Technical Summary

MPC603E/D
Rev. 2 11/2001

MPC603e RISC Microprocessor Technical Summary

This document provides an overview of features for the MPC603e microprocessor and PowerPC architecture, and information about how the MPC603e implementation complies with the architectural definitions. Note that the MPC603e microprocessor is implemented in both a 2.5-volt version (PID 0007t MPC603e microprocessor, abbreviated as PID7t-603e) and a 3.3-volt version (PID 0006 MPC603e microprocessor, abbreviated as PID6-603e). However, the PID6-603e is end-of-life and not recommended for new designs.

This document is divided into two parts:

- Part I, "MPC603e Microprocessor Overview," provides an overview of the MPC603e features, including a block diagram showing the major functional components.
- Part II, "MPC603e Microprocessor Architecture Implementation," describes the PowerPC architecture in general, as well as providing specific details about the implementation of the MPC603e as a low-power, 32-bit member of this processor family.

Part I
MPC603e Microprocessor Overview

This section describes the detailed features of the MPC603e, provides a block diagram showing the major functional units (see Figure 1), and briefly describes how these units interact. Any differences between the PID6-603e and PID7t-603e implementations are noted.

The MPC603e is a low-power implementation of this microprocessor family of reduced instruction set computing (RISC) microprocessors. The MPC603e 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 MPC603e provides four software controllable power-saving modes. Three of the modes (the doze, nap, and sleep modes) are static in nature, and progressively reduce the amount of power dissipated by the processor. The fourth is a dynamic power management mode that causes the functional units in the MPC603e to automatically enter a low-power mode when the functional units are idle without affecting operational performance, software execution, or any external hardware.

The MPC603e is a superscalar processor that can issue and retire as many as three instructions per clock cycle. Instructions can execute out of program order for increased performance; however, the MPC603e makes completion appear sequential.

The MPC603e 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). These execution units are (shown in Figure 1) operate independently and in parallel. The ability to execute five instructions in parallel and the use of simple instructions with rapid execution times yield high efficiency and throughput for MPC603e-based systems. Most integer instructions execute in one clock cycle. On the MPC603e, the FPU is pipelined so a single-precision multiply-add instruction can be issued and completed every clock cycle. Note that Figure 1 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 *MPC603e RISC Microprocessor User's Manual*.

The MPC603e provides address translation and protection facilities, including an ITLB, DTLB, and instruction and data BAT arrays. Instruction fetching and issuing is handled in the instruction unit. Translation of addresses for cache or external memory accesses are handled by the MMUs. Both units are discussed in more detail in Section 1.2, "Instruction Unit," and Section 1.4.1, "Memory Management Units (MMUs)."

The MPC603e 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 MPC603e 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 MPC603e has a selectable 32- or 64-bit data bus and a 32-bit address bus. The MPC603e interface protocol allows multiple masters to compete for system resources through a central external arbiter. The MPC603e 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 MPC603e supports single-beat and burst data transfers for memory accesses, and supports memory-mapped I/O operations.

The MPC603e is fabricated using an advanced CMOS process technology and is fully compatible with TTL devices.

1.1 MPC603e Microprocessor Features

This section describes the major features of the MPC603e noting where the PID6-603e and PID7t-603e implementations 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 (IQ) that provides lookahead capability
  - Independent pipelines with feed-forwarding that reduces data dependencies in hardware
  - 16-Kbyte data cache and 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 and 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 (PID7t-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 (PID7t-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

The following features are specific to the PID7t-603e:

- Enhancements to the register set
  - The PID7t-603e adds two HID0 bits:
    - The address bus enable (ABE) bit, HID0[28], gives the PID7t-603e microprocessor the ability to broadcast **dcbf**, **dcbi**, and **dcbst** onto the 60x bus.
    - The instruction fetch enable M (IFEM) bit, HID0[24], allows the PID7t-603e to reflect the value of the M bit during instruction translation onto the 60x bus .
- 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 copy-back L2 cache.
  - All of the cache control instructions (**icbi**, **dcbi**, **dcbf**, and **dcbst**, excluding **dcbz**) require that the HID0[ABE] configuration bit be set in order to execute.
- Exceptions
  - The PID7t-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 PID7t-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 are selected by the unused encodings of PLL_CFG[0:3]. Bus multipliers of 1x and 1.5x are not supported by PID7t-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, **divwu**[**o**][**.**] and **divw**[**o**][**.**], execute in 20 clock cycles;

execution of these instructions in the PID6-603e takes 37 clock cycles.

- Support for single-cycle store
- An adder/comparator added to system register unit that allows dispatch and execution of multiple integer add and compare instructions on each cycle.

1.2 Instruction Unit

As shown in Figure 1, the MPC603e instruction unit, containing a fetch unit, instruction queue, dispatch unit, and BPU, provides centralized control of instruction flow to the execution units. The instruction unit determines the address of the next instruction to be fetched based on information from the sequential fetcher and from the BPU.

The instruction unit fetches the instructions from the instruction cache into the instruction queue (IQ). The BPU receives branch instructions from the fetcher and uses static branch prediction to allow fetching from a predicted instruction stream while a conditional branch is evaluated. Branch folding occurs when a branch is predicted taken or is resolved taken(as in the case with unconditional branches). The cycle after branch execution, the BPU folds the branch out of the instruction stream (removes the branch from the IQ) and also flushes instructions that are in the IQ after the branch.

Instructions issued beyond a predicted branch cannot complete execution until the branch is resolved, preserving the programming model of sequential execution. If any of these are branch instructions, they are decoded but not issued. Instructions to be executed by the FPU, IU, LSU, and SRU are issued and allowed to progress up to the register write-back stage. Write-back is allowed when a correctly predicted branch is resolved, and execution continues along the predicted path.

If branch prediction is incorrect, the instruction unit flushes all predicted path instructions, and instructions are issued from the correct path.

