Freescale Semiconductor
Addendum

Document Number: e500CORERMAD
Rev. 1.2, 09/2011

Errata to PowerPC™ e500 Core Family Reference Manual, Rev. 1

Supports
e500v1
e500v2

This errata document describes corrections to the *PowerPC™ e500 Core Family Reference Manual*, Revision 1. For convenience, the section number and page number of the errata item in the reference manual are provided. Items in bold are new since the last revision of this document.

To locate any published updates for this document, visit our website listed on the back cover of this document.

Section, Page No.	Changes
1.2, 1-5	Replace Table 1-1 with the following:

Table 1-1. Revision Level-to-Device Marking Cross-Reference

SoC Revision	Core Version	Core Revision	Processor Version Register (PVR)	System Version Register (SVR)
SoC-dependent value	e500v1	1.0	0x8020_0010	SoC-dependent value
SoC-dependent value	e500v1	2.0	0x8020_0020	SoC-dependent value
SoC-dependent value	e500v2	1.0	0x8021_0010	SoC-dependent value
SoC-dependent value	e500v2	2.0	0x8021_0020	SoC-dependent value

2.7.2.3, 2-22 A new register, MCARU, machine check address register upper, has been added to support 36-bit physical addressing in the e500v2 core. Replace the entire section with the following:

2.7.2.3 Machine Check Address Register (MCAR/MCARU)

When the core complex takes a machine check interrupt, it updates MCAR, shown in Figure 2-13, to indicate the address of the data associated with the machine check. Note that if a machine check interrupt is caused by a signal, MCAR contents are not meaningful. Errors that cause MCAR contents to be updated are implementation dependent.

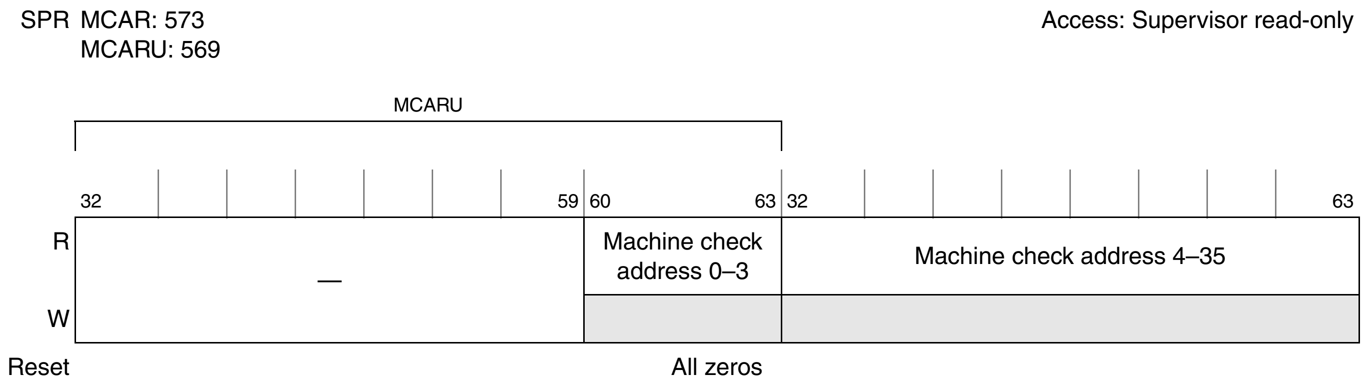

SPR MCAR: 573
MCARU: 569
Access: Supervisor read-only
MCARU
32
59
60
63
32
63
R
W
—
Machine check address 0–3
Machine check address 4–35
Reset
All zeros

Figure 2-13. Machine Check Address Register (MCAR/MCARU)

In the e500v2, MCARU[60–63] contain the 4 highest-order bits of the machine check address.

