IBM PowerPC® 750FX RISC Microprocessor

Technical Summary

Version: 1.0

Preliminary

May 30, 2002

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both.

IBM	IBM Logo
PowerPC	PowerPC Logo
PowerPC Architecture	PowerPC 750
PowerPC 750CX	PowerPC 750CXe
PowerPC 750FX	

Other company, product and service names may be trademarks or service marks of others.

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in implantation or other life support, space, nuclear, or military applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change IBM product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. All information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.

While the information contained herein is believed to be accurate, such information is preliminary, and should not be relied upon for accuracy or completeness, and no representations or warranties of accuracy or completeness are made.

Note: This document contains information on products in the sampling and/or initial production phases of development. This information is subject to change without notice. Verify with your IBM field applications engineer that you have the latest version of this document before finalizing a design.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY 12533-6351

The IBM home page can be found at
http://www.ibm.com

The IBM Microelectronics Division home page
can be found at http://www.chips.ibm.com

Title_750FX_Tech_Sum.fm.1.0
May 30, 2002

Contents

List of Tables

List of Figures

1. IBM PowerPC® 750FX Technical Summary

The IBM PowerPC® 750FX RISC microprocessor is an implementation of the PowerPC Architecture™ with enhancements based on the IBM PowerPC 750™ and 750CXe™ RISC microprocessor designs. This section provides an overview of the PowerPC 750FX microprocessor features, including a block diagram showing the major functional components. It also provides information about how PowerPC 750FX implementation complies with the PowerPC Architecture definition.

Note: In this document PowerPC 750FX RISC Microprocessor is abbreviated as PowerPC 750FX or 750FX.

1.1 Introduction

The 750FX processor is a 32-bit implementation of the PowerPC Architecture in a 0.13 micron CMOS technology with six levels of copper interconnect. The 750FX is designed for high performance (up to 1GHz) and low power consumption. It provides a super set of functionality to the PowerPC 750 processor, including a complete 60x bus interface, and enhancements such as an onboard 512KB L2 cache.

2. Features

The following is a brief list of some of the key features of the 750FX processor:

- Implements full PowerPC 32-bit architecture
- Four (4) stage Pipeline control
  - Fetch
  - Dispatch and decode
  - Execute
  - Complete/Write back
- Dual Issue superscalar control
  - A maximum of two instructions completed plus one branch folded per cycle
  - Two instructions dispatched in one of the combinations shown in *Table 2-1*

Table 2-1. Instructions

inst0 \ inst1	fxu1	fxu2	lsu	fpu	sysu
fxu1		X	X	X	X
fxu2	X		X	X	X
lsu	X	X		X	X
fpu	X	X	X		X
sysu	X	X	X	X	

- Branch folding
- Buffers for architectural registers to support fast speculation correction
- Precise interrupt handling
- Branch processing unit
  - Four instructions fetched per clock
  - One branch processed per cycle (plus resolving two speculations)
  - Up to one speculative stream in execution, one additional speculative stream in fetch
  - 512-entry branch history table (BHT) for dynamic prediction
  - 64-entry, 4-way set associative branch target instruction cache (BTIC) for eliminating branch delay slots
- Dispatch unit
  - Full hardware detection of dependencies (resolved in the execution units)
  - Dispatch two instructions to six independent units (system, branch, load/store, fixed-point unit 1, fixed-point unit 2, or floating-point)
  - 4-stage pipeline: fetch, dispatch, execute, and complete
  - Serialization control (predispatch, postdispatch, execution, serialization)
- Decode
  - Register file access
  - Forwarding control
  - Partial instruction decode

- Load/store unit
  - One cycle load or store cache access (byte, half-word, word, double-word)
  - Effective address generation
  - Non-blocking cache (up to two outstanding misses)
  - Single-cycle misaligned access within double word boundary
  - Alignment, zero padding, sign extend for integer register file
  - Floating-point internal format conversion (alignment, normalization)
  - Sequencing for load/store multiples and string operations
  - Store gathering
  - Cache and TLB instructions
  - Big- and little-endian byte addressing supported
  - Misaligned little-endian support in hardware
- Fixed-point units
  - Fixed-point unit 1 (FXU1): multiply, divide, shift, rotate, arithmetic, logical
  - Fixed-point unit 2 (FXU2): shift, rotate, arithmetic, logical
  - Single-cycle arithmetic, shift, rotate, logical
  - Multiply and divide support (multi-cycle)
  - Early out multiply
  - Thirty-two, 32-bit general purpose registers
- Floating-point unit
  - Support for IEEE-754 standard single and double-precision floating-point arithmetic
  - Optimized for single-precision multiply/add
  - Thirty-two, 64-bit floating point registers
  - Enhanced reciprocal estimates
  - 3-cycle latency, 1-cycle throughput, single-precision multiply-add
  - 3-cycle latency, 1-cycle throughput, double-precision add
  - 4-cycle latency, 2-cycle throughput, double-precision multiply-add
  - Hardware support for divide
  - Hardware support for denormalized numbers
  - Time deterministic non-IEEE mode
- System unit
  - Executes CR logical instructions and miscellaneous system instructions
  - Special register transfer instructions
- L1 Cache structure
  - 32KB, 32-byte line, 8-way set associative instruction cache
  - 32KB, 32-byte line, 8-way set associative data cache
  - Single-cycle cache access
  - Pseudo-LRU replacement
  - Copy-back or write-through data cache (on a page per page basis)
  - 3-state (MEI) memory coherency
  - Hardware support for data coherency
  - Non-blocking instruction cache (one outstanding miss)
  - Non-blocking data cache (two outstanding misses)[1]
  - No snooping of instruction cache
  - Parity added for L1 tags and caches[1]

1. Not supported under DD 1.X.

- Memory management unit
  - 128 entry, two way set associative instruction (data) TLB
  - Hardware reload for TLB's
  - 8 instruction BATs and 8 data BATs
  - Virtual memory support for up to 4 exabytes (2^{52}) virtual memory
  - Real memory support for up to 4 gigabytes (2^{32}) of physical memory
- Level 2 (L2) cache
  - 512KB, two-way set associative unified L2 cache
  - Copy-back or write-through data cache on a page basis, or for all L2
  - 64-byte line size, two sectors per line
  - L2 frequency at core speed
  - On-board ECC[1]
    - Parity added for L2 tags
  - Supports up to 2 outstanding misses (1 data and 1 instruction)
  - Supports up to 2 outstanding misses (2 data)[1]
  - Cache locking by way[1]
- Bus interface
  - Compatible with 60x
  - 32-bit address bus
  - 64-bit data bus (or 32-bit mode)
  - Enhanced 60x bus: pipelines back-to-back reads to a depth of 2
  - Core-to-bus frequency multipliers of 2x, 2.5x, 3x, 3.5x, 4x, 4.5x, 5x, 5.5x, 6x, 6.5x, 7x, 7.5x, 8x, 8.5x, 9x, 9.5x, 10x, 11x, 12x, 13x, 14x, 15x, 16x, 17x, 18x, 19x, and 20x supported
- Power
  - Dual PLLs for seamless frequency switching
  - 3 static power saving modes: doze, nap, and sleep
  - Dynamic power management (DPM)[1]
- Reliability and serviceability
  - Parity checking on 60x busses
  - Parity checking on internal arrays[1]
  - ECC checking on L2 cache
  - Parity on the L1 arrays[1]
  - Parity on the L1 and L2 tags[1]
- Testability
  - LSSD scan design for high percentage stuck-fault test coverage
  - AC LSSD scan design for signal transition fault coverage
  - Programmable array built-in self test for complete test coverage of large arrays
  - Powerful diagnostic and test interface through Common On-Chip Processor (COP) and IEEE 1149.1 (JTAG) interface

1. Not supported under DD 1.X.

3. Summary of Enhancements

This section provides a summary of the major enhancements for the PowerPC 750.

3.1 Performance

The performance enhancements included in the 750FX microprocessor are:

- Up to 1GHz operating frequency
- 512KB on-board L2 cache with locking by way[1]
- L1 data cache availability improvements
- Bus pipelining (consecutive data reads)
- Additional floating point unit (FPU) reservation station and improved reciprocal estimates
- Eight (8) data BATs and instruction BATs

3.1.1 L2 Cache

The 750FX has an internal L2 cache capacity of 512KB. The cache is two-way set associative; each way contains 4096 blocks and each block consists of two 32-byte sectors. Array read and write operations execute in one processor cycle. Writes to the array are 64 bits wide, while reads are 256 bits wide.

In addition, the L2 cache has an 8-bit ECC for every 64-bit word in memory that can be used to correct a majority of single bit errors and detect multiple bit errors.

The L2 tags also support parity and locking by way.[1]

3.1.2 L1 Data Cache

The 750FX L1 data cache supports miss-under-miss access, meaning that with one miss outstanding, the cache can continue to be accessed until a second miss occurs. Additionally, the 750FX L1 data cache allows the second miss to initiate a transaction in the bus interface unit, while the first miss is pending[1].

In previous PowerPC 750 microprocessor implementations the data bus width for bus interface unit (BIU) accesses of the L1 data cache array was 64 bits. To cast out or to reload a 256-bit cache line required four access cycles. On the 750FX, this bus has been expanded to 256 bits. As a result, cache line data bursts can be read from or written to the cache array in a single cycle, reducing cache contention between the BIU and the load-store unit.

3.1.3 Bus Pipelining

The 60x bus has decoupled address and data busses, allowing transactions to be pipelined on the bus. That is, the address operation for a second transaction can be initiated before the data operation for a previous transaction is complete. The 750FX BIU allows pipelined read transactions to a depth of two back-to-back reads.

1. Not supported under DD 1.X.

3.1.4 Floating Point Unit Reservation Station

The floating point unit (FPU) consists of a three-stage pipeline. Formerly, there was a restriction that if the pipeline was full with three (3) floating point instructions, a single cycle stall would occur before a new instruction could be loaded into the floating point unit. The 750FX has added a second reservation station so the floating point unit can now accept a floating point instruction every cycle.

3.2 Power

The 750FX design includes two PLLs (PLL0 and PLL1), allowing the processor clock frequency to be changed on-the-fly to match processing requirements.

During reset PLL0 is selected to provide the internal processor (i.e. core) clock. The external clock to core clock multiplier is selected using external pins. Thereafter, PLL0 and PLL1 may be controlled using software. The HID1 register contains fields that specify the frequency range of each PLL, the clock multiplier for each PLL, external or internal control of PLL0, and a bit to choose which PLL is selected as the source of the processor clock at any given time.

3.3 Reliability

Parity is implemented for the following arrays: I-Cache, I-Tag, D-Cache, D-Tag, and L2 Tag. Parity errors will result in a machine check interrupt which is not recoverable.[1]

3.4 Pin Interfaces

3.4.1 60x Interface

The 750FX processor defines a versatile bus interface which allows for a wide range of system performance versus system complexity trade-offs to be exploited. The interface defines a 72-bit data bus (64 bits of data and 8 bits of parity), a 36-bit address bus (32 bits of address and 4 bits of parity) along with sufficient control signals to allow for unique system level optimizations. The interface allows for address-only transactions as well as address and data transactions. The 750FX initiates an address-only transaction for a **dcbz** cache control instruction, but can be configured (via HID0) to initiate address-only transactions for other data cache control instructions, **eieio**, and **sync**. The **dcbz** is the only address-only transaction that is snooped by the 750FX. The 750FX will generate or accept either one beat or four beat data transactions, although it is possible for other bus participants to perform longer data transfers.

In addition, the 750FX processor supports cache snooping. The bus interface defines the signals required to build systems which require a coherent memory system. It is envisioned that a wide range of system implementations can be constructed from the defined interface.

3.4.2 Configuration and Power Pins

Signals introduced in the 750FX design include PLL_CFG[4] and PLL_RNG[0:1]. PLL_CFG[4] provides an additional PLL configuration bit to support additional processor to bus clock frequency ratios. PLL_RNG[0:1] are used to select a frequency range for the processor clock. This allows a wider range of frequencies to be supported by the PLLs. In addition, a second analog voltage pin, AVdd1, has been added to power the second PLL.

I/O voltage is selectable through using the BVSEL pin and L1_TSTCLK pin.

