MC9S12XDP512, Mask 0L15Y

Introduction

This errata sheet applies to the following devices:

MC9S12XDP512, MC9S12XDT512, MC9S12XA512, MC9S12XDT384, MC9S12XDQ526, MC9S12XDT256, MC9S12XD256, MC9S12XB256, MC9S12XA256, MC9S12XDG128, MC9S12XD128, MC9S12XA128, MC9S12XB128

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 1K79X. 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 "0201" indicates the first week of the year 2002.

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.

Errata System Tracking Numbers

MUCtsXXXXX is the tracking number for device errata. It can be used with the mask set and date code to identify a specific erratum.

Errata Summary

Errata Number	Module affected	Brief Description	Work-around
MUCts01812	s12x_cpu	False tagged breakpoint hits may be reported by the DBG module	YES
MUCts01816	s12x_dbg	Back to back accesses to DBG trace buffer may return incorrect data	YES
MUCts01818	s12x_bdm	BDM hardware read command may be corrupted by entering stop mode	NO
MUCts01974	pim_9xd	PIM: ECLK divider can be activated in emulation modes	YES
MUCts02366	s12x_mmc	Incorrect 23-Bit Program Counter Generated for DBG on global accesses	YES
MUCts02409	eetx	EEPROM: Protection Disabled on EPROT Read During Command Write Sequence	YES
MUCts02539	spi	SPI: Slave entering stop in wait mode, pending rx data not rejected	YES
MUCts02582	s12x_dbg	S12X_DBG: Indexed jump loop1 mode trace buffer entries may be missed	YES
MUCts02667	spi	SPI in slave mode (CPHA=0) and SS line not deasserted between transmissions may lead to corrupted rx data	YES
MUCts02963	mscan	msCAN: Potential byte corruption when FIFO full	YES
MUCts02988	xgate	XGATE: Carry-Flag may be falsely set by SSEM instruction	YES

MUCts03017	ect_16b8c	ECT: TCNT counter resets in Input Capture Mode	YES
MUCts03018	ect_16b8c	ECT: CxF flag clears following a read to TCx on an OC event with TFFCA=1	YES
MUCts03019	ect_16b8c	ECT: CxF flag clears following a wite to TCx on an IC event with TFFCA=1	YES
MUCts03037	ect_16b8c	ECT: Forced OC on PT7 occured even when TIOS7 = 0	NO
MUCts03039	ect_16b8c	ECT:Faulty OC event with OM/OL=0, OC7Mx=1, TIOSx=1	YES
MUCts03112	s12x_bdm	BDM: Incomplete Memory Access on misaligned access due to BDM features	YES
MUCts03332	ftx	FTX: Blind Spot in Data Compress Command Algorithm	YES
MUCts03390	atd_10b16c	ATD: Abort of an A/D conversion sequence with write to ATDxCTL0/1/2/3 may not work	YES
MUCts03391	atd_10b8c	ATD: Abort of an A/D conversion sequence with write to ATDxCTL0/1/2/3 may not work	YES
MUCts03453	mscan	MSCAN: Corrupt ID may be sent in early-SOF condition	YES
MUCts03617	s12x_dbg	DBG 'outside range' mode databus qualification ignored	YES
MUCts03620	s12x_dbg	DBG No address match if next transaction is misaligned word access	YES
MUCts03621	s12x_dbg	DBG Range Mode TAGB and TAGD influence in range modes	YES
MUCts03634	eetx	eetx4k An interrupt following immediately after execution of a STOP instruction may disable read access to the EEPROM array	YES
MUCts03646	vreg_3v3	vreg_3v3.05.00: Possible incorrect operation if device is wakened from stop mode within 4.7µs of stop mode entry	NO
MUCts03686	atd_10b8c	ADC: conversion does not start with 2 consecutive writes to ATDCTL5	YES
MUCts03689	atd_10b16c	ADC: conversion does not start with 2 consecutive writes to ATDCTL5	YES
MUCts03761	s12x_dbg	DBG: State flags and counter corrupted by simultaneous arm and disarm	YES
MUCts03870	s12x_cpu	CPU: Breakpoint missed at simultaneous taghits	YES
MUCts03977	pwm_8b8c	PWM: Emergency shutdown input can be overruled	YES
MUCts03996	ftx	Flash: Burst programming issue if bus clock frequency is higher than oscillator clock frequency	YES
MUCts03997	pim_9xd	PIM: Edge-sensitive mode of IRQ-pin may cause incorrect interrupt vector fetch	YES
MUCts04095	ect_16b8c	ECT: Channel 0 - 3 Input Capture interrupts inhibited	YES

		when BUFEN=1, LATQ=0 and NOVWx=1	
MUCts04135	pwm_8b8c	PWM: Wrong output level after shutdown restart in 16bit concatenated channel mode	YES
MUCts04136	pwm_8b8c	PWM: Wrong output value after restart from stop or wait mode	YES
MUCts04155	ect_16b8c	ECT_16B8C: Output compare pulse is inaccurate	YES
MUCts04232	ftx	FTX: Flash Command influenced by Backdoor Key write	YES

False tagged breakpoint hits may be reported by the DBG module — MUCts01812

Description

If the device executes the BACKGROUND command (received over BDM) in the

cycle when it is about to execute a tagged instruction the DBG module

may indicate that a tagged breakpoint was hit even though the tagged

instruction itself was not executed yet.

Workaround

If the DBG module indicates a tagged breakpoint hit after the BACKGROUND

command was issued, verify contents of PC to determine whether the

tagged breakpoint was hit or not.

Back to back accesses to DBG trace buffer may return incorrect data — MUCts01816

Description

When the trace buffer is unlocked for reading and data is read from the

trace buffer register in the next bus cycle (back-to-back accesses),

wrong data may be read out.

Workaround

When using the core or Xgate to read the contents of the trace buffer,

do not use back-to-back bus accesses.

OR

Only use the BDM to access contents of the trace buffer.

BDM hardware read command may be corrupted by entering stop mode — MUCts01818

Description

The BDM can generate invalid data when the BDM is processing a hardware

(HW) read command with the handshake feature of the BDM enabled and the

CPU enters STOP mode with the device not reaching system STOP mode

because of XGATE activity or set-up (as described below).

The BDM read data is cleared in the BDM shift register because the CPU

enters STOP mode but the corresponding ACK pulse of the HW read command

is not prevented. The occuring ACK pulse indicates to the host that

valid data is ready, which is not the case. Instead of valid data a data

value of $FF will be received.

This behaviour occurs only if the CPU enters into STOP mode after a BDM

read access has occured on the internal bus and before the corresponding

ACK pulse of the HW read command is sent.

It only occurs when the CPU enters into STOP mode while XGATE is not

idle or the XGFACT bit is set. In this case only CPU clocks are stopped.

This behaviour does not occur when a HW read command is sent while the

CPU is already in STOP mode.

It does not occur if the device reaches system STOP mode (all clocks

stopped), which happens when the CPU enters STOP mode and XGATE is in

the idle state and the XGFACT bit is cleared.

This is a debug only issue with a low probability of occurance because:

1) The timing window during which the CPU transition (from run mode to

STOP mode) must occur is very short. (Timing window means the time from

the BDM read access on the internal bus until the BDM internal ACK pulse

control signal is set - in the case of no clock switching this is two

bus clock cycles).

2) Only a transition of the CPU from run mode to STOP mode during the

time window will cause the issue.

3) The issue occurs only if system STOP is not reached.

Workaround

None

PIM: ECLK divider can be activated in emulation modes MUCts01974

Description

The ECLK does not run at bus clock rate in emulation modes if an EDIV value greater than zero is selected. In this case the ECLK will be divided as specified or stopped if NECLK is set. This may affect emulation systems which rely on constant ECLK rate.

Workaround

Always keep the NECLK and EDIV value at zero in case constant ECLK rate

is required.

In order to pass the required divider settings to the emulation system,

the EDIV settings should alternatively be mapped to the reserved bits ECLKCTL[3:2]. The emulation system should utilize these bits instead of

EDIV to control the ECLK generator for the target application. This software modification needs to be undone when the application runs without the emulator. The emulator can monitor values written into ECLKCTL[1:0] and notify the user when non-zero value can cause the emulator to function incorrectly.

Note:

Please note that this erratum only concerns the ECLK signal. Signal

ECLKX2 is not affected by EDIV values when operating in emulation modes.

Incorrect 23-Bit Program Counter Generated for DBG on global accesses — MUCts02366

Description

An incorrect trace buffer entry occurs when the DBG module attempts to trace a Change Of Flow (COF) instruction that follows a global access. This only happens for COF instructions where the source address should be stored to the trace buffer.

This errata only affects the functionality of the DBG module, causing incorrect DBG trace buffer entries.

During global instruction accesses, the program counter bits [22:16] are derived from the global address bus bits [22:16] which, at that moment, contain the address of the global access.

The program counter should always provide the global opcode address and should be independent of addresses associated with memory accesses. In the errata case, the page part of the opcode address contains the data access page address.

The program counter bits [15:0] are not affected by this problem.

Workaround

In debug mode, put a NOP before the global instruction.

EEPROM: Protection Disabled on EPROT Read During Command Write Sequence — MUCts02409

Description

If EPROT EOPEN bit is set with EPROT EPDIS bit clear (i.e. protection is

on) and EPROT is read during a command write sequence then protection is

disabled until the next reset or write of the EPROT register.

Workaround

Do not read EPROT register during a command write sequence.

SPI: Slave entering stop in wait mode, pending rx data not rejected — MUCts02539

Description

In slave mode, pending data in the receive shift register is not

rejected when entering Wait mode with the SPISWAI bit set.

When the SPI stops on entering Wait mode (executing WAI with SPISWAI bit

set), data pending in the receive shift register should be rejected

(lost). This is to prevent pending data in the receive shift register

from being corrupted by SCK cycles while halted in Wait mode.

Workaround

Workaround #1: Ensure that the receive buffer queue is empty before entering Wait mode.

Workaround #2: Make sure SCK does not toggle while in Wait mode.

Workaround #3: Do not use the 'stop in Wait mode' feature.

S12X_DBG: Indexed jump loop1 mode trace buffer entries may be missed — MUCts02582

Description

Loop1 Mode inhibits consecutive duplicate source address entries that would typically be stored in most tight looping constructs. It does not inhibit repeated entries of destination addresses. However, the logic prevents the storage of valid indexed jump destination addresses if consecutive indexed jumps have the same opcode address.