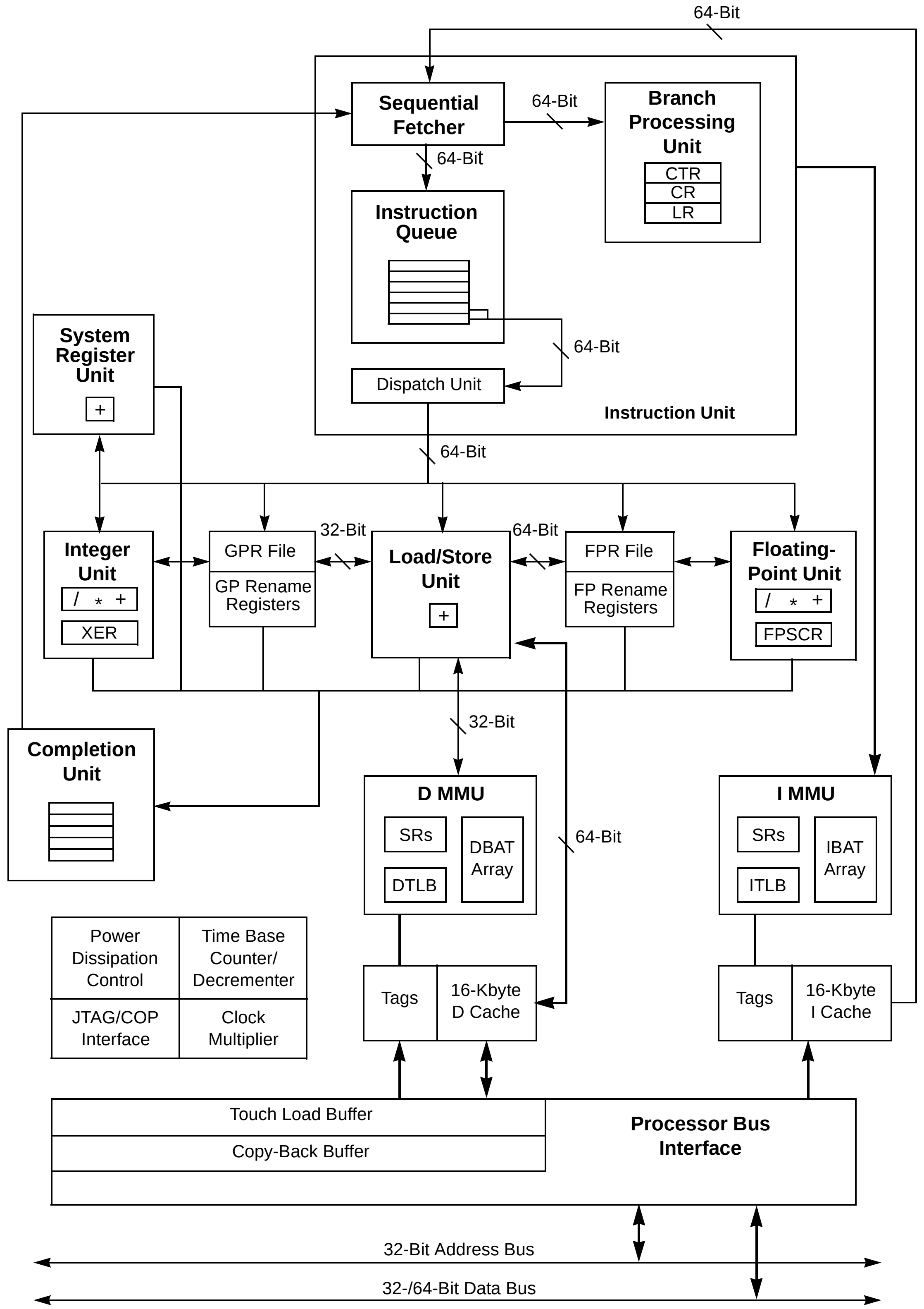


Figure 1. Block Diagram of the MPC603e

1.2.1 Instruction Queue and Dispatch Unit

The instruction queue (IQ), shown in Figure 1, holds as many as six instructions and loads up to two instructions from the instruction unit during a single cycle. The instruction fetch unit continuously loads as many instructions as space in the IQ allows. Instructions are dispatched to their respective execution units from the dispatch unit at a maximum rate of two instructions per cycle. Dispatching is facilitated to the IU, FPU, LSU, and SRU by the provision of a reservation station at each unit. The dispatch unit performs source and destination register dependency checking, determines dispatch serializations, and inhibits subsequent instruction dispatching as required.

For a more detailed overview of instruction dispatch, see Section 2.7, “Instruction Timing.”

1.2.2 Branch Processing Unit (BPU)

The BPU receives branch instructions from the fetch unit and performs CR lookahead operations on conditional branches to resolve them early, achieving the effect of a zero-cycle branch in many cases.

The BPU uses a bit in the instruction encoding to predict the direction of the conditional branch. Therefore, when an unresolved conditional branch instruction is encountered, the MPC603e fetches instructions from the predicted target stream until the conditional branch is resolved.

The BPU contains an adder to compute branch target addresses and three user-control registers—the link register (LR), the count register (CTR), and the CR. The BPU calculates the return pointer for subroutine calls and saves it into the LR for certain types of branch instructions. The LR also contains the branch target address for the Branch Conditional to Link Register (**bclr***x*) instruction. The CTR contains the branch target address for the Branch Conditional to Count Register (**bcctr***x*) instruction. The contents of the LR and CTR can be copied to or from any GPR. Because the BPU uses dedicated registers rather than GPRs or FPRs, execution of branch instructions is largely independent from execution of integer and floating-point instructions.

1.3 Independent Execution Units

The PowerPC architecture’s support for independent execution units allows implementation of processors with out-of-order instruction execution. For example, because branch instructions do not depend on GPRs or FPRs, branches can often be resolved early, eliminating stalls caused by taken branches.

In addition to the BPU, the MPC603e provides four other execution units and a completion unit, which are described in the following sections.

1.3.1 Integer Unit (IU)

The IU executes all integer instructions. The IU executes one integer instruction at a time, performing computations with its arithmetic logic unit (ALU), multiplier, divider, and XER register. Most integer instructions are single-cycle instructions. The 32 GPRs hold integer operands. Stalls due to contention for GPRs are minimized by the automatic allocation of rename registers. The MPC603e writes the contents of the rename registers to the appropriate GPR when integer instructions are retired by the completion unit.

1.3.2 Floating-Point Unit (FPU)

The FPU contains a single-precision multiply-add array and the floating-point status and control register (FPSCR). The multiply-add array allows the MPC603e to efficiently implement multiply and multiply-add operations. The FPU is pipelined so that single- and double-precision instructions can be issued back-to-back. The 32 FPRs are provided to support floating-point operations. Stalls due to contention for FPRs are minimized by the automatic allocation of rename registers. The MPC603e writes the contents of the rename registers to the appropriate FPR when floating-point instructions are retired by the completion unit.

The MPC603e supports all IEEE 754 floating-point data types (normalized, denormalized, NaN, zero, and infinity) in hardware, eliminating the latency incurred by software exception routines.

1.3.3 Load/Store Unit (LSU)

The LSU executes all load and store instructions and provides the data transfer interface between the GPRs, FPRs, and the cache/memory subsystem. The LSU calculates effective addresses, performs data alignment, and provides sequencing for load/store string and multiple instructions.

Load and store instructions are issued and executed in program order; however, the memory accesses can occur out of order. Synchronizing instructions are provided to enforce strict ordering.

Cacheable loads, when free of data dependencies, execute in an out-of-order manner with a maximum throughput of one per cycle and a two-cycle total latency. Data returned from the cache is held in a rename register until the completion logic commits the value to a GPR or FPR. Stores cannot be executed in a predicted manner and are held in the store queue until the completion logic signals that the store operation is to be completed to memory. The MPC603e executes store instructions with a maximum throughput of one per cycle and a three-cycle total latency. The time required to perform the actual load or store depends on whether the operation involves the cache, system memory, or an I/O device.

1.3.4 System Register Unit (SRU)

The SRU executes various system-level instructions, including condition register logical operations and move to/from special-purpose register instructions, and also executes integer add/compare instructions. In order to maintain system state, most instructions executed by the SRU are completion-serialized; that is, the instruction is held for execution in the SRU until all prior instructions issued have completed. Results from completion-serialized instructions executed by the SRU are not available or forwarded for subsequent instructions until the instruction completes.

1.3.5 Completion Unit

The completion unit tracks instructions from dispatch through execution and then retires, or completes, them in program order. Completing an instruction commits the MPC603e to any architectural register changes caused by that instruction. In-order completion ensures the correct architectural state when the MPC603e must recover from a mispredicted branch or any exception.

Instruction state and other information required for completion is kept in a five-entry FIFO completion queue. A single completion queue entry is allocated for each instruction that enters the dispatch unit. An available completion queue entry is a required resource for dispatch; if no completion entry is available, dispatch stalls. A maximum of two instructions per cycle are completed in order from the queue.

1.4 Memory Subsystem Support

The MPC603e provides both separate instruction and data caches and MMUs. The MPC603e also provides an efficient processor bus interface to facilitate access to main memory and other bus subsystems. The memory subsystem support functions are described in the following sections.

1.4.1 Memory Management Units (MMUs)

The MPC603e MMUs support up to 4 Petabytes (2^{52}) of virtual memory and 4 Gigabytes (2^{32}) of physical memory (referred to as real memory in the architecture specification) for instruction and data. The MMUs also control access privileges for these spaces on block and page granularities. Referenced and changed status is maintained by the processor for each page to assist implementation of a demand-paged virtual memory system. A key bit is implemented to provide information about memory protection violations prior to page table search operations.

The LSU calculates effective addresses (EAs) for data loads and stores, performs data alignment to and from cache memory, and provides the sequencing for load and store string and multiple word instructions. The instruction unit calculates effective addresses for instruction fetching.