Table 3-1. I/O Voltage Selection Settings

Voltage Selection	Bvsel	L1tstclk (Second Vsel Signal)
Reserved	0	0
1.2, 1.5, or 1.8V	0	1
2.5V	1	1
3.3V	1	0

1. Not support under DD 1.X.

4. The 750FX Functional Units

This section presents a brief overview and high-level block diagrams of each functional unit of the processor. The 750FX processor is partitioned into the following subsystems:

- Control Unit
- Fixed Point and Floating Point Units
- Load/Store Unit
- Cache and Memory Management Unit
- Bus Interface Unit

4.1 Principles of Operation

The block diagram of the 750FX is shown in *Figure 4-1 on page 19*. The instruction fetch unit prefetches instructions from the instruction cache into the instruction buffers. The branch unit identifies any branch instructions in the instruction buffers and folds them out if possible.

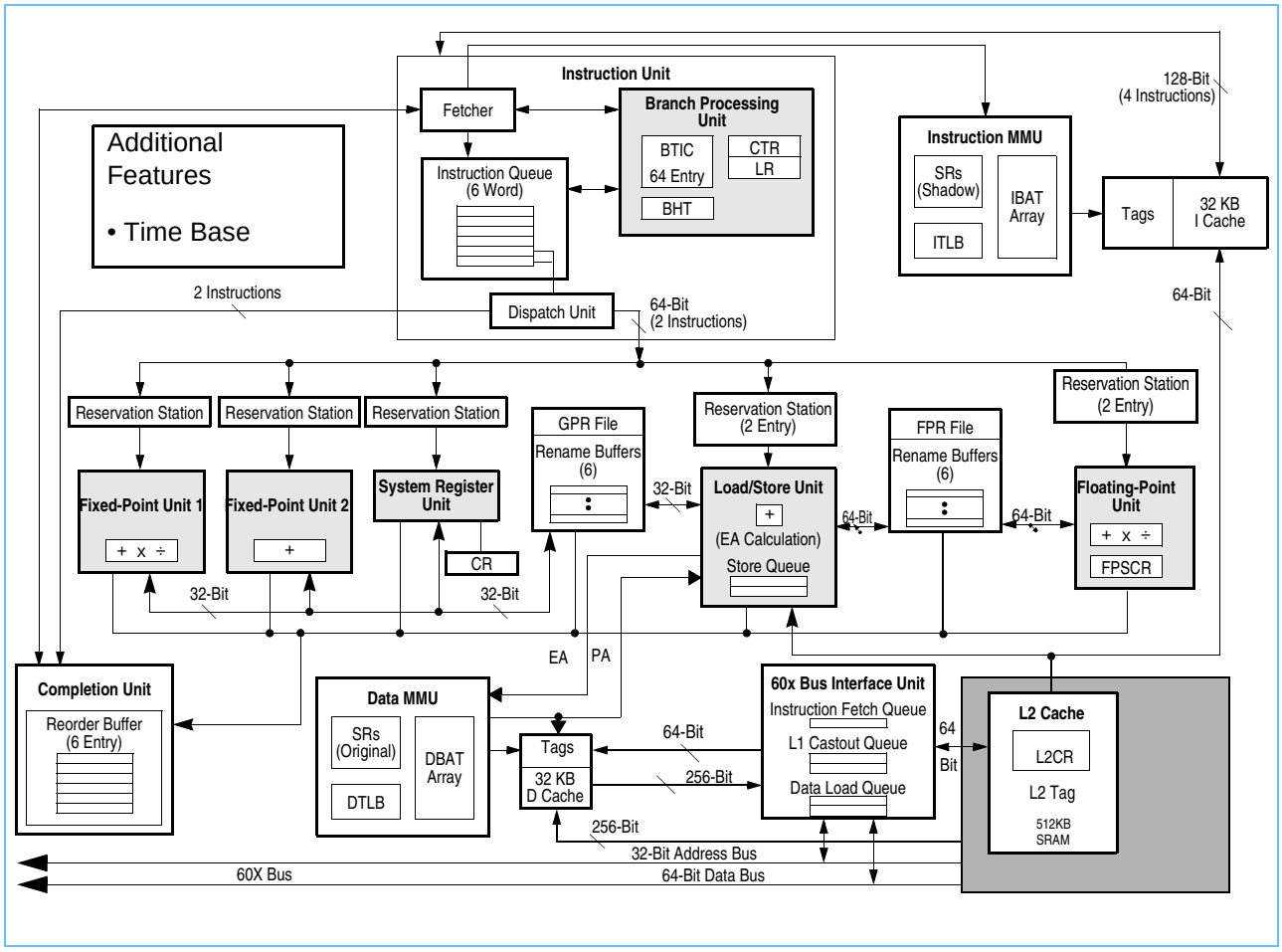
The dispatcher decodes the two instructions from the instruction buffer and decides whether they can be dispatched. The dispatcher also reads the operands from the register files. If any operand is not yet available for any reason, the dispatcher supplies a rename tag for that operand. The dispatched instructions are placed on the instruction dispatch busses along with their execution unit assignment and rename register assignment for destination results.

The execution units store the operands/tags that have been assigned to them from the dispatch busses. When all operands are available, the instruction is executed.

The execution unit writes the result of a finished instruction on the proper rename bus and into the rename register.

The completion logic retires an instruction when all instructions ahead of it have been retired and the instruction has finished execution. Retirement will write the instruction's result into the proper architectural register and remove it from the rename buffers. It also updates any other machine state that is affected by this instruction. Up to two instructions can be simultaneously retired.

Figure 4-1. 750FX Block Diagram



4.2 Control Unit

This section describes the control unit of the 750FX. The control unit contains the instruction fetch unit, branch unit, instruction dispatch unit, completion unit, interrupt handling logic, and the system execution unit.

4.3 Instruction Fetch

The instruction fetch unit controls the flow of instructions from the cache to the instruction buffers and dispatch buffer.

4.4 Branch Processing

The branch unit executes branch instructions, predicts conditional branches and provides addresses for instruction fetches.

4.5 Instruction Dispatch

The dispatch unit includes the dispatch buffer which is two instructions wide and the issue logic. The major functions of the issue logic are:

- Opcode decoding to determine the instruction class and resource requirements for each instruction in the dispatch buffer.
- Rename buffer assignment for fixed and float instruction destinations.
- Source and destination register dependency checking.
- Execution unit assignment.
- Determine post-dispatch serialization and inhibit subsequent instruction dispatching.

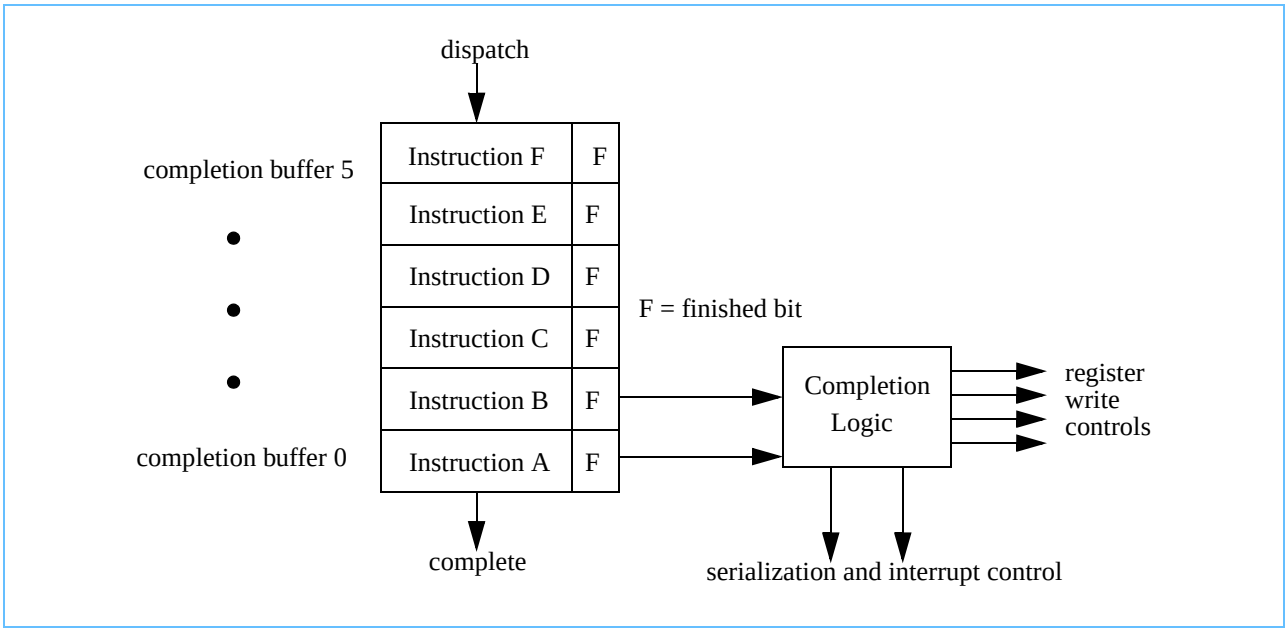
The dispatch system operates in a single processor clock cycle.

4.6 Instruction Completion

Completion provides a mechanism to track instructions from dispatch through execution, then retire or “complete” them in program order. Completing an instruction implies committing to the machine any architectural register changes caused by that instruction. In-order completion ensures the correct architectural state when the machine must recover from a mispredicted branch or any interrupt. Due to the possibility of out-of-order instruction execution, some state of each instruction must be maintained to allow in-order completion.

Instruction state and all information required for completion is kept in a FIFO queue of six completion buffers. A single completion buffer is allocated for each instruction once it leaves dispatch. A completion buffer is a required resource for dispatch, i.e., if there are no completion buffers available, dispatch will stall. Two instructions per cycle are completed in order from the queue, while interrupts (including mispredicted branches) are monitored.

Figure 4-2. Completion Buffer Queue Block Diagram



Writes to the General Purpose or Floating Point Registers are tracked in their respective rename buffers but are initiated by completion. Other selected architectural register updates are renamed in the completion buffers. The completion logic also signals the functional units when to finish completion-serialized instructions.

The completion buffer queue and associated logic provide:

- Instruction tracking and peak completion of two instructions per cycle.
- Completion of instructions in program order while supporting out-of-order instruction execution, completion serialization and all instruction flow changes.

4.7 Interrupt Handling

The interrupt handling unit includes logic to handle interrupts and traps.

4.8 System Execution Unit

The System unit executes several miscellaneous instructions, including the Condition Register Logical Operations and Move to/from Special Purpose Register instructions. In order to maintain system state, numerous instructions executed by the system unit are serialized. These instructions access/modify system registers. In addition, due to their infrequent occurrence, the System unit may only hold one instruction for execution.

4.9 Data Execution Units

750FX data execution units consist of two fixed point units, the load/store unit, and the floating point unit. Included in the execution units section are the 32 by 32-bit General Purpose Registers (GPRs) and the 32- by 64-bit Floating Point Registers (FPRs). With all execution units busy a maximum of three instructions may speculatively finish in a single cycle but actual completion will support only two per cycle.

The execution units will accept instructions from the dispatcher when not busy. Instructions with data dependencies begin execution when all such dependencies are resolved.

4.10 Load / Store Unit

The load/store unit provides the data transfer interface between the cache memory and the internal GPRs and the FPRs. This unit consists of all logic necessary to calculate effective addresses, handle data alignment to and from the cache memory, and provide the sequences for load and store strings and multiples. *Figure 4-1 on page 19* shows the block diagram for the load/store unit.

4.11 Caches

4.11.1 750FX Level 1 Cache Description

- 32KB Instruction Cache (8-way set associative)
- 32KB Data Cache (8-way set associative)
- Physically addressed cache and tag array
- 32 byte line size
- Three-state coherency protocol
- Pseudo Least Recently Used (PLRU) replacement algorithm

4.11.1.1 Coherency Support

The 750FX supports a three state coherency protocol which supports the Invalid, Exclusive Unmodified and the Exclusive Modified cache states. This protocol is a compatible subset of the MESI four state protocol and will operate coherently in systems which contain four state caches. The 750FX initiates a broadcast of dcbz (to M=1, coherency required page) and snoops a dcbz broadcast on the 60x bus. All other data cache control instructions are not broadcast unless broadcast is enabled through the HID0[ABE] configuration bit. Instruction cache and TLB control instructions are not broadcast. Except for dcbz all other cache operations are not snooped. The cache operations are intended primarily for the management of the internal cache but not for other caches in the system. However, the optional data cache-op broadcast feature is necessary to allow proper management of a copyback L2.

4.11.1.2 Cache Loads

The 750FX cache lines are loaded in four beats from the 60x bus (64 bits each) or 1 beat from the L2 (256 bits). The burst load is performed critical double word first. The cache which is being loaded is not blocked to internal accesses while the load completes (hits under misses). The critical double word is simultaneously written to the cache reload buffer and forwarded to the requesting unit, thus minimizing stalls due to load delays. The 750FX dcache miss-under-miss feature can be enabled or disabled by HID0. With this enabled one miss can be serviced while a current miss is in progress. Misses can be either load or store type.