Considering the code below; the first occurence of JMP 0,X stores MARK5

to the trace buffer. Since the code returns to MARK4 with an unconditional branch, the next trace buffer entry should be the next destination address of JMP 0,X (MARK6 the second time around).

This entry is missed because further COFs from MARK4 are masked out.

```
        LDX     #MARK5
MARK4   JMP     0,X
        NOP
MARK5   LDX     #MARK6
        BRA     MARK4
MARK6   NOP
```

This does not affect XGATE loop mode tracing

Workaround

If code contains consecutive indexed jumps from an idential opcode

address, then normal mode tracing can be temporarily used to confirm the

incidence of consecutive indexed jumps from the same address.

Subsequently loop1 mode can be enabled to continue further debugging.

SPI in slave mode (CPHA=0) and SS line not deasserted between transmissions may lead to corrupted rx data — MUCts02667

Description

When the SPI is in slave mode with both the SPI data register (SPIDR)

and the receive shift register containing pending data, on reception of

further data the contents of the shift register should be discarded and

the new data byte shifted in. Reading the SPIDR should not cause the

shift register contents to be transferred to the SPIDR until the latest

data byte is completely received.

In the erratum condition, the shift register pending state is not

discarded and reading the SPIDR during the reception of a new data byte

will cause the shift register contents to be immediately transferred to

the SPIDR register causing data corruption.

This condition only occurs in slave mode with CPHA = 0 and if SS is not

de-asserted between transmissions.

Workaround

Workaround #1: Deassert SS for minimum idle time (0.5*Tsck) between

transmissions.

Workaround #2: Ensure that the receive buffer queue is empty before

starting a new transmission.

---

msCAN: Potential byte corruption when FIFO full MUCts02963

Description

When messages received by the msCAN controller are not serviced in

time, such that the five-stage FIFO becomes full, one data byte of

the oldest message in the receive buffer FIFO may contain invalid data

with the value 0x7F.

The data lengths of the oldest and the newest message in the FIFO

determine if this occurs, and if so, the position of the affected Data

Segment Register (DSR) byte:

DSR(n+2) is affected in oldest message if newest message has data

length n.

CONDITIONS UNDER WHICH THE ERROR OCCURS

This issue occurs only when the message buffer FIFO runs full:

The FIFO has been filled with four messages 'a' (oldest), 'b', 'c', and 'd', when a new message 'e' is received that is protocol compliant (no errors) and that passes the acceptance filter.

and the oldest and newest message lengths meet the following critera:

The data length codes, La and Le, of messages 'a' and 'e' are such that:

3 <= La <= 8, and

0 <= Le <= 5, and

Le <= La - 3.

Example

Consider the case where La = 8, and Le = 4.

Here n = Le = 4, therefore byte DSR(n+2) = DSR6 (i.e. the seventh data byte of message 'a') gets written to 0x7F.

PROBABILITY OF THE ERROR OCCURRING

A. The FIFO runs full without service (near-overrun situation) - Pa:

Application software will typically avoid this possible overrun situation, minimizing the probability of this condition occurring - hence this is typically a very low probability (that is dependent on the application implementation).

B. Message length dependency - Pb:

If all Rx messages have the same length, or are 6, 7, or 8 bytes long, there is no possibility of the error occurring, and Pb = 0.

For random non-0 data lengths, Pb = 0.25 (approximately).

Overall, the probability of the error occurring is P = Pa x Pb.

Workaround

The following precautions can be taken to avoid the problem:

- Do not allow the receive FIFO buffer to become full.
  > If necessary, raise the Rx interrupt priority so that condition A is avoided.
- Use constant data lengths (any value) for the Rx messages
  > E.g. use eight bytes only, and pad shorter messages with dummy bytes
- Use (mixed) data lengths in the range 6 to 8.
- Use (mixed) data lengths in the range 1 to 3.
- Use a parity or checksum scheme to detect a corrupted byte.
- Check for the 'critical' value 0x7F in bytes 3 to 8.

XGATE: Carry-Flag may be falsely set by SSEM instruction — MUCts02988

Description

If the S12X_CPU and the XGATE attempt to lock a semaphore at the same time the S12X_CPU will obtain the semaphore, but the Carry-Flag will be set for the XGATE (falsely indicating that the XGATE has locked the semaphore).

Workaround

Execute two consecutive "SSEM" instructions to set a semaphore. Ignore result of the first "SSEM" instruction. Carry-Flag will indicate correctly whether XGATE has allocated the semaphore after the second "SSEM" instruction is executed.

ECT: TCNT counter resets in Input Capture Mode — MUCts03017

Description

Normal Operation:

Timer Free Running Counter, also called as Timer Counter resets to 0x0000 on a Channel 7 Output Compare event when Timer Counter Reset Enable (TCRE) bit of TSCR2 registers is set to 1. This assumes that TIOS7 bit of TIOS register is set to a 1 (Output Compare mode).

Issue: Erroneously this behaviour was found to occur even when

TIOS7 bit of TIOS register was set to a "0" (Input Capture mode)

Workaround

Reseting Free Running Counter alias Timer Counter can be avoided by

setting Timer Counter Reset Enable (TCRE) bit of TSCR2 register to a "0".

ECT: CxF flag clears following a read to TCx on an OC event with TFFCA=1 — MUCts03018

Description

Problem:

Normal Operation:

With TIOS = 1 (Output Compare mode) and TFFCA = 1 (Fast Flag Clearing

mechanism) a write to TCx register following an output compare event

clears the flag.

Erroneous Operation:

A read to TCx register following an Output Compare event clears Cxf flag

in TFLG1 register.

Workaround

Customer should avoid reading TCx register following an Ouput Compare event.

ECT: CxF flag clears following a wite to TCx on an IC event with TFFCA=1 — MUCts03019

Description

Normal Operation:

With TIOS = 0 (Input Capture (IC) mode) and TFFCA = 1 (Fast Flag

Clearing mechanism) a read to TCx register following an Input Capture

(IC) event clears CxF (flag) bit of TFLG1 register.

Erroneous Operation:

A write to TCx register following an Input Capture (IC) event clears Cxf

(flag) bit of TFLG1 register.

Workaround

Customer should avoid writing to TCx register following an Input Capture

(IC) event.

ECT: Forced OC on PT7 occured even when TIOS7 = 0 MUCts03037

Description

Correct Operation:

Forced Output Compare (OC) operation on channel 7 requires TIOS7 bit be

set to a “1” for a successful Ouput Compare (OC) event.

Faulty Operation:

Forced Ouput Compare (OC) action was noticed on Channel 7 inspite of

TIOS7 being set to a “0”.

Workaround

None.

ECT:Faulty OC event with OM/OL=0, OC7Mx=1, TIOSx=1 — MUCts03039

Description

Correct Operation:

When timer is enabled (TEN bit of TSCR1 register is set to 1) and Output

Compare functionality is selected (TIOSx bit of TIOS register is set) on

a match (TCNT = Tcx) an output compare event(dictated by OM/OL bits of

TCTL1/TCTL2 register) is generated. Please not that the Output Compare

is registered internally.

Faulty Operation:

Consider a case, when an normal Output Compare event (driving port to 1)

was performed. You intend to dis-connect output compare logic from pin

logic. Hence you configure (OM/OL bits to a “0”), but your TIOSx bit is

still set to a “1”. Also, you intend to use masking feature of output

compare so you set TIOS7=1 and you have OC7Mx bit set to a “1”. If the

next output compare match(TCNT = Tcx) happens to be normal output

compare event, the default state of internal register (0) was noticed

on

the channel.

In reality the output port should not have toggled since OM/OL were set

to 0.

This is an erroneous behaviour.

Workaround

Make sure that OC7Mx bit is set to "0" if not using OC7M feature for

that channel. If using masking feature, please remember to set OC7Mx bit

to "0" as soon as you have a channel 7 Compare event.

BDM: Incomplete Memory Access on misaligned access due to BDM features MUCts03112

Description

If a misaligned word write access is directly followed by an attempted

entry into active BDM, then the second byte access may be performed on a

different target due to the memory map switching for BDM active.

Depending on the type of access this can lead to the following situations...

1. When writing in the address range from $7F_FEFF to $7F_FFFD in the

external space the MMC splits the accesses into two 8-bit accesses. It

is able to complete the first access, but just before the second access

the BDM firmware becomes mapped into this address location and the

second byte access goes to the firmware.

This can only occur if the last cycle of the access instruction is a

word write to an odd address AND the instruction is followed by BGND or

a tagged breakpoint.

2. On a misaligned word write with a global store (GSTD,GSTS,GSTX,GSTY)

in the address range $XX_0FFF to $XX_3FFD, the first split access is

performed correctly in the external space, but in the second byte cycle

the MMC is instructed to interpret the address coming from the CPU as a

local address, which is the internal RAM.

This can only occur if the CPU executes a global store instruction which

writes to an odd address in the last instruction cycle AND the

instruction is followed by BGND or a tagged breakpoint.

3. Generally on a misaligned global store (GSTD,GSTS,GSTX,GSTY) the

lower 16-bits of the global address are interpreted as a local address

for the second byte access. This can result in unintended accesses to

registers or external RAM.

Workaround

To prevent scenarios (2) and (3) of the errata occurring, avoid setting

breakpoints after a GSTD.

When not debugging, remove all instances of BGND from code.

FTX: Blind Spot in Data Compress Command Algorithm — MUCts03332

Description

If the range of Flash addresses to be compressed is 32K or greater, the

data at one of the addresses will be effectively ignored. The address

affected is 32K from the upper address read in the data compress

algorithm, e.g., for an address range of 32K, the first data read in the

algorithm will not affect the final signature provided by the algorithm.

Workaround

Limit range of addresses to be compressed to less than 32K addresses.

Execute multiple data compress commands to compress larger Flash address

ranges.

ATD: Abort of an A/D conversion sequence with write to ATDxCTL0/1/2/3 may not work — MUCts03390

Description

Starting a conversion with a write to ATDxCTL5 or on an external trigger

event, and aborting immediately afterwards with a write to ATDxCTL0,

ATDCTL1, ATDxCTL2 or ATDxCTL3 can fail to stop the conversion process.

Workaround

Only write to ATDxCTL4 to abort an ongoing conversion sequence.

Use the recommended start and abort procedures from the Block Guide.