After an EA is generated, its higher-order bits are translated by the appropriate MMU into physical address bits. The lower-order EA bits are the same on the physical address and form the index into the four-way set-associative tag array. After translating the address, the MMU passes the higher-order physical address bits to the cache and the cache lookup completes. For caching-inhibited accesses or accesses that miss in the cache, the untranslated lower-order address bits are concatenated with the translated higher-order address bits; the resulting 32-bit physical address is then used by the memory unit and the system interface to access external memory.

The MMU also directs the address translation and enforces the protection hierarchy programmed by the operating system in relation to the supervisor/user privilege level of the access and in relation to whether the access is a load or store.

For instruction fetches, the MMU looks for the address in the ITLB and in the IBAT array. If an address hits both, the IBAT array translation is used. Data accesses cause a lookup in the DTLB and DBAT array. In most cases, the translation is in a TLB and the physical address bits are readily available to the on-chip cache. Note that the BAT entry is chosen if the translation hits in both a BAT and a TLB.

When the EA misses in the TLBs, the MPC603e provides hardware assistance for software to perform a search of the translation tables in memory. The hardware assist consists of the following features:

- Automatic storage of the missed effective address in IMISS and DMISS
- Automatic generation of the primary and secondary hashed real address of the page table entry group (PTEG), which are readable from the HASH1 and HASH2 register locations.

  The HASH data is generated from the contents of the IMISS or DMISS register. The register that is selected depends on the miss (instruction or data) that was last acknowledged.
- Automatic generation of the first word of the page table entry (PTE) of the tables being searched
- A real page address (RPA) register that matches the format of the lower word of the PTE
- TLB access instructions (**tlbli** and **tlbld**) that are used to load an address translation into the instruction or data TLBs
- Shadow registers for GPR0–GPR3 that allow miss code to execute without corrupting the state of any of the existing GPRs. These shadow registers are used only for servicing a TLB miss.

See Section 2.6.2, “Implementation-Specific Memory Management,” for more information about memory management for the MPC603e.

1.4.2 Cache Units

The MPC603e provides independent 16-Kbyte, four-way set-associative instruction and data caches. The cache block is 32 bytes long. The caches adhere to a write-back policy, but the PowerPC architecture allows control of cacheability, write policy, and memory coherency at the page and block levels. The caches use an LRU replacement policy.

As shown in Figure 1, the caches provide a 64-bit interface to the instruction fetch unit and LSU. The surrounding logic selects, organizes, and forwards the requested information to the requesting unit. Write operations to the cache can be performed on a byte basis, and a complete read-modify-write operation to the cache can occur in each cycle.

The load/store and instruction fetch units provide the caches with the address of the data or instruction to be fetched. In the case of a cache hit, the cache returns two words to the requesting unit.

Because the data cache tags are single ported, simultaneous load or store and snoop accesses cause resource contention. Snoop accesses have the highest priority and are given first access to the tags, unless the snoop access coincides with a tag write; in this case the snoop is retried and must rearbitrate for cache access. Loads or stores deferred due to snoop accesses are performed on the clock cycle following the snoop.

1.5 Processor Bus Interface

Because the caches are on-chip, write-back caches, the most common transactions are burst-read memory operations, burst-write memory operations, and single-beat (noncacheable or write-through) memory read and write operations. There can also be address-only operations, variants of the burst and single-beat operations, (for example, global memory operations that are snooped and atomic memory operations), and address retry activity (for example, when a snooped read access hits a modified cache block).

Memory accesses can occur in single-beat (1–8 bytes) and four-beat burst (32 bytes) data transfers when the bus is configured as 64 bits, and in single-beat (1–4 bytes), two-beat (8 bytes), and eight-beat (32 bytes) data transfers when the bus is configured as 32 bits. The address and data buses operate independently to support pipelining and split transactions during memory accesses. The MPC603e can pipeline its own transactions to a depth of one level.

Access to the system interface is granted through an external arbitration mechanism that allows devices to compete for bus mastership. This arbitration is flexible, allowing the MPC603e to be integrated into systems that implement various fairness and bus parking procedures to avoid arbitration overhead.

Typically, memory accesses are weakly ordered—sequences of operations, including load/store string and multiple instructions, do not necessarily complete in the order they begin—maximizing the efficiency of the bus without sacrificing coherency of the data. The MPC603e allows read operations to precede store operations (except when a dependency exists, or in cases where a non-cacheable access is performed), and provides support for a write operation to proceed a previously queued read data tenure (for example, allowing a snoop push to be enveloped by the address and data tenures of a read operation). Because the processor can dynamically optimize run-time ordering of load/store traffic, overall performance is improved.

1.6 System Support Functions

The MPC603e implements several support functions that include power management, time base/decrementer registers for system timing tasks, an IEEE 1149.1 (JTAG)/common on-chip processor (COP) test interface, and a phase-locked loop (PLL) clock multiplier. These system support functions are described in the following sections.

1.6.1 Power Management

The MPC603e provides four power modes, selectable by setting the appropriate control bits in the machine state register (MSR) and hardware implementation register 0 (HID0). The four power modes are as follows:

- Full-power—This is the default power state of the MPC603e. The MPC603e 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 MPC603e 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, system management interrupt, decrementer exception, hard or soft reset, or machine check brings the MPC603e into the full-power state. The MPC603e 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 MPC603e returns to the full-power state upon receipt of an external asynchronous interrupt, system management interrupt, decrementer exception, hard or soft reset, or the assertion of the machine check input ($\overline{\text{MCP}}$) signal. A return to the full-power state from the nap state takes only a few processor clock cycles.
- Sleep—Sleep mode reduces power consumption to a minimum by disabling all internal functional units; then external system logic may disable the PLL and SYSCLK. Returning the MPC603e to the full-power state requires the enabling of the PLL and SYSCLK, followed by the assertion of an external asynchronous interrupt, system management interrupt, hard or soft reset, or $\overline{\text{MCP}}$ signal after the time required to relock the PLL.

The PID7t-603e implementation offers the following enhancements to the MPC603e family:

- Lower-power design
- 2.5-volt core and 3.3-volt I/O

1.6.2 Time Base/Decrementer

The time base is a 64-bit register (accessed as two 32-bit registers) that is incremented once every four bus clock cycles; external control of the time base is provided through the time base enable (TBEN) signal. The decrementer is a 32-bit register that generates a decrementer interrupt exception after a programmable delay. The contents of the decrementer register are decremented once every four bus clock cycles, and the decrementer exception is generated as the count passes through zero.

1.6.3 IEEE 1149.1 (JTAG)/COP Test Interface

The MPC603e provides IEEE 1149.1 and COP functions for facilitating board testing and chip debugging. The IEEE 1149.1 test interface provides a means for boundary-scan testing the MPC603e and the attached board. The COP function shares the IEEE 1149.1 test port, providing a means for executing test routines, and facilitating chip and software debugging.

1.6.4 Clock Multiplier

The internal clocking of the MPC603e is generated from and synchronized to the external clock signal, SYSCLK, by means of a voltage-controlled oscillator-based PLL. The PLL provides programmable internal processor clock rates of 1x, 1.5x, 2x, 2.5x, 3x, 3.5x, and 4x multiples of the externally supplied clock frequency. The bus clock is the same frequency and is synchronous with SYSCLK. The configuration of the PLL can be read by software from the hardware implementation register 1 (HID1).

Part II
MPC603e Microprocessor Architecture 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 exception model. Implementations that conform to the OEA also adhere to the UISA and VEA.

The PowerPC architecture allows a wide range of designs for such features as cache and system interface implementations.

2.1 Implementation-Specific Information

The PowerPC architecture is derived from the IBM POWER architecture (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 scaleable to take advantage of future technological gains.

