

Freescale Semiconductor

MPC8xx Exception Processing



Exception Terms

User Mode The Privilege Level that Applications run in.

Supervisor Mode The Privilege Level that the Operating System

runs in. Also called "Privileged Mode"

Exception An event which causes deviation from normal

processing.

Examples:

- Interrupt (internal or external)

- Resets

- Bus error

Ordered Exception No program state is lost after the exception (the

machine state is saved).

Unordered Exception Program state may be lost after the exception.

Includes reset, machine check and other non-

maskable exceptions.

Asynchronous Exception Exception not caused by an instruction.

Synchronous Exception Exception caused by an instruction.

Precise Exception The exact processor context when the exception

occurred is available, and the exact cause of the

exception is always known.

Processor backs the machine up to the

instruction which caused the exception

The exact processor context is not known when

the exception is processed, because concurrent operations have affected the information that

comprises the processor context.

Maskable Exception May be masked by the operating system.

Imprecise Exception



Exception Classes

CLASS	EXCEPTION TYPE
ASYNCHRONOUS, UNORDERED	RESET, NON-MASKABLE
SYNCHRONOUS, UNORDERED	MACHINE CHECK (BUS ERROR)
ASYNCHRONOUS, ORDERED	EXTERNAL INTERRUPT DECREMENTER, PIT INTERRUPTS
SYNCHRONOUS (ORDERED, PRECISE)	INSTRUCTION -CAUSED EXCEPTIONS

ORDERED EXCEPTIONS - When the Exception is taken, No program state is lost.

UNORDERED EXCEPTIONS - When the Exception is taken, the program state is unrecoverable.

Reset and **Machine Check** Exceptions are unrecoverable, if they occur during the servicing of another exception.

PRECISE EXCEPTIONS - When the exception is taken, the processor backs the machine up to the instruction causing the exception. The instruction causing the exception may not have begun execution, may partially be completed, or may have completed execution.

The Core implements all storage associated interrupts as precise interrupts. This means that a load/store instruction is not complete until all possible error indications have been sampled from the Load/Store Bus.



Exception Definitions (1 of 2)

System Reset Interrupt	Occurs when the NMI pin is asserted, SWT times out, Hard or Soft Reset pins are asserted
Machine Check Interrupt	The accessed address does not exist or a data error was detected
Data Storage Interrupt	Never generated by the hardware The software may branch to this location as a result of either
	Implementation Specific Data TLB error interrupt or Implementation Specific Data TLB miss int.
Instruction Storage Interrupt	Never generated by the hardware The software may branch to this location as a result of an Implementation Specific Instruction TLB error interrupt
Alignment Interrupt	Occurs as a result of one of the following cases: The operand of a Floating-Point load or store is not word aligned. The operand of Load/Store multiple is not word aligned.
	The operand of lwarx or stwcx. is not word aligned. The operand of Load/Store individual scalar instruction is not naturally aligned when $MSR_{LE} = 1$. An attempt to execute multiple/string instruction is made when $MSR_{LE} = 1$
Program Interrupt	Floating-Point Enabled Exception type Program interrupt is not generated by the MPC8xx core.
	Illegal Instruction type program interrupt is not generated by the Core, an Implementation Dependent Software Emulation Interrupt is generated instead
	Privileged instruction type Program interrupt is generated for on core valid SPR field or any SPR encoded as an external to the core special register if spr ₀ =1 and MSR _{PR} =1, as well as an attempt to execute privileged instruction when MSR _{PR} =1.
Floating Point Unavailable Interrupt	Not generated by the MPC8xx core. An Implementation Dependent Software Emulation Interrupt will be taken on any attempt to execute Floating-Point instruction regardless of MSR _{FP}



Exception Definitions (2 of 2)

