Freescale Semiconductor
Application Note

Document Number: AN2580
Rev. 1, 1/2007

L3ITCR Registers of the L3 Cache Interface of the MPC7450 Family

by *Michael Everman*
CPD Applications
Freescale Semiconductor, Inc.
Austin, TX

This document describes how to use the L3ITCR*n* registers on the PowerPC™ MPC7450 family of microprocessors. Because the implementation of these registers is similar on all members of the MPC7450 family, this document refers to the MPC7450 except where specific differences exist. These registers are described as factory-only debugging registers in the *MPC7450 RISC Microprocessor Family User's Manual* and are not characterized or described in the *MPC7450 Hardware Specifications*. However, system designers may find the registers helpful when debugging potential AC timing issues because they allow the input AC timing of the L3 interface to be adjusted in software.

While the MPC7455 and MPC7457 processors allow the output AC timing of the L3 interface to be adjusted in software by means of the L3OH[0–1] bits or the L3OHCR register, respectively, these registers have no impact on the input AC timing of the L3 interface. Ordinarily, detailed analyses during layout and corrections made by adjusting trace lengths should resolve any AC timing issues, as described in *Understanding the MPC7450-family L3 Cache Hardware Interface* (AN2468). However, altering the L3 input AC timing on an existing board, especially during early system development and debugging, may be desirable or necessary. Use the L3ITCR*n* registers for this alteration.

Contents

freescale™ semiconductor

1 Function of the L3ITCR*n* Registers

To understand how to use the L3ITCR*n* registers, it is helpful to know how the processor uses them—especially how data and echo clocks are returned from DDR SRAM. Figure 1 shows that the SRAM drives echo clock edges concurrently with data transitions (that is, the clock phase envelopes the data). To latch the data correctly, the processor delays the echo clock edge internally so that it is aligned within the valid data window.

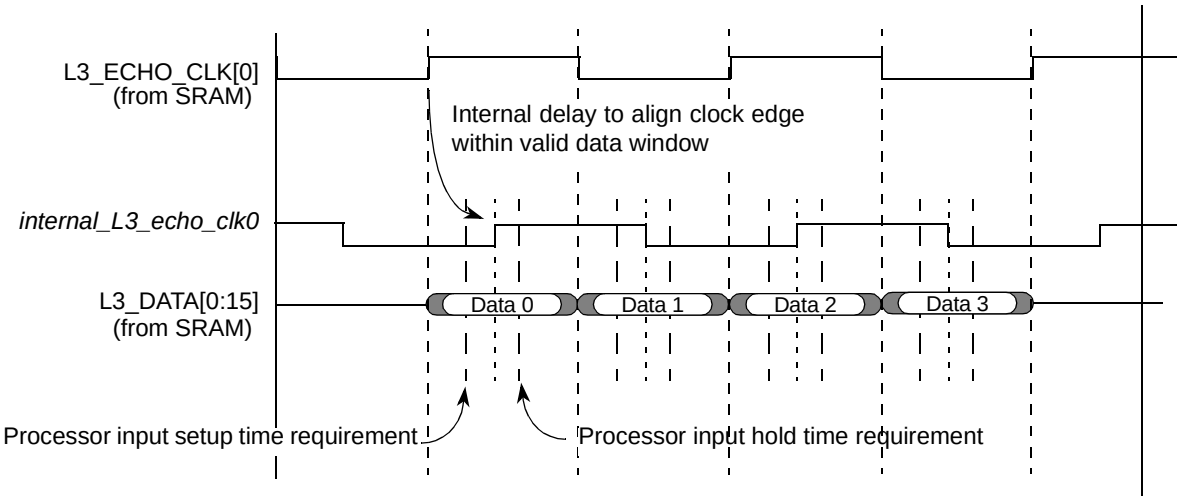

Figure 1. Internal Delay of L3 Echo Clock

Because the amount of delay needed is proportional to the clock period, a static delay cannot be used. Furthermore, because the interface is asynchronous, no internal clock such as the core or VCO clocks can be used to generate the delay. Instead, the processor uses a special circuit to determine the necessary delay automatically. The circuit selects a numerical value that in turn selects the proper delay from a delay chain. Figure 2 shows this operation for the MPC7450, MPC7451, and MPC7455, and Figure 3 shows it for the MPC7457.