This section describes the PowerPC architecture in general, and specific details about the implementation of the MPC603e as a low-power, 32-bit member of this processor family. The main topics addressed are as follows:

- Section 2.2, “PowerPC Registers and Programming Model,” describes the registers for the operating environment architecture common among processors and describes the programming model. It also describes the additional registers that are unique to the MPC603e.
- Section 2.3, “Instruction Set and Addressing Modes,” describes the PowerPC instruction set and addressing modes, and defines and describes the instructions implemented in the MPC603e.
- Section 2.4, “Cache Implementation,” describes the cache model that is defined generally for processors that implement the VEA. It also provides specific details about the MPC603e cache implementation.
- Section 2.5, “Exception Model,” describes the exception model of the PowerPC OEA and the differences in the MPC603e exception model.
- Section 2.6, “Memory Management,” describes generally the conventions for memory management among the processors. This section also describes the MPC603e implementation of the 32-bit PowerPC memory management specification.
- Section 2.7, “Instruction Timing,” provides a general description of the instruction timing provided by the superscalar, parallel execution supported by the PowerPC architecture and the MPC603e.

- Section 2.8, “System Interface,” describes the signals implemented on the MPC603e.

The MPC603e is a high-performance, superscalar 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 processors that implement the PowerPC architecture.

The following sections summarize the features of the MPC603e, including both those that are defined by the architecture and those that are unique to the various MPC603e implementations.

Specific features of the MPC603e are listed in Section 1.1, “MPC603e Microprocessor Features.”

2.2 PowerPC Registers and Programming Model

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.

The 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 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 MPC603e 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 sections describe the PID7t-603e implementation-specific features as they apply to registers.

2.2.1 General-Purpose Registers (GPRs)

The PowerPC architecture defines 32 user-level GPRs. These registers are either 32 bits wide in 32-bit microprocessors or 64 bits wide in 64-bit microprocessors. The GPRs serve as the data source or destination for all integer instructions.

2.2.2 Floating-Point Registers (FPRs)

The PowerPC architecture also defines 32 user-level, 64-bit FPRs. The FPRs serve as the data source or destination for floating-point instructions. These registers can contain data objects of either single- or double-precision floating-point formats.

2.2.3 Condition Register (CR)

The CR is a 32-bit user-level register that provides a mechanism for testing and branching. It consists of eight 4-bit fields that reflect the results of certain operations, such as move, integer and floating-point compare, arithmetic, and logical instructions.

2.2.4 Floating-Point Status and Control Register (FPSCR)

The user-level FPSCR contains all floating-point exception signal bits, exception summary bits, exception enable bits, and rounding control bits needed for compliance with the IEEE 754 standard.

2.2.5 Machine State Register (MSR)

The MSR is a supervisor-level register that defines the state of the processor. The contents of this register are saved when an exception is taken and restored when the exception handling completes. The MPC603e implements the MSR as a 32-bit register; 64-bit processors implement a 64-bit MSR.

2.2.6 Segment Registers (SRs)

For memory management, 32-bit microprocessors implement sixteen 32-bit SRs. To speed access, the MPC603e implements the SRs as two arrays; a main array (for data memory accesses) and a shadow array (for instruction memory accesses). Loading a segment entry with the Move to Segment Register (**mtsr**) instruction loads both arrays.

2.2.7 Special-Purpose Registers (SPRs)

The PowerPC OEA defines numerous SPRs that serve a variety of functions, such as providing controls, indicating status, configuring the processor, and performing special operations. During normal execution, a program can access the registers, as shown in Figure 2, depending on the program's access privilege (supervisor or user, determined by the privilege-level bit, MSR[PR]). Note that GPRs and FPRs are accessed through operands that are part of the instructions. Access to registers can be explicit (that is, through the use of specific instructions for that purpose such as Move to Special-Purpose Register (**mtspr**) and Move from Special-Purpose Register (**mfspr**) instructions) or implicit, as the part of the execution of an instruction. Some registers are accessed both explicitly and implicitly.

In the MPC603e, all SPRs are 32 bits wide.

2.2.7.1 User-Level SPRs

The following MPC603e SPRs are accessible by user-level software:

- Link register (LR)—The LR can be used to provide the branch target address and to hold the return address after branch and link instructions. The LR is 32 bits wide in 32-bit implementations.
- Count register (CTR)—The CTR is decremented and tested automatically as a result of branch-and-count instructions. The CTR is 32 bits wide in 32-bit implementations.
- XER register—The 32-bit XER contains the summary overflow bit, integer carry bit, overflow bit, and a field specifying the number of bytes to be transferred by a Load String Word Indexed (**lswx**) or Store String Word Indexed (**stswx**) instruction.

2.2.7.2 Supervisor-Level SPRs

The MPC603e also contains SPRs that can be accessed only by supervisor-level software. These registers consist of the following:

- The DSISR defines the cause of data access and alignment exceptions.
- The data address register (DAR) holds the address of an access after an alignment or DSI exception.
- Decrementer register (DEC) is a 32-bit decrementing counter that provides a mechanism for causing a decrementer exception after a programmable delay.
- SDR1 specifies the page table format used in virtual-to-physical address translation for pages. (Note that physical address is referred to as real address in the architecture specification.)
- The machine status save/restore register 0 (SRR0) is used for saving the address of the instruction that caused the exception, and the address to return to when a Return from Interrupt (**rfi**) instruction is executed.
- The machine status save/restore register 1 (SRR1) is used to save machine status on exceptions and to restore machine status when an **rfi** instruction is executed.
- The SPRG0–SPRG3 registers are provided for operating system use.
- The external access register (EAR) controls access to the external control facility through the External Control In Word Indexed (**eciwx**) and External Control Out Word Indexed (**ecowx**) instructions.
- The time base register (TB) is a 64-bit register that maintains the time of day and operates interval timers. It consists of two 32-bit fields—time base upper (TBU) and time base lower (TBL).
- The processor version register (PVR) is a read-only register that identifies the version (model) and revision level of the processor.
- Block address translation (BAT) arrays—The PowerPC architecture defines 16 BAT registers—four pairs of data BATs (DBATs) and four pairs of instruction BATs (IBATs). See Figure 2 for a list of the SPR numbers for the BAT arrays.

The following supervisor-level SPRs are implementation-specific to the MPC603e:

- DMISS and IMISS are read-only registers that are loaded automatically upon an instruction or data TLB miss.
- HASH1 and HASH2 registers contain the physical addresses of the primary and secondary page table entry groups (PTEGs).
- ICMP and DCMP contain a duplicate of the first word in the page table entry (PTE) for which the table search is looking.
- The required physical address (RPA) register is loaded by the processor with the second word of the correct PTE during a page table search.
- The hardware implementation (HID0 and HID1) registers provide the means for enabling MPC603e checkstops and features, and allows software to read the configuration of the PLL configuration signals.
- The instruction address breakpoint register (IABR) is loaded with an instruction address that is compared to instruction addresses in the dispatch queue. When an address match occurs, an instruction address breakpoint exception is generated.