_	
Trace Interrupt	Occurs If $MSR_{SE} = 1$ and any instruction except rfi is successfully completed, or $MSR_{BE} = 1$ and a branch is completed
Floating Point Assist Interrupt	Not generated by the MPC8xx An Implementation Dependent Software Emulation Interrupt will be taken on any attempt to execute Floating-Point instruction
Implementation Dependent	Occurs as a result of one of the following cases:
Software Emulation Interrupt	•When there is an attempt to execute any non implemented
	instruction. (This include all Illegal and unimplemented optional instructions and all floating point instructions).
	•When there is an attempt to execute a mtspr or mfspr which specifies on core non implemented register. (regardless of spr ₀).
	•When there is an attempt to execute a mtspr or mfspr which specifies off core non implemented register and spr ₀ =0 or MSR _{PR} =0
Implementation Specific	Occurs when MSR _{IR} =1 and there is an attempt to fetch an instruction
Instruction TLB Miss Interrupt	from a page that its Effective Page Number can not be translated by
	the Instruction TLB
Implementation Specific	Occurs in the following cases:
Instruction TLB Error Interrupt	•The effective address cannot be translated (either Segment valid bit or
	Page valid bit of this page are cleared in the translation table)
	The fetch access violates storage protection The fetch access is to Guarded storage and MSR _{IR} =1
Implementation Specific	Occurs when MSR _{DR} =1 and there is an attempt to access a page
Data TLB Miss Interrupt	that its Effective Page Number can not be translated by the Data TLB
Implementation Specific	Occurs in the following cases:
Data TLB Error Interrupt	•The offective address of a load Store iski daha dahat dahat
	The effective address of a Load, Store, icbi, dcbz, dcbst, dcbf or dcbi instruction cannot be translated (either Segment valid bit or Page valid bit of this page are cleared in the translation table)
	•The access violates the storage protection
	•An attempt to write to a page with negated Change Bit



Interrupt Priority Mapping

#	Interrupt Type	Caused By
#1	Development non-maskable interrupt	Signal from the Development Port
#2	System reset	NMI_L assertion
#3	Instruction Related Interrupts	Instruction Processing
#4	Peripheral breakpoint request or Dev Port maskable interrupt	Breakpoint signal from any peripheral
#5	External Interrupt	Signal from the interrupt controller
#6	Decrementer interrupt	Decrementer request

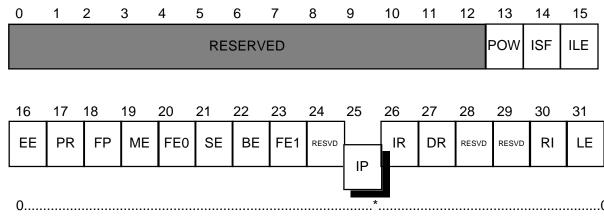
Instruction Related Interrupt Detection Order

#	Interrupt Type	Caused by
#1	Trace	Trace bit asserted
#2	Implementation Dependent Instruction TLB miss	Instruction MMU TLB Miss
#3	Implementation Dependent Instruction TLB error	Instruction MMU protection/translation error
#4	Machine Check Interrupt	Fetch Error
#5	Debug I- Breakpoint	Match detection
#6	Implementation Dependent Software Emulation Interrupt	Attempt to invoke un-implemented feature
#7	Floating-Point Unavailable	Attempt is made to execute Floating-Point instruction and MSRFP=0
#8	Privileged Instruction	Attempt to execute priviledged instruction in problem mode
	Alignment Interrupt	Load store checking
	System Call Interrupt	SC Instruction
	Trap	Trap Instruction
#9	Implementation Dependent Data TLB miss	Data MMU TLB Miss
#10	Implementation Dependent Data TLB error	Dat MMU TLB Protection/translation error
#11	Machine Check Interrupt	Load or store access error
#12	Debug -L Breakpoint	Match detection