4.11.1.3 Cache Replacements

Cache lines are selected for replacement based on an PLRU (Pseudo Least Recently Used) algorithm. Each time a cache line is accessed, it is tagged as the most recently used line of the set. When a miss occurs, if all lines in the set are valid (occupied), then the least recently used line is replaced with the new data. The LRU bits in the cache are updated each time a cache hit occurs.

When the data to be replaced is in the XM (exclusive modified) state, that data is written into a writeback buffer while the missed data is being accessed on the bus. When the load completes, the 750FX then pushes the replaced line from the writeback buffer to the L2 if L2 is enabled or to main memory if L2 is disabled.

4.11.2 750FX Level 2 Cache Features.

The 750FX includes an internal level 2 (L2) cache with the following features:

- 512KB 2-way set associative
- Built-in error correction (ECC)
- Single cycle transfer to dcache (32 bytes)
- Handles miss-under-miss
  - One from the icache and one from the dcache
  - Two (2) from the dcache
- Dual reload buffers
- Two (2) data load request queues to BIU
- Cache locking by way with locked side managed by software and not snooped
- Parity added for L2 tags[1]

4.12 Bus Interface Unit

The BIU queues up bus requests from the instruction, data, and L2 caches, and runs the requests to the 60x bus. It includes address register queues, prioritization logic, and bus control units. The BIU also captures snoop addresses for snooping in the caches and in the address register queues and snoops for reservations. All data storage for the address register queues (load and store data queues) are located in the cache section. The data queues are considered temporary storage for the cache and not part of the BIU. For the 750FX a second data load queue was added. This bus pipelining feature is needed for support of the cache miss-under-miss capability.

1. Not supported under DD 1.X.

5. PowerPC 750FX Microprocessor Implementation

The PowerPC Architecture is derived from the 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 scalable to take advantage of future technological gains.

This section describes the PowerPC Architecture in general, and provides details about the implementation of PowerPC 750FX as a low-power, 32-bit member of the PowerPC processor family.

For a description of...	See...
Registers and programming model	*Section 6* on page 25 describes the registers for the operating environment architecture common among PowerPC processors and describes the programming model. It also describes the registers that are unique to PowerPC 750FX.
Instruction set and addressing modes	*Section 7* on page 27 describes the PowerPC instruction set and addressing modes for the PowerPC operating environment architecture, defines the PowerPC instructions implemented in PowerPC 750FX, and describes new instruction set extensions to improve the performance of single-precision floating-point operations and the capability of data transfer.
Cache implementation	*Section 8* on page 29 describes the cache model that is defined generally for PowerPC processors by the virtual environment architecture. It also provides specific details about PowerPC 750FX L2 cache implementation.
Exception model	*Table 9* on page 31 describes the exception model of the PowerPC operating environment architecture and the differences in PowerPC 750FX exception model.
Memory management	*Table 10* on page 34 describes generally the conventions for memory management among the PowerPC processors. This section also describes PowerPC 750FX's implementation of the 32-bit PowerPC memory management specification.
Instruction timing	*Section 11* on page 36 provides a general description of the instruction timing provided by the superscalar, parallel execution supported by the PowerPC Architecture and PowerPC 750FX.
Power management	*Section 12* on page 39 describes how the power management can be used to reduce power consumption when the processor, or portions of it, are idle.
Thermal management	*Section 13* on page 41 describes how the thermal management unit and its associated registers (THRM1–THRM3) and exception can be used to manage system activity in a way that prevents exceeding system and junction temperature thresholds. This is particularly useful in high-performance portable systems, which cannot use the same cooling mechanisms (such as fans) that control overheating in desktop systems.
Performance monitor	*Section 14* on page 42 describes the performance monitor facility, which system designers can use to help bring up, debug, and optimize software performance.

6. PowerPC Registers and Programming Model

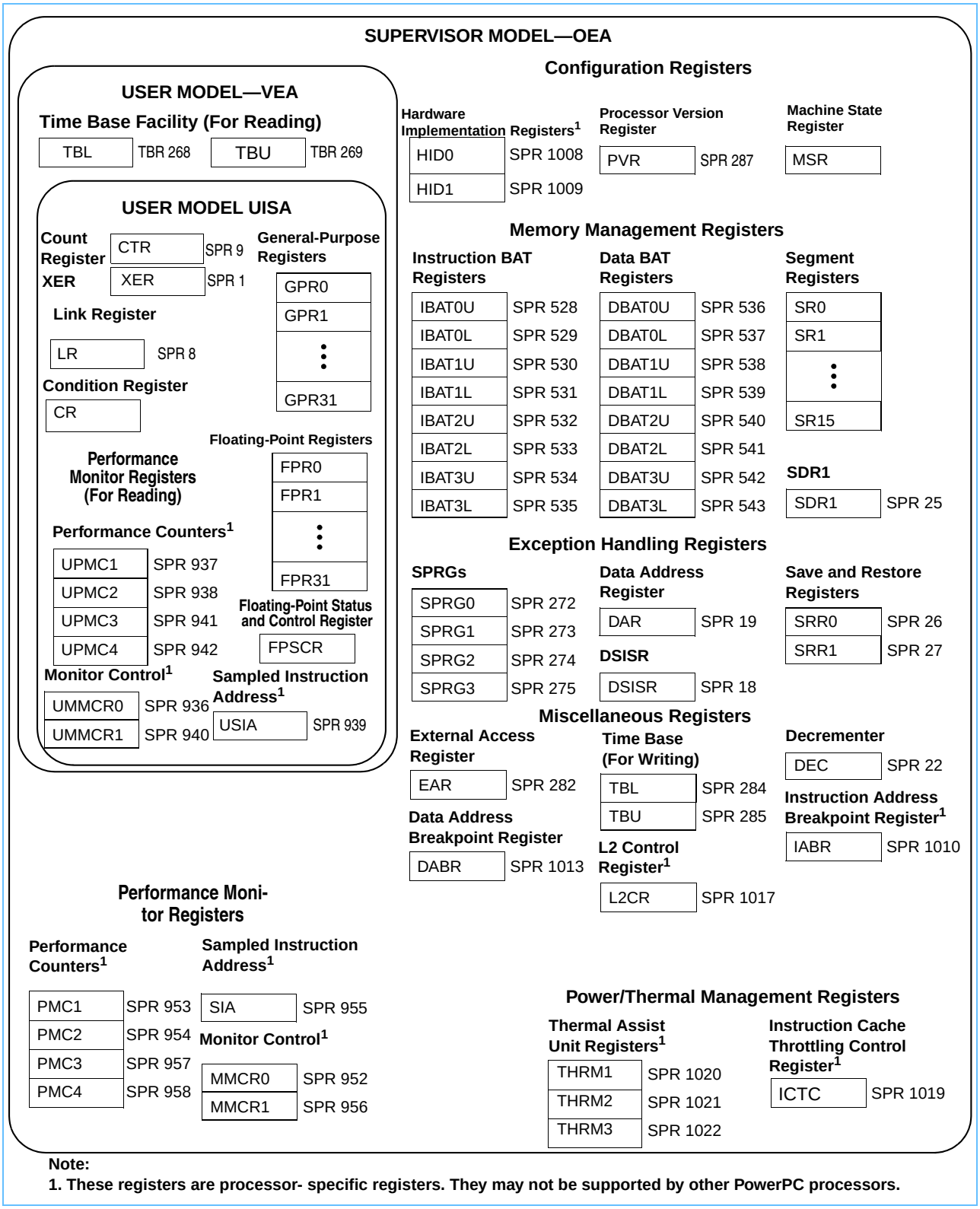
The PowerPC Architecture defines register-to-register operations for most computational instructions. Source operands for these instructions are accessed from the registers or are provided as immediate values embedded in the instruction itself. The three-register instruction formats allow specification of a target register distinct from the two source operands. Only 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, it is also called problem state). 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-dependent (HID) registers.

While running in supervisor mode the operating system is able to execute all instructions and access all registers defined in the PowerPC Architecture. In this mode the operating system establishes all address translations and protection mechanisms, loads all processor state registers. and sets up all other control mechanisms defined on the PowerPC 750CX processor. While running in user mode (problem state) many of these registers and facilities are not accessible and any attempt to read or write these register results in a program exception.

Figure 6-1 on page 26 shows all the PowerPC 750FX 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.

Figure 6-1. PowerPC 750FX Microprocessor Programming Model — Registers



Note:
1. These registers are processor- specific registers. They may not be supported by other PowerPC processors.

7. Instruction Set

All PowerPC instructions are encoded as single-word (32-bit) instructions. Instruction formats are consistent among all instruction types (primary op-code is always 6 bits, register operands always specified in the same bit fields in the instruction), permitting efficient decoding to occur in parallel with operand accesses. This fixed instruction length and consistent format greatly simplify instruction pipelining.

7.1 PowerPC Instruction Set

The PowerPC instructions are divided into the following categories.

1. Integer instructions
   These include computational and logical instructions.
   - Integer arithmetic instructions
   - Integer compare instructions
   - Integer logical instructions
   - Integer rotate and shift instructions
2. 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
3. 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)
4. 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 (sets conditions for branches)
   - System Call
5. 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 (processor and memory system)
   - Instruction synchronize
   - Order loads and stores
6. Memory control instructions
   To provide control of caches, TLBs, and SRs.
   - Supervisor-level cache management instructions
   - User-level cache instructions

- Segment register manipulation instructions
- Translation lookaside buffer management instructions

This grouping 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 (two words) floating-point operands. The PowerPC Architecture uses instructions that are four bytes long and word-aligned. It provides for integer byte, half-word, and word operand loads and stores between memory and a set of 32 GPRs. It also provides for single and double-precision loads and stores between memory and a set of 32 floating-point registers (FPRs).

Computational instructions do not access 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 using three or more instructions.

PowerPC 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 type of exception will cause the associated exception handler to be invoked.

Effective address computations for both data and instruction accesses use 32-bit signed two’s complement binary arithmetic. A carry from bit 0 and overflow are ignored.

8. On-Chip Cache Implementation

This section describes the PowerPC Architecture treatment of cache in general, and the PowerPC 750FX-specific implementation.

8.1 PowerPC Cache Model

The PowerPC Architecture does not define hardware aspects of cache implementations. For example, PowerPC processors can have unified caches, separate instruction and data caches (Harvard architecture), or no cache at all. PowerPC microprocessors control the following memory access modes on a virtual page or block (BAT) basis

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

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 PowerPC Architecture defines the term ‘cache block’ as the cacheable unit. The VEA and OEA define cache management instructions that a programmer can use to affect cache contents.

8.2 PowerPC 750FX Microprocessor Cache Implementation

8.2.1 Description of 750FX L1 Caches

- 32K byte, 8-way set associative, L1 instruction cache (pseudo LRU replacement)
- 32K byte, 8-way set associative, L1 data cache (pseudo LRU replacement)
- 32 byte line size with each line having one valid bit and three possible states:
  - Exclusive
  - Modified
  - Invalid
- Physically addressed cache and cache directory. Physical address tag stored in the cache directory.

8.2.2 Description of 750FX L2 Cache

The 750FX 512KB L2 cache is implemented with an internal two-way set-associative tag memory with 4096 tags per way, and an internal 512KB SRAM for data storage. The tags are sectored to support two cache blocks per tag entry (two 32-byte sectors totalling 64 bytes). Each sector (32-byte L1 cache block) in the L2 cache has its own valid and modified bits. In addition, the SRAM includes an 8-bit ECC for every double-word. The ECC logic corrects most single-bit errors and detects double-bit errors as data is read from the SRAM. The L2 cache maintains cache coherency through snooping and is normally configured to operate in copy-back mode.

The L2 cache control register (L2CR) allows control of the following:

- L2 cache configuration
- Double bit error machine check

- Global invalidation of L2 contents
- Write-through operation
- L2 test support
- L2 locking by way[1]

8.2.2.1 L2 Cache Operation

The L2 cache for the 750FX is a combined instruction and data cache that receives memory requests from both L1 instruction and L1 data caches independently. The L1 requests are generally the result of instruction fetch misses, data load or store misses, write-through operations, or cache management instructions. Each L1 request generates an address lookup in the L2 tags. If a hit occurs, the instructions or data are forwarded to the L1 cache. A miss in the L2 tags causes the L1 request to be forwarded to the 60x bus interface. The cache block received from the bus is forwarded to the L1 cache immediately, and is also loaded into the L2 cache with the tag marked valid and unmodified. If the cache block loaded into the L2 causes a new tag entry to be allocated and the current tag entry is marked valid modified, the modified sectors of the tag to be replaced are castout from the L2 cache to the 60x bus.