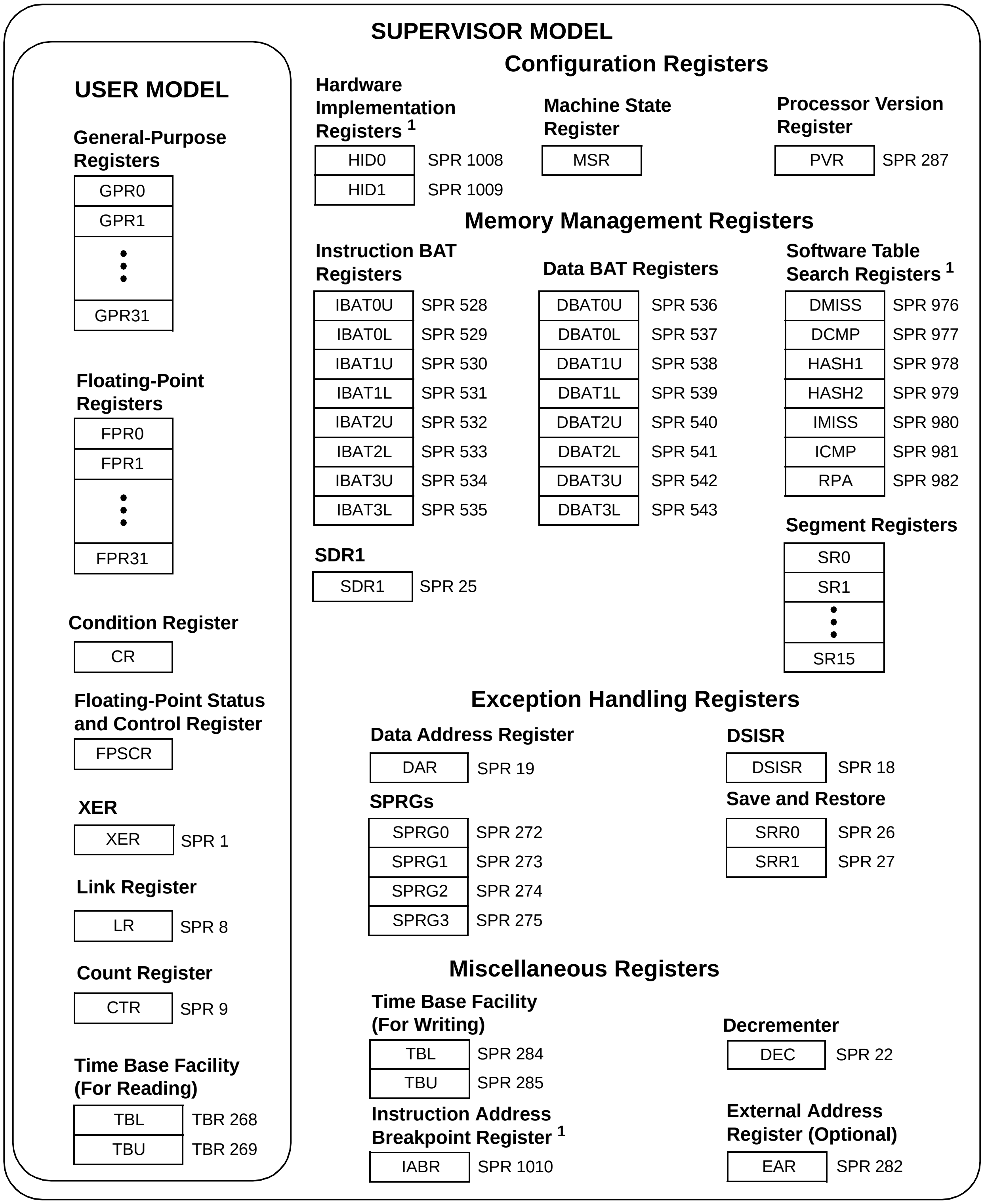
Figure 2 shows all the MPC603e registers available at the user and supervisor level. The numbers to the right of the SPRs indicate the number that is used in the syntax of the instruction operands to access the register.

2.2.7.3 Processor Version Register (PVR)

The processor version number is 6 for the PID6-603e and 7 for the PID7t-603e. The processor revision level starts at 0x0100 and changes for each chip revision. The revision level is updated on all silicon revisions.

2.2.7.4 Hardware Implementation Register 0 (HID0)

PID7t-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 MPC603e checkstops and features, and allows software to read the configuration of the PLL configuration signals.



[1] These registers are MPC603e-specific (PID6-603e and PID7t-603e). They may not be supported by other processors.

Figure 2. Programming Model—Registers

The HID0 bits with changed bit assignments are shown in Table 1. The HID0 bits that are not shown here are implemented as shown in Section 2.2.7.4, “Hardware Implementation Register 0 (HID0).”

Table 1. Additional/Changed HID0 Bits

Bits	Description
24	Enable M bit on bus for instruction fetches (IFEM) (PID7t-603e only). 0 M bit does not reflected on bus for instruction feaches. Instruction fetches are treated as nonglobal on the bus. 1 Instruction fetches reflect the M bit from the WIM settings.
25–26	Reserved
28	Address broadcast enable. Controls whether certain address-only operations (such as cache operations) are broadcast on the 60x bus. 0 Address-only operations affect only local caches and are not broadcast. 1 Address-only operations are broadcast on the 60x bus. Affected instructions are **dcbi**, **dcbf**, and **dcbst**. Note that these cache control instruction broadcasts are not snooped by the PID7t-603e. Refer to Section 2.4, “Cache Implementation.”
29–30	Reserved

2.3 Instruction Set and Addressing Modes

The following sections describe the PowerPC instruction set and addressing modes in general.

2.3.1 PowerPC Instruction Set and Addressing Modes

All PowerPC instructions are encoded as single-word (32-bit) opcodes. Instruction formats are consistent among all instruction types, permitting efficient decoding to occur in parallel with operand accesses. This fixed instruction length and consistent format greatly simplifies instruction pipelining.

2.3.1.1 PowerPC Instruction Set

The PowerPC instructions are divided into the following categories:

- Integer instructions—These include computational and logical instructions.
  - Integer arithmetic instructions
  - Integer compare instructions
  - Integer logical instructions
  - Integer rotate and shift instructions
- Floating-point instructions—These include floating-point computational instructions, as well as instructions that affect the FPSCR.
  - Floating-point arithmetic instructions
  - Floating-point multiply/add instructions
  - Floating-point rounding and conversion instructions
  - Floating-point compare instructions
  - Floating-point status and control instructions
- Load/store instructions—These include integer and floating-point load and store instructions.

- — Integer load and store instructions
- — Integer load and store multiple instructions
- — Floating-point load and store
- — Primitives used to construct atomic memory operations (**lwarx** and **stwcx.** instructions)

- Flow control instructions—These include branching instructions, condition register logical instructions, trap instructions, and other instructions that affect the instruction flow.
  - — Branch and trap instructions
  - — Condition register logical instructions
- Processor control instructions—These instructions are used for synchronizing memory accesses and management of caches, TLBs, and the segment registers.
  - — Move to/from SPR instructions
  - — Move to/from MSR
  - — Synchronize
  - — Instruction synchronize
- Memory control instructions—These instructions provide control of caches, TLBs, and segment registers.
  - — Supervisor-level cache management instructions
  - — User-level cache instructions
  - — Segment register manipulation instructions
- Translation lookaside buffer management instructions

Note that this grouping of instructions does not indicate the execution unit that executes a particular instruction or group of instructions.

Integer instructions operate on byte, half-word, and word operands. Floating-point instructions operate on single-precision (one word) and double-precision (one double word) floating-point operands. The PowerPC architecture uses instructions that are 4 bytes long and word-aligned. It provides for byte, half-word, and word operand loads and stores between memory and a set of 32 GPRs. It also provides for word and double-word operand loads and stores between memory and a set of 32 FPRs.

Computational instructions do not modify memory. To use a memory operand in a computation and then modify the same or another memory location, the memory contents must be loaded into a register, modified, and then written back to the target location with distinct instructions.

The processors follow the program flow when they are in the normal execution state. However, the flow of instructions can be interrupted directly by the execution of an instruction or by an asynchronous event. Either kind of exception may cause one of several components of the system software to be invoked.

2.3.2 Calculating Effective Addresses

The effective address (EA) is the 32-bit address computed by the processor when executing a memory access or branch instruction or when fetching the next sequential instruction.

The PowerPC architecture supports two simple memory addressing modes:

- EA = (**r**A|0) + offset (including offset = 0) (register indirect with immediate index)
- EA = (**r**A|0) + **r**B (register indirect with index)

These simple addressing modes allow efficient address generation for memory accesses. Calculation of the effective address for aligned transfers occurs in a single clock cycle.

For a memory access instruction, if the sum of the effective address and the operand length exceeds the maximum effective address, the memory operand is considered to wrap around from the maximum effective address to effective address 0.

Effective address computations for both data and instruction accesses use 32-bit unsigned binary arithmetic. A carry from bit 0 is ignored in 32-bit implementations.