Vector Table

MSR - MACHINE STATE REGISTER



VECTOR OFFSET (HEX)		= 0 Vector Table Address at 0x00000000 = 1 Vector Table Address at 0xFFF00000.
0 0000	RESERVED	
0 0100	SYSTEM RESET	HARD & SRESETS
0 0200	MACHINE CHECK	TEA (BUS ERROR)
0 0300	DATA STORAGE	
0 0400	INSTRUCTION STORAGE	
0 0500	EXTERNAL INTERRUPT	IRQ[1:7], PIT, TB, RTC, PCMCIA, CPM
0 0600	ALIGNMENT	ALIGNMENT ERROR
0 0700	PROGRAM	INSTR. TRAPS,ERRORS, ILLEGAL, PRIVILEGED MSR[FP]=0 & F.P. INSTRUCTION ENCOUNTERED
0 0800	FLOATING-POINT UNAVAILIABLE	month lead in the meaning the second state of
0 0900	DECREMENTER	DECREMENTER REGISTER
0 0A00	RESERVED	_
0 0B00	RESERVED	
0 0000	SYSTEM CALL	'SC' INSTRUCTION
0 0D00	TRACE*	SINGLE-STEP OR BRANCH TRACING
0 0E00	FLOATING-POINT ASSIST*	SOFTWARE ASSIST FOR INFREQUENT & COMPLEX FP OPERATIONS
0 1000	IMPLEMENTATION DEPENDENT SOFTWARE EMULATION	
0 1100	IMPLEMENTATION DEPENDENT INSTRUCTION TLB MISS	
0 1200	IMPLEMENTATION DEPENDENT DATA TLB MISS	
0 1300	IMPLEMENTATION DEPENDENT INSTRUCTION TLB ERROR	
0 1400	IMPLEMENTATION DEPENDENT DATA TLB ERROR	
0 1500 - 01BFF	RESERVED	
0 1C00	IMPLEMENTATION DEPENDENT DATA BREAKPOINT	
0 1D00	IMPLEMENTATION DEPENDENT INSTRUCTION BREAKPOINT	
0 1E00	IMPLEMENTATION DEPENDENT PERIPHERAL BREAKPOINT	
0 1F00	IMPLEMENTATION DEPENDENT NON MASKABLE DEVELOPMENT PORT	



MSR After Hard Reset

MSR - MACHINE STATE REGISTER

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
					RES	SERVI	ΞD					F	POW	ISF	ILE
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
EE	PR	FP	ME	FE0	SE	BE	FE1	RESVD	IP	IR	DR	RESVD	RESVD	RI	LE

MSR AFTER RESET:

DOW	0	Decree Management Disable
POW	0	Power Management Disable
ISF	0	Implementation Specific Function
ILE	0	Interrupt Little Endian Mode Disabled
EE	0	External and DEC Interrupt are disabled
PR	0	Privilege Level is Supervisor.
FP	0	Floating Point Unit not available
ME	0	Machine Check Disabled: If transfer error acknowledge (<i>TEA</i>) occurs, the Chip will go to Checkstop State. The SIU may assert <i>reset</i> in order to recover.
FE0	0	Floating-Point Exception Mode 0(has no effect).
SE	0	Single Step Trace Disabled.
BE	0	Branch Trace Disabled.
DL	U	Dianch Trace Disabled.
FE1	0	Floating-Point Exception Mode 1(has no effect).
IP	*	Interrupt Prefix . Vector Table Located
		at 0x000n - nnnn or at 0xFFFn - nnnn for a value
		of a "0" or a "1" respectively.
IR	0	Instruction Relocate
DR	0	Data Relocate
RI	0	Recoverable Interrupt Mode is Disabled.
	_	

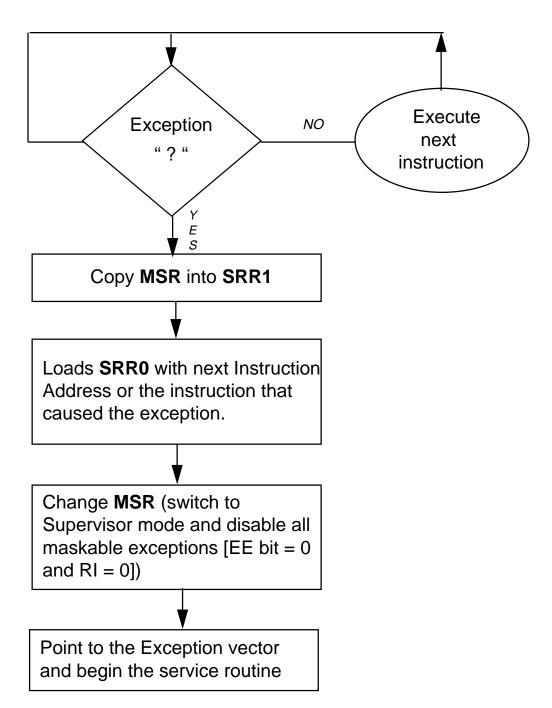
0

LE

Normal Processing is set for Big Endian Mode.