At any given time the L1 instruction cache may have one instruction fetch request, and the L1 data cache may have two loads and two stores requesting L2 cache access. The L2 cache also services snoop requests from the 60x bus. When there are multiple pending requests to the L2 cache, snoop requests have highest priority, followed by data load and store requests (serviced on a first-in, first-out basis). Instruction fetch requests have the lowest priority in accessing the L2 cache when there are multiple accesses pending.

If read requests from both the L1 instruction and L1 data caches are pending, the L2 cache can perform hit-under-miss and supplies the available instruction or data while a bus transaction for the previous L2 cache miss is being performed. The L2 cache supports miss under miss. Up to two outstanding misses are supported: one miss from the instruction cache and one from the data cache, or two data cache misses. Requests that hit will be serviced even while misses are in progress on the bus.

All requests to the L2 cache that are marked cacheable (even if the respective L1 cache is disabled or locked) cause tag lookup and will be serviced if the instructions or data are in the L2 cache. Burst and single-beat read requests from the L1 caches that hit in the L2 cache are forwarded to the L1 caches as instructions or data, and the L2 LRU bit for that tag is updated. Burst writes from the L1 data cache due to a castout or replacement copyback are written only to the L2 cache, and the L2 cache sector is marked modified.

If the L2 cache is configured as write-through, the L2 sector is marked unmodified, and the write is forwarded to the 60x bus. If the L1 castout requires a new L2 tag entry to be allocated and the current tag is marked modified, any modified sectors of the tag to be replaced are cast out of the L2 cache to the 60x bus.

Single-beat read requests from the L1 caches that miss in the L2 cache do not cause any state changes in the L2 cache and are forwarded on the 60x bus interface. Cacheable single-beat store requests marked copy-back that hit in the L2 are allowed to update the L2 cache sector, but do not cause L2 cache sector allocation or deallocation. Cacheable, single-beat store requests that miss in the L2 are forwarded to the 60x bus. Single-beat store requests marked write-through (through address translation or through the configuration of L2CR[L2WT]) are written to the L2 cache if they hit and are written to the 60x bus independent of the L2 hit/miss status. If the store hits in the L2 cache, the modified/unmodified status of the tag remains unchanged. All requests to the L2 cache that are marked cache-inhibited by address translation (through either the MMU or by default WIMG configuration) bypass the L2 cache and do not cause any L2 cache tag state change.

1. Not supported in DD1.X

9. Exception Model

The section describes the PowerPC exception model and the PowerPC 750FX implementation.

9.1 PowerPC Exception Model

The PowerPC exception mechanism allows the processor to interrupt the instruction flow to handle certain situations caused by external signals, errors, or unusual conditions arising from the instruction execution. 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. System software must complete the saving of the processor state prior to servicing the exception. Exception processing proceeds 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 MSR, DSISR and the FPSCR contain status bits which farther identify the exception condition. Additionally, some exception conditions can be explicitly enabled or disabled by software.

The PowerPC Architecture requires that exceptions be handled in specific priority and program order; therefore, although a particular implementation may recognize exception conditions out of order, they are handled in program order. When an instruction-caused exception is recognized, any unexecuted instructions that appear earlier in the instruction stream, including any that are undispatched, are required to complete before the exception is taken, and any exceptions those instructions cause must also be handled first; likewise, asynchronous, precise exceptions are recognized when they occur but are not handled until the instructions currently in the completion queue successfully retire or generate an exception, and the completion queue is emptied.

Unless a catastrophic condition causes a system reset or machine check exception, only one exception is handled at a time. For example, if one instruction encounters multiple exception conditions, those conditions are handled sequentially in priority order. After the exception handler completes, the instruction processing continues until the next exception condition is encountered. Recognizing and handling exception conditions sequentially guarantees system integrity.

When an exception is taken, information about the processor state before the exception was taken is saved in SRR0 and SRR1. Exception handlers must save the information stored in SRR0 and SRR1 early to prevent the program state from being lost due to a system reset and machine check exception or due to an instruction-caused exception in the exception handler, and before re-enabling external interrupts. The exception handler must also save and restore any GPR registers used by the handler.

The PowerPC Architecture supports four types of exceptions.

1. 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.

2. Synchronous, imprecise
   The PowerPC Architecture defines two imprecise floating-point exception modes, recoverable and nonrecoverable. Even though PowerPC 750FX provides a means to enable the imprecise modes, it implements these modes identically to the precise mode (that is, enabled floating-point exceptions are always precise).
3. Asynchronous, maskable
   The PowerPC Architecture defines external and decrementer interrupts as 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 no instructions are in the execution units, the exception is taken immediately upon determination of the correct restart address (for loading SRR0). As shown in the *Table 9-1* on page 32 the PowerPC 750FX implements additional asynchronous, maskable exceptions.
4. 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. Exceptions report recoverability through the MSR[RI] bit.

9.2 PowerPC 750FX Microprocessor Exception Implementation

The PowerPC 750FX exception classes described above are shown in the *Table 9-1 PowerPC 750FX Microprocessor Exception Classifications* Although exceptions have other characteristics, such as priority and recoverability, *Table 9-1* describes categories of exceptions PowerPC 750FX handles uniquely. *Table 9-1* includes no synchronous imprecise exceptions; although the PowerPC Architecture supports imprecise handling of floating-point exceptions, PowerPC 750FX implements these exception modes precisely.

Table 9-1. PowerPC 750FX Microprocessor Exception Classifications

Synchronous/Asynchronous	Precise/Imprecise	Exception Type
Asynchronous, nonmaskable	Imprecise	Machine check, system reset
Asynchronous, maskable	Precise	External, decrementer, system management, performance monitor, and thermal management interrupts
Synchronous	Precise	Instruction-caused exceptions

Table 9-2 Exceptions and Conditions lists the PowerPC 750FX exceptions and conditions that cause them. Exceptions specific to PowerPC 750FX are indicated.

Table 9-2. Exceptions and Conditions

Exception Type	Vector Offset (hex)	Causing Conditions
Reserved	00000	—
System reset	00100	Assertion of either $\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$ or at power-on reset
Machine check	00200	Assertion of $\overline{\text{TEA}}$ during a data bus transaction, assertion of $\overline{\text{MCP}}$, an address, data or L2 double bit error. MSR[ME] must be set.
DSI	00300	As specified in the PowerPC Architecture. (e.g., page fault occurs)
ISI	00400	As defined by the PowerPC Architecture. (e.g., page fault occurs)

1. PowerPC 750FX-specific

Table 9-2. Exceptions and Conditions (Continued)

Exception Type	Vector Offset (hex)	Causing Conditions
External interrupt	00500	MSR[EE] = 1 and $\overline{\text{INT}}$ is asserted.
Alignment	00600	A floating-point load/store, **stmw, stwcx, lmw**, **lwarx**, **eciwx** or **ecowx** instruction operand is not word-aligned. A multiple/string load/store operation is attempted in little-endian mode. The operand of **dcbz** is in memory that is write-through-required or caching-inhibited or the cache is disabled
Program	00700	As defined by the PowerPC Architecture.
Floating-point unavailable	00800	As defined by the PowerPC Architecture.
Decrementer	00900	As defined by the PowerPC Architecture, when the most significant bit of the DEC register changes from 0 to 1 and MSR[EE] = 1.
Reserved	00A00–00BFF	—
System call	00C00	Execution of the System Call (**sc**) instruction.
Trace	00D00	MSR[SE] = 1 or a branch instruction completes and MSR[BE] = 1. Unlike the architecture definition, **isync** does not cause a trace exception
Reserved	00E00	PowerPC 750FX does not generate an exception to this vector. Other PowerPC processors may use this vector for floating-point assist exceptions.
Reserved	00E10–00EFF	—
Performance monitor[1]	00F00	The limit specified in a PMC register is reached and MMCR0[ENINT] = 1
Instruction address breakpoint[1]	01300	IABR[0–29] matches EA[0–29] of the next instruction to complete, IABR[TE] matches MSR[IR], and IABR[BE] = 1.
Reserved	01400–016FF	—
Thermal management interrupt[1]	01700	Thermal management is enabled, the junction temperature exceeds the threshold specified in THRM1 or THRM2, and MSR[EE] = 1.
Reserved	01800–02FFF	—

1. PowerPC 750FX-specific

10. Memory Management

This section describes the memory management features of the PowerPC Architecture, and the PowerPC 750FX implementation.

10.1 PowerPC Memory Management Model

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 PowerPC 750FX that require address translation—instruction fetches, and data accesses to memory generated by load, store, and cache control instructions.

The PowerPC Architecture defines different resources for 32 and 64-bit processors; PowerPC 750FX implements the 32-bit memory management model. The memory-management unit provides two types of memory access models: Block Address Translate (BAT) model and a virtual address model. The BAT block sizes range from 128Kbyte to 256Mbyte and are selectable from high order effective address bits and have priority over the virtual model. The virtual model employs a 52-bit virtual address space made up by a 24-bit segment address space and a 28-bit effective address space. The virtual model utilizes a demand paging method with a 4Kbyte page size. In both models address translation is done completely by hardware, in parallel with cache accesses, with no additional cycles incurred.

The PowerPC 750FX MMU 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 128Kbytes to 256Mbytes. The BAT arrays are maintained by system software. Instructions and data share the same virtual address model but could operate in separate segment spaces.

The PowerPC 750CX 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 for data and instructions are loaded into physical memory from system disk only when they are required by an executing program. Infrequently used pages in memory are returned to disk or discarded if they have not been modified.

The hashed page table is a fixed-sized data structure (size should be determined by the amount of physical memory available to the system) that contains 8-byte entries (PTEs) that define the mapping between virtual pages and physical pages. The page table size is a power of 2 and is boundary aligned in memory based on the size of the table. The page table contains a number of page table entry groups (PTEGs). A PTEG contains eight page table entries (PTEs) of eight bytes each; therefore, each PTEG is 64 bytes long. PTEG addresses are entry points for table search operations. A given page translation can be found in one of two possible PTEG’s. The size and location in memory of the page table is defined in the SDR1 register.

Setting MSR[IR] enables instruction address translations and MSR[DR] enables data address translations. If the bit is cleared, the respective effective address is used as the physical address.

10.2 PowerPC 750FX Microprocessor Memory Management Implementation

The 750FX implements a virtual memory management scheme compliant with the PowerPC specification for 32-bit microprocessors. The organization of the Memory Management Unit (MMU) hardware is as follows;

- 64-entry, 2-way set associative data TLB (total of 128)
- 64-entry, 2-way set associative instruction TLB (total of 128)
- Eight (8) entry, fully associative Instruction BAT
- Eight (8) entry, fully associative Data BAT
- Sixteen segment registers
- Hardware tablewalks

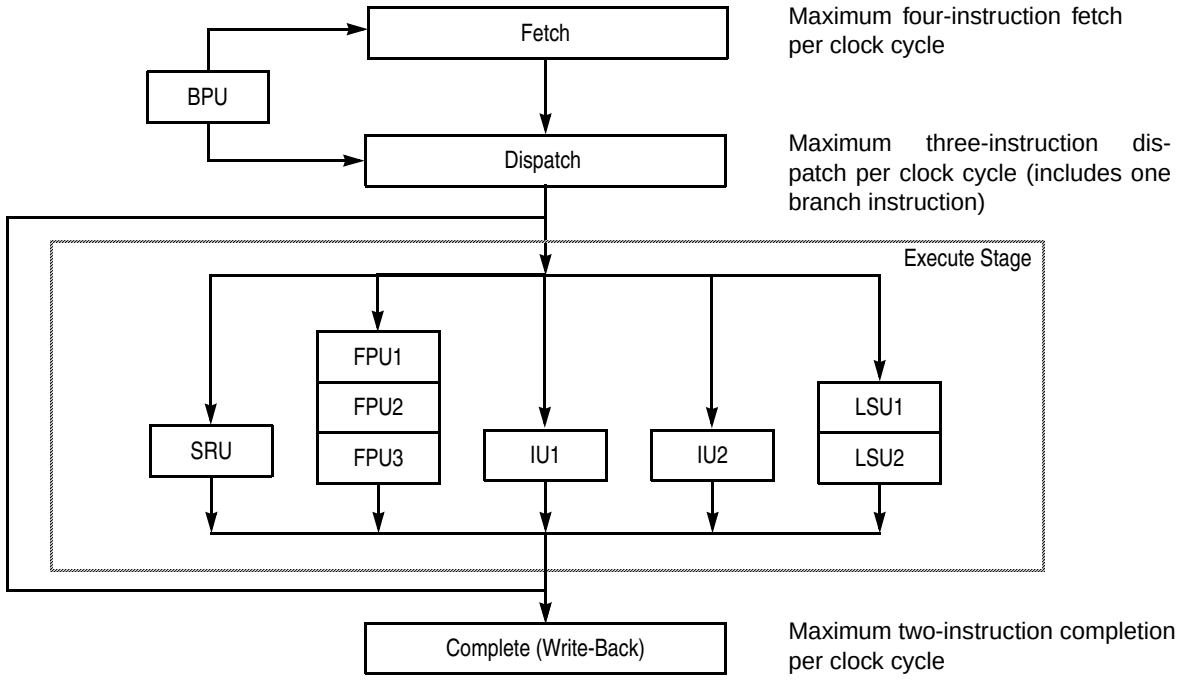
11. Instruction Timing

The PowerPC 750FX is a pipelined, superscalar processor. A pipelined processor is one in which instruction processing is divided into discrete stages, allowing work to be done on multiple instructions in each stage. For example, after an instruction completes one stage, it can pass on to the next stage leaving the previous stage available to a subsequent instruction. This improves overall instruction throughput.