2.3.3 Implementation-Specific Instruction Set

The MPC603e instruction set is defined as follows:

- The MPC603e provides hardware support for all 32-bit PowerPC instructions.
- The MPC603e provides two implementation-specific instructions used for software table search operations following TLB misses:
  - Load Data TLB Entry (**tlbld**)
  - Load Instruction TLB Entry (**tlbli**)
- The MPC603e implements the following instructions which are defined as optional by the PowerPC architecture:
  - External Control In Word Indexed (**eciwx**)
  - External Control Out Word Indexed (**ecowx**)
  - Floating Select (**fsel**)
  - Floating Reciprocal Estimate Single-Precision (**fres**)
  - Floating Reciprocal Square Root Estimate (**frsqrte**)
  - Store Floating-Point as Integer Word (**stfiwx**)

2.4 Cache Implementation

The following sections describe the general cache characteristics as implemented in the PowerPC architecture, and the MPC603e implementation, specifically. PID7t-603e specific information is noted where applicable.

2.4.1 PowerPC Cache Characteristics

The PowerPC architecture does not define hardware aspects of cache implementations. These 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 MPC603e, 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.

2.4.2 Implementation-Specific Cache Implementation

The MPC603e 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 MPC603e, the block size is equivalent to a cache line. A block diagram of the data cache organization is shown in Figure 3.

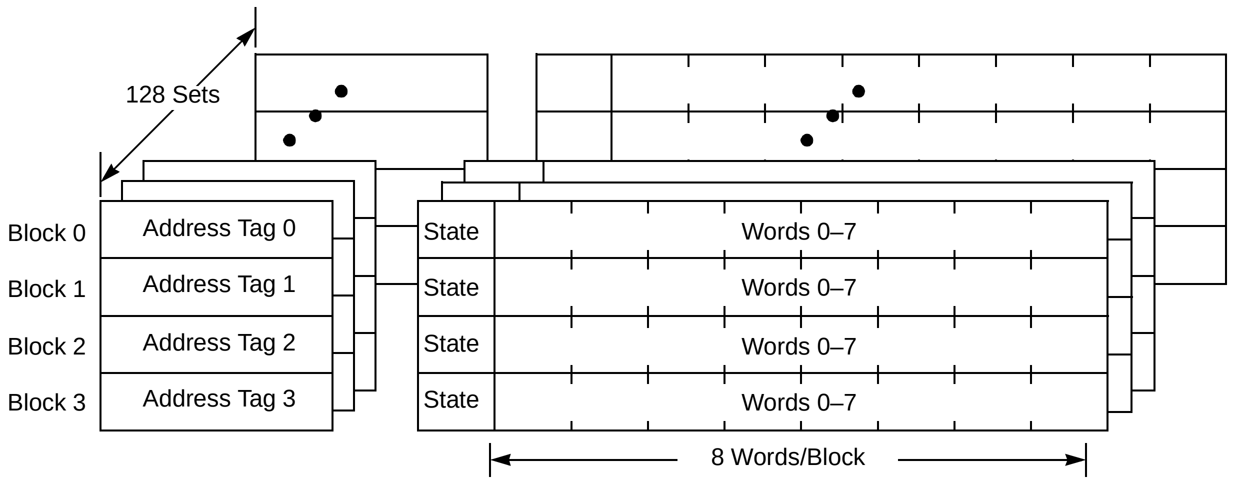


Figure 3. Data Cache Organization

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 PID7t-603e, the instruction cache is blocked only until the critical load completes. The PID7t-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 MPC603e cache blocks are loaded in four beats of 64 bits each when the MPC603e is configured with a 64-bit data bus. When the MPC603e 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 PID7t-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 MPC603e implements the MEI protocol. The following three states indicate the state of the cache block:

- 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. Because the MPC603e data cache tags are single-ported, a simultaneous load or store and snoop access represents 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.

2.5 Exception Model

This section describes the PowerPC exception model and the MPC603e implementation, specifically. PID7t-603e-specific information is noted where applicable.

2.5.1 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 differs 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 type. 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 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 MPC603e 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 MPC603e).
- 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 on 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 MSR[RI].

2.5.2 Implementation-Specific Exception Model

As specified by the PowerPC architecture, all MPC603e 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 MPC603e, are caused by instructions. The MPC603e exception classes are shown in Table 2.

Table 2. 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 MPC603e handles uniquely. Note that Table 2 includes no synchronous imprecise instructions. While the PowerPC architecture supports imprecise handling of floating-point exceptions, the MPC603e implements floating-point exception modes as precise exceptions.

The MPC603e 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 $\overline{\text{TEA}}$ 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 **eciwx** or **ecowx** 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 **eciwx** or **ecowx** 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: • 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. • The fetch access is to a direct-store segment (indicated by SRR1[3] set). • 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.
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 MPC603e cannot perform a memory access for any of the reasons described below: • The operand of a floating-point load or store instruction is not word-aligned. • The operand of **lmw**, **stmw**, **lwarx**, and **stwcx.** instructions are not aligned. • The operand of a single-register load or store operation is not aligned, and the MPC603e is in little-endian mode (PID6-603e only). • The execution of a floating-point load or store instruction to a direct-store segment. • 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. • Execution of a misaligned **eciwx** or **ecowx** instruction (PID7t-603e only). • The instruction is **lmw**, **stmw**, **lswi**, **lswx**, **stswi**, **stswx** and the MPC603e is in little-endian mode. • The operand of **dcbz** is in memory that is write-through-required or caching-inhibited.

Table 3. Exceptions and Conditions (continued)

Exception Type	Vector Offset (hex)	Causing Conditions
Program	00700	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: • Floating-point enabled exception—A floating-point enabled exception condition is generated when the following condition is met: (MSR[FE0] | MSR[FE1]) & 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. • 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 MPC603e), or when execution of an optional instruction not provided in the MPC603e is attempted (these do not include those optional instructions that are treated as no-ops). • 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 MPC603e, this exception is generated for **mtspr** or **mfspr** with an invalid SPR field if SPR[0] = 1 and MSR[PR] = 1. This may not be true for all processors. • Trap—A trap type program exception is generated when any of the conditions specified in a trap instruction is met.
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 DEC[31] changes from 0 to 1. Must also be enabled with MSR[EE].
Reserved	00A00–00BFF	—
System call	00C00	A system call exception occurs when a System Call (**sc**) 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 MPC603e does not generate an exception to this vector. Other 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 IABR[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	—

2.6 Memory Management

The following sections describe the memory management features of the PowerPC architecture and the MPC603e implementation, respectively.

2.6.1 PowerPC Memory Management

The primary functions of the MMU are to translate logical (effective) addresses to physical addresses for memory accesses, and to provide access protection on blocks and pages of memory.

There are two types of accesses generated by the MPC603e that require address translation— instruction accesses, and data accesses to memory generated by load and store instructions.

The PowerPC MMU and exception model support demand-paged virtual memory. Virtual memory management permits execution of programs larger than the size of physical memory; demand-paged implies that individual pages are loaded into physical memory from system memory only when they are first accessed by an executing program.

The hashed page table is a variable-sized data structure that defines the mapping between virtual page numbers and physical page numbers. The page table size is a power of two, and its starting address is a multiple of its size.

The page table contains a number of page table entry groups (PTEGs). A PTEG contains eight page table entries (PTEs) of 8 bytes each; therefore, each PTEG is 64 bytes long. PTEG addresses are entry points for table search operations.

Address translations are enabled by setting bits in the MSR—MSR[IR] enables instruction address translations and MSR[DR] enables data address translations.

2.6.2 Implementation-Specific Memory Management

The instruction and data memory management units in the MPC603e provide 4 Gbytes of logical address space accessible to supervisor and user programs with a 4-Kbyte page size and 256-Mbyte segment size. Block sizes range from 128 Kbytes to 256 Mbytes and are software selectable. In addition, the MPC603e uses an interim 52-bit virtual address and hashed page tables for generating 32-bit physical addresses. The MMUs in the MPC603e rely on the exception processing mechanism for the implementation of the paged virtual memory environment and for enforcing protection of designated memory areas.