Section : Initialization/Application Information

Subsection: Setting up and starting an A/D conversion

Subsection: Aborting an A/D conversion

ATD: Abort of an A/D conversion sequence with write to ATDxCTL0/1/2/3 may not work — MUCts03391

Description

Starting a conversion with a write to ATDxCTL5 or on an external

trigger event, and aborting immediately afterwards with a write to

ATDxCTL0, ATDCTL1, ATDxCTL2 or ATDxCTL3 can fail to stop the

conversion process.

Workaround

Only write to ATDxCTL4 to abort an ongoing conversion sequence.

Use the recommended start and abort procedures from the Block Guide.

Section : Initialization/Application Information

Subsection: Setting up and starting an A/D conversion

Subsection: Aborting an A/D conversion

MSCAN: Corrupt ID may be sent in early-SOF condition MUCts03453

Description

The initial eight ID bits will be corrupted if a message is set up for

transmission during the third bit of INTERMISSION and a dominant bit is

sampled leading to an early-SOF*.

The CRC is calculated from the resulting bit stream so that the

receiving nodes will still validate the message.

An early-SOF condition may only occur if the oscillators in the network

operate at a tolerance range which could lead to a cumulated phase error

after 11 bit times larger than phase segment 2.

In case arbitration is lost during transmission of the corrupt

identifier, a non-corrupted ID will be sent with the next attempt if the

transmit request remains active.

*The CAN protocol condition referred to as 'early-SOF' in this erratum

is detailed in "Bosch CAN Specification Version 2.0" Part A, section 9,

and a Note to section 3.2.5 INTERFRAME SPACING – INTERMISSION in Part B.

Workaround

Due to increased oscillator tolerance a transmission start in the third

bit of intermission is possible and allowed. The errata can be avoided

when calculating the maximum oscillator tolerance of the overall CAN

system. The phase error after 11 bit times due to the oscillator

tolerance should be smaller than phase segment 2.

If an early-SOF cannot be avoided the following methods will provide

prevention:

- Assigning the same value to all upper eight ID bits in the network

- Allocating dedicated data length codes (DLC) to every identifier used

  in the network and checking for correspondence after reception

- Assigning only IDs (x) which do not consist of a combination of other

  assigned IDs (y,z) and using the acceptance filters to reject

  erroneous messages, i.e.

  - for standard frames: IDx[11:0] != {IDy[11:3], IDz[2:0]}

  - for extended frames: IDx[28:21] != {IDy[28:21],IDz[20:0]}

DBG 'outside range' mode databus qualification ignored MUCts03617

Description

When using a comparator pair for a range comparison, the databus can also be used for qualification by using the comparator A/C data and data mask registers. This does not work correctly.

Scenario:

With CompA/CompB in 'outside range' mode an access to a memory location

with an address above the high boundary of CompB always generates a comparator match. The data qualification is not carried out.

If the accessed memory is located below the address defined in CompA, the behavior is correct.

CompC/CompD behavior is the same.

Workaround

Using 2 comparator pairs configured for inside range mode an outside

range match with databus qualification can be generated.

```
MAP
0x000000 |COMPA|
         |     | Inside Range A/B with DB qualification
ADDRESSX |COMPB|
         |     |
ADDRESSY |COMPC|
         |     | Inside Range C/D with DB qualification
0x7FFFFF |COMPD|
```

DBG No address match if next transaction is misaligned word access — MUCts03620

Description

Memory accesses in successive bus cycles must both be able to generate forced triggers if both accesses match.

If accesses occur in successive cycles whereby the second access is a misaligned word access, then a comparator match is lost. This could cause a forced breakpoint to be missed if the state sequencer is dependent on both the successive matches to reach final state.
Example...

```
        LDX    #WORD_MISALIGNED

        STD    0,X               ; First access M0->State2
        LDD    WORD_MISALIGNED   ; Second access M0->Final State
```

If the STD last bus cycle is a write and the LDD first bus cycle is a read then only one state sequencer transition occurs.

In range modes, this can occur when 2 different addresses within/outside the specified range are accessed in successive bus cycles, otherwise it can only happen if the same address is written and then read in successive bus cycles.

Workaround

Insert a NOP instruction before misaligned word accesses if they can follow accesses to the predefined comparator range.

```
        LDX    #WORD_MISALIGNED-1
        STD    0,X              ; First access in range M0->State2
        NOP                     ; NOP insertion
        LDD    WORD_MISALIGNED  ; Second access in range M0->Final State
```

DBG Range Mode TAGB and TAGD influence in range modes — MUCts03621

Description

The comparator A and C TAG bits are used to tag range comparisons for the AB and CD ranges respectively. The comparator B and D TAG bits should have no effect in range modes. However the TAGB/TAGD bits do

have

an effect.

If the A/C TAG bit is 0 but the paired B/D TAG bit is set, a valid

match

may be missed.

Workaround

Clear TAGB when configured for forced range mode comparisons using CompAB

Clear TAGD when configured for forced range mode comparisons using CompCD

---

eetx4k An interrupt following immediately after execution of a STOP instruction may disable read access to the EEPROM array — MUCts03634

Description

An interrupt request immediately following execution of a STOP

instruction may cause the EEPROM array to return incorrect data when

read. The problem will only occur if all of the following conditions

are

met:

1) S bit in the CCR register is cleared (stop mode enabled)

2) I bit in the CCR register is cleared (interrupts enabled)

3) A STOP instruction is executed by the microcontroller

4) An interrupt becomes pending between 0 and 0.5 bus clock cycles

after beginning of the STOP instruction execution.

In this case the microcontroller will wake-up correctly from the stop mode, but the EEPROM array will return incorrect data when a read operation is performed.

Access (read or write) to any of the EEPROM registers or write into the EEPROM array (to start a command sequence) will restore the read access. The EEPROM will return the correct data afterwards.

Workaround

If the STOP instruction is executed while interrupts are enabled, read or write any of the EEPROM registers after wake-up from stop mode to ensure correct operation. Alternatively a write into the EEPROM array (as part of a command sequence) can be performed instead of the register access.

---

vreg_3v3.05.00: Possible incorrect operation if device is wakened from stop mode within 4.7µs of stop mode entry — MUCts03646

Description

It is possible that after the device enters Stop or Pseudo-Stop mode it may reset rather than wake up normally upon reception of the wake-up signal.

CONDITIONS:

This event will only happen provided ALL of the following conditions

are

met:

1) Device is powered by the on-chip voltage regulator.

2) Device enters stop or pseudo-stop mode (see Stop mode entry description below)

3) The wake-up signal is activated within a specific and very short window (typically 11ns long, not longer than 20ns). The position of the window varies between different devices, however it never starts sooner than 1.6µs and never ends later than 4.7µs after the stop mode entry. This narrow width of the susceptible window makes the erratum unlikely to ever show in the applications life.

Stop or Pseudo-Stop mode entry are:

1) Execution of STOP instruction by the CPU (provided the S-bit in CCR is cleared)

NOTE: The part enters stop mode either after 12 oscillator clock cycles with the PLL disengaged or 3 PLL clock cycles and 8 oscillator clock cycles with the PLL engaged after the STOP command is executed.

2) End of XGATE thread (providing the CPU is in stop or pseudo-stop mode)

The incorrect behavior will never occur if ANY of the wake-up conditions are met at the time when the stop mode entry is attempted (an enabled interrupt is pending).

EFFECT:

If this incorrect behavior occurs, the device will Reset and indicate a

Low Voltage Reset (LVR) as the reset source.

The device will operate normally after the reset.

Workaround

None.

--

Asynchronous Low Voltage Resets are possible in any microcontroller

application (due to power supply drops) and the integrated LVR and LVI

features and dedicated LVR reset vector are provided to manage this fact

cleanly. For best practice, the application's software should be written

to recover from a Low Voltage Reset in a controlled manner.

An application software written to deal with valid Low Voltage Resets

should correctly manage erroneous LVR events.

It can also be possible to avoid erroneous Low Voltage Resets from

synchronous wake-up events by configuring the application software to

ensure that the entry into stop occurs at such a time, in relation to

the wake-up event timer, that a wake-up event does not occur within

1.6µs to 4.7µs after Stop/Pseudo-Stop entry.

ADC: conversion does not start with 2 consecutive writes to ATDCTL5 — MUCts03686

Description

When the ATD is started with write to ATDCTL5

and, which is very unusual and not necessary,

within a certain period again started with write

to ATDCTL5. The conversion will not start at all.

This does only happen if the two consecutive writes to ATDCTL5 occur

within one "ATD clock period". An ATD clock period is defined by a full

rollover of the ATD clock prescaler. That is for example PRS[4:0] = 2 -

> (2+1)*2 = within 6 bus cycles.

Workaround

Only write once to ATDCTL5 when starting a conversion.

Use the recommended start and abort procedures from the Block Guide.

Section : Initialization/Application Information Subsection: Setting up

and starting an A/D conversion Subsection: Aborting an A/D conversion

---

ADC: conversion does not start with 2 consecutive writes to ATDCTL5 — MUCts03689

Description

When the ATD is started with write to ATDCTL5

and, which is very unusual and not necessary,

within a certain period again started with write

to ATDCTL5. The conversion will not start at all.

This does only happen if the two consecutive writes to ATDCTL5 occur

within one "ATD clock period". An ATD clock period is defined by a full

rollover of the ATD clock prescaler. That is for example PRS[4:0] = 2 -

> (2+1)*2 = within 6 bus cycles.

Workaround

Only write once to ATDCTL5 when starting a conversion.

Use the recommended start and abort procedures from the Block Guide.

Section : Initialization/Application Information Subsection: Setting up

and starting an A/D conversion Subsection: Aborting an A/D conversion

---

DBG: State flags and counter corrupted by simultaneous arm and disarm — MUCts03761

Description

Simultaneous disarming (hardware) and arming (software) results in

status and counter register (DBGSR, DBGCNT) corruption.

Hardware disarming initiated by an internal trigger or by tracing

completion takes 2 clock cycles. If a write to DBGC1 with the ARM bit

set occurs in the final clock cycle of this disarming process, arming

is suppressed but the DBGSR register is initialized to state1 and

DBGCNT is initialized to zero.

The result is that the DBG module is disarmed by hardware but DBGSR indicates state1.

NOTE: DBGC1 is typically only written to whilst armed to set the TRIG bit or update the COMRV bits to map different registers to the address map window.

Generally during debugging, after arming the DBG module and returning to application code a further write access of DBGC1 over the BDM interface requires considerable time relative to application code execution, therefore in many cases the breakpoint may be reached before a DBGC1 update is attempted. Furthermore the probability of hitting the same cycle is seen to be very low.

