MSE912D60C_1M35Z
Rev. 0, 6/2011

Mask Set Errata for 68HC912D60C, Mask 1M35Z

Introduction

This mask set errata applies to this 68HC912D60C MCU mask set:

- 1M35Z

MCU Device Mask Set Identification

The mask set is identified by a 5-character code consisting of a version number, a letter, two numerical digits, and a letter, for example 2J88Y. All standard devices are marked with a mask set number and a date code.

MCU Device Date Codes

Device markings indicate the week of manufacture and the mask set used. The date is coded as four numerical digits where the first two digits indicate the year and the last two digits indicate the work week. For instance, the date code “0401” indicates the first week of the year 2004.

MCU Device Part Number Prefixes

Some MCU samples and devices are marked with an SC, PC, or XC prefix. An SC prefix denotes special/custom device. A PC prefix indicates a prototype device which has undergone basic testing only. An XC prefix denotes that the device is tested but is not fully characterized or qualified over the full range of normal manufacturing process variations. After full characterization and qualification, devices will be marked with the MC or SC prefix.

freescale™
semiconductor

Errata number	Module affected	Description
AR_700	CPU	Asserting an XIRQ interrupt can prevent the CPU from generating the vector request signal for the IRQ

XIRQ interrupt and IRQ — Errata Number: HC12_AR_700

Description

If all of the following conditions are met, the XIRQ asynchronous path can prevent the CPU from generating the vector request signal for the IRQ:

- Using an MCU in the HC12 Family (not the HCS12 Family)
- Using XIRQs (X-bit is cleared in the CCR by software)
- Asserting an XIRQ interrupt (through the XIRQ pin)
- XIRQ interrupt occurs at the start of an IRQ interrupt exception processing

Because XIRQs interrupt IRQs, the XIRQ stack will follow the IRQ stack. The lack of the IRQ vector request signal will cause the XIRQ stack to have an invalid return address. As soon as the XIRQ finishes executing the XIRQ interrupt service routine, the XIRQ RTI (return from interrupt) causes the CPU to use that invalid return address, leading to code runaway.

The potential failure window is only a few nanoseconds and varies with process, temperature, design, etc.

Workaround

There are two identified workarounds: one hardware and one software.

Hardware

Because the failure window is small and occurs near the T1 cycle, the external XIRQ signal could be gated to the rising edge of ECLK.

Software

Because the XIRQ interrupt service routine (ISR) still executes correctly, code can be added to the XIRQ ISR to determine whether the error may have occurred and use software to work around the situation. Because the problem only occurs if the XIRQ interrupts an IRQ before any ISR instructions are executed, the CCR in the XIRQ stack could be checked to determine whether the I-bit was set and two stack frames were created (first one for the IRQ and second one for the XIRQ). Further checks can then be done to determine whether the two stacks are identical except for the return address. If they are, use the IRQ stack as the return address for the XIRQ.

Here is an example of that code:

```
ALL_ISRs:
              pshy                    ;First instruction of the ALL ISRs need to push something
              inx                     ; (Y for example) onto stack to separate the stack frames
                                      ; to help to determine if any instructions from the ISR were
                                      ; run.  That will determine if the workaround need to be
                                      ; done when in the XIRQ ISR.
                     ;Note: this must be done in all ISRs, also adding the inx makes it less likely
                     ; you would falsely think you fell into the erratum. Increment what you tend
                     ; to not use in ISRs first, you could also increment D and Y for even further
                     ; security that the software does not falsely think it fell into the erratum.

                     ;Normal user ISR code here [except no RTI (yet)].

              leas 2,SP      ; return SP to adjust for the pshy

              rti     ; normal user rti

XIRQ_ISR:
                                      ;normal user ISR code here [except no RTI (yet)].

              brset 0,SP,#$10,Check;If CCR had I-bit set in the stack, this is the first part
                                      ;of the workaround to determine if the XIRQ interrupted
                                      ;an IRQ or a section of code that had the I bit set
                                      ;If not just return since no problem.
Okrti:
              rti                     ;Normal user code (unstack registers, etc.)


Check:                       ;The I bit was set in the XIRQ stack so we need
                                      ; to further check and see if we should do the
                                      ; workaround.  Need to check to see if there are two
                                      ; nearly identical stack frames with the exception of the
                                      ; I-bit and the return address. If so adjust the Stack Pointer
                                      ; to point to the ISR stack frame before the XIRQ RTI.
                                      ;Note, non interruptible code that gets an XIRQ
                                      ; will also go here.  If X or Y was not used in the XIRQ
                                      ; ISR then this code could be reduced in size, (by removing
                                      ; the appropriate loads from below)

              ldd 1,SP                ;Load ACCD from XIRQ stack frame ACCB:ACCA value
                                      ;Note the values of A and B are interchanged from a
                                      ;normal pull for easier checking.
              cpd 10,SP       ;Compare ACCD from suspect IRQ stack frame with XIRQ
                                      ; stack. Note the values of A and B are still interchanged.
              bne Okrti       ;Not the same, so not in erratum, just return (RTI).
              ldx 3,SP                ;Load X with XIRQ stack frame X value.
              cpx 12,SP       ;Compare X from suspect IRQ stack frame with XIRQ stack.
              bne Okrti       ;Not the same, so not in erratum, just return (RTI).
              ldy 5,SP                ;Load Y with XIRQ stack frame Y value.
              cpy 14,SP       ;Compare Y from suspect IRQ stack frame with XIRQ stack.
              bne Okrti       ;Not the same, so not in erratum, just return (RTI).

                                      ;Next we check the CCR to see if they are the same except
                                      ; for the I bit which should be different.
              ldaa 0,SP       ;Load the CCR from the XIRQ stack into ACCA
              eora 9,SP       ;Exclusive OR XIRQ CCR with suspect IRQ CCR
              anda #$EF       ;AND with the I bit mask to not check the I bit.
              bne Okrti       ;Not the same, so not in erratum, just return (RTI).
                                      ;if all checks the same could be in the erratum
                                      ;could check return address from IRQ ISR as a further check
```

```
                                ;to make sure it is a normal ISR program space
                                ;could also check to make sure room for SP to back up
        leas 9,SP       ;add 9 to SP (to point to IRQ stack frame)

    rti                         ;Return using IRQ stack (unstack registers, etc)
```

As with most code workarounds, there are a few situations where there still may be an issue. For example, if you pushed information on to the stack that exactly matched the stack frame before you did the XIRQ and that XIRQ occurred while the I bit was set, the software could falsely think it was the ISR stack. This is a very rare situation.

How to Reach Us:

USA/Europe/Locations not listed:
Freescale Semiconductor Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

Japan:
Freescale Semiconductor Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu
Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

Asia/Pacific:
Freescale Semiconductor H.K. Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
852-26668334

Learn More:
For more information about Freescale Semiconductor products, please visit
http://www.freescale.com

MSE912D60C_1M35Z
Rev. 0, 6/2011

freescale™ semiconductor