A superscalar processor is one that issues multiple independent instructions to separate execution units in a single cycle, allowing multiple instructions to execute in parallel. The PowerPC 750FX has six independent execution units, two for integer instructions, and one each for floating-point instructions, branch instructions, load and store instructions, and system register instructions. Having separate GPRs and FPRs allows integer, floating-point calculations, and load and store operations to occur simultaneously without interference. Additionally, rename buffers are provided to allow operations to post completed results to be use by subsequent instructions without committing them to the architected FPR and GPR register files.

The common pipeline of the PowerPC 750FX, as shown in *Figure 11-1* on page 36, has four stages through which all instructions must pass: fetch, decode/dispatch, execute, and complete/write back. Instructions flow sequentially through each stage. However, at dispatch a position is made available in the completion queue at the same time it enters the execution stage. This simplifies the completion operation when instructions are retired in program order. Both the load/store and floating-point units have multiple stages to execute their instructions. An instruction occupies only one stage at a time in all execution units. At each stage an instruction may proceed without delay or may stall. Stalls are caused by the requirement of additional processing or other events. For example divide instructions require multiple cycles to complete the operation, load and store instructions may stall waiting for address translation (TLB reload, page fault, and so forth).

Figure 11-1. Pipeline Diagram



Note: This diagram does not show features that reduce stalls and improve instruction throughput (for example reservation stations and rename buffers).

Note: *Figure 11-1* on page 36 does not show features that reduce stalls and improve instruction throughput (for example reservation stations and rename buffers).

The is a listing of the four major instruction pipeline stages in the PowerPC 750FX.

1. Fetch
   The fetch pipeline stage primarily involves fetching instructions from the memory system and keeping the instruction queue full. The BPU decodes branches after they are fetched and removes (folds out) those that do not update CTR or LR from the instruction stream. If the branch is taken or predicted as taken the fetch unit is informed of the new address and fetching resumes along the taken patch. For branches not taken or predicted as not taken sequential fetching continues.
2. Dispatch
   The dispatch unit is responsible for taking instructions from the bottom two locations of the instruction queue and delivering them to an execution unit for farther processing. Dispatch is responsible for decoding the instructions and determining which instructions can be dispatched. To qualify for dispatch, a reservation station, a rename buffer and a position in the completion queue all must be available. A branch instruction could be processed by the BPU on the same clock cycle for a maximum of three-instruction dispatch per cycle.

   The dispatch stage accesses operands, assigns a rename buffer for an operand(s) that updates an architected register(s) (GPR, FPR, CR, etc.) and delivers the instruction to the reservation registers of the respective execution units. If a source operand is not available (a previous instruction is updating the item via a rename buffer) dispatch provides a tag that indicates which rename buffer will supply the operand when it becomes available. At the end of the dispatch stage, the instructions are removed from the instructions queue, latched into reservation stations at the appropriate execution unit and assigned positions in the completion buffers in sequential program order.
3. Execution
   The execution units process instructions from their reservations stations using the operands provided from dispatch and notifies the completion stage when the instruction has finished execution. With the exception of multiply and divide integer instructions complete execution in a single cycle.

   FPU has three stages for processing floating-point arithmetic. The FPU stages are multiply, add, and normalize. All single precision arithmetic (add, subtract, multiply and multiply/add) instructions are processed without stalls at each stage. They have a one cycle through put and a three cycle latency. Three different arithmetic instructions can be in execution at one time with one instruction completing execution each cycle. Double-precision arithmetic multiply requires two cycles in the multiply stage and one cycle in add, and one in normalize yielding a two cycle through put and a 4 cycle latency. All divide instructions require multiple cycles in the first stage for processing.

   The load/store unit has two reservation registers and two pipeline stages. The first stage is for effective address calculation and the second stage is for MMU translation and accessing the L1 data cache. Load instructions have a one cycle through put and a two cycle latency.

   In the case of an internal exception, the execution unit reports the exception to the completion pipeline stage and (except for the FPU) discontinues instruction execution until the exception is handled. The exception is not signaled until it is determined that all previous instruction have completed to a point where they will not signal an exception.
4. Completion
   The completion unit retires instruction from the bottom two positions of the completion queue in program order. This maintains the correct architectural machine state and transfers execution results from the rename buffers to the GPRs and FPRs (and CTR and LR, for some instructions) as instructions are

retired. If completion logic detects an instruction causing an exception, all following instructions are cancelled, their execution results in rename buffers are discarded, and instructions are fetched from the appropriate exception vector.

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

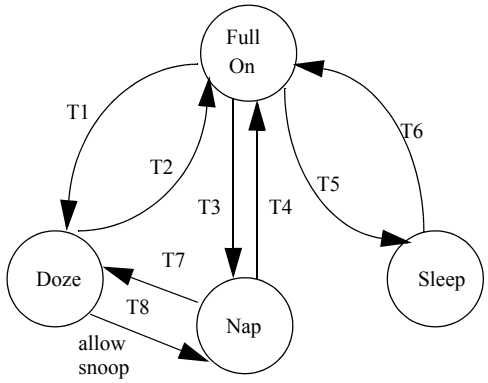
For a detailed discussion of instruction timing with examples and a table of latencies for each execution unit, see *Section 11 Instruction Timing* on page 36.

12. Power Management

The four power states for the 750FX are shown in *Figure 12-1 750FX Power States*. The three power saving modes provide different levels of power savings. Sleep mode provides the greatest power savings, followed by Nap mode and then Doze mode.

Use software to enter the 750FX power saving modes by setting the appropriate control bits in the HID0 register and the Machine State Register (MSR) power management bit (MSR[POW]) as shown in *Figure 12-1 750FX Power States*.

Figure 12-1. 750FX Power States



T1: HID0(Doze) = 1 and MSR[POW] transitions from MSR[POW] = 0 to MSR[POW] = 1
T2: HRESET_, SRESET_, INT_, SMI_, MCP_, DEC, PFM, machine check interrupts
T3: HID0(Nap) = 1 and MSR[POW] transitions from MSR[POW] = 0 to MSR[POW] = 1
T4:HRESET_, SRESET_, INT_, SMI_, MCP_, DEC
T5: HID0(Sleep) = 1 and MSR[POW] transitions MSR[POW] = 0 to MSR[POW] = 1
T6: HRESET_, SRESET_, INT_, SMI_, MCP_.
T7: QACK_ transitions from QACK_ = 0 to QACK_ = 1
T8: QACK_ transitions from QACK_ = 1 to QACK_ = 0

12.1 Full On

This is the default power state of the processor. The 750FX is fully powered and the internal functional units are operating at full processor clock speed.

12.2 Doze

In this power saving mode, all the 750FX functional units are disabled except for the Time Base/Decrementer, data cache, L2 cache, and the bus snooping logic. In Doze mode, the bus snooping logic, the data cache. and the L2 cache are fully functional to maintain memory coherency. Once the Doze mode is entered, an external asynchronous interrupt, a system management interrupt, a decrementer interrupt, a reset (hard or soft), a performance monitor interrupt, thermal management interrupt or machine checks will bring 750FX out of the Doze mode and into the Full-On state. In Doze mode, the PLL is required to be running and locked to SYSCLK. The transition to the Full-On state takes no more than a few processor cycles.

12.3 Nap

Further power savings can be achieved through the Nap mode in which only the Time Base and Thermal unit is operating. The processor will not enter Nap mode until QACK_ has been asserted. To maintain cache coherency, the system must de-assert QACK_ at least 8 bus cycles before issuing a snoop to the processor. Reasserting QACK_ after the snoop will put the processor back into Nap mode. 750FX will wake up to the Full-On state by an external asynchronous interrupt, a system management interrupt, a decrementer interrupt, a reset (hard or soft) or a machine check input (MCP_). As in the Doze mode, the PLL is required to be running and locked to SYSCLK. The transition to the Full-On state takes no more than a few processor cycles.

12.4 Sleep

Sleep mode provides further power savings compared to the Nap mode. In sleep mode no functional units are operating and the PLL and SYSCLK input may be disabled by the system logic. The PLL can be disabled by configuring the PLL_CFG(0:4) pins into the PLL bypass mode. Note that, simply forcing SYSCLK to be static will not disable the PLL which will still be running at an undetermined frequency. When recovering from sleep mode with the PLL disabled, the system logic has to re-enable the PLL and SYSCLK first, and then assert an external asynchronous interrupt, a system management interrupt, a reset (hard or soft) or a machine check input (MCP_) after 100usec of PLL re-lock time. Select HRESET if PLL loses lock.

Section 12 Power Management on page 39 and *Section 13 Thermal Management* on page 41 provide information about power saving and thermal management modes for the PowerPC 750FX.

13. Thermal Management

The PowerPC 750FX's thermal assist unit (TAU) provides a way to control heat dissipation. This ability is particularly useful in portable computers, which, due to power consumption and size limitations, cannot use desktop cooling solutions such as fans. Therefore, better heat sink designs coupled with intelligent thermal management is of critical importance for high performance portable systems.

Implementation Note: Before finalizing a design, contact your IBM sales and technical support for TAU technical specifications.

Primarily, the thermal management system monitors and regulates the system's operating temperature. For example, if the temperature is about to exceed a set limit, the system can be made to slow down or even suspend operations temporarily in order to lower the temperature.

The thermal management facility also ensures that the processor's junction temperature does not exceed the operating specification. To avoid the inaccuracies that arise from measuring junction temperature with an external thermal sensor, PowerPC 750FX on-chip thermal sensor and logic tightly couples the thermal management implementation.

The TAU consists of a thermal sensor, digital-to-analog convertor, comparator, control logic, and the dedicated SPRs described in *Section 6 PowerPC Registers and Programming Model* on page 25. The TAU does the following.

- Compares the junction temperature against user-programmable thresholds.
- Generates a thermal management interrupt if the temperature crosses the threshold.
- Enables the user to estimate the junction temperature by way of a software successive approximation routine.

The TAU is controlled through the privileged **mtspr**/**mfspr** instructions to the three SPRs provided for configuring and controlling the sensor control logic, which function as follows.

- THRM1 and THRM2 provide the ability to compare the junction temperature against two user-provided thresholds. Having dual thresholds gives the thermal management software finer control of the junction temperature. In single threshold mode, the thermal sensor output is compared to only one threshold in either THRM1 or THRM2.
- THRM3 is used to enable the TAU and to control the comparator output sample time. The thermal management logic manages the thermal management interrupt generation and time multiplexed comparisons in the dual threshold mode as well as other control functions.

Instruction cache throttling provides the PowerPC 750FX with overall junction temperature control by determining the interval at which instructions are fetched. This feature is accessed through the ICTC register.

14. Performance Monitor

Facilities are incorporated into the 750FX to monitor its selected behavioral characteristics. These facilities are collectively called the performance monitor.

Some motivations for incorporating a performance monitor into the 750FX:

- To increase system performance with efficient software, memory hierarchy behavior must be monitored and studied in order to develop algorithms that schedule tasks (and perhaps partition them) and that structure and distribute data optimally.
- To improve processor architecture, the detailed behavior of the 750FX structure must be known and understood in many software environments. Some environments may not easily be characterized by a benchmark or trace.
- The performance monitor will help system developers to bring up and debug their systems.