Workaround

If the fault condition is caused by writing to DBGC1 to set the TRIG bit (to request an immediate breakpoint), then the application code may be rerun again without the attempted setting of TRIG. The software trigger (TRIG) is unnecessary in any case at the same point point in time as an internal hardware trigger occurs.

Development tool vendors should avoid COMRV updates while the DBG is

armed.

Users that observe the problem due to development tool COMRV updates can add a NOP to code and rerun to shift the disarm cycle, thereby preventing a collision with the COMRV updates.

CPU: Breakpoint missed at simultaneous taghits MUCts03870

Description

The CPU execution priority encoder evaluates taghits and then generates a breakpoint if a taghit must lead to an immediate breakpoint as determined by the DBG module.

If the DBG module indicates that this taghit leads to an immediate breakpoint then the CPU loads the execution stage with SWI or BGND thus
generating the breakpoint.

At simultaneous taghits the lowest channel has priority.
If taghits on 2 channels simultaneously, whereby the lower channel tag must be ignored, but the higher channel tag must cause a breakpoint, then the breakpoint request is erroneously missed.
Thus if channel[1:0] taghits occur simultaneously whereby channel[0] must be ignored but channel[1] must cause a breakpoint, then the breakpoint request on channel[1] is erroneously missed and no breakpoint generated.

The DBG module recognises the taghit and the state sequencer transitions accordingly. Furthermore the taghit causes the DBG module to request a forced breakpoint, which means that a late CPU breakpoint occurs. If the tagged instruction is a single cycle instruction, the breakpoint occurs at the second instruction following the tagged instruction.
Otherwise the breakpoint occurs at the instruction immediately following the tagged instruction.

This bug requires that separate tags are placed on the same instruction. This is not a typical case when using exact tag addresses.
It is more relevant in debugging environments using range modes, where tags may cover a whole range. In this case a tagged range may cover a tag or another tagged range, making simultaneous taghits possible.

This is a debugging issue only.

Workaround

Do not attach multiple tags to the same exact address.
When attaching multiple tags to the same address by overlapping tag ranges in range modes, or covering a tag with a tag range in range modes, then the missed tag can be avoided by mapping the final state change to the lower channel number.

For example the case
Channel[0] tags a range; channel[3] tags an instruction within that

range.

From DBG State1 Taghit[0] leads to State2. (DBGSCR1=$6)

From DBG State2 Taghit[3] leads to FinalState.(DBGSCR2=$5)

In State2 a simultaneous Taghit[3,0] scenario would miss the breakpoint.

This can be avoided by using the alternative DBG configuration

Channel[2] tags a range; channel[0] tags an instruction within that range.

From DBG State1 Taghit[2] leads to State2. (DBGSCR1=$3)

From DBG State2 Taghit[0] leads to FinalState.(DBGSCR2=$7)

PWM: Emergency shutdown input can be overruled MUCts03977

Description

If the PWM emergency shutdown feature is enabled (PWM7ENA=1) and PWM channel 7 is disabled (PWME7=0) another lower priority function available on the related pin can take control over the data direction. This does not lead to a problem if input mode is maintained. If the alternative function switches to output mode the shutdown function may unintentionally be triggered by the output data.

Workaround

When using the PWM emergency shutdown feature the GPIO function on the pin associated with PWM channel 7 should be selected as an input.

In the case that this pin is selected as an output or where an

alternative function is enabled which could drive it as an output, enable PWM channel 7 by setting the PWME7 bit. This prevents an active shutdown level driven on the (output) pin from resulting in an emergency shutdown of the enabled PWM channels.

Flash: Burst programming issue if bus clock frequency is higher than oscillator clock frequency — MUCts03996

Description

If S12X is running at a bus clock frequency higher than the oscillator clock frequency (Fbus > Fosc), flash words may remain not programmed after a program burst sequence. Software methods can be used to avoid this problem. If burst programming is not used, no errors occur.

The root cause of this issue is related to flash internal state machine which needs about 2 to 3 oscillator clock periods to be ready to accept a new flash command after the assertion of the CBEIF flag.

Note that this latency of flash state machine is related to the assertion of CBEIF and the start of a new command sequence (i.e. a write to a flash word). There is no latency after the assertion of the CCIF flag.

Workaround

Adding a time delay between the check of CBEIF flag and the start of the

next flash program command sequence will ensure that all words will be

programmed. Add a delay of (3 * (Fbus/Fosc)) Bus clock cycles after

CBEIF is set, that can be achieved by the addition of NOPs (one NOP

instruction takes one bus cycle to execute).

Note that since a flash word program operation takes much longer than 3

oscillator clock periods, there is no impact in the total programming

time of a long program burst sequence by adding a delay of 3 clocks.

The example below illustrates the proposed workaround:

Code below executes a program burst sequence by launching a new flash

program command right after the assertion of the CBEIF flag.

```
1    LDX     #(PGM_ADDR_START+PGM_SIZE) ;Load X with last addr+1

2    STX     TMP_VAR                    ;Store last programmed addr+1 at

tmp_w1 var

3    LDX     #PGM_ADDR_START            ;Load X with start addr

4

5  LOOPGM:                              ;Loop Program

6    BRCLR   FSTAT, #$80, *             ;Wait for buffer empty (CBEIF = 1)

7

8    ; -> Time delay of 3 Osc clock periods must be inserted at this point.
```

```
9

10   MOVW    #DATA      2,X+           ;Write DATA to address pointed
by

index X

11   MOVB    #FCMD_PGM  FCMD           ;Write PGM command code to FCMD

register

12   MOVB    #$80       FSTAT          ;Launch command

13   CPX     TMP_VAR


14   BLO     LOOPGM
```

The minimum time delay for the code above can be found by the equation

below:

Time delay (in Bus clock cycles) = 3*(Fbus/Fosc) - (bus clock cycles

need by BRCLR at line 6) - (bus clock cycles needed by MOVW at line 10)

Considering that the BRCLR instruction at line 6 takes 3 bus clock cycles after the assertion of the CBEIF and that the MOVW takes 3 bus clock cycles to be executed, the equation above can be written as:

Time delay (in Bus clock cycles) = 3*(Fbus/Fosc) - 6

For the case of Fosc=4MHz and Fbus=12MHz, a time delay of 3*(12MHz/4MHz) - 6 = 3 bus clock periods is needed. So, at least 3 NOP instructions must be inserted at line 8 for proper operation, in this example.

Depending on the configuration of Bus clock frequency and oscillator clock frequency, and due to the actual code used in the application, the required time delay of 3 oscillator clock cycles may be already spent inside of the programming routine and the problem will not be detected.

Under certain conditions described by the equations and explanation above, adding NOPs may not be required.

PIM: Edge-sensitive mode of IRQ-pin may cause incorrect interrupt vector fetch — MUCts03997

Description

Where the IRQ interrupt is being used in edge-sensitive mode and a lower priority interrupt is already pending when an IRQ edge event occurs, if the IRQ edge event occurs inside a very narrow (<3ns) window just as the pending interrupt vector is being fetched, then a different vector other than that relating to either the pending interrupt or IRQ will be taken.

In the case that a programmed interrupt vector is fetched both the originally pending interrupt and the IRQ interrupt request will remain pending and will be serviced once the erroneously called service routine has completed (and RTI has been executed).

In the case that the incorrect vector fetch is from an unprogrammed vector table entry (i.e. erased state = 0xFFFF) then erroneous execution from the start of the register space will occur most often resulting in an illegal memory access reset, COP reset or Unimplemented Instruction Trap occurring.

In the less likely case that one of the three reset vectors is incorrectly fetched then execution will jump to the appropriate reset code.

The following vectors are not affected will not cause erroneous behavior if pending:

$F0 - RTI

$F6 - SWI

$E2 – ECT channel 6

$B2 – CAN0 Rx

$72 – XGATE software trigger 0

This issue is limited to the edge-sensitive mode of the IRQ input only - applications not using IRQ interrupts or configured for level-sensitive IRQ input are not affected.

There is no issue where a pending interrupt has higher priority than the IRQ request.

Workaround

Where using IRQ in edge-sensitive mode then configure the interrupt priority levels of all interrupts being used to ensure that the IRQ request always has the lowest priority.

For new designs, where possible use the IRQ input in level-sensitive mode or alternatively use a key-interrupt port.

There are a number of ‘best practices’ and features of the S12X which can help minimize the impact of this errata in the case of it occurring:

1) As ‘best practice’ initialize all unused and unimplemented/reserved interrupt vector table locations to point to a dummy interrupt service routine (terminated with an RTI).

2) Where possible, check for appropriate asserted flags in interrupt service routines and return / flag a system error if no request flag is set.

3) Support is provided on the MCU for managing the following system conditions:

* COP watchdog reset
* Illegal access reset
* Unimplemented instruction trap

For ‘best practice’ the application's software should be written to recover from any of these conditions in a controlled manner.

4) In the case of erroneous code execution jumping to unused Flash the typical practice of filling all unused Flash and RAM space with the op-code for the SWI instruction will help manage this. SWI exception routine should be written in this case to manage this event.

ECT: Channel 0 - 3 Input Capture interrupts inhibited when BUFEN=1, LATQ=0 and NOVWx=1 — MUCts04095

Description

Channel 0 – 3 Input Capture interrupts are inhibited when BUFEN=1, LATQ=0 and NOVWx=1 if an Input Capture edge occurs during or between a read of TCx and TCxH or between a read of TCx/TCxH and clearing of CxF.

Details:

When any of the buffered input capture channels 0 - 3 are configured for buffered/queue mode (BUFEN=1, LATQ=0) each of the channel's input capture holding registers and each channel's associated pulse accumulator and its holding register are enabled. When the input capture channel is enabled by writing to a channel's EDGxB and EDGxA bits, both the input capture and input capture holding register are considered empty. The first valid edge received after enabling a channel will latch the ECT's free running counter into the input capture register (TCx) without setting the channel's associated CxF

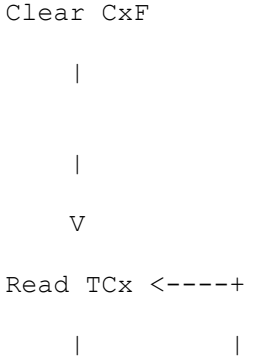
interrupt flag. The second valid edge received will transfer the value of the input capture register, TCx, into the channel's TCxH holding register, latch the current value of the free running timer into the input capture register and set the channel's associated CxF interrupt flag. In this condition, both the TCx and TCxH registers are considered 'full'.