2.9.2, 2-25 The SPR of the branch buffer target address register should be 514. The register should appear in the manual as follows:

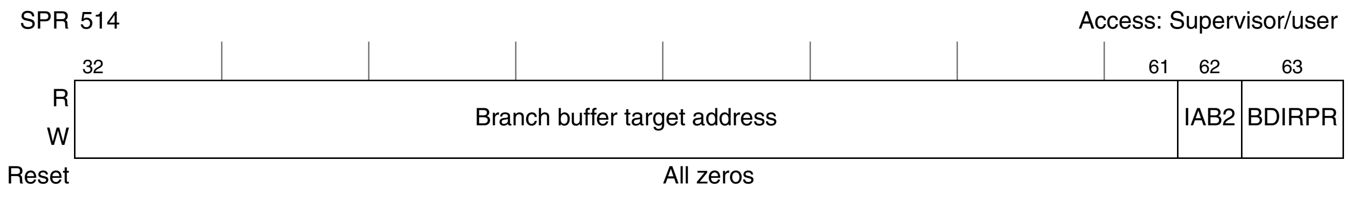


Figure 2-16. Branch Buffer Target Address Register (BBTAR)

Section, Page No.	Changes
2.10.2, 2-29	The size of HID1[PLL_CFG] is incorrectly shown. Replace the register figure and the affected rows of the field descriptions as follows:

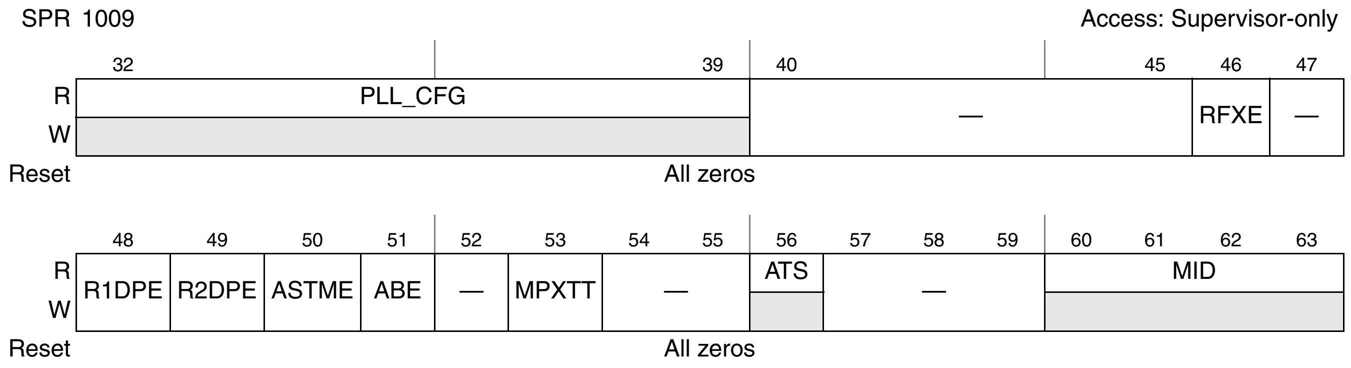


Figure 2-19. Hardware Implementation-Dependent Register 1 (HID1)

Table 2-15. HID1 Field Descriptions

Bits	Name	Description
32–39	PLL_CFG	Reflected directly from the PLL_CFG input pins (read-only)
40–45	—	Reserved, should be cleared.
. . .		

Section, Page No.	Changes
2.10.2, 2-29	In Table 2-15, revise the description of HID1[RFXE] as follows:

46	RFXE	Read fault exception enable. Enables the core to internally generate a machine check interrupt when *core_fault_in* is asserted. Depending on the value of MSR[ME], this results in either a machine check interrupt or a checkstop. 0 Assertion of *core_fault_in* cannot cause a machine check. The core does not execute any instructions from a faulty instruction fetch and does not execute any load instructions that get their data from a faulty data fetch. On the e500v2, if these instructions are eventually required by the sequential programming model (that is, they are not in a speculative execution path), the e500v2 stalls until an asynchronous interrupt is taken. The e500v1 does not stall when faulty instructions or data are received, as described in the following note. **Note:** The e500v1 does not stall when faulty instructions or data are received. Instead, it continues processing with faulty instructions or data. The only reliable way to prevent such behavior is to set RFXE, which causes a machine check before the faulty instructions or data are used. To avoid the use of faulty instructions or data and to have good error determination, software must set RFXE and program the PIC to interrupt the processor when errors occur. As a result, software must deal with multiple interrupts for the same fundamental problem. 1 Assertion of *core_fault_in* causes a machine check if MSR[ME] = 1 or a checkstop if MSR[ME] = 0. The *core_fault_in* signal is asserted to the core when logic outside of the core has a problem delivering good data to the core. For example, the front-side L2 cache asserts *core_fault_in* when an ECC error occurs and ECC is enabled. As a second example, it is asserted when there is a master abort on a PCI transaction. See Section 13.8, "Proper Reporting of Bus Faults." The RFXE bit provides flexibility in error recovery. Typically, devices outside of the core have some way other than the assertion of *core_fault_in* to signal the core that an error occurred. Usually, this is done by channeling interrupt requests through a programmable interrupt controller (PIC) to the core. In these cases, the assertion of *core_fault_in* is used only to prevent the core from using bad data before receiving an interrupt from the PIC (for example, an external or critical input interrupt). Possible combinations of RFXE and PIC configuration are as follows: • RFXE = 0 and the PIC is configured to interrupt the processor. In this configuration, the assertion of *core_fault_in* does not trigger a machine check interrupt. The core does not use the faulty instructions or data and may stall. The PIC interrupts the core so that error recovery can begin. This configuration allows the core to query the PIC and the rest of the system for more information about the cause of the interrupt, and generally provides the best error recovery capabilities. • RFXE = 1 and the PIC is not configured to interrupt the processor. This configuration provides quick error detection without the overhead of configuring the PIC. When the PIC is not configured, setting RFXE avoids stalling the core when *core_fault_in* is asserted. Determination of the root cause of the problem may be somewhat more difficult than it would be if the PIC were enabled. • RFXE = 1 and the PIC is configured to interrupt the processor. In this configuration, the core may receive two interrupts for the same fundamental error. The two interrupts may occur in any order, which may complicate error handling. Therefore, this is usually not an interesting configuration for a single-core device. This may, however, be an interesting configuration for multi-core devices in which the PIC may steer interrupts to a processor other than the one that attempted to fetch the faulty data. • RFXE = 0 and the PIC is not configured to interrupt the processor. This is not a recommended configuration. The processor may stall indefinitely due to an unreported error.

2.12.1, 2-36 In the PID*n* registers, revise the Process ID field of the register as follows:

SPR 48 (PID0: PID in Book E);
SPR 633 (PID1: e500-specific);
SPR 634 (PID2: e500-specific)

Access: Supervisor-only

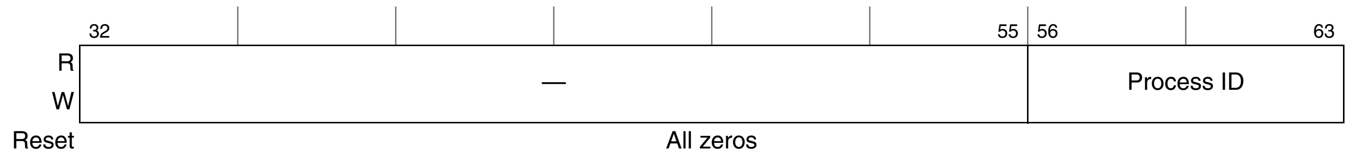
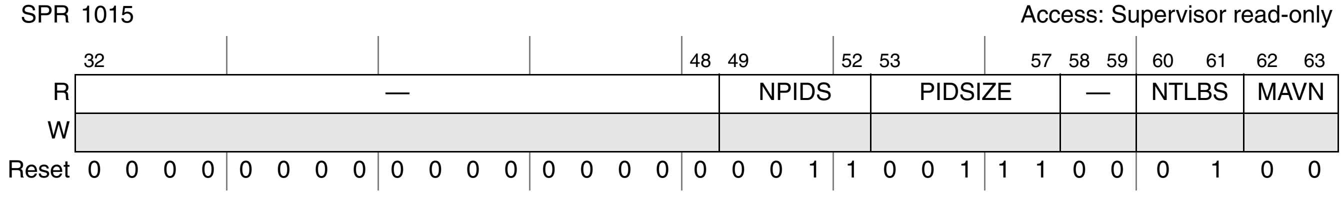