Instruction and data TLBs provide address translation in parallel with the on-chip cache access, incurring no additional time penalty in the event of a TLB hit. A TLB is a cache of the most recently used page table entries. Software is responsible for maintaining the consistency of the TLB with memory. The MPC603e TLBs are 64-entry, two-way set-associative caches that contain instruction and data address translations. The MPC603e provides hardware assist for software table search operations through the hashed page table on TLB misses. Supervisor software can invalidate TLB entries selectively.

The MPC603e also provides independent four-entry BAT arrays for instructions and data that maintain address translations for blocks of memory. These entries define blocks that can vary from 128 Kbytes to 256 Mbytes. The BAT arrays are maintained by system software.

As specified by the PowerPC architecture, the hashed page table is a variable-sized data structure that defines the mapping between virtual page numbers and physical page numbers. The page table size is a power of two, and its starting address is a multiple of its size.

Also as specified by the PowerPC architecture, the page table contains a number of PTEGs. A PTEG contains 8 PTEs of 8 bytes each; therefore, each PTEG is 64 bytes long. PTEG addresses are entry points for table search operations.

2.7 Instruction Timing

The MPC603e is a pipelined superscalar processor. The processing of an instruction is performed in discrete stages by a pipelined processor. 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 MPC603e instruction pipeline 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, if possible, the BPU decodes branches during the fetch stage and folds out branch instructions before the dispatch stage.
- 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.
- In the execute pipeline stage, each execution unit with 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 when the execution has finished. 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 FPU pipeline stages are multiply, add, and round-convert. The LSU has two pipeline stages. The first stage is for effective address calculation and MMU translation, and the second is for accessing 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 issues multiple independent instructions into multiple pipelines allowing instructions to execute in parallel. The MPC603e 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 PID7t-603e has been improved, with the **divwu***x* and **divw***x* instructions executing in 20 clock cycles, instead of the 37 cycles required in the PID6-603e.

The MPC603e 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.

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

2.8 System Interface

The system interface is specific for each microprocessor implementation.

The MPC603e provides a versatile system interface that allows for a wide range of implementations. The interface includes a 32-bit address bus, a 32- or 64-bit data bus, and 56 control and information signals (see Figure 4). The system interface allows for address-only transactions, as well as address and data transactions. The MPC603e control and information signals include the address arbitration, address start, address transfer, transfer attribute, address termination, data arbitration, data transfer, data termination, and processor state signals. Test and control signals provide diagnostics for selected internal circuits.

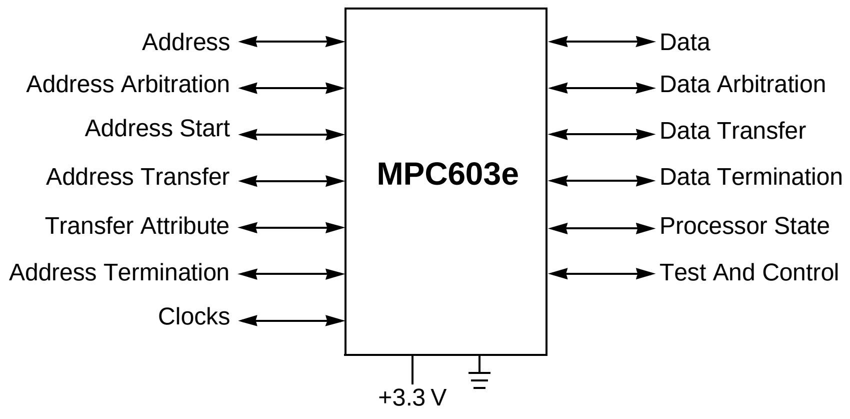


Figure 4. System Interface

The system interface supports bus pipelining, allowing the address tenure of one transaction to overlap the data tenure of another. The extent of the pipelining depends on external arbitration and control circuitry. Similarly, the MPC603e supports split-bus transactions for systems with multiple potential bus masters—one device can have mastership of the address bus while another has mastership of the data bus.

Allowing multiple bus transactions to occur simultaneously increases the available bus bandwidth for other activity, and as a result, improves performance.

The MPC603e supports multiple masters through a bus arbitration scheme that allows various devices to compete for the shared bus resource. The arbitration logic can implement priority protocols, such as fairness, and can park masters to avoid arbitration overhead. The MEI protocol ensures coherency among multiple devices and system memory. Also, the MPC603e on-chip caches, TLBs, and optional second-level caches can be controlled externally.

The MPC603e clocking structure allows the bus to operate at integer multiples of the processor cycle time.

The following sections describe the MPC603e bus support for memory operations. Note that some signals perform different functions depending on the addressing protocol used.

2.8.1 Memory Accesses

The MPC603e data bus is configured at power-up to either a 32- or 64-bit width. When the MPC603e is configured with a 32-bit data bus, memory accesses allow transfer sizes of 8, 16, 24, or 32 bits in one bus clock cycle. Data transfers occur in either single-beat transactions, two-beat or eight-beat burst transactions, with a single-beat transaction transferring as many as 32 bits. Single- or double-beat transactions are caused by noncached accesses that access memory directly (that is, reads and writes when caching is disabled, caching-inhibited accesses, and stores in write-through mode). Eight-beat burst transactions, which always transfer an entire cache line (32 bytes), are initiated when a line is read from or written to memory.

When the MPC603e is configured with a 64-bit data bus, memory accesses allow transfer sizes of 8, 16, 24, 32, 40, 48, 56, or 64 bits in one bus clock cycle. Data transfers occur in either single-beat transactions or four-beat burst transactions. Single-beat transactions are caused by noncached accesses that access memory directly (that is, reads and writes when caching is disabled, caching-inhibited accesses, and stores in write-through mode). Four-beat burst transactions, which always transfer an entire cache line (32 bytes), are initiated when a line is read from or written to memory.

2.8.2 Signals

The MPC603e signals are grouped as follows:

- Address arbitration signals—The MPC603e uses these signals to arbitrate for address bus mastership.
- Address transfer start signals—These signals indicate that a bus master has begun a transaction on the address bus.
- Address transfer signals—These signals, consisting of address bus, address parity, and address parity error signals, are used to transfer the address and to ensure the integrity of the transfer.
- Transfer attribute signals—These signals provide information about the type of transfer, such as the transfer size and whether the transaction is bursted, write-through, or caching-inhibited.
- Address transfer termination signals—These signals are used to acknowledge the end of the address phase of the transaction. They also indicate whether a condition exists that requires the address phase to be repeated.
- Data arbitration signals—The MPC603e uses these signals to arbitrate for data bus mastership.
- Data transfer signals—These signals, consisting of data bus, data parity, and data parity error signals, are used to transfer the data and to ensure the integrity of the transfer.

- Data transfer termination signals—Data termination signals are required after each data beat in a data transfer. In a single-beat transaction, the data termination signals also indicate the end of the tenure. In burst accesses, the data termination signals apply to individual beats and indicate the end of the tenure only after the final data beat. They also indicate whether a condition exists that requires the data phase to be repeated.
- System status signals—These signals include the interrupt signal, checkstop signals, and soft- and hard-reset signals. They are used to interrupt and, under various conditions, to reset the processor.
- Processor state signals—These signals indicate the state of the reservation coherency bit, enable the time base, provide machine quiescence control, and can be used to cause a machine halt on execution of a **tlbsync** instruction.
- IEEE 1149.1 (JTAG)/COP interface signals—The IEEE 1149.1 test unit and common on-chip processor (COP) unit are accessed through a shared set of input, output, and clocking signals. The IEEE 1149.1/COP interface provides a means for boundary scan testing and internal debugging of the MPC603e.
- Test interface signals—These signals are used for production testing.
- Clock signals—These signals determine the system clock frequency and can be used to synchronize multiprocessor systems.

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.

2.8.3 Signal Configuration

Figure 5 illustrates the MPC603e logical pin configuration, showing how the signals are grouped.