If a corresponding channel's NOVWx bit in the ICOVW register is set, the capture register or its holding register cannot be written by a valid edge at the input pin unless they are first emptied by reading

the TCx and TCxH registers. The act of reading the TCx and TCxH registers and clearing the channel's associated CxF interrupt flag involves three separate operations. Two 16-bit read operations and an 8-

bit write operation.

If a channel's associated CxF interrupt flag is cleared before reading the TCx and TCxH registers and if a valid input edge occurs during or between the reading of the capture and holding register, a channel's associated CxF interrupt flag will no longer be set as the result of valid input edges. For example:



```
    |<--------+--- Valid Input Edge Occurs
    V         |
Read TCxH <---+
```

If the TCx and TCxH registers are read before a channel's associated CxF interrupt flag is cleared and if a valid input edge occurs between the reading of TCx/TCxH and the clearing of a channel's associated CxF interrupt flag, a channel's associated CxF interrupt flag will no longer be set as the result of valid input edges. For example:

```
Clear CxF
    |
    |
    V
Read TCx
    |
    |<------------ Valid Input Edge Occurs
    V
Read TCxH
```

Systems that service the interrupt request and read the TCx and TCxH registers before the next valid edge occurs at a channel's associated input pin will avoid the conditions under which the errata will occur.

Workaround

A simple workaround exists for this errata:

1. Clear the input capture channel’s associated CxF bit.

2. Disable the input capture function by writing 0:0 to a

   channel’s EDGxB and EDGxA bits.

3. Read TCx

4. Read TCxH

5. Re-enable the input capture function by writing to a channel’s EDGxB

   and EDGxA bits.

Code Example:

```
unsigned char ICSave;

unsigned int TC0Val;
unsigned int TC0HVal;

ICSave = TCTL4 & 0x03;  /* save state of EDG0B and EDG0A */
TFLG1 = 0x01;           /* clear ECT Channel 0 flag */
TCTL4 &= 0xfc;          /* disable Channel 0 input capture function */
TC0Val = TC0;           /* Read value of TC0 */
TC0HVal = TC0H;         /* Read value of TC0H */
TCTL4 |= ICSave;        /* Restore Channel 0 input capture function */
```

PWM: Wrong output level after shutdown restart in 16bit concatenated channel mode — MUCts04135

Description

When the PWM is used in 16-bit (concatenation) channel and the emergency

shutdown feature is being used, after de-asserting PWM channel 7

(note:PWMRSTRT should be set) the PWM channels do

not show the state which is set by PWMLVL bit when the 16-bit counter is

non-zero.

Workaround

If emergency shutdown mode is required:

In 16-bit concatenation mode, user can disable the related PWM

channels and set the corresponding general-purpose IO to be the PWM

LVL value. After a intend period, restart the PWM channels.

PWM: Wrong output value after restart from stop or wait mode — MUCts04136

Description

In low power modes (P-STOP/STOP/WAIT mode) and during PWM7

de-assert and when PWM counter reaching 0, the PWM channel outputs

cannot keep the state which is set by PWMLVL bit.

Workaround

Before entering low power modes, user can disable the related PWM channels and set the corresponding general-purpose IO to be the PWM LVL value. After a intend period, restart the PWM channels.

ECT_16B8C: Output compare pulse is inaccurate MUCts04155

Description

The pulse width of an output compare (which resets the free running counter when TCRE = 1) will measure one more bus clock cycle than expected.

Workaround

The specification has been updated. Please refer to revision 02.05 (04

May 2010) or later.

In description of bitfield TCRE in register TSCR2,a note has been added:

TCRE=1 and TC7!=0, the TCNT cycle period will be TC7 x "prescaler

counter width" + "1 Bus Clock". When TCRE is set and TC7 is not equal to

0, then TCNT will cycle from 0 to TC7. When TCNT reaches TC7 value, it

will last only one bus cycle then reset to 0.

FTX: Flash Command influenced by Backdoor Key write MUCts04232

Description

When executing a flash erase verify (0x05) command sequence to a flash

block different from the block where the backdoor keys are written to,

both blocks will be erase verified. Any programmed location in either

block will terminate the operation preventing the FSTAT.BLANK flag from

setting.

When executing a flash data compress (0x06) command sequence to a flash

block different from the block where the backdoor keys are written to,

a given number of words from both blocks will be compressed, this

number will be equal to the value written at last key's address. The signature from the block containing the backdoor keys will affect the signature returned in the FDATA register.

When executing a flash program (0x20) command sequence to a flash block

different from the block where the backdoor keys are written to, both blocks will be programmed at the same relative address with the unselected block being programmed to a data value equal to the last key

written. Setting protection at the location in the block where the backdoor keys are written will not prevent the flash command from executing.

When executing a flash sector erase (0x40) command sequence to a flash block different from the block where the backdoor keys are written to, both blocks will receive the erase at the sector address provided in the flash sector erase command sequence. Setting protection in the location where the backdoor keys are written to will not prevent the flash command from executing.

When executing a flash mass erase (0x41) command sequence to a flash block different from the block where the backdoor keys are written to, both blocks will be erased. Setting protection in the block where the

backdoor keys are written to will not prevent the flash command from executing.

The flash sector erase abort (0x47) command is not impacted as all

active sector erase operations will be terminated if successfully aborted.

Workaround

Write 0x30 to FSTAT register (ACCERR = 1, PVIOL = 1) prior to executing any flash command sequence when backdoor keys have been written. This step can be done in conjunction with or instead of checking the FSTAT register as shown in the flash command sequence flow in the reference manual.

MC9S12XDP512, Mask 0M23S

Introduction

This errata sheet applies to the following devices:

MC9S12XDP512, MC9S12XDT512, MC9S12XA512, MC9S12XDT384, MC9S12XDQ526, MC9S12XDT256, MC9S12XD256, MC9S12XB256, MC9S12XA256, MC9S12XDG128, MC9S12XD128, MC9S12XA128, MC9S12XB128

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 1K79X. 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 "0201" indicates the first week of the year 2002.

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.

Errata System Tracking Numbers

MUCtsXXXXX is the tracking number for device errata. It can be used with the mask set and date code to identify a specific erratum.

Errata Summary

Errata Number	Module affected	Brief Description	Work-around
MUCts01812	s12x_cpu	False tagged breakpoint hits may be reported by the DBG module	YES
MUCts01816	s12x_dbg	Back to back accesses to DBG trace buffer may return incorrect data	YES
MUCts01818	s12x_bdm	BDM hardware read command may be corrupted by entering stop mode	NO
MUCts01974	pim_9xd	PIM: ECLK divider can be activated in emulation modes	YES
MUCts02366	s12x_mmc	Incorrect 23-Bit Program Counter Generated for DBG on global accesses	YES
MUCts02409	eetx	EEPROM: Protection Disabled on EPROT Read During Command Write Sequence	YES
MUCts02539	spi	SPI: Slave entering stop in wait mode, pending rx data not rejected	YES
MUCts02582	s12x_dbg	S12X_DBG: Indexed jump loop1 mode trace buffer entries may be missed	YES
MUCts02667	spi	SPI in slave mode (CPHA=0) and SS line not deasserted between transmissions may lead to corrupted rx data	YES
MUCts02988	xgate	XGATE: Carry-Flag may be falsely set by SSEM instruction	YES
MUCts03017	ect_16b8c	ECT: TCNT counter resets in Input Capture Mode	YES

MUCts03018	ect_16b8c	ECT: CxF flag clears following a read to TCx on an OC event with TFFCA=1	YES
MUCts03019	ect_16b8c	ECT: CxF flag clears following a wite to TCx on an IC event with TFFCA=1	YES
MUCts03037	ect_16b8c	ECT: Forced OC on PT7 occured even when TIOS7 = 0	NO
MUCts03039	ect_16b8c	ECT:Faulty OC event with OM/OL=0, OC7Mx=1, TIOSx=1	YES
MUCts03112	s12x_bdm	BDM: Incomplete Memory Access on misaligned access due to BDM features	YES
MUCts03332	ftx	FTX: Blind Spot in Data Compress Command Algorithm	YES
MUCts03390	atd_10b16c	ATD: Abort of an A/D conversion sequence with write to ATDxCTL0/1/2/3 may not work	YES
MUCts03391	atd_10b8c	ATD: Abort of an A/D conversion sequence with write to ATDxCTL0/1/2/3 may not work	YES
MUCts03453	mscan	MSCAN: Corrupt ID may be sent in early-SOF condition	YES
MUCts03617	s12x_dbg	DBG 'outside range' mode databus qualification ignored	YES
MUCts03620	s12x_dbg	DBG No address match if next transaction is misaligned word access	YES
MUCts03621	s12x_dbg	DBG Range Mode TAGB and TAGD influence in range modes	YES
MUCts03634	eetx	eetx4k An interrupt following immediately after execution of a STOP instruction may disable read access to the EEPROM array	YES
MUCts03686	atd_10b8c	ADC: conversion does not start with 2 consecutive writes to ATDCTL5	YES
MUCts03689	atd_10b16c	ADC: conversion does not start with 2 consecutive writes to ATDCTL5	YES
MUCts03761	s12x_dbg	DBG: State flags and counter corrupted by simultaneous arm and disarm	YES
MUCts03870	s12x_cpu	CPU: Breakpoint missed at simultaneous taghits	YES
MUCts03977	pwm_8b8c	PWM: Emergency shutdown input can be overruled	YES
MUCts03996	ftx	Flash: Burst programming issue if bus clock frequency is higher than oscillator clock frequency	YES
MUCts03997	pim_9xd	PIM: Edge-sensitive mode of IRQ-pin may cause incorrect interrupt vector fetch	YES
MUCts04095	ect_16b8c	ECT: Channel 0 - 3 Input Capture interrupts inhibited when BUFEN=1, LATQ=0 and NOVWx=1	YES
MUCts04135	pwm_8b8c	PWM: Wrong output level after shutdown restart in 16bit concatenated channel mode	YES

MUCts04136	pwm_8b8c	PWM: Wrong output value after restart from stop or wait mode	YES
MUCts04155	ect_16b8c	ECT_16B8C: Output compare pulse is inaccurate	YES
MUCts04232	ftx	FTX: Flash Command influenced by Backdoor Key write	YES

False tagged breakpoint hits may be reported by the DBG module — MUCts01812

Description

If the device executes the BACKGROUND command (received over BDM) in the

cycle when it is about to execute a tagged instruction the DBG module

may indicate that a tagged breakpoint was hit even though the tagged

instruction itself was not executed yet.