The Performance Monitor on the 750FX processor incorporates:

- Event Counting
- Interrupt Generation

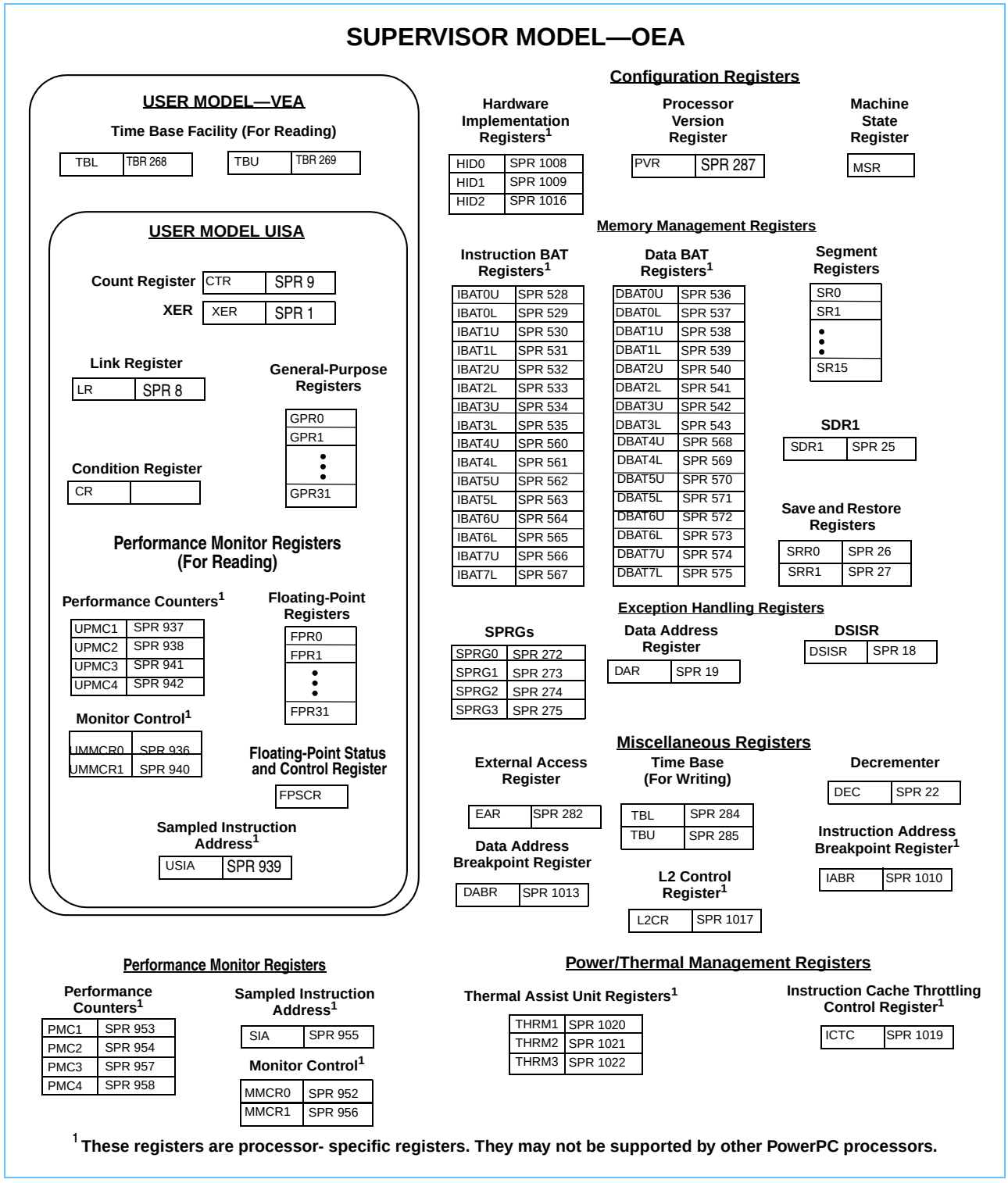
15. PowerPC Architecture™ Compliance

The 750FX implements the PowerPC Architecture™ as defined in the *PowerPC Programming Environments*.

These documents provide a strong framework which defines the PowerPC Architecture. However, they also recognize that different implementations may require clarifications, extensions or deviations relative to the architectural descriptions provided. This section provides additional information about the PowerPC Architecture as it relates specifically to the 750FX.

15.1 Implementation-Specific MSR-bits for 750FX

Figure 15-1. PowerPC 750FX Microprocessor Programming Model—Registers



[1] **These registers are processor- specific registers. They may not be supported by other PowerPC processors.**

15.1.1 Machine State Register

Table 15-1 on page 45 list the 750FX-specific bits that are implemented in the Machine State Register (MSR).

Table 15-1. Machine State Register 750FX-Specific Bits

Bit	Name	Function
13	POW	Activates power management
29	PM	Marks a process for the performance monitor.
Note: The 750FX will clear the POW bit when it leaves a power saving mode.		

The **mtmsr** must be followed by a context-synchronizing instruction.

The MSR[PM] bit is used by the Performance Monitor to help determine when it should count events.

15.1.2 Implementation-Specific SRR1 bits for 750FX

Table 15-2. SRR1 750FX-Specific Bits for Machine Check

Bit	Name	Function
11	L2DP	Set by a double bit ECC error in the L2
12	MCpin	Machine Check pin
13	TEA	TEA error
14	DP	Data Parity error
15	AP	Address Parity error

When the 750FX takes the machine check exception, it will set one or more error bits in SRR1, in HID02, or L2CR. A data parity error on the 60X bus is indicated by the DP bit. The MCpin bit indicates that the machine check pin was activated. The TEA bit indicates the machine check was caused by a TEA response on the 60X bus. An address parity error on the 60X bus will set the AP bit.

15.1.3 Processor Version Register

The Processor Version Number is x'7000' for the 750FX. The Processor Revision Level is described in the *IBM PowerPC 750FX RISC Microprocessor Datasheet*.

15.1.4 Implementation-Specific SPRs

The 750FX implements the implementation-specific Special Purpose Registers (SPRs) shown in *Table 15-3 750FX Implementation-Specific SPRs* on page 45.

Table 15-3. 750FX Implementation-Specific SPRs

Decimal	SPR		r/w	Privileged	Register Name	Function
	spr_{5-9}	spr_{0-4}				
528	10000	10000	r/w	yes	IBAT0U	Instruction BATs
529	10000	10001	r/w	yes	IBAT0L	Instruction BATs
530	10000	10010	r/w	yes	IBAT1U	Instruction BATs

Table 15-3. 750FX Implementation-Specific SPRs (Continued)

Decimal	SPR		r/w	Privileged	Register Name	Function
	spr_{5-9}	spr_{0-4}				
531	10000	10011	r/w	yes	IBAT1L	Instruction BATs
532	10000	10100	r/w	yes	IBAT2U	Instruction BATs
533	10000	10101	r/w	yes	IBAT2L	Instruction BATs
534	10000	10110	r/w	yes	IBAT3U	Instruction BATs
535	10000	10111	r/w	yes	IBAT3L	Instruction BATs
536	10000	11000	r/w	yes	DBAT0U	Data BATs
537	10000	11001	r/w	yes	DBAT0L	Data BATs
538	10000	11010	r/w	yes	DBAT1U	Data BATs
539	10000	11011	r/w	yes	DBAT1L	Data BATs
540	11110	11100	r/w	yes	DBAT2U	Data BATs
541	11110	11101	r/w	yes	DBAT2L	Data BATs
542	11110	11110	r/w	yes	DBAT3U	Data BATs
543	11110	11111	r/w	yes	DBAT3L	Data BATs
560	10001	10000	r/w	yes	IBAT4U	Instruction BATs
561	10001	10001	r/w	yes	IBAT4L	Instruction BATs
562	10001	10010	r/w	yes	IBAT5U	Instruction BATs
563	10001	10011	r/w	yes	IBAT5L	Instruction BATs
564	10001	10100	r/w	yes	IBAT6U	Instruction BATs
565	10001	10101	r/w	yes	IBAT6L	Instruction BATs
566	10001	10110	r/w	yes	IBAT7U	Instruction BATs
567	10001	10111	r/w	yes	IBAT7L	Instruction BATs
568	10001	11000	r/w	yes	DBAT4U	Data BATs
569	10001	11001	r/w	yes	DBAT4L	Data BATs
570	10001	11010	r/w	yes	DBAT5U	Data BATs
571	10001	11011	r/w	yes	DBAT5L	Data BATs
572	10001	11100	r/w	yes	DBAT6U	Data BATs
573	10001	11101	r/w	yes	DBAT6L	Data BATs
574	10001	11110	r/w	yes	DBAT7U	Additional BATs
575	10001	11111	r/w	yes	DBAT7L	Data BATs
952	11101	11000	r/w	yes	MMCR0	Performance monitor control register
953	11101	11001	r/w	yes	PMC1	Performance monitor counter
954	11101	11010	r/w	yes	PMC2	Performance monitor counter
955	11101	11011	r/w	yes	SIA	Sampled instruction address
956	11101	11100	r/w	yes	MMCR1	Performance monitor control register
957	11101	11101	r/w	yes	PMC3	Performance monitor counter
958	11101	11110	r/w	yes	PMC4	Performance monitor counter

Table 15-3. 750FX Implementation-Specific SPRs (Continued)

Decimal	SPR spr 5-9	SPR spr 0-4	r/w	Privileged	Register Name	Function
936	11101	01000	r	no	UMMCR0	Performance monitor control register
937	11101	01001	r	no	UPMC1	Performance monitor counter
938	11101	01010	r	no	UPMC2	Performance monitor counter
939	11101	01011	r	no	USIA	Sampled instruction address
940	11101	01100	r	no	UMMCR1	Performance monitor control register
941	11101	01101	r	no	UPMC3	Performance monitor counter
942	11101	01110	r	no	UPMC4	Performance monitor counter
1008	11111	10000	r/w	yes	HID0	Checkstop and misc. enables[1]
1009	11111	10001	ro	yes	HID1	Clock Configuration
1010	11111	10010	r/w	yes	IABR	Instruction breakpoint register[2]
1013	11111	10101	r/w	yes	DABR	Data address breakpoint reg[2]
1016	11111	11000	r/w	yes	HID2	Parity/Power management
1017	11111	11001	r/w	yes	L2CR	L2 Cache control register
1019	11111	11011	r/w	yes	ICTC	L cache Throttling control register
1020	11111	11100	r/w	yes	THRM1	Thermal Management register
1021	11111	11101	r/w	yes	THRM2	Thermal Management register
1022	11111	11110	r/w	yes	THRM3	Thermal Management register

[1]See *Table 15-4 HID0 Bit Functions* on page 47 and *Table 15-5 HID1 Bit Functions* on page 48.
[2]See *Section 15.2.10 Instruction Address Breakpoint Exception* on page 53.

15.1.5 Hardware Implementation Dependent Register 0 (HID0) Bit Functions

Table 15-4. HID0 Bit Functions

Bit	Name	Function
0	EMCP	machine check pin enable
1	DBP	Disable 60x bus address and data parity generation.
2	EBA	Enable 60x bus address parity checking
3	EBD	Enable 60x bus data parity checking
4	Reserved	Defined as CLK_OUT pin output enable and clock type selection on earlier 750-type processors.
5	Reserved	Defined as EICE on earlier 603-type processors.
6	Reserved	Defined as CLK_OUT pin output enable and clock type selection on earlier 750-type processors.
7	PAR	disable precharge of ARTRY and shared signals
8	DOZE	Select low-power doze.
9	NAP	Select low-power nap.
10	SLEEP	Select low-power sleep.
11	DPM	Dynamic Power Management enable

Table 15-4. HID0 Bit Functions (Continued)

Bit	Name	Function
12	RISEG	Read I SEG (For test only)
13	Reserved	—
14	Reserved	—
15	NHR	Not Hard Reset (software use only)
16	ICE	Instruction Cache enable
17	DCE	Data Cache enable
18	ILOCK	Instruction cache LOCK
19	DLOCK	Data cache LOCK
20	ICFI	Instruction Cache Flash Invalidate
21	DCFI	Data Cache Flash Invalidate
22	SPD	DCache and ICache speculative access disable
23	IFEM	Enable M bit on bus for instruction fetches
24	SGE	Store Gathering enable
25	DCFA	Data Cache Flush Assist. (Force data cache to ignore invalid sets on miss replacement selection.)
26	BTIC	Branch Target Instruction Cache enable.
27	Reserved	Defined as FBIOB on earlier 603-type processors.
28	ABE	Address Broadcast Enable (for cache ops, **eieio**, **sync**)
29	BHT	Branch History Table enable
30	Reserved	—
31	NOOPTI	No-op the **dcbt**/**dcbtst** instructions

15.1.6 Hardware Implementation Dependent Register 1 (HID1) Bit Functions

Table 15-5. HID1 Bit Functions

Bit	Name	Function
0-4	PCE	PLL external configuration bits (read-only)
5-6	PRE	PLL external range bits (read-only)
7	PSS	Processor Switch Status
8	ECLK	Set to b’1’ to enable the CLKOUT pin.
9-11	—	Select the internal clock to be output on the CLKOUT pin with the following decode: b’000’ = Internal core clock b’001’ = PLL0 core clock b’010’ = PLL0 rcv clock b’011’ = PLL1 core clock b’100’ = PLL1 rcv clock Other = reserved
12-13	-	Reserved

Table 15-5. HID1 Bit Functions

Bit	Name	Function
14	PI0	PLL 0 internal configuration select 0 Select external configuration and range bits to control PLL 0 1 Select internal fields in HID1 to control PLL0
15	PS	PLL select 0 Select PLL 0 as source for processor clock 1 Select PLL 1 as source for processor clock
16-20	PC0	PLL 0 configuration bits
21-22	PR0	PLL 0 range select bits
23	-	Reserved
24-28	PC1	PLL 1 configuration bits
29-30	PR1	PLL 1 range bits
31	-	Reserved
Note: These clock configuration bits reflect the state of the PLL_CFG[0:4] pins.		

