**Freescale Semiconductor**
Application Note

Document Number: AN2540
Rev. 1, 11/2006

# Synchronizing Instructions for PowerPC™ Instruction Set Architecture

*by* *Freescale Semiconductor, Inc.*
*Austin, TX*

The architecture for the PowerPC™ instruction set provides two types of synchronizing instructions:

- Context-synchronizing
- Execution-synchronizing

This application note presents a high-level definition and description of each type of synchronizing instruction. All information in this application note is covered in more detail in *PowerPC Microprocessor Family: The Programming Environments for 32-Bit Processors*.

**Contents**



## 1 Context-Synchronizing Instructions

Context-synchronizing instructions for PowerPC include **isync**, **sc**, and **rfi**. In addition, exceptions can cause context synchronization. These instructions can be used to ensure that the effects of all previously issued instructions are in place before a context switch and that the context switch takes effect for instructions after the switch. Context synchronization should be performed when the values in certain fields of certain processor registers are changed, as shown in Table 1.

The context-synchronizing instructions ensure that the following conditions occur:

- Instruction dispatching is halted. The **sc** instruction also ensures that no higher-priority exception exists.
- All previously issued instructions have completed, at least to a point where they can no longer cause an exception. However, memory accesses that these instructions cause need not have completed with respect to other processors and mechanisms. The **sync** instruction can be used to ensure that memory accesses are complete.
- Previously-issued instructions complete in the context in which they were issued (privilege, protection, address translation).
- Instructions issued after the synchronizing instruction execute in the new context.

To ensure that context changes occur for instructions after the synchronization, the instruction queue is flushed and all these instructions are refetched with the new context in place.

**NOTE**

All context-synchronizing instructions are execution synchronizing.

# 2 Execution-Synchronizing Instructions

Context-synchronizing instructions are useful for ensuring that context switches take effect at the right time. Execution-synchronizing instructions are useful for providing a sequencing function in a program and for synchronizing memory accesses on multiple-processor implementations. These instructions can ensure that previous instructions completed before subsequent instructions execute without forcing a flush of prefetched instructions in the instruction queue. This feature is often useful for ordering programs and for debugging. However, the instruction uses a significant amount of time to complete and should not be used indiscriminately. For many applications that specifically order loads and stores, the **eieio** instruction may provide the desired effect at lower cost to performance.

Execution-only synchronizing instructions (those that are not also context-synchronizing) differ from context-synchronizing instructions in that they do not ensure that subsequent instructions execute in the context that the synchronizing instruction establishes. (When the instruction queue is not flushed, instructions after the synchronization in the queue execute in the old context). The new context takes effect sometime after the execution-synchronizing instruction completes. Like context-synchronizing instructions, all execution-synchronizing instructions halt dispatching and ensure that previous instructions have completed to the point where they cannot cause an exception. Execution-synchronizing instructions include **sync**, **mtmsr**, and all context-synchronizing instructions.

In addition to synchronizing execution, the **sync** instruction, can broadcast addresses on the bus and is sometimes used for synchronizing coherent memory with other processors. Unlike **isync**, **sync** forces all external accesses to complete with respect to other processors and mechanisms that access memory.

## 2.1 Synchronization Requirements for Instructions that Alter Context

Table 1 summarizes the synchronization requirements for instructions that alter context. Individual processors occasionally have additional synchronization requirements or restrictions. For complete

information, see *PowerPC Microprocessor Family: The Programming Environments for 32-Bit Processors*, section 2.3.17, “Synchronization Requirements for Special Registers and for Lookaside Buffers” and your specific processor manual.

**Table 1. Synchronization Requirements for Instructions that Alter Context**

| Instruction | Synchronization Required Before | Synchronization Required After |
|---|---|---|
| mtmsr[PR] | none | context |
| mtmsr[FP](instr access) | none | context |
| mtmsr[FE0,FE1](instr access) | none | context |
| mtmsr[SE,BE](instr access) | none | context |
| mtmsr[ME] | none | context |
| mtmsr[LE] | implementation dependent | implementation dependent |
| mtmsr[DR](data access) | none | context |
| mtmsr[IR](instr access) | none | context |
| mtmsr[POW](instr access) | implementation dependent | implementation dependent |
| \|mtmsr[DABR](data access) | implementation dependent | implementation dependent |
| mtsr(data access) | context | context |
| mtsr(instr access) | none | context |
| mtsrin(data access) | context | context |
| mtsrin(instr access) | none | context |
| mtspr[SDR1] | **sync** | context |
| mtspr[DBAT](data access) | context | context |
| mtspr[IBAT](instr access) | none | context |
| mtspr[EAR](data access) | context | context |
| tlbie(data access) | context | context or **sync** |
| tlbie(instr access) | none | context or **sync** |
| tlbia(data access) | context | context or **sync** |
| tlbia(instr access) | none | context or **sync** |

## 2.2 Hardware Implementation Registers (HIDs)

In addition to the instructions listed in Table 1, many processors have bits in the hardware implementation registers (HIDs) that require synchronization when they are changed. For example, many PowerPCs require synchronization before changing the bits that control cache enabling and locking for the instruction and data caches. Changing the DCE and DLOCK bits in HID0 for many processors requires a **sync** before a change, and setting the ICE and ILOCK bits in HID0 necessitates an **isync** before the change. For details on synchronization requirements when bits are changed in the hardware implementation registers, carefully read the manual for your PowerPC processor.

**NOTE**

Some register fields require synchronization differently for data or instruction accesses.

# 3 Sources of Information

More information on this topic can be found in the Freescale PowerPC library and in the following publications:

- *PowerPC Microprocessor Family: The Programming Environments for 32-Bit Processors*
  - Section 4.1.5 Synchronizing Instructions
  - Section 6.1.2 Synchronization
  - Section 2.3.17 Synchronization Requirements for Special Registers and for Lookaside Buffers
  - Section 8.2 PowerPC Instruction Set
  - Appendix E Synchronization Programming Examples (see listings for **sync**, **isync**, **rfi**, **sc**, **mtmsr**, and **eieio**)
- *MPC603e and EC603e User's Manual*
  - Section 2.3.2.4 Synchronization
  - Section 2.3.5.2 Memory Synchronization
  - Section 6.3.3.2 Instruction Serialization
- *MPC750 User's Manual*
  - Section 2.3.2.4 Synchronization
  - Section 2.3.4.7 Memory Synchronization Instruction
  - Section 4.4 Process Switching
  - Section 6.3.2.2 Instruction Serialization

The manuals for the individual processors that are not listed contain details for a specific processor also. This information would be too cumbersome to list in this document. See your processor manual for complete information.

# 4 Revision History

Table 2 provides a revision history for this application note.

**Table 2. Document Revision History**

| Rev. No. | Substantive Change(s) |
|---|---|
| 0 | Initial release |
| 1 | Nontechnical reformatting |

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

***How to Reach Us:***

**Home Page:**
www.freescale.com

**email:**
support@freescale.com

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
1-800-521-6274
480-768-2130
support@freescale.com

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064, Japan
0120 191014
+81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

**For Literature Requests Only:**
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com





Document Number: AN2540
Rev. 1
11/2006