Figure 2-24. Process ID Registers (PID0–PID2)

Section, Page No.	Changes
2.12.3, 2-37	In the MMU configuration register (MMUCFG), update the reset value for the NPIDS field to 0011. The register should appear as follows:



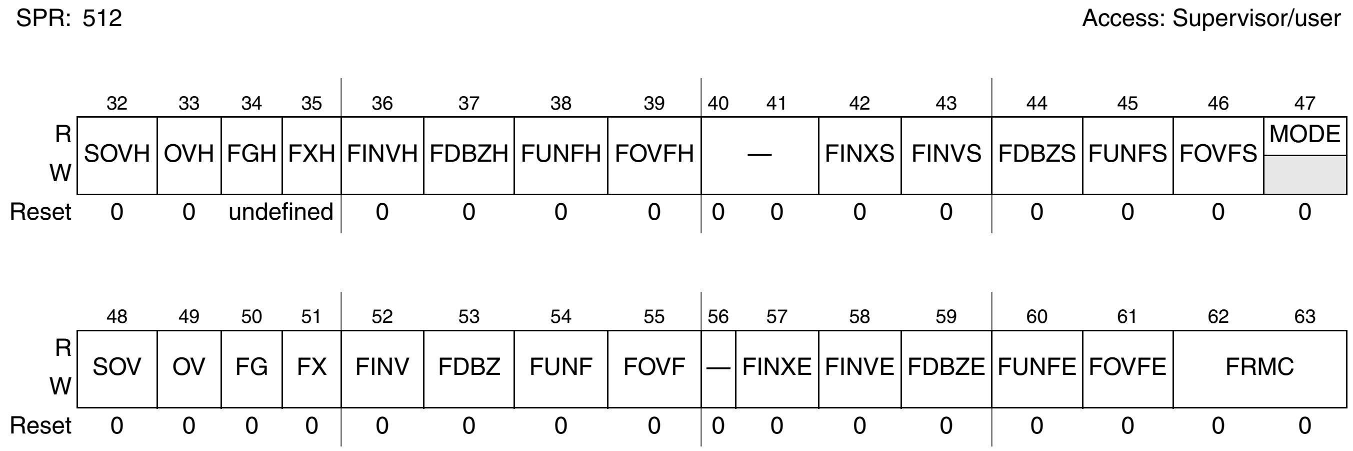
SPR 1015
Access: Supervisor read-only
32
48 49
52 53
57 58 59
60 61
62 63
R
W
—
NPIDS
PIDSIZE
—
NTLBS
MAVN
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 1 0 0

Figure 2-26. MMU Configuration Register (MMUCFG)

Section, Page No.	Changes
2.13.1.1, 2-46	In Table 2-31, add the DBCR0[EDM] field description, as follows:

Bits	Name	Description
32	EDM	External debug mode. Indicates whether the processor is in external debug mode. (Read only by software.) 0 The processor is not in external debug mode. 1 The processor is in external debug mode. An external debug agent has locked the following debug registers: DBCR0–DCBR2, DAC1–DAC2, and IAC1–IAC2.

Section, Page No.	Changes
2.14.1, 2-50	The signal processing and embedded floating-point status and control register (SPEFSCR) is shown as supervisor only. The register is user accessible. The register should appear as follows:



SPR: 512
Access: Supervisor/user
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
R
W
SOVH OVH FGH FXH FINVH FDBZH FUNFH FOVFH — FINXS FINVS FDBZS FUNFS FOVFS MODE
Reset 0 0 undefined 0 0 0 0 0 0 0 0 0 0 0 0
48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
R
W
SOV OV FG FX FINV FDBZ FUNF FOVF — FINXE FINVE FDBZE FUNFE FOVFE FRMC
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 2-38. Signal Processing and Embedded Floating-Point Status and Control Register (SPEFSCR)

Section, Page No.	Changes
3.8.1.4, 3-58	Add the following notes to Table 3-36, “SPE APU Vector Instructions.” The affected rows and the notes should appear as follows:

Table 3-36. SPE APU Vector Instructions

Instruction	Mnemonic	Syntax
. . .		
Vector Multiply Word Low Signed, Modulo, Integer and Accumulate in Words [1]	**evmwlsmiaaw**	**r**D,**r**A,**r**B
Vector Multiply Word Low Signed, Modulo, Integer and Accumulate Negative in Words [1]	**evmwlsmianw**	**r**D,**r**A,**r**B
Vector Multiply Word Low Signed, Saturate, Integer and Accumulate in Words [1]	**evmwlssiaaw**	**r**D,**r**A,**r**B
Vector Multiply Word Low Signed, Saturate, Integer and Accumulate Negative in Words [1]	**evmwlssianw**	**r**D,**r**A,**r**B
. . .		
Vector Multiply Word Low Unsigned, Modulo, Integer and Accumulate in Words [1]	**evmwlumiaaw**	**r**D,**r**A,**r**B
Vector Multiply Word Low Unsigned, Modulo, Integer and Accumulate Negative in Words [1]	**evmwlumianw**	**r**D,**r**A,**r**B
Vector Multiply Word Low Unsigned, Saturate, Integer and Accumulate in Words [1]	**evmwlusiaaw**	**r**D,**r**A,**r**B
Vector Multiply Word Low Unsigned, Saturate, Integer and Accumulate Negative in Words [1]	**evmwlusianw**	**r**D,**r**A,**r**B
. . .		
Vector Multiply Word Signed, Saturate, Fractional [2]	**evmwssf**	**r**D,**r**A,**r**B
Vector Multiply Word Signed, Saturate, Fractional and Accumulate [2]	**evmwssfa**	**r**D,**r**A,**r**B
Vector Multiply Word Signed, Saturate, Fractional and Accumulate [2, 3]	**evmwssfaa**	**r**D,**r**A,**r**B
Vector Multiply Word Signed, Saturate, Fractional and Accumulate Negative [2, 3]	**evmwssfan**	**r**D,**r**A,**r**B
. . .		

[1] On the e500, if the intermediate product cannot be represented in 32 bits, the final result is undefined. The e500 sets status bits indicating an overflow.

[2] The architecture specifies that if the final result cannot be represented in 64 bits, SPEFSCR[OV] should be set (along with the SOV bit, if it is not already set). The e500 violates the architectural specification for these instructions because it sets the overflow bit in cases where there is no overflow.

[3] Although the e500 records any overflow resulting from the addition/subtraction portion of these instructions, a saturate value is not saved to **r**D or the accumulator. The architecture specifies that the intermediate result should be saturated if it cannot be represented in 64 bits. It also specifies that the final result should be saturated if it cannot be represented in 64 bits. The e500 does not saturate in either case.

3.8.1.4, 3-59	Immediately before Table 3-37, add the following note:

NOTE

The e500 does not generate NaNs or infinities as the result of computations involving normalized or denormalized floating-point numbers. Such values would have to be generated by using non–floating-point instructions.

Section, Page No.	Changes
3.8.1.4, 3-60	To Table 3-37, "Vector and Scalar SPFP APU Floating-Point Instructions," add notes 2 and 3. The affected rows and the notes should appear as follows:

Table 3-37. Vector and Scalar Floating-Point APU Instructions

Instruction	Single-Precision Scalar	Double-Precision Scalar (e500v2)	Vector	Syntax
Floating-Point Compare Equal [2]	**efscmpeq**	**efdcmpeq**	**evfscmpeq**	**crD**,**rA**,**rB**
Floating-Point Compare Greater Than [2]	**efscmpgt**	**efdcmpgt**	**evfscmpgt**	**crD**,**rA**,**rB**
Floating-Point Compare Less Than [2]	**efscmplt**	**efdcmplt**	**evfscmplt**	**crD**,**rA**,**rB**
Floating-Point Subtract [3]	**efssub**	**efdsub**	**evfssub**	**rD**,**rA**,**rB**
. . .				

Note: On the e500v1, floating-point operations that produce a result of zero may generate an incorrect sign.

[1] Exception detection for these instructions is implementation dependent. On the e500, Infinities, NaNs, and Denorms are always treated as Norms. No exceptions are taken if SPEFSCR[FINVE] = 1.