The implementations are very similar. The main difference is that the MPC7457 implements a separate circuit for each echo clock input, whereas the other devices have one circuit that sets the delays for all echo clocks, implementing only L3ITCR0. Additionally, the MPC7457 uses a larger bit field for L3DC*n*. These diagrams show that either of the following values can be used:

- Value that an automatic delay calculation circuit determines (called the *natural* value)
- Value written in L3ITCR*n*[L3DC*n*] (called the *override* value)

The natural value, which is the default, is selected when L3ITCR*n*[L3DCO*n*] is cleared. The override value is selected when L3DCO*n* is set. The automatic calculation circuit is completely disabled when L3ITCR0[L3DCDIS0] is set. Note that on the MPC7457, L3DCDIS0 disables the automatic calculation circuit for all echo clocks. Because the natural value is not used at all when L3DCO*n* is set, L3DCDIS*n* should not be set. We recommend leaving these bits cleared, even when using an override value to program the internal echo clock delay.

NOTE

It is impossible to use software to read the override value in the L3ICTR registers because attempting to read these registers always returns the natural value and does not read the override value contained in the register itself. The override values can be read only by means of COP. Natural values cannot be read using COP.

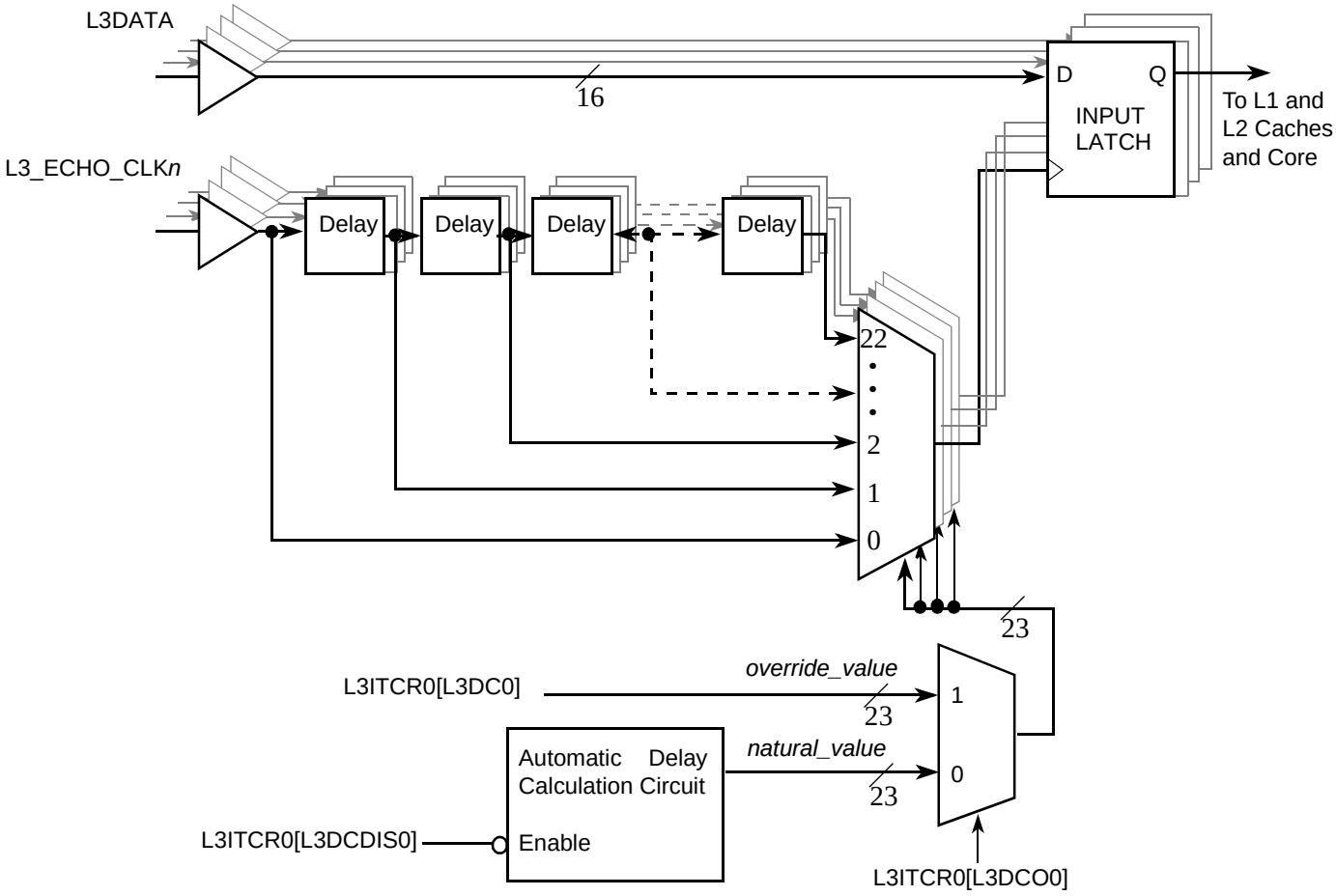

Figure 2. L3 Echo Clock Internal Delay Circuit for MPC7450, MPC7451, and MPC7455

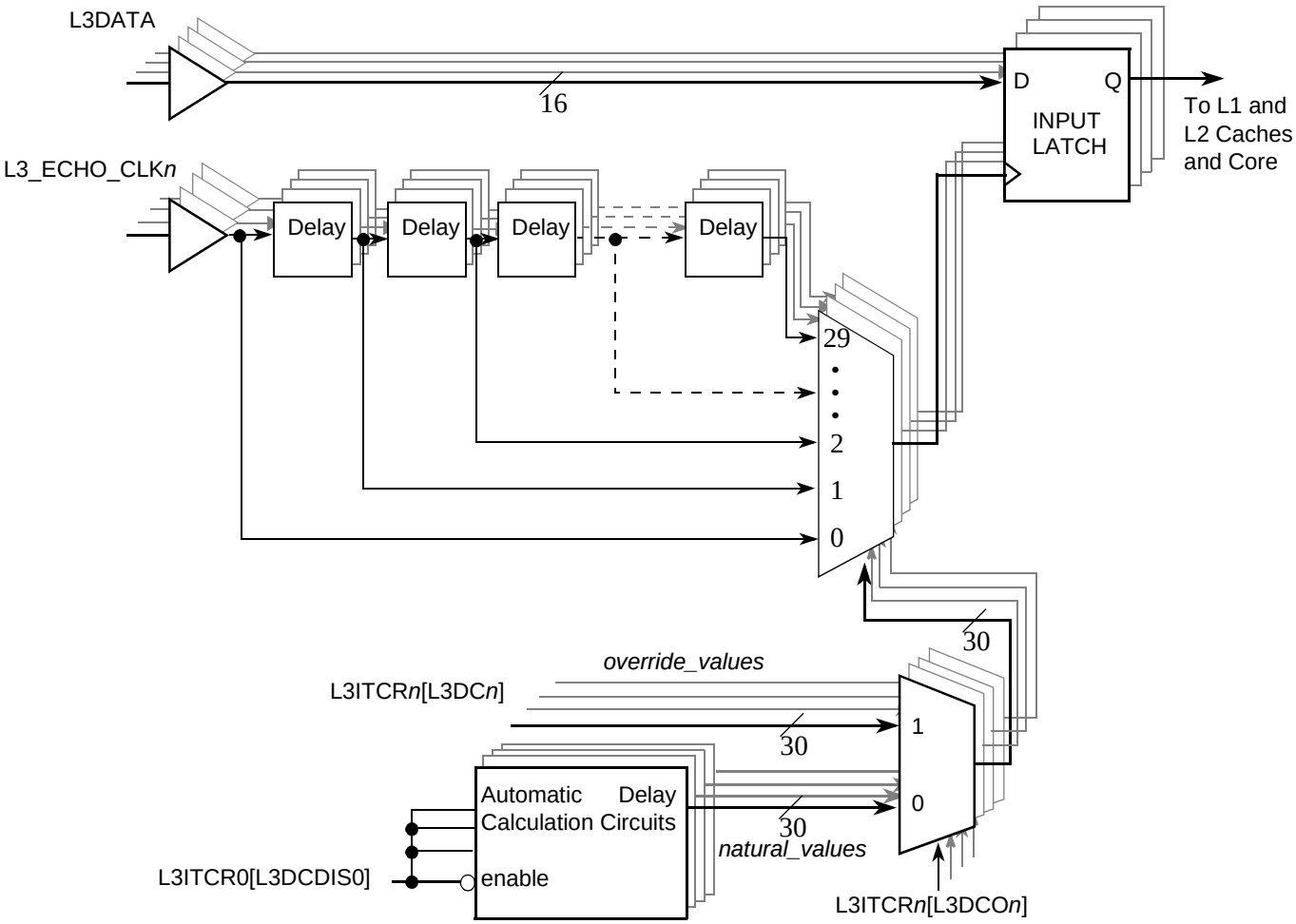

Figure 3. L3 Echo Clock Internal Delay Circuit for MPC7457

2 Adjusting the Delay

Adjusting L3ITCR[0–3] causes the L3 echo clock phases to be shifted internally, thereby changing the input timing characteristics of the L3 interface. As Figure 2 and Figure 3 show, the state of the L3ITCR*n* registers determines the amount of delay added to received L3_ECHO_CLK signals. By default, L3DCO*n* = 0 and the natural value determine the amount of delay added to align clocks inside the valid data window. When L3DCO*n* = 1, the natural value is overridden and the value of L3DC*n* is used instead. Because the natural value is a baseline for L3 AC timing specifications, we recommend using the natural value as a starting point and shifting this value by one or more bits to determine the override values. The amount of delay that each tap in the delay chain adds depends on process, voltage, and temperature, and may vary among devices. Automatic delay calculation circuits can correct the differences. Using the natural value as the baseline value and shifting helps ensure that the algorithm is relatively independent of these conditions. The correct algorithm for adjusting the delay is as follows:

1. Read the natural value by reading L3DC.
2. Generate the override value by shifting the natural value by an appropriate amount.

3. Write the override value to L3DC.

The direction in which the register needs to be shifted depends upon the type of AC timing problem that is being corrected. As Figure 4 shows, the SRAM drives the L3_DATA and L3_ECHO_CLK pins simultaneously. To correct a problem related to insufficient input setup time, shifting L3DC right increases the clock delay, effectively improving the processor's input setup time margin. Note that this improvement causes a decreased output hold time margin. Similarly, shifting L3DC left improves input hold time margin at the cost of the input setup time margin.

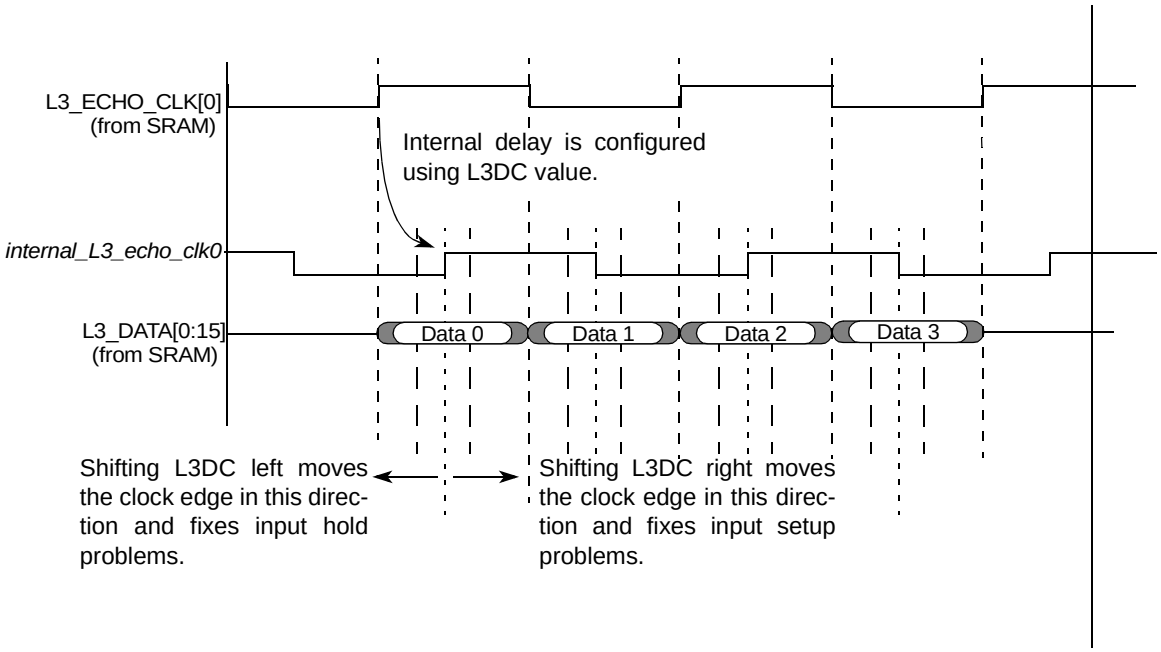

Figure 4. Shifting L3DC to Adjust Input AC Timing

Often, an AC timing problem during data read operations may be suspected without clearly indicating whether the problem is related to insufficient input setup or input hold time. In these cases, use trial-and-error to determine the optimal shift amount, as follows:

1. After enabling L3 clocks (but before enabling the L3 cache or private memory), read the value of L3DC*n* and use a right-shift/OR function shift in a '1' on L3DC*n*.
2. Set L3DCO*n*.
3. Check for proper L3 operation and note the outcome of tests.
4. Repeat steps 1-3, noting the outcome of each test iteration.
5. Eventually a shift amount where the L3 does not operate correctly is reached, indicating that the echo clocks were shifted too far, which causes the AC timing violation.
6. Return the register to its original value.
7. Read and shift L3DC*n* left by one bit (shift in '0' at the end).
8. Check for proper L3 operation.
9. Set L3DCO*n*.
10. Repeat steps 7–9, noting the outcome of each test iteration.

Eventually a shift amount is reached where the L3 does not operate correctly, which indicates that the echo clocks were shifted too far in the other direction. This problem causes the AC timing violation. When these steps are completed, a range of shift amounts for each L3ITCR register is produced to make the L3 interface operate properly. A shift value in the middle of each range should then be chosen for each register. In all cases, you should read and shift the value of the register rather than writing a hardcoded value because the natural values may vary from device to device and depend partly on operating conditions. We also recommend conducting tests across the entire range of expected operating conditions of the device in the system before determining the shift amounts that are ultimately used. Also, the override bit (L3DCO*n*) must be set to cause the processor to use the override value instead of the natural value.

NOTE

An improper override value can introduce AC timing problems and cause erratic behavior on the L3 interface.