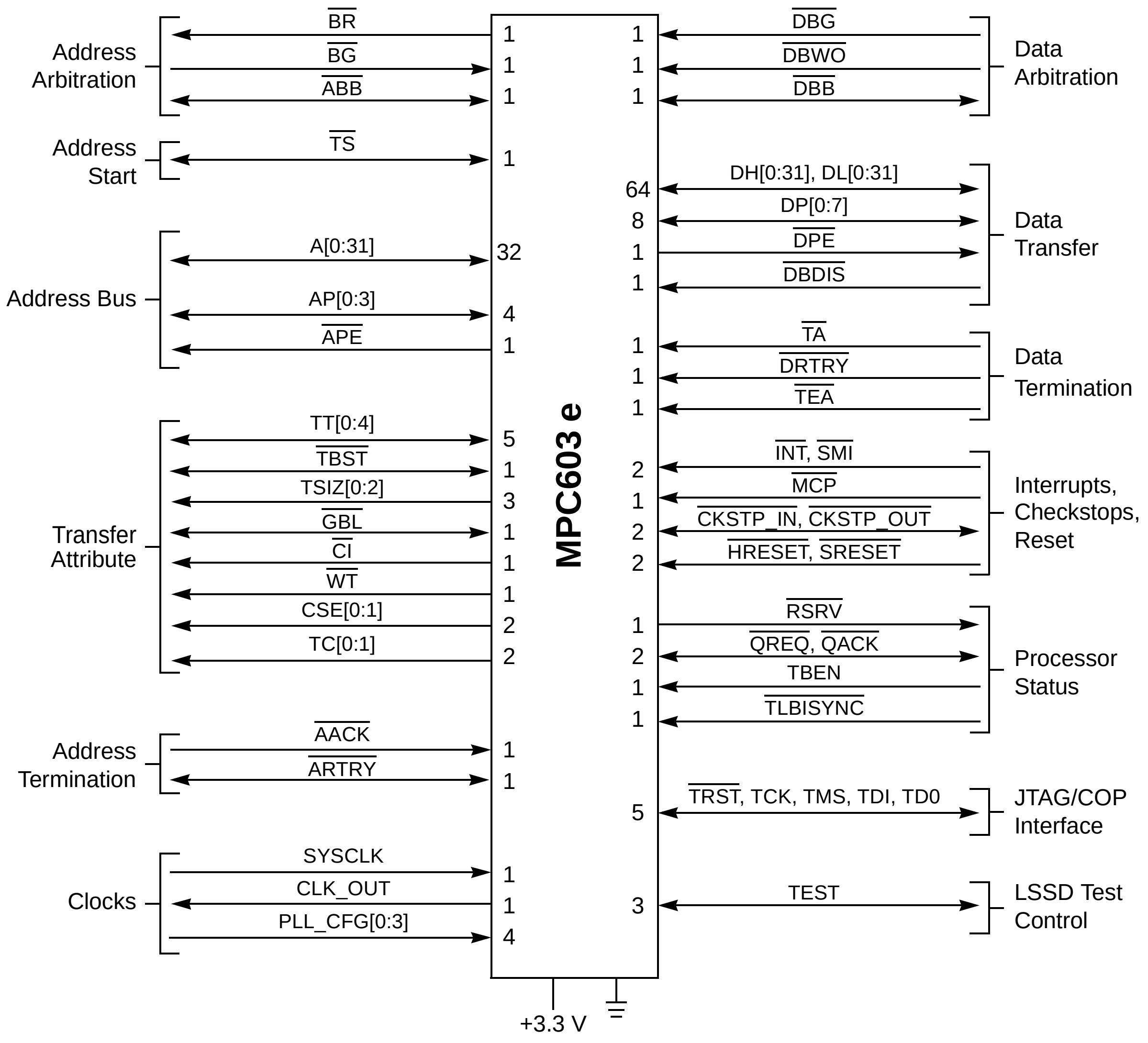


Figure 5. Signal Groups

2.9 System Design and Programming Considerations

The MPC603e is a low-power dissipation, low cost, and high performance device. It also provides the system designer additional capabilities through higher processor clock speeds (to 100 MHz), increases in cache size (16-Kbyte instruction and data caches) and set associativity (four-way), and greater system clock flexibility compared with the MPC603e. The following sections describe the differences between the MPC603 and MPC603e that affect the system designer and programmer already familiar with the operation of the MPC603.

The design enhancements to the MPC603e are described in the following sections as changes that can require a modification to the hardware or software configuration of a system designed for the MPC603.

2.9.1 Hardware Features

The following hardware features of the MPC603e may require modifications to MPC603 systems.

2.9.1.1 Replacement of $\overline{\text{XATS}}$ Signal by CSE1 Signal

The MPC603e employs four-way set associativity for both the instruction and data caches, in place of the two-way set associativity used in the MPC603 . This change requires the use of an additional cache set entry (CSE1) signal to indicate which member of the cache set is being loaded during a cache line fill. CSE1 on the MPC603e is in the same pin location as $\overline{\text{XATS}}$ on the MPC603 . Note that $\overline{\text{XATS}}$ is no longer needed by the MPC603e because support for access to direct-store segments has been removed.

Table 4 shows the CSE[0:1] signal encoding indicating the cache set element selected during a cache load operation.

Table 4. CSE[0:1] Signals

CSE[0:1]	Cache Set Element
00	Set 0
01	Set 1
10	Set 2
11	Set 3

2.9.1.2 Addition of Half-Clock Bus Multipliers

Some of the reserved clock configuration signal settings of the MPC603 are redefined to allow more flexible selection of higher internal and bus clock frequencies. The MPC603e provides programmable internal processor clock rates of 1x, 1.5x, 2x, 2.5x, 3x, 3.5x, and 4x multiples of the externally supplied clock frequency. For additional information, refer to the appropriate device-specific hardware specifications.

2.9.2 Software Features

The features of the MPC603e described in the following sections affect software originally written for the MPC603 .

2.9.2.1 16-Kbyte Instruction and Data Caches

The MPC603e instruction and data caches are 16 Kbytes, twice the size of the MPC603 caches. The increase in cache size may require modification of cache flush routines. The increase is also reflected in four-way set associativity of both caches in place of the two-way set associativity in the MPC603.

2.9.2.2 Clock Configuration Available in HID1 Register

HID1[0–3] of the MPC603e provides software read-only access to the configuration of the PLL_CFG signals. HID1 is not implemented in the MPC603.

2.9.2.3 Performance Enhancements

The following enhancements improve performance without requiring changes to software (other than compiler optimization) or hardware designed for the MPC603:

- Support for single-cycle store

- Addition of adder/comparator in the SRU allows dispatch and execution of multiple integer add and compare instructions on each cycle.
- Addition of SRR1[KEY] to provide information about memory protection violations prior to page table search operations. This bit is set when the combination of the settings in the appropriate SR[Kx] and in the MSR[PR] bit indicate that when the PTE[PP] bits are either 00 or 01, a protection violation exists. If this is the case for a data write operation with a DTLB miss, the changed (C) bit in the page tables should not be updated (see Table 5). This reduces the time required to execute the page table search routine because the software no longer has to explicitly read both SR[Kx] and MSR[PR] to determine whether a protection violation exists before updating the C bit.

Table 5. Generated SRR1 [KEY] Bit

Segment Register [Ks, Kp]	MSR[PR]	SRR1[KEY] Generated on DTLB Misses
0x	0	0
x0	1	0
1x	0	1
x1	1	1

Note: SRR1[KEY] indicates a protection violation if the PTE[pp] bits are 00 or 01.

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

Motorola Japan Ltd.;
SPS, Technical Information Center,
3-20-1, Minami-Azabu Minato-ku,
Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.;
Silicon Harbour Centre, 2 Dai King Street,
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

http://www.motorola.com/semiconductors

DOCUMENT COMMENTS:

FAX (512) 933-2625,
Attn: RISC Applications Engineering

MOTOROLA

MPC603E/D