Workaround

If the DBG module indicates a tagged breakpoint hit after the BACKGROUND

command was issued, verify contents of PC to determine whether the

tagged breakpoint was hit or not.

Back to back accesses to DBG trace buffer may return incorrect data — MUCts01816

Description

When the trace buffer is unlocked for reading and data is read from the

trace buffer register in the next bus cycle (back-to-back accesses),

wrong data may be read out.

Workaround

When using the core or Xgate to read the contents of the trace buffer,

do not use back-to-back bus accesses.

OR

Only use the BDM to access contents of the trace buffer.

BDM hardware read command may be corrupted by entering stop mode MUCts01818

Description

The BDM can generate invalid data when the BDM is processing a hardware

(HW) read command with the handshake feature of the BDM enabled and the

CPU enters STOP mode with the device not reaching system STOP mode

because of XGATE activity or set-up (as described below).

The BDM read data is cleared in the BDM shift register because the CPU

enters STOP mode but the corresponding ACK pulse of the HW read command

is not prevented. The occuring ACK pulse indicates to the host that

valid data is ready, which is not the case. Instead of valid data a data

value of $FF will be received.

This behaviour occurs only if the CPU enters into STOP mode after a BDM

read access has occured on the internal bus and before the corresponding

ACK pulse of the HW read command is sent.

It only occurs when the CPU enters into STOP mode while XGATE is not idle or the XGFACT bit is set. In this case only CPU clocks are stopped.

This behaviour does not occur when a HW read command is sent while the CPU is already in STOP mode.

It does not occur if the device reaches system STOP mode (all clocks stopped), which happens when the CPU enters STOP mode and XGATE is in the idle state and the XGFACT bit is cleared.

This is a debug only issue with a low probability of occurance because:

1) The timing window during which the CPU transition (from run mode to STOP mode) must occur is very short. (Timing window means the time from the BDM read access on the internal bus until the BDM internal ACK pulse control signal is set - in the case of no clock switching this is two bus clock cycles).

2) Only a transition of the CPU from run mode to STOP mode during the time window will cause the issue.

3) The issue occurs only if system STOP is not reached.

Workaround

None

PIM: ECLK divider can be activated in emulation modes — MUCts01974

Description

The ECLK does not run at bus clock rate in emulation modes if an EDIV value greater than zero is selected. In this case the ECLK will be divided as specified or stopped if NECLK is set. This may affect emulation systems which rely on constant ECLK rate.

Workaround

Always keep the NECLK and EDIV value at zero in case constant ECLK rate

is required.

In order to pass the required divider settings to the emulation system,

the EDIV settings should alternatively be mapped to the reserved bits

ECLKCTL[3:2]. The emulation system should utilize these bits instead of

EDIV to control the ECLK generator for the target application. This

software modification needs to be undone when the application runs

without the emulator. The emulator can monitor values written into

ECLKCTL[1:0] and notify the user when non-zero value can cause the

emulator to function incorrectly.

Note:

Please note that this erratum only concerns the ECLK signal. Signal

ECLKX2 is not affected by EDIV values when operating in emulation modes.

Incorrect 23-Bit Program Counter Generated for DBG on global accesses — MUCts02366

Description

An incorrect trace buffer entry occurs when the DBG module attempts to trace a Change Of Flow (COF) instruction that follows a global access. This only happens for COF instructions where the source address should be stored to the trace buffer.

This errata only affects the functionality of the DBG module, causing incorrect DBG trace buffer entries.

During global instruction accesses, the program counter bits [22:16] are derived from the global address bus bits [22:16] which, at that moment, contain the address of the global access.

The program counter should always provide the global opcode address and should be independent of addresses associated with memory accesses. In the errata case, the page part of the opcode address contains the data access page address.

The program counter bits [15:0] are not affected by this problem.

Workaround

In debug mode, put a NOP before the global instruction.

EEPROM: Protection Disabled on EPROT Read During Command Write Sequence — MUCts02409

Description

If EPROT EOPEN bit is set with EPROT EPDIS bit clear (i.e. protection is

on) and EPROT is read during a command write sequence then protection is

disabled until the next reset or write of the EPROT register.

Workaround

Do not read EPROT register during a command write sequence.

SPI: Slave entering stop in wait mode, pending rx data not rejected — MUCts02539

Description

In slave mode, pending data in the receive shift register is not

rejected when entering Wait mode with the SPISWAI bit set.

When the SPI stops on entering Wait mode (executing WAI with SPISWAI bit

set), data pending in the receive shift register should be rejected

(lost). This is to prevent pending data in the receive shift register

from being corrupted by SCK cycles while halted in Wait mode.

Workaround

Workaround #1: Ensure that the receive buffer queue is empty before

entering Wait mode.

Workaround #2: Make sure SCK does not toggle while in Wait mode.

Workaround #3: Do not use the 'stop in Wait mode' feature.

S12X_DBG: Indexed jump loop1 mode trace buffer entries may be missed MUCts02582

Description

Loop1 Mode inhibits consecutive duplicate source address entries that would typically be stored in most tight looping constructs. It does not inhibit repeated entries of destination addresses. However, the logic prevents the storage of valid indexed jump destination addresses if consecutive indexed jumps have the same opcode address.

Considering the code below; the first occurence of JMP 0,X stores MARK5

to the trace buffer. Since the code returns to MARK4 with an unconditional branch, the next trace buffer entry should be the next destination address of JMP 0,X (MARK6 the second time around).

This entry is missed because further COFs from MARK4 are masked out.

```
        LDX     #MARK5
MARK4   JMP     0,X
        NOP
MARK5   LDX     #MARK6
        BRA     MARK4
MARK6   NOP
```

This does not affect XGATE loop mode tracing

Workaround

If code contains consecutive indexed jumps from an idential opcode address, then normal mode tracing can be temporarily used to confirm the incidence of consecutive indexed jumps from the same address.

Subsequently loop1 mode can be enabled to continue further debugging.

SPI in slave mode (CPHA=0) and SS line not deasserted between transmissions may lead to corrupted rx data — MUCts02667

Description

When the SPI is in slave mode with both the SPI data register (SPIDR) and the receive shift register containing pending data, on reception of further data the contents of the shift register should be discarded and the new data byte shifted in. Reading the SPIDR should not cause the shift register contents to be transferred to the SPIDR until the latest data byte is completely received.

In the erratum condition, the shift register pending state is not discarded and reading the SPIDR during the reception of a new data byte will cause the shift register contents to be immediately transferred to the SPIDR register causing data corruption.

This condition only occurs in slave mode with CPHA = 0 and if SS is not de-asserted between transmissions.

Workaround

Workaround #1: Deassert SS for minimum idle time (0.5*Tsck) between transmissions.

Workaround #2: Ensure that the receive buffer queue is empty before starting a new transmission.

XGATE: Carry-Flag may be falsely set by SSEM instruction MUCts02988

Description

If the S12X_CPU and the XGATE attempt to lock a semaphore at the same time the S12X_CPU will obtain the semaphore, but the Carry-Flag will be set for the XGATE (falsely indicating that the XGATE has locked the semaphore).

Workaround

Execute two consecutive "SSEM" instructions to set a semaphore. Ignore result of the first "SSEM" instruction. Carry-Flag will indicate correctly whether XGATE has allocated the semaphore after the second "SSEM" instruction is executed.

ECT: TCNT counter resets in Input Capture Mode MUCts03017

Description

Normal Operation:

Timer Free Running Counter, also called as Timer Counter resets to 0x0000 on a Channel 7 Output Compare event when Timer Counter Reset Enable (TCRE) bit of TSCR2 registers is set to 1. This assumes that TIOS7 bit of TIOS register is set to a 1 (Output Compare mode).

Issue: Erroneously this behaviour was found to occur even when TIOS7 bit of TIOS register was set to a "0" (Input Capture mode)

Workaround

Reseting Free Running Counter alias Timer Counter can be avoided by setting Timer Counter Reset Enable (TCRE) bit of TSCR2 register to a "0".

ECT: CxF flag clears following a read to TCx on an OC event with TFFCA=1 — MUCts03018

Description

Problem:

Normal Operation:

With TIOS = 1 (Output Compare mode) and TFFCA = 1 (Fast Flag Clearing mechanism) a write to TCx register following an output compare event clears the flag.

Erroneous Operation:

A read to TCx register following an Output Compare event clears Cxf flag in TFLG1 register.

Workaround

Customer should avoid reading TCx register following an Ouput Compare event.

ECT: CxF flag clears following a wite to TCx on an IC event with TFFCA=1 — MUCts03019

Description

Normal Operation:

With TIOS = 0 (Input Capture (IC) mode) and TFFCA = 1 (Fast Flag Clearing mechanism) a read to TCx register following an Input Capture (IC) event clears CxF (flag) bit of TFLG1 register.

Erroneous Operation:

A write to TCx register following an Input Capture (IC) event clears Cxf (flag) bit of TFLG1 register.

Workaround

Customer should avoid writing to TCx register following an Input Capture (IC) event.

ECT: Forced OC on PT7 occured even when TIOS7 = 0 — MUCts03037

Description

Correct Operation:

Forced Output Compare (OC) operation on channel 7 requires TIOS7 bit be

set to a “1” for a successful Ouput Compare (OC) event.

Faulty Operation:

Forced Ouput Compare (OC) action was noticed on Channel 7 inspite of

TIOS7 being set to a “0”.

Workaround

None.

ECT:Faulty OC event with OM/OL=0, OC7Mx=1, TIOSx=1 — MUCts03039

Description

Correct Operation:

When timer is enabled (TEN bit of TSCR1 register is set to 1) and Output

Compare functionality is selected (TIOSx bit of TIOS register is set) on

a match (TCNT = Tcx) an output compare event(dictated by OM/OL bits of

TCTL1/TCTL2 register) is generated. Please not that the Output Compare

is registered internally.

Faulty Operation:

Consider a case, when an normal Output Compare event (driving port to 1)

was performed. You intend to dis-connect output compare logic from pin logic. Hence you configure (OM/OL bits to a “0”), but your TIOSx bit is still set to a “1”. Also, you intend to use masking feature of output compare so you set TIOS7=1 and you have OC7Mx bit set to a “1”. If the next output compare match(TCNT = Tcx) happens to be normal output compare event, the default state of internal register (0) was noticed on the channel.

