision is 1.0; the PVR reads as 0x00060100.
2.1.1, Page 2-6	Add the following sentence at the end of the first bullet item in the miscellaneous registers section:  The TB is incremented once every four bus clock cycles.
3.3.2, Page 3-7	Replace the first sentence with the following:  The 603e uses an LRU replacement algorithm to determine which of the four possible cache locations should be used for a cache update on a cache miss.
3.7, Page 3-23	Add the following new section before Section 3.7.1:  <b>3.7.1a Data Cache Block Invalidate (dcbi) Instruction</b>  If the block containing the byte addressed by the EA is in the data cache, the cache block is invalidated regardless whether the block is in the exclusive or modified state. If HID0[ABE] is set on a PID7v-603e when a <b>dcbi</b> instruction is executed, the PID7v-603e will perform an address-only bus transaction. The <b>dcbi</b> instruction can only be executed when the 603e is in the supervisor state.
8.6.1, Page 8-39	Replace Figure 8-22 with the following figure; $\overline{\text{DBB}}$ signal is asserted for three clock cycles.



**Figure 8-22. 32-Bit Data Bus Transfer (Two-Beat Burst with  $\overline{DRTRY}$ )**

9.2.2, Page 9-4 Replace the last sentence with the following:

To ensure a clean transition into and out of the power management mode, set the MSR[EE] bit and execute the following code sequence:

```
sync
mtmsr[POW = 1]
isync
loop:    b loop
```

C.1.6, C.1.7, C.1.8 Add the following new sections:

### **C.1.6 PLL Configuration (PLL\_CFG0–PLL\_CFG3)— Input**

The 603 operates as described in Section 7.2.12.3 of the *PowerPC 603e RISC Microprocessor User's Manual*, except for the following:

To avoid incorrect operation of the PLL, the clock input to the SYCLK signal input should be stable and within the frequency range specified for the selected PLL\_CFG configuration during power-up, during normal operation, or when exiting the sleep power-saving mode.

### C.1.7 Address Pipelining and Split-Bus Transactions

The 603 operates as described in Section 8.2.2 of the *PowerPC 603e RISC Microprocessor User's Manual*, except for the following:

Note that in multiprocessor systems, addresses associated with cache line loads are not snooped between the third and fourth beat during the data tenure when the system is configured for 64-bit bus operation.

When configured for 32-bit bus operation, cache line loads are not snooped between the sixth and eighth beats. To ensure memory coherency, multiprocessor systems should avoid pipelined operation, or disallow snooping during the last data beat of a cache load operation.

### C.1.8 Data Bus Arbitration

The 603 operates as described in Section 8.4.1 of the *PowerPC 603e RISC Microprocessor User's Manual*, except for the following:

When the 603 is configured for 1:1 processor to bus clock operation and  $\overline{\text{DBG}}$  is always held asserted, multiple single-beat writes will cause incorrect data to be written to memory. The  $\overline{\text{DBG}}$  signal should only be asserted when the data tenure can be started on the following bus cycle.

C.2, Page C-15 Add the following bullet item:

- The PVR value for the 603 is 0x0003.

C.2.4, C.2.5, C.2.6 Add the following new sections at the end of Appendix C:

### C.2.4 Machine Check Exception (0x00200)

The 603 operates as described in Section 4.5.2 of the *PowerPC 603e RISC Microprocessor User's Manual*, with the exception of the following:

To ensure memory coherency following the assertion of  $\overline{\text{TEA}}$ , the instruction cache should be invalidated by setting and clearing  $\text{HID0}[\text{ICFI}]$ , and flushing the data cache before performing any load or store operations, or executing any data cache management instructions other than **dcbf**.

Note that an assertion of  $\overline{\text{TEA}}$  during an instruction fetch will result in an immediate instruction refetch before the machine check exception is taken, which will result in a second assertion of the  $\overline{\text{TEA}}$  signal. The second assertion of  $\overline{\text{TEA}}$  while the machine check exception is pending from the previous  $\overline{\text{TEA}}$  assertion will result in the 603 entering the checkstop state instead of taking the machine check exception.

### C.2.5 Instruction Address Breakpoint Exception (0x01400)

The 603 operates as described in Section 4.5.15 of the *PowerPC 603e RISC Microprocessor User's Manual*, with the exception of the following:

To avoid spurious IABR exceptions, the IABR special-purpose register should not be loaded with an address that falls within the same cache line as a disabled, but matching IABR address.

### C.2.6 Cache Control Instructions

The 603 operates as described in Section 3.7 of the *PowerPC 603e RISC Microprocessor User's Manual*, with the exception of the following:

Note that loop structures that contain long sequences of **dcbz** or **dcbi** instructions may cause snoop performance degradation. Programmers can improve snoop performance by inserting no-op instructions (**ori** 0,0,0) between **dcbz** or **dcbi** instructions, replacing the **dcbz** or **dcbi** instructions with a sequence of write-through store operations, using the decremter to generate a periodic exception to allow snoop activity, or mapping the address space where the **dcbz** or **dcbi** instructions execute as global ( $M = 1$ ).

Note that the use of the **dcbz** instruction in a multiprocessor system can result in loss of data coherency if the **dcbz** instruction is executed in memory space marked as global ( $M = 1$ ). Programmers should use software coherency protocols to ensure that no processor can perform a kill operation to memory used by another processor.

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 603e microprocessor embodies 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 601, PowerPC 603, and PowerPC 603e 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/>



**MOTOROLA**