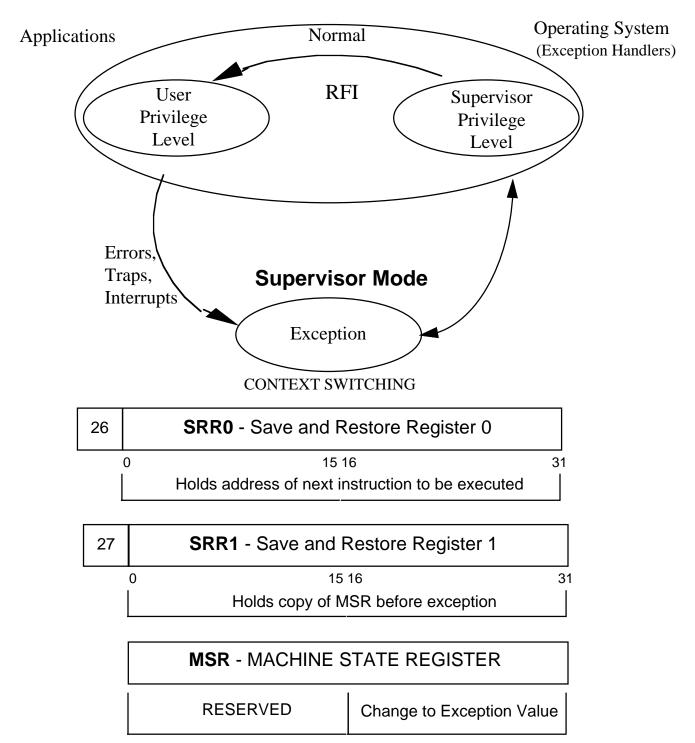
Exception Processing Sequence (1 of 2)



- For most exceptions, the machine state is saved only in SRR0 and SRR1.
- Some exceptions will save other information in DSISR and DAR:
 - **DSISR** (Source Instruction Service Register) 7 bit field identifies which instruction caused the exception.
 - DAR (Data Address Register): Contains the effective address o the load or store for misaligned exceptions.



Exception Processing Sequence (2 of 2)



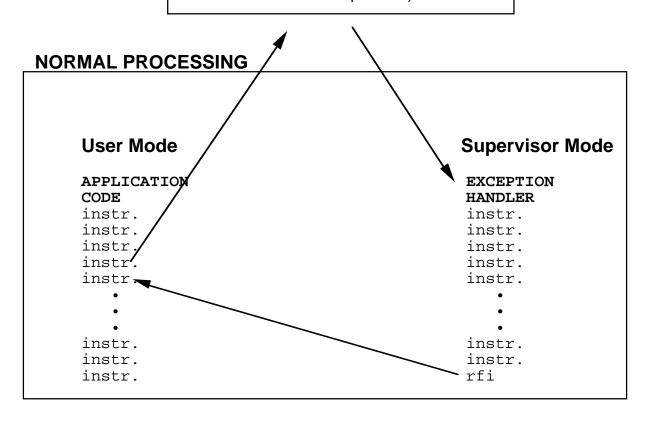
- SRR0, SRR1 and MSR are changed after every exception
- All exceptions cause the core to enter the supervisor mode.
- The RFI instruction restores the Machine State back to User Mode.
- The RFI instruction is usually the last instruction in the exception handler.

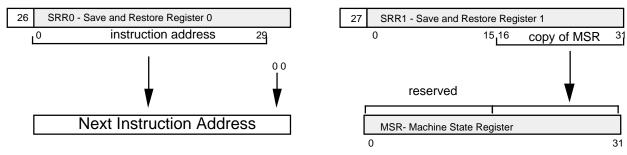


How the RFI Instruction Operates

EXCEPTION PROCESSING

- 1. HW Saves Instr. Addr to SRR0
- 2. HW Saves MSR to SRR1
- **3. HW changes MSR** (change to Supervisor Mode, mask other maskable exceptions...)





RFI Instruction is Supervisor-only used to restore previous Machine State.