In reality the output port should not have toggled since OM/OL were set to 0.

This is an erroneous behaviour.

Workaround

Make sure that OC7Mx bit is set to "0" if not using OC7M feature for that channel. If using masking feature, please remember to set OC7Mx bit to "0" as soon as you have a channel 7 Compare event.

BDM: Incomplete Memory Access on misaligned access due to BDM features MUCts03112

Description

If a misaligned word write access is directly followed by an attempted entry into active BDM, then the second byte access may be performed on a different target due to the memory map switching for BDM active.

Depending on the type of access this can lead to the following

situations...

1. When writing in the address range from $7F_FEFF to $7F_FFFD in the external space the MMC splits the accesses into two 8-bit accesses. It is able to complete the first access, but just before the second access the BDM firmware becomes mapped into this address location and the second byte access goes to the firmware.
This can only occur if the last cycle of the access instruction is a word write to an odd address AND the instruction is followed by BGND or a tagged breakpoint.

2. On a misaligned word write with a global store (GSTD,GSTS,GSTX,GSTY) in the address range $XX_0FFF to $XX_3FFD, the first split access is performed correctly in the external space, but in the second byte cycle the MMC is instructed to interpret the address coming from the CPU as a local address, which is the internal RAM.
This can only occur if the CPU executes a global store instruction which writes to an odd address in the last instruction cycle AND the instruction is followed by BGND or a tagged breakpoint.

3. Generally on a misaligned global store (GSTD,GSTS,GSTX,GSTY) the lower 16-bits of the global address are interpreted as a local address for the second byte access. This can result in unintended accesses to registers or external RAM.

Workaround

To prevent scenarios (2) and (3) of the errata occurring, avoid setting

breakpoints after a GSTD.

When not debugging, remove all instances of BGND from code.

FTX: Blind Spot in Data Compress Command Algorithm — MUCts03332

Description

If the range of Flash addresses to be compressed is 32K or greater, the

data at one of the addresses will be effectively ignored. The address

affected is 32K from the upper address read in the data compress

algorithm, e.g., for an address range of 32K, the first data read in the

algorithm will not affect the final signature provided by the algorithm.

Workaround

Limit range of addresses to be compressed to less than 32K addresses.

Execute multiple data compress commands to compress larger Flash address

ranges.

ATD: Abort of an A/D conversion sequence with write to ATDxCTL0/1/2/3 may not work — MUCts03390

Description

Starting a conversion with a write to ATDxCTL5 or on an external trigger

event, and aborting immediately afterwards with a write to ATDxCTL0,

ATDCTL1, ATDxCTL2 or ATDxCTL3 can fail to stop the conversion process.

Workaround

Only write to ATDxCTL4 to abort an ongoing conversion sequence.

Use the recommended start and abort procedures from the Block Guide.

Section : Initialization/Application Information

Subsection: Setting up and starting an A/D conversion

Subsection: Aborting an A/D conversion

ATD: Abort of an A/D conversion sequence with write to ATDxCTL0/1/2/3 may not work — MUCts03391

Description

Starting a conversion with a write to ATDxCTL5 or on an external

trigger event, and aborting immediately afterwards with a write to ATDxCTL0, ATDCTL1, ATDxCTL2 or ATDxCTL3 can fail to stop the conversion process.

Workaround

Only write to ATDxCTL4 to abort an ongoing conversion sequence.

Use the recommended start and abort procedures from the Block Guide.

Section : Initialization/Application Information

Subsection: Setting up and starting an A/D conversion

Subsection: Aborting an A/D conversion

MSCAN: Corrupt ID may be sent in early-SOF condition — MUCts03453

Description

The initial eight ID bits will be corrupted if a message is set up for transmission during the third bit of INTERMISSION and a dominant bit is sampled leading to an early-SOF*.

The CRC is calculated from the resulting bit stream so that the receiving nodes will still validate the message.

An early-SOF condition may only occur if the oscillators in the network

operate at a tolerance range which could lead to a cumulated phase error

after 11 bit times larger than phase segment 2.

In case arbitration is lost during transmission of the corrupt

identifier, a non-corrupted ID will be sent with the next attempt if the

transmit request remains active.

*The CAN protocol condition referred to as 'early-SOF' in this erratum

is detailed in "Bosch CAN Specification Version 2.0" Part A, section 9,

and a Note to section 3.2.5 INTERFRAME SPACING – INTERMISSION in Part B.

Workaround

Due to increased oscillator tolerance a transmission start in the third

bit of intermission is possible and allowed. The errata can be avoided

when calculating the maximum oscillator tolerance of the overall CAN

system. The phase error after 11 bit times due to the oscillator

tolerance should be smaller than phase segment 2.

If an early-SOF cannot be avoided the following methods will provide

prevention:

- Assigning the same value to all upper eight ID bits in the network
- Allocating dedicated data length codes (DLC) to every identifier

used

in the network and checking for correspondence after reception

- Assigning only IDs (x) which do not consist of a combination of other

assigned IDs (y,z) and using the acceptance filters to reject

erroneous messages, i.e.

- for standard frames: IDx[11:0] != {IDy[11:3], IDz[2:0]}

- for extended frames: IDx[28:21] != {IDy[28:21],IDz[20:0]}

DBG 'outside range' mode databus qualification ignored — MUCts03617

Description

When using a comparator pair for a range comparison, the databus can also be used for qualification by using the comparator A/C data and data mask registers. This does not work correctly.

Scenario:

With CompA/CompB in 'outside range' mode an access to a memory location

with an address above the high boundary of CompB always generates a comparator match. The data qualification is not carried out.

If the accessed memory is located below the address defined in CompA, the behavior is correct.

CompC/CompD behavior is the same.

Workaround

Using 2 comparator pairs configured for inside range mode an outside range match with databus qualification can be generated.

```
MAP
0x000000 |COMPA|
         |     | Inside Range A/B with DB qualification
ADDRESSX |COMPB|
         |     |
ADDRESSY |COMPC|
         |     | Inside Range C/D with DB qualification
0x7FFFFF |COMPD|
```

DBG No address match if next transaction is misaligned word access — MUCts03620

Description

Memory accesses in successive bus cycles must both be able to generate forced triggers if both accesses match.

If accesses occur in successive cycles whereby the second access is a misaligned word access, then a comparator match is lost. This could cause a forced breakpoint to be missed if the state sequencer is dependent on both the successive matches to reach final state.

Example...

```
        LDX    #WORD_MISALIGNED
        STD    0,X              ; First access M0->State2
```

```
        LDD    WORD_MISALIGNED   ; Second access M0->Final State
```

If the STD last bus cycle is a write and the LDD first bus cycle is a read then only one state sequencer transition occurs.

In range modes, this can occur when 2 different addresses within/outside the specified range are accessed in successive bus cycles, otherwise it can only happen if the same address is written and then read in successive bus cycles.

Workaround

Insert a NOP instruction before misaligned word accesses if they can follow accesses to the predefined comparator range.

```
        LDX    #WORD_MISALIGNED-1
        STD    0,X               ; First access in range M0->State2
        NOP                      ; NOP insertion
        LDD    WORD_MISALIGNED   ; Second access in range M0->Final State
```

DBG Range Mode TAGB and TAGD influence in range modes — MUCts03621

Description

The comparator A and C TAG bits are used to tag range comparisons for

the AB and CD ranges respectively. The comparator B and D TAG bits should have no effect in range modes. However the TAGB/TAGD bits do have
an effect.
If the A/C TAG bit is 0 but the paired B/D TAG bit is set, a valid match
may be missed.

Workaround

Clear TAGB when configured for forced range mode comparisons using CompAB

Clear TAGD when configured for forced range mode comparisons using CompCD

---

eetx4k An interrupt following immediately after execution of a STOP instruction may disable read access to the EEPROM array — MUCts03634

Description

An interrupt request immediately following execution of a STOP instruction may cause the EEPROM array to return incorrect data when read. The problem will only occur if all of the following conditions are
met:

1) S bit in the CCR register is cleared (stop mode enabled)
2) I bit in the CCR register is cleared (interrupts enabled)
3) A STOP instruction is executed by the microcontroller
4) An interrupt becomes pending between 0 and 0.5 bus clock cycles

after beginning of the STOP instruction execution.

In this case the microcontroller will wake-up correctly from the stop mode, but the EEPROM array will return incorrect data when a read operation is performed.

Access (read or write) to any of the EEPROM registers or write into the
EEPROM array (to start a command sequence) will restore the read access. The EEPROM will return the correct data afterwards.

Workaround

If the STOP instruction is executed while interrupts are enabled, read or write any of the EEPROM registers after wake-up from stop mode to ensure correct operation. Alternatively a write into the EEPROM array (as part of a command sequence) can be performed instead of the register access.

ADC: conversion does not start with 2 consecutive writes to ATDCTL5 — MUCts03686

Description

When the ATD is started with write to ATDCTL5
and, which is very unusual and not necessary,
within a certain period again started with write
to ATDCTL5. The conversion will not start at all.
This does only happen if the two consecutive writes to ATDCTL5 occur within one "ATD clock period". An ATD clock period is defined by a full

rollover of the ATD clock prescaler. That is for example PRS[4:0] = 2
-

> (2+1)*2 = within 6 bus cycles.

Workaround

Only write once to ATDCTL5 when starting a conversion.

Use the recommended start and abort procedures from the Block Guide.

Section : Initialization/Application Information Subsection: Setting up

and starting an A/D conversion Subsection: Aborting an A/D conversion

---

ADC: conversion does not start with 2 consecutive writes to ATDCTL5 — MUCts03689

Description

When the ATD is started with write to ATDCTL5

and, which is very unusual and not necessary,

within a certain period again started with write

to ATDCTL5. The conversion will not start at all.

This does only happen if the two consecutive writes to ATDCTL5 occur

within one "ATD clock period". An ATD clock period is defined by a full

rollover of the ATD clock prescaler. That is for example PRS[4:0] = 2
-

> (2+1)*2 = within 6 bus cycles.

Workaround

Only write once to ATDCTL5 when starting a conversion.

Use the recommended start and abort procedures from the Block Guide.