15.1.7 Hardware Implementation Dependent Register 2 (HID2) Bit Functions

Parity is enabled with a new register, HID2. The HID2 bits are defined in the following *Table 15-6 HID2 Bit Definitions* on page 49.

Table 15-6. HID2 Bit Definitions

HID2 Bit[1]	Description	Name
0-24	Reserved	—
25	I-Cache/I-Tag Parity Error Status	ICPS
26	D-Cache/D-Tag Parity Error Status	DCPS
27	L2 Tag Parity Error Status	L2PS
28	Reserved	—
29	Enable I-Cache/I-Tag parity checking	ICPE
30	Enable D-Cache/D-Tag parity checking	DCPE
31	Enable L2 Tag parity checking	L2PE

Note:
1. Not supported in DD 1.X.

HID2 SPR number is 1016 decimal, (spr[5-9] = 11111, spr[0-4] = 11000).

The status bits (25-27) are set when a parity error is detected and cleared by writing 0 to each bit.

15.1.8 Illegal and Invalid SPR Operations

Any **mtspr**, **mfspr**, or **mftb** with an invalid SPR field will cause an illegal type program exception. An access to a privileged SPR while in the Problem State (MSR[PR]=1) will result in a privileged instruction type program exception. A **mtspr** executing in privileged state (MSR[PR]=0) with the SPR field specifying HID1 or PVR (read-only registers) will execute as a no-op.

With the above definition, the 750FX treats **mtspr** and **mfspr** instructions which reference SPRs defined for Power and not for PowerPC as illegal.

15.1.9 TLB and BAT Software Reset Requirement

The TLBs are not automatically invalidated on reset. They must be invalidated under program control. The **tlbie** instruction is to be used to invalidate the TLBs. The BATs are cleared by writing to them with the **mtspr** instruction.

15.2 Exceptions

15.2.1 Machine Check Exception

A machine check exception occurs if MSR[ME]=1 and a 60x bus error (TEA_), 60x bus address parity error, 60x bus data parity error, an L2 ECC double bit error, locked L2 snoop hit[1], or a cache/tag parity error occurs[1]. This exception may also be generated with the machine check pin (MCP_). The priority of this exception is second only to a system reset exception due to a hard reset request.

For a machine check the 750FX will make no attempt to force recoverability. Furthermore, the GPR, L1 and L2 caches may contain corrupt data. The 750FX guarantees that the machine check exception will always be taken immediately upon request, with a non-speculative address saved in SRR0, regardless of the current machine state. Any pending stores in the completed store queue are drained (completed) before a machine check exception is taken.

Note: One case in which software can use the machine check exception in a recoverable mode is for checking bus configuration. For this case a **sync**, load, **sync** instruction sequence is used. A subsequent machine check exception at the load address indicates a bus configuration problem and the processor will be in a recoverable state.

The machine check exception uses the normal machine check offset of 0x00200.

See *Table 15-7 Sources and Priorities of Externally-Generated Exceptions* on page 51 for relative priorities and recoverability of machine check, system reset and other externally-generated exceptions.

15.2.2 System Reset Exception

The system reset exception is a non-maskable exception that is signaled to the 750FX through the assertion of an input signal to the chip (HRESET_, SRESET_). This exception is considered asynchronous in the manner in which it is received and in the manner in which it breaks the pipeline to handle the exception.

1. Not supported in DD 1.X.

If a hard reset request occurs (HRESET_), the processor will immediately branch to the system reset exception vector without attempting to reach a recoverable state. A hard reset forces the system exception vector address to be 0xFFF00100. A system reset exception due to a hard reset has the highest exception priority and will always be non-recoverable. The address saved in SRR0 upon taking this exception will always be guaranteed non-speculative.

If a soft reset request is made (SRESET_), the processor will first progress to a recoverable state. To do this, the 750FX will allow any instruction at the point of completion to either complete or except, block completion of any following instructions and allow the completed store queue to drain. The state before the exception occurred is then saved as specified in the PowerPC Architecture and the system reset exception vector is taken. The vector address on a soft reset depends on the MSR[IP] bit and will be either 0x00000100 or 0xFFF00100. This exception case is third in priority, following hard reset and machine check. This exception will be recoverable provided attaining a recoverable state does not generate a machine check.

See *Table 15-7 Sources and Priorities of Externally-Generated Exceptions* on page 51 and *Table 15-9 750FX Exception Priorities* on page 55 for relative priorities and recoverability of machine check, system reset, and other externally-generated exceptions.

Table 15-7. Sources and Priorities of Externally-Generated Exceptions

Priority	Exception	Cause	Recoverability
0	System Reset	HRESET_, POR	nonrecoverable, all cases
1	Machine Check	TEA_, 60X bus address parity, 60X bus data parity, MCP_	nonrecoverable, most cases
2	System Reset	SRESET_	recoverable unless Machine Check occurs
3	SMI	SMI_	recoverable unless a Machine Check or System Reset occurs
4	EI	INT_	recoverable unless a Machine Check or System Reset occurS

15.2.3 System Management Exception

A system management exception is enabled if MSR[EE]=1 and is signaled to the 750FX by the assertion of an input signal pin (SMI_). It is expected that once this signal is asserted, it remains asserted until the 750FX takes the system management exception. Failure to do so may lead to unexpected results.

Once an enabled system management exception request is detected, the 750FX generates a recoverable halt to instruction completion. The 750FX will require the next instruction in program order to complete or cause an exception, block completion of any following instructions and allow the completed store queue to drain. If any higher priority exceptions are encountered in this process, those exceptions are taken first and the system management exception is delayed until a recoverable halt is achieved. At this time the 750FX will save state and take the system management exception. The system management exception behaves like an external exception except that a dedicated vector will be taken as shown in *Table 15-8 750FX Implementation-Specific Exception Vector Offsets* on page 53.

15.2.4 External Exception

An external exception is enabled if MSR[EE]=1 and is signaled to the 750FX by the assertion of an input signal pin (INT_). It is expected that once this signal is asserted, it remains asserted until the 750FX takes the external exception. Failure to do so may lead to unexpected results.

Once an enabled external exception request is detected, the 750FX generates a recoverable halt to instruction completion. The 750FX will require the next instruction in program order to complete or cause an exception, block completion of any following instructions and allow the completed store queue to drain. If any other exceptions are encountered in this process, those exceptions are taken first and the external exception is delayed until a recoverable halt is achieved. At this time the 750FX will save state and take the external exception as defined in the PowerPC Architecture.

15.2.5 Thermal Management Exception

A thermal management exception is generated by the Thermal Management Assist Unit (TAU) when the junction temperature crosses a programmed threshold. The exception is enabled by the TIE bit of THRM[1,2] and the exception is maskable by the MSR[EE] bit. Once the exception occurs, it remains asserted until cleared by the exception unit upon taking the exception.

Once an enabled thermal management exception request is detected, the 750FX generates a recoverable halt to instruction completion. The 750FX will require the next instruction in program order to complete or cause an exception, block completion of any following instructions and allow the completed store queue to drain. If any other exceptions are encountered in this process, those exceptions are taken first and the thermal management exception is delayed until a recoverable halt is achieved. At this time the 750FX will save state and take the thermal management exception. The thermal management exception behaves like an external exception except that a dedicated vector will be taken as shown in *Table 15-8 750FX Implementation-Specific Exception Vector Offsets* on page 53.

15.2.6 Alignment Exception

The 750FX will initiate an alignment exception when it detects any of the following conditions.

- The operand of a floating point load or store is not word aligned.
- The operand of **lmw**, **stmw**, **lwarx**, or **stwcx** is not word aligned.
- The operand of **dcbz** is in a page that is write-through or cache-inhibited, for a virtual mode access or an attempt to execute **dcbz** when the cache is disabled.
- **eciwx** or **ecowx** that is not word aligned.
- A multiple or string access with the Little Endian bit set.

15.2.7 Data Storage Exception

A data storage exception will occur for all cases specified in the PowerPC Architecture. In the case of a TLB miss for a load, store or cache operation, a hardware table walk is performed and a DSI will be taken if the table walk results in a page fault. Also, a DSI will be taken when any load/store instruction accesses T=1 space.

System Implementation Note: In the 750FX a floating-point load or store to a direct-store segment (T=1) space causes a Data Storage Exception rather than a Alignment Exception as specified in the PowerPC Operating Environment Architecture Book III version 1.07. This is consistent with 603e/603ev processors.

15.2.8 Implementation-Specific Exception Vector Offsets

750FX implementation-specific exception offsets are given in *Table 15-8 750FX Implementation-Specific Exception Vector Offsets* on page 53.

Table 15-8. 750FX Implementation-Specific Exception Vector Offsets

Offset (hex)	Exception type
0F00	Performance Monitor Exception
1300	Instruction Address Breakpoint
1400	System Management Exception
1700	Thermal Management Exception

15.2.9 Trace Exception

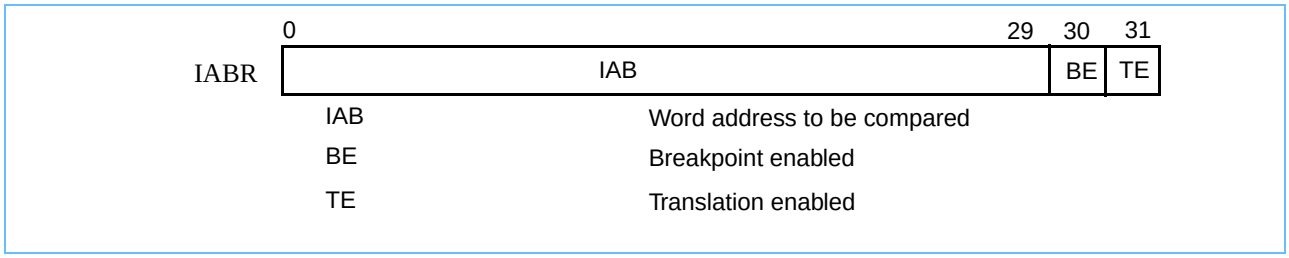
The trace exception is implemented in the 750FX and will be taken if MSR[SE]=1 or if MSR[BE]=1 and the currently completing instruction is a branch. Each instruction considered during trace mode will complete before a trace exception is taken. The 750FX deviates from the architecture by not taking trace exceptions on **isync** instructions.

15.2.10 Instruction Address Breakpoint Exception

The instruction address breakpoint exception is implemented in the 750FX and will be taken if the following are met: the instruction breakpoint address IABR[0:29] matches the EA[0:29] of the next instruction to complete in program order, the $IABR[TE]_{31}$ translation enable bit matches the MSR[IR] bit, and the $IABR[BE]_{30}$ enable bit is set. The address match will also be reported to the jtag/cop block which may subsequently generate a soft stop or hard stop for this condition. The instruction tagged with the match will not be completed before the breakpoint exception is taken. The vector offset for this exception is shown in *Table 15-8 750FX Implementation-Specific Exception Vector Offsets* on page 53.