How to Make the ESR Recoverable

```
asm (" stwu r9,-12(r1);
                                                    /* SAVE R9
 Making the
            asm (" mfspr r9,26");
                                                    /* PUSH SRR0 ONTO STACK */
      ESR
            asm (" stw r9,4(r1)");
Recoverable
            asm (" mfspr r9,27");
                                                    /* PUSH SRR1 ONTO STACK */
            asm (" stw r9,8(r1)");
            asm (" mtspr 80,0");
                                                 /* ENABLE INTERRUPTS
                                                                                * /
            asm (" mtspr 82,0");
                                                 /* MAKE NON-RECOVERABLE
                                                                                * /
Before ESR
            asm (" lwz r9,8(r1)");
                                                 /* PULL SRR1 FROM STACK
                                                                                * /
     Exit
            asm (" mtspr 27,r9");
            asm (" lwz r9,4(r1)");
                                                 /* PULL SRR0 FROM STACK
            asm (" mtspr 26,r9");
            asm (" lwz r9,0(r1)");
                                                 /* PULL R9 FROM STACK
            asm (" addi r1, r1, 12");
                  2
                       3
                            4
                                 5
                                           7
                                                8
                                                    9
                                                                          14
                                                                               15
         0
             1
                                      6
                                                         10
                                                              11 12
                                                                     13
                                                                     POW
                                                                          ISF
                                 RESERVED
                                                                               ILE
        16
             17
                  18
                       19
                            20
                                 21
                                     22
                                         23
                                              24
                                                   25
                                                        26
                                                             27
                                                                 28
                                                                      29
                                                                          30
                                                                               31
             PR
                  FΡ
                      ME
                           FE<sub>0</sub>
                                SE
                                     ΒE
                                         FE1
                                                        IR
                                                            DR
                                                                               LE
                                              RESVD
                                                   IΡ
                                                                 RESVD
                                                                     RESVD
        EE
                                                                           RΙ
```

Mnemonic	MSR _{EE}	MSR _{RI}	Used For
EIE (80)	1	1	External Interrupt Enable
EID (81)	0	1	External Interrupt Disable, but other interrupts are recoverable
NRI(82)	0	0	Non-Recoverable Interrupt



Freescale Technical Training - MPC860 Course Phoenix, Arizona

Title: ex1.c

Handling a System Call Exception

<u>Creation Date:</u> Jan. 10, 1996 <u>From:</u> 68360 Course

AuthorBob Bra tt

Description:

The results of this routine are:

- 1. Initializes the exception vector area with a service routine to increment an LED counter each time a system call instruction is executed.
- 2. The exception service routine is made recoverable. Assumptions:
- 1. Reset conditions exist.

Objective:

If the program executes properly, the LED counter has a count of 1.

Equipment:

MPC860ADS board and UDLP1.

UDLP1 Switch Settings: N/A

Connections: MPC860ADS board and UDLP1 are connected at P13.

<u>Updates:</u>



ex1.c (1 of 2)

```
/* (EX1.C)
                                /* DUAL PORT RAM EQUATES*/
#include "mpc860.h"
struct dprbase *pdpr;
                                  /* PNTR TO DUAL PORT RAM*/
main()
                                   /* EXCEPTION SERVICE RTN */
  void esr();
   int *ptrs,*ptrd;
                                   /* SOURCE & DEST POINTERS*/
  pdpr = (struct dprbase *) (getimmr() & 0xFFFF0000);
                                   /* INIT PNTR TO DPRBASE
  ptrs = (int *) esr;
                                   /* INIT SOURCE POINTER
                                                            * /
  ptrd = (int *)(getevt() + 0xC00); /* INIT DEST POINTER
                                                            * /
                                                            * /
                                   /* MOVE ESR TO EVT
      *ptrd++ = *ptrs;
                                   /* MOVE UNTIL
   /* CLEAR PORT D DATA REG */
   pdpr->PDDAT = 0;
                                  /* MAKE PORT D8-15 OUTPUT*/
  pdpr->PDDIR = 0xff;
   asm(" sc");
                                   /* SYSTEM CALL
#pragma interrupt esr
void esr()
   asm (" stwu r9,-12(r1)");
                                 /* PUSH GPR9 ONTO STACK
                                                            * /
                                 /* PUSH SRRO ONTO STACK
  asm (" mfspr r9,26");
   asm (" stw r9,4(r1)");
   asm (" mfspr r9,27");
                                 /* PUSH SRR1 ONTO STACK
   asm (" stw r9,8(r1)");
                                 /* ENABLE INTERRUPTS
  asm (" mtspr 80,0");
  pdpr->PDDAT += 1;
                                 /* MAKE NON-RECOVERABLE
  asm (" mtspr 82,0");
                                 /* PULL SRR1 FROM STACK
  asm (" lwz r9,8(r1)");
                                                            * /
  asm (" mtspr 27,r9");
                               /* PULL SRR0 FROM STACK
  asm (" lwz r9,4(r1)");
  asm (" mtspr 26,r9");
asm (" lwz r9,0(r1)");
asm (" addi r1,r1,12");
                                /* PULL GPR9 FROM STACK
                                 /* RESTORE STACK POINTER
}
getimmr()
   asm(" mfspr 3,638");
getevt()
                                   /* GET EVT LOCATION
                                   /* IF MSR.IP IS 0
   if ((getmsr() \& 0x40) == 0)
                                   /* THEN EVT IS IN LOW MEM*/
     return (0);
                                                            * /
   else
                                   /* ELSE
      return (0xFFF00000);
                                   /* EVT IS IN HIGH MEM
}
```