Section : Initialization/Application Information Subsection: Setting up

and starting an A/D conversion Subsection: Aborting an A/D conversion

---

DBG: State flags and counter corrupted by simultaneous arm and disarm — MUCts03761

Description

Simultaneous disarming (hardware) and arming (software) results in status and counter register (DBGSR, DBGCNT) corruption.

Hardware disarming initiated by an internal trigger or by tracing completion takes 2 clock cycles. If a write to DBGC1 with the ARM bit set occurs in the final clock cycle of this disarming process, arming is suppressed but the DBGSR register is initialized to state1 and DBGCNT is initialized to zero.

The result is that the DBG module is disarmed by hardware but DBGSR indicates state1.

NOTE: DBGC1 is typically only written to whilst armed to set the TRIG bit or update the COMRV bits to map different registers to the address map window.

Generally during debugging, after arming the DBG module and returning to application code a further write access of DBGC1 over the BDM

interface requires considerable time relative to application code execution, therefore in many cases the breakpoint may be reached before a DBGC1 update is attempted. Furthermore the probability of hitting the same cycle is seen to be very low.

Workaround

If the fault condition is caused by writing to DBGC1 to set the TRIG bit (to request an immediate breakpoint), then the application code may be rerun again without the attempted setting of TRIG. The software trigger (TRIG) is unnecessary in any case at the same point point in time as an internal hardware trigger occurs.

Development tool vendors should avoid COMRV updates while the DBG is armed.

Users that observe the problem due to development tool COMRV updates can add a NOP to code and rerun to shift the disarm cycle, thereby preventing a collision with the COMRV updates.

CPU: Breakpoint missed at simultaneous taghits MUCts03870

Description

The CPU execution priority encoder evaluates taghits and then generates a breakpoint if a taghit must lead to an immediate

breakpoint as determined by the DBG module.

If the DBG module indicates that this taghit leads to an immediate breakpoint then the CPU loads the execution stage with SWI or BGND thus generating the breakpoint.

At simultaneous taghits the lowest channel has priority.
If taghits on 2 channels simultaneously, whereby the lower channel tag must be ignored, but the higher channel tag must cause a breakpoint, then the breakpoint request is erroneously missed.
Thus if channel[1:0] taghits occur simultaneously whereby channel[0] must be ignored but channel[1] must cause a breakpoint, then the breakpoint request on channel[1] is erroneously missed and no breakpoint generated.

The DBG module recognises the taghit and the state sequencer transitions accordingly. Furthermore the taghit causes the DBG module to request a forced breakpoint, which means that a late CPU breakpoint occurs. If the tagged instruction is a single cycle instruction, the breakpoint occurs at the second instruction following the tagged instruction.
Otherwise the breakpoint occurs at the instruction immediately following the tagged instruction.

This bug requires that separate tags are placed on the same instruction. This is not a typical case when using exact tag addresses.

It is more relevant in debugging environments using range modes, where tags may cover a whole range. In this case a tagged range may cover a tag or another tagged range, making simultaneous taghits possible.

This is a debugging issue only.

Workaround

Do not attach multiple tags to the same exact address.
When attaching multiple tags to the same address by overlapping tag ranges in range modes, or covering a tag with a tag range in range modes, then the missed tag can be avoided by mapping the final state change to the lower channel number.

For example the case
Channel[0] tags a range; channel[3] tags an instruction within that range.
From DBG State1 Taghit[0] leads to State2. (DBGSCR1=$6)
From DBG State2 Taghit[3] leads to FinalState.(DBGSCR2=$5)
In State2 a simultaneous Taghit[3,0] scenario would miss the breakpoint.

This can be avoided by using the alternative DBG configuration
Channel[2] tags a range; channel[0] tags an instruction within that range.
From DBG State1 Taghit[2] leads to State2. (DBGSCR1=$3)
From DBG State2 Taghit[0] leads to FinalState.(DBGSCR2=$7)

PWM: Emergency shutdown input can be overruled MUCts03977

Description

If the PWM emergency shutdown feature is enabled (PWM7ENA=1) and PWM channel 7 is disabled (PWME7=0) another lower priority function available on the related pin can take control over the data direction. This does not lead to a problem if input mode is maintained. If the alternative function switches to output mode the shutdown function may unintentionally be triggered by the output data.

Workaround

When using the PWM emergency shutdown feature the GPIO function on the pin associated with PWM channel 7 should be selected as an input.

In the case that this pin is selected as an output or where an alternative function is enabled which could drive it as an output, enable PWM channel 7 by setting the PWME7 bit. This prevents an active shutdown level driven on the (output) pin from resulting in an emergency shutdown of the enabled PWM channels.

Flash: Burst programming issue if bus clock frequency is higher than oscillator clock frequency — MUCts03996

Description

If S12X is running at a bus clock frequency higher than the oscillator clock frequency (Fbus > Fosc), flash words may remain not programmed after a program burst sequence. Software methods can be used to avoid

this problem. If burst programming is not used, no errors occur.

The root cause of this issue is related to flash internal state machine

which needs about 2 to 3 oscillator clock periods to be ready to accept

a new flash command after the assertion of the CBEIF flag.

Note that this latency of flash state machine is related to the

assertion of CBEIF and the start of a new command sequence (i.e. a write

to a flash word). There is no latency after the assertion of the CCIF flag.

Workaround

Adding a time delay between the check of CBEIF flag and the start of the

next flash program command sequence will ensure that all words will be

programmed. Add a delay of (3 * (Fbus/Fosc)) Bus clock cycles after

CBEIF is set, that can be achieved by the addition of NOPs (one NOP

instruction takes one bus cycle to execute).

Note that since a flash word program operation takes much longer than 3

oscillator clock periods, there is no impact in the total programming

time of a long program burst sequence by adding a delay of 3 clocks.

The example below illustrates the proposed workaround:

Code below executes a program burst sequence by launching a new flash program command right after the assertion of the CBEIF flag.

```
1    LDX     #(PGM_ADDR_START+PGM_SIZE) ;Load X with last addr+1
2    STX     TMP_VAR                    ;Store last programmed addr+1
at
tmp_w1 var
3    LDX     #PGM_ADDR_START            ;Load X with start addr
4
5  LOOPGM:                              ;Loop Program
6    BRCLR   FSTAT, #$80, *             ;Wait for buffer empty (CBEIF
= 1)
7
8    ; -> Time delay of 3 Osc clock periods must be inserted at this
point.
9
10   MOVW    #DATA      2,X+            ;Write DATA to address pointed
by
index X
11   MOVB    #FCMD_PGM  FCMD            ;Write PGM command code to FCMD
register
12   MOVB    #$80       FSTAT           ;Launch command
13   CPX     TMP_VAR
14   BLO     LOOPGM
```

The minimum time delay for the code above can be found by the equation below:

Time delay (in Bus clock cycles) = 3*(Fbus/Fosc) - (bus clock

cycles

need by BRCLR at line 6) - (bus clock cycles needed by MOVW at line 10)

Considering that the BRCLR instruction at line 6 takes 3 bus clock cycles after the assertion of the CBEIF and that the MOVW takes 3 bus clock cycles to be executed, the equation above can be written as:

Time delay (in Bus clock cycles) = 3*(Fbus/Fosc) - 6

For the case of Fosc=4MHz and Fbus=12MHz, a time delay of 3*(12MHz/4MHz) - 6 = 3 bus clock periods is needed. So, at least 3 NOP instructions must be inserted at line 8 for proper operation, in this example.

Depending on the configuration of Bus clock frequency and oscillator clock frequency, and due to the actual code used in the application, the

required time delay of 3 oscillator clock cycles may be already spent inside of the programming routine and the problem will not be detected.

Under certain conditions described by the equations and explanation above, adding NOPs may not be required.

PIM: Edge-sensitive mode of IRQ-pin may cause incorrect interrupt vector fetch — MUCts03997

Description

Where the IRQ interrupt is being used in edge-sensitive mode and a lower priority interrupt is already pending when an IRQ edge event occurs, if the IRQ edge event occurs inside a very narrow (<3ns) window just as the pending interrupt vector is being fetched, then a different vector other than that relating to either the pending interrupt or IRQ will be taken.

In the case that a programmed interrupt vector is fetched both the originally pending interrupt and the IRQ interrupt request will remain pending and will be serviced once the erroneously called service routine has completed (and RTI has been executed).

In the case that the incorrect vector fetch is from an unprogrammed vector table entry (i.e. erased state = 0xFFFF) then erroneous execution from the start of the register space will occur most often resulting in an illegal memory access reset, COP reset or Unimplemented Instruction Trap occurring.

In the less likely case that one of the three reset vectors is incorrectly fetched then execution will jump to the appropriate reset code.

The following vectors are not affected will not cause erroneous behavior if pending:

$F0 - RTI

$F6 - SWI

$E2 – ECT channel 6

$B2 – CAN0 Rx

$72 – XGATE software trigger 0

This issue is limited to the edge-sensitive mode of the IRQ input only –

applications not using IRQ interrupts or configured for level-sensitive IRQ input are not affected.

There is no issue where a pending interrupt has higher priority than the IRQ request.

Workaround

Where using IRQ in edge-sensitive mode then configure the interrupt priority levels of all interrupts being used to ensure that the IRQ request always has the lowest priority.

For new designs, where possible use the IRQ input in level-sensitive mode or alternatively use a key-interrupt port.

There are a number of ‘best practices’ and features of the S12X which can help minimize the impact of this errata in the case of it occurring:

1) As ‘best practice’ initialize all unused and unimplemented/reserved interrupt vector table locations to point to a dummy interrupt service routine (terminated with an RTI).

2) Where possible, check for appropriate asserted flags in interrupt service routines and return / flag a system error if no request flag

is

set.

3) Support is provided on the MCU for managing the following system conditions:

* COP watchdog reset

* Illegal access reset

* Unimplemented instruction trap

For ‘best practice’ the application's software should be written to recover from any of these conditions in a controlled manner.

4) In the case of erroneous code execution jumping to unused Flash the typical practice of filling all unused Flash and RAM space with the op-

code for the SWI instruction will help manage this. SWI exception routine should be written in this case to manage this event.

ECT: Channel 0 - 3 Input Capture interrupts inhibited when BUFEN=1, LATQ=0 and NOVWx=1 — MUCts04095

Description

Channel 0 – 3 Input Capture interrupts are inhibited when BUFEN=1, LATQ=0 and NOVWx=1 if an Input Capture edge occurs during or between a read of TCx and TCxH or between a read of TCx/TCxH and clearing of CxF.