The 750FX requires that a **mtspr**(IABR) be followed by a context synchronizing instruction. The 750FX cannot generate a breakpoint response for that context synchronizing instruction if the breakpoint was enabled by the **mtspr**(IABR) immediately preceding it. The 750FX also cannot block a breakpoint response on the context synchronizing instruction if the breakpoint was disabled by the **mtspr**(IABR) immediately preceding it. See *Section 15.4 Synchronization Requirements for SPRs and Segment Registers* on page 58 for more information on this requirement. The format of the IABR register is shown in the figure below:

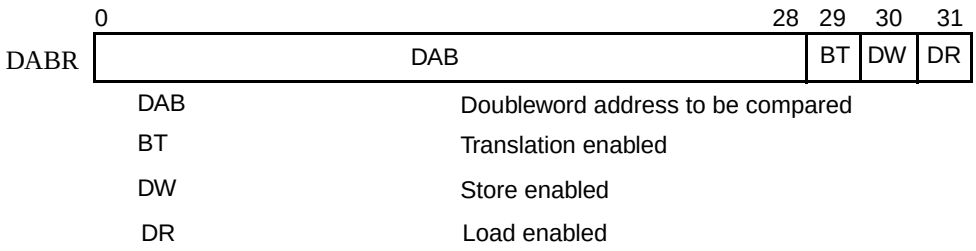
Figure 15-2. IABR Register Diagram



15.2.11 Data Address Breakpoint Exception

The Data Address Breakpoint Register is a special purpose register which can cause a Data Storage Exception (DSI). When enabled, data addresses will be compared with an effective address that is stored in the DABR (bits 0:28). The granularity of these compares will be a double-word. Bit 29 is the Translation enable bit and is compared with the MSR[DR] bit. Bit 30 is a Store enable. Bit 31 is a Load enable. The DABR is enabled by setting either the DW or DR bit. The format of the DABR register is shown in the figure below.

Figure 15-3. DABR Register Diagram



15.2.12 Soft Stops

Both trace and breakpoint exception conditions will generate a soft stop instead of an exception if soft stop has been enabled by the jtag/cop logic. If trace and instruction breakpoint conditions occur simultaneously, instruction breakpoint will take priority over trace in both the exception and soft stop enabled cases.

A soft stop may also be generated with a request from the COP. This request will be treated similarly to an external exception, except that it will be non-maskable and will generate a soft stop instead of an exception.

If soft stop is enabled, only one soft stop will be generated before completion of an instruction with an IABR match. This holds true if a soft stop is generated before that instruction for any other reason, such as trace mode on for the preceding instruction or a COP soft stop request.

15.2.13 Performance Monitor Exception

A performance monitor exception is taken when one of the specified PMCs saturates from counting events.

15.2.14 Floating Point Assist Exception

The floating point assist exception is not implemented in the 750FX.

15.2.15 Floating Point Unavailable

As defined in PowerPC Architecture.

15.2.16 Program Exception

As defined in PowerPC Architecture.

15.2.17 System Call Exception

As defined in PowerPC Architecture.

15.2.18 Instruction Storage Exception

As defined in PowerPC Architecture.

15.2.19 Decrementer Exception

As defined in PowerPC Architecture.

15.2.20 Exception Latencies

Latencies for taking various exceptions are variable based on the state of the machine when conditions to produce an exception occur. The shortest latency possible is one cycle. In this case an exception is signaled the cycle following appearance of the conditions which generated that exception. In most cases, a hard reset or machine check will have a single-cycle latency to exception. The only situation which may prevent this is when a speculative instruction is next to complete. This case, which produces an extra two-cycle minimum, three-cycle maximum delay, will only occur if the branch guess which forced this instruction to be speculative was resolved to be incorrect.

Another latency variable is introduced for a soft reset exception -- recoverability. The time to reach a recoverable state may depend on the time needed to complete or except an instruction at the point of completion, the time needed to drain the completed store queue or the time waiting for a correct empty state so that a valid IP may be saved. For other externally-generated exceptions, a further delay may be incurred waiting for another exception, generated while reaching a recoverable state, to be serviced.

Further delays are possible for other types of exceptions depending on the number and type of instructions that must be completed before that exception may be serviced. See *Section 15.2.22 Summary of Front-End Exception Handling* on page 56 to determine possible maximum latencies for different exceptions.

15.2.21 Exception Priorities

Table 15-9. 750FX Exception Priorities

Priority	Exception	Cause
Asynchronous Exceptions		
0	System Reset	HRESET_, POR
1	Machine Check	TEA_, 60x Address Parity Error, 60x Data Parity Error, L2 ECC Double Bit Error, MCP_, L2-Tag Parity Error, D-Tag Parity Error, I-Tag Parity Error, I-Cache Parity Error, D-Cache Parity Error, or locked L2 snoop hit[1]
2	System Reset	SRESET_
3	SMI	SMI_ (System Management Exception)
4	EI	INT_ (External Exception)
5	PFM	Performance Monitor Exception
6	DEC	Decrementer Exception
7	TMI	Thermal Management Exception
Instruction Fetch Exceptions		
0	ISI	Instruction Storage Exception
Instruction Dispatch/Execution Exceptions		
0	IABR	Instruction Address Breakpoint Exception
1	PI	Program Exception due to: 1. Illegal instruction 2. Privileged instruction 3. Trap

Table 15-9. 750FX Exception Priorities

Priority	Exception	Cause
2	SC	System Call
3	FPA	Floating Point Unavailable Exception
4	PI	Program Exception due to: 1. Floating point enabled exception
5	DSI	Data Storage Exception due to: 1. **eciwx**, **ecowx** with EAR(E)=0 (bit 11 of DSISR)
6	Alignment	Alignment Exception due to: 1. Floating point not word aligned 2. **lmw**, **stmw**, **lwarx**, **stwcx** not word-aligned 3. **eciwx** or **ecowx** not word-aligned 4. Multiple or string access with the Little Endian bit set. 5. **dcbz** to write thru or cache-inhibited page or cache is disabled.
7	DSI	Data Storage Exception due to: 1. BAT page protection violation
8	DSI	Data Storage Exception due to: 1. Any access except cache operations to T=1 (bit 5 of DSISR). 2. T=0 -> T=1 crossing ((bit 5 of DSISR).
9	DSI	Data Storage Exception due to: 1. TLB page protection violation
10	DSI	Data Storage Exception due to: DABR address match
11	Trace	Trace Exception due to: 1. MSR[SE]=1 2. MSR[BE]=1 for branches

Notes:
1. Not supported in DD.1X.
2. Even though DSISR(5) and DSISR(11) are set by different priority exceptions, both bits can be set at the same time.

15.2.22 Summary of Front-End Exception Handling

The following list of exception categories describes how the 750FX handles exceptions up to the point of signaling the appropriate exception to occur. A recoverable state is reached in the 750FX if the completed store queue is empty (drained, not canceled) and any instruction which is next in program order and has been signaled to complete has completed. If MSR[RI] = 0, the 750FX is in a nonrecoverable state by default. Also, completion of an instruction is defined as performing all architectural register writes associated with that instruction, and then removing that instruction from the completion buffer queue.

- Asynchronous Non-maskable Nonrecoverable: System Reset for HRESET_
  Has highest priority and is taken immediately regardless of other pending exceptions or recoverability. A non-speculative address is guaranteed.
- Asynchronous Maskable Nonrecoverable: Machine Check
  Takes priority over any other pending exception except System Reset for HRESET_ or POR. Taken immediately regardless of recoverability. A non-speculative address is guaranteed.
- Asynchronous Non-maskable Recoverable: System Reset for SRESET_
  Takes priority over any other pending exception except System Reset for HRESET_ or POR or Machine Check. Taken immediately when a recoverable state is reached.

- Asynchronous Maskable Recoverable: SMI, EI, DEC
  Before handling this type of exception, the next instruction in program order must complete or except. If this action causes another type of exception, that exception is taken and the Asynchronous Maskable Recoverable (AMR) exception remains pending. Once an instruction is able to complete without causing an exception, while the AMR exception is enabled, further instruction completion is halted. The AMR exception is then taken once a recoverable state is reached.
- Instruction Fetch: ISI
  Once this type of exception is detected, dispatch is halted and the current instruction stream is allowed to drain out of the machine. If completing any of the instructions in this stream causes an exception, that exception is taken and the Instruction Fetch exception is forgotten. Otherwise, once the machine is empty and a recoverable state is reached, the Instruction Fetch exception is taken.
- Instruction Dispatch/Execution: Program, DSI, Alignment, FPA, SC, IABR, DABR
  This type of exception is determined at dispatch or execution of an instruction. The exception will remain pending until all instructions in program order before the exception-causing instruction are completed. The exception will then be taken without completing the exception-causing instruction. If any other exception condition is created in completing these previous instructions in the machine, that exception will take priority over the pending Instruction Dispatch/Execution exception, which will then be forgotten.
- Post Instruction Execution: Trace
  This type of exception is generated following execution and completion of an instruction while a trace mode is enabled. If executing the instruction produces conditions for another type of exception, that exception is taken and the Post Instruction Execution exception is forgotten for that instruction.

15.3 THRM1, THRM2, THRM3 Bit Field Settings

The bit fields in the THRM1 and THRM2 SPRs are described in *Table 15-10*. The bit fields in the THRM3 SPR are described in *Table 15-11*.

Table 15-10. THRM1 and THRM2 SPR Bit Field Settings

Bits	Field	Description
0	TIN	Thermal management interrupt bit. Read only. This bit is set if the thermal sensor output crosses the threshold specified in the SPR. The state of this bit is valid only if TIV is set. The interpretation of the TIN bit is controlled by the TID bit.
1	TIV	Thermal management interrupt valid. Read only. This bit is set by the thermal assist logic to indicate that the thermal management interrupt (TIN) state is valid.
2–8	Threshold	Threshold value that the output of the thermal sensor is compared to. The threshold range is between 0° and 127° C, and each bit represents 1° C. This is not the resolution of the thermal sensor.
9–28	—	Reserved. System software should clear these bits to 0.

Table 15-10. THRM1 and THRM2 SPR Bit Field Settings

Bits	Field	Description
29	TID	Thermal management interrupt direction bit. Selects the result of the temperature comparison to set TIN. If TID is cleared to 0, TIN is set and an interrupt occurs if the junction temperature exceeds the threshold. If TID is set to 1, TIN is set and an interrupt is indicated if the junction temperature is below the threshold.
30	TIE	Thermal management interrupt enable. Enables assertion of the thermal management interrupt signal. The thermal management interrupt is maskable by the MSR[EE] bit. If TIE is cleared to 0 and THRM*n* is valid, the TIN bit records the status of the junction temperature vs. threshold comparison without asserting an interrupt signal. This feature allows system software to make a successive approximation to estimate the junction temperature.
31	V	SPR valid bit. This bit is set to indicate that the SPR contains a valid threshold, TID, and TIE controls bits. Setting THRM1/2[V] and THRM3[E] to 1 enables operation of the thermal sensor.

Table 15-11. THRM3 Bit Field SPR Settings

Bits	Name	Description
0-17	—	Reserved for future use. System software should clear these bits to 0.
18–30	SITV	Sample interval timer value. Number of elapsed processor clock cycles before a junction temperature vs. threshold comparison result is sampled for TIN bit setting and interrupt generation. This is necessary due to the thermal sensor, DAC, and the analog comparator settling time being greater than the processor cycle time. The value should be configured to allow a sampling interval of 20 microseconds.
31	E	Enables the thermal sensor compare operation if either THRM1[V] or THRM2[V] is set to 1.

15.4 Synchronization Requirements for SPRs and Segment Registers

Altering certain registers requires software synchronization to honor register dependencies for following instructions. A context-synchronizing operation must follow any instruction which affects instruction fetch or data access dependencies by altering any of the registers listed in *Table 15-12 Instructions and Registers Requiring a Context-Synchronization Operation* on page 58.

Table 15-12. Instructions and Registers Requiring a Context-Synchronization Operation

Instruction Fetch Dependencies	Data Access Dependencies	Other
• Segment Registers • IBATs • SDR1 • MSR bits: PR, FP, FE0, FE1, IR, LE • IABR	• Segment Registers • DBATs • SDR1 • EAR • MSR bits: PR, DR, LE • DABR	• MSR bits: POW, FP • HID0

Context-synchronizing operations which may be used are **isync**, **sc**, **rfi** and any exception other than System Reset and Machine Check.

In order to insure fetching in the new endian-mode, it is required to use an **rfi** when changing the MSR[LE] to guarantee a solid context boundary. Placing an **isync** after a **mtmsr** which updates the MSR[LE] does not guarantee fetching in the new endian-mode because, due to the endian-mode change, the **isync** itself may not get fetched.

The alteration of the MSR[POW] bit must be followed by a context synchronizing operation.

Revision Log

Revision Date	Contents of Modification
May 30, 2002	Initial preliminary release version 1.0.