ex1.c (2 of 2)

Freescale Technical Training - MPC860 Course Phoenix, Arizona
Title:
——— ex2.c Handling a Alignment Error Exception
Creation Date: Jan. 10, 1996 From: 68360 Course
Author: Bob Bratt
Description:
The results of this routine are: 1. Initializes the exception vector area with a service routine to increment an LED counter each time an alignment error occurs. 2. The exception service routine is made recoverable. Assumptions: 1. Reset conditions exist.
Objective:
If the program executes properly, the LED counter contains a random count.
Equipment:
MPC860ADS board and a UDLP1.
UDLP1 Switch Settings: N/A
Connections: MPC860ADS board and a UDLP1 are connected through P13.
Updates:



ex2.c (1 of 2)

```
/* (EX2.C)
main()
                                        /* EXCEPTION SERVICE RTN */
   void esr();
   int *ptrs,*ptrd;
                                        /* SOURCE & DEST POINTERS*/
   pdpr = (struct dprbase *) (getimmr() & 0xFFFF0000);
                                         /* INIT PNTR TO DPRBASE */
   ptrs = (int *) esr;
                                         /* INIT SOURCE POINTER
                                                                      * /
   ptrd = (int *)(getevt() + 0x600); /* INIT DEST POINTER
                                                                      * /
                                         /* MOVE ESR TO EVT
                                                                      * /
       *ptrd++ = *ptrs;
                                         /* MOVE UNTIL
                                                                      * /
   while (*ptrs++ != 0x4c000064); /* RFI INTRUCTION
                                                                * /
   pdpr->PDDAT = 0; /* CLEAR PORT D DATA REG */
pdpr->PDDIR = 0xff; /* MAKE PORT D8-15 OUTPUT*/
asm(" li r21,0x1001"); /* INIT r21 TO UNALIGNED */
asm(" lwarx r20,r0,r21"); /* ALIGNMENT INTERRUPT */
#pragma interrupt esr
void esr()
   asm (" stwu r9,-12(r1)");
asm (" mfspr r9,26");
                                     /* PUSH GPR9 ONTO STACK
                                       /* PUSH SRRO ONTO STACK
   asm (" stw r9,4(r1)");
asm (" mfspr r9,27");
                                      /* PUSH SRR1 ONTO STACK
   asm (" stw r9,8(r1)");
asm (" mtspr 80,0");
                                   /* ENABLE INTERRUPTS
                                                                      * /
   pdpr->PDDAT += 1;
asm (" mtspr 82,0");
   pdpr->PDDAT += 1;
   asm (" mtspr 82,0"); /* MAKE NON-RECOVERABLE
asm (" lwz r9,8(r1)"); /* PULL SRR1 FROM STACK
                                                                      * /
   asm (" mtspr 27,r9");
   asm (" lwz r9,4(r1)"); /* PULL SRRO FROM STACK
   asm (" mtspr 26,r9");
asm (" lwz r9,0(r1)"); /* PULL GPR9 FROM STACK */
asm (" addi r1,r1,12"); /* RESTORE STACK POINTER */
getimmr()
   asm(" mfspr 3,638");
                                        /* GET EVT LOCATION
getevt()
   if ((getmsr() \& 0x40) == 0)
                                      /* IF MSR.IP IS 0
      return (0);
                                        /* THEN EVT IS IN LOW MEM*/
                                        /* ELSE
   else
      return (0xFFF00000);
                                        /* EVT IS IN HIGH MEM */
```



ex2.c (2 of 2)

How to Reach Us:

Home Page:

www.freescale.com

E-mail:

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 or +1-480-768-2130 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 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 303-675-2140 Fax: 303-675-2150

LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document. Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