3 Example Code

The following is an example of code for shifting the L3ITCR registers:

```
/* Adjust L3 input timing via L3ITCR registers
 * Here we hardcode the shift amount for each of the L3ITCR registers
 * --NOTE: Always shift--never write a hard-coded value to the registers! */
// L3ITCR0; found on all MPC745x devices
.equ     L3ITCR0_S,        TAP0_S            //shift amount
.equ     L3ITCR0_MS,       TAP0_MS           //mask start
.equ     L3ITCR0_ME,       TAP0_ME           //mask end
.equ     L3ITCR0_M,        TAP0_M            //OR mask value
// L3ITCR1; MPC7457-only register
.equ     L3ITCR1_S,        TAP0_S            //shift amount
.equ     L3ITCR1_MS,       TAP0_MS           //mask start
.equ     L3ITCR1_ME,       TAP0_ME           //mask end
.equ     L3ITCR1_M,        TAP0_M            //OR mask value
// L3ITCR2; MPC7457-only register
.equ     L3ITCR2_S,        TAP0_S            //shift amount
.equ     L3ITCR2_MS,       TAP0_MS           //mask start
.equ     L3ITCR2_ME,       TAP0_ME           //mask end
.equ     L3ITCR2_M,        TAP0_M            //OR mask value
// L3ITCR3; MPC7457-only register
.equ     L3ITCR3_S,        TAP0_S            //shift amount
.equ     L3ITCR3_MS,       TAP0_MS           //mask start
.equ     L3ITCR3_ME,       TAP0_ME           //mask end
```

```
.equ     L3ITCR3_M,        TAP0_M            //OR mask value
// Set up table of mask and shift values to use above
// add no taps (i.e. leave L3ITCRn at natural value)
.equ     TAP0_S            ,         0       //shift amount
.equ     TAP0_MS           ,         0       //mask start
.equ     TAP0_ME           ,         31      //mask end
.equ     TAP0_M            ,         0       //OR mask value
// add 2 taps
.equ     TAP2_S            ,         30
.equ     TAP2_MS           ,         2
.equ     TAP2_ME           ,         31
.equ     TAP2_M            ,         0xC000
// add 3 taps
.equ     TAP3_S            ,         29
.equ     TAP3_MS           ,         3
.equ     TAP3_ME           ,         31
.equ     TAP3_M            ,         0xE000
// add 4 taps
.equ     TAP4_S            ,         28
.equ     TAP4_MS           ,         4
.equ     TAP4_ME           ,         31
.equ     TAP4_M            ,         0xF000
// add 5 taps
.equ     TAP5_S            ,         27
.equ     TAP5_MS           ,         5
.equ     TAP5_ME           ,         31
.equ     TAP5_M            ,         0xF800
// and so on...
// subtract 1 tap
.equ     TAPn1_S           ,         1
.equ     TAPn1_MS ,        0
.equ     TAPn1_ME ,        30
.equ     TAPn1_M           ,         0x0
// subtract 2 taps
.equ     TAPn2_S           ,         2
.equ     TAPn2_MS ,        0
```

```
.equ     TAPn2_ME ,        29
.equ     TAPn2_M           ,        0x0
// subtract 3 taps
.equ     TAPn3_S           ,        3
.equ     TAPn3_MS ,        0
.equ     TAPn3_ME ,        28
.equ     TAPn3_M           ,        0x0
// subtract 4 taps
.equ     TAPn4_S           ,        4
.equ     TAPn4_MS ,        0
.equ     TAPn4_ME ,        27
.equ     TAPn4_M           ,        0x0
// and so on....
/*******************************
 * config_L3ITCR
 *     input: hard coded above
 *
 *     output: L3ITCR registers
*******************************/
config_L3ITCR:
// L3ITCR0 (all MPC745x devices, but note differences in MPC7457)
        sync
        mfspr   r5,984                      // Get L3ITCR0
        rlwinm  r5,r5,L3ITCR0_S,L3ITCR0_MS,L3ITCR0_ME                  // shift
        oris    r5,r5,L3ITCR0_M             // OR w/ mask to set bits shifted in
         // comment next instruction for MPC7450, MPC7451, and MPC7455
         // and uncomment for MPC7457
         ori     r5,r5,0x0001               // set override bit for MPC7457
         // comment next instruction for MPC7457
         // and uncomment for MPC7450, MPC7451, and MPC7455
//       ori     r5,r5,0x0080               //  set override bit
                                            //       for MPC7450, MPC7451, MPC7455
        mtspr   984,r5                      // write L3ITCR0
        isync
// L3ITCR1 (MPC7457 only)
```

```
        sync
        mfspr   r5,1001                     // Get L3ITCR1
        rlwinm  r5,r5,L3ITCR1_S,L3ITCR1_MS,L3ITCR1_ME                  // shift
        oris    r5,r5,L3ITCR1_M             // OR w/ mask to set bits shifted in
        ori     r5,r5,0x0001                // set override bit
        mtspr   1001,r5                     // write L3ITCR1
        isync
// L3ITCR2 (MPC7457 only)
        sync
         mfspr   r5,1002                    // Get L3ITCR2
        rlwinm  r5,r5,L3ITCR2_S,L3ITCR2_MS,L3ITCR2_ME                  // shift
        oris    r5,r5,L3ITCR2_M             // OR w/ mask to set bits shifted in
        ori     r5,r5,0x0001                // set override bit
        mtspr   1002,r5                     // write L3ITCR2
        isync
// L3ITCR3 (MPC7457 only)
        sync
        mfspr   r5,1003                     // Get L3ITCR3
        rlwinm  r5,r5,L3ITCR3_S,L3ITCR3_MS,L3ITCR3_ME                  // shift
        oris    r5,r5,L3ITCR_M              // OR w/ mask to set bits shifted in
        ori     r5,r5,0x0001                // set override bit
        mtspr   1003,r5                     // write L3ITCR3
        isync
```

4 References

The reference materials shown in Table 1 may be useful.

Table 1. Reference Documentation

Title	Author	Document
MPC7450 RISC Microprocessor Family User's Guide	—	Order #: MPC7450UM/D
Understanding the MPC7450-family L3 Cache Hardware Interface	Michael Everman	Order #: AN2468

5 Revision History

Table 2 shows the history of changes to this document.

Table 2. Document Revision History Table

Revision	Date	Substantive Changes
Rev 0	Dec. 2003	Initial release
Rev 1	Jan. 2007	Non-substantive formatting.

THIS PAGE INTENTIONALLY LEFT BLANK

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or
+1-480-768-2130
www.freescale.com/support

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)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064
Japan
0120 191014 or
+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 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: AN2580
Rev. 1
1/2007