[2] The e500 indicates that the result of a floating-point compare of zero with a denormalized number is EQ (equal) rather than GT (greater than) or less than (LT) as defined by the architecture.

[3] When subtracting a single-precision normalized number from a negative NaN. the e500 does not saturate the result.

3.8.8, 3-60 — Replace Table 3-40 with the following two tables (note the new introductory text):

The Freescale Book E implementation standards define a set of register resources used exclusively by the performance monitor. PMRs are similar to the SPRs defined in the Book E architecture and are accessed by **mtpmr** and **mfpmr**, which are also defined by the EIS. Table 3-40 lists supervisor-level PMRs. User-level software that attempts to read or write supervisor-level PMRs causes a privilege exception.

Table 3-40. Performance Monitor Registers—Supervisor Level

Abbreviation	Register Name	PMR Number	pmr[0–4]	pmr[5–9]	Section/Page
PMGC0	Performance monitor global control register 0	400	01100	10000	2.15.1/2-64
PMLCa0	Performance monitor local control a0	144	00100	10000	2.15.3/2-66
PMLCa1	Performance monitor local control a1	145	00100	10001	
PMLCa2	Performance monitor local control a2	146	00100	10010	
PMLCa3	Performance monitor local control a3	147	00100	10011	
PMLCb0	Performance monitor local control b0	272	01000	10000	2.15.5/2-67
PMLCb1	Performance monitor local control b1	273	01000	10001	
PMLCb2	Performance monitor local control b2	274	01000	10010	
PMLCb3	Performance monitor local control b3	275	01000	10011	
PMC0	Performance monitor counter 0	16	00000	10000	2.15.7/2-68
PMC1	Performance monitor counter 1	17	00000	10001	
PMC2	Performance monitor counter 2	18	00000	10010	
PMC3	Performance monitor counter 3	19	00000	10011	

User-level PMRs in Table 3-41 are read-only and are accessed with **mfpmr**. Attempting to write user-level registers in supervisor or user mode causes an illegal instruction exception.

Table 3-41. Performance Monitor Registers—User Level (Read-Only)

Abbreviation	Register Name	PMR Number	pmr[0–4]	pmr[5–9]	Section/Page
UPMGC0	User performance monitor global control register 0	384	01100	00000	2.15.2/2-65
UPMLCa0	User performance monitor local control a0	128	00100	00000	2.15.4/2-67
UPMLCa1	User performance monitor local control a1	129	00100	00001	
UPMLCa2	User performance monitor local control a2	130	00100	00010	
UPMLCa3	User performance monitor local control a3	131	00100	00011	
UPMLCb0	User performance monitor local control b0	256	01000	00000	2.15.6/2-68
UPMLCb1	User performance monitor local control b1	257	01000	00001	
UPMLCb2	User performance monitor local control b2	258	01000	00010	
UPMLCb3	User performance monitor local control b3	259	01000	00011	
UPMC0	User performance monitor counter 0	0	00000	00000	2.15.8/2-69
UPMC1	User performance monitor counter 1	1	00000	00001	
UPMC2	User performance monitor counter 2	2	00000	00010	
UPMC3	User performance monitor counter 3	3	00000	00011	

11.1, 11-1 Replace the bulleted statement about instruction cache parity with the following:

- Both L1 caches support parity generation and checking (enabled through L1CSR0 and L1CSR1 bits), as follows:
  - — Instruction cache: 1 parity bit per word of instruction
  - — Data cache: 1 parity bit per byte of data

11.2.2, 11-7 Following Figure 11-3, “L1 Instruction Cache Organization,” replace the statement about instruction cache parity with the following:

Each block consists of eight instructions, 1 status bit, 1 lock bit, and an address tag. Also, although it is not shown in Figure 11-3, the instruction cache has 1 parity bit/word; 8 parity bits for each line.

D.1, D-14 In Table D-1, remove instructions *evmwhusiaaw* and *evmwhusianw.*

D.3, D-46 In Table D-3, remove instructions *evmwhusiaaw* and *evmwhusianw.*

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 China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
1-800 441-2447 or
+1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Document Number: e500CORERMAD
Rev. 1.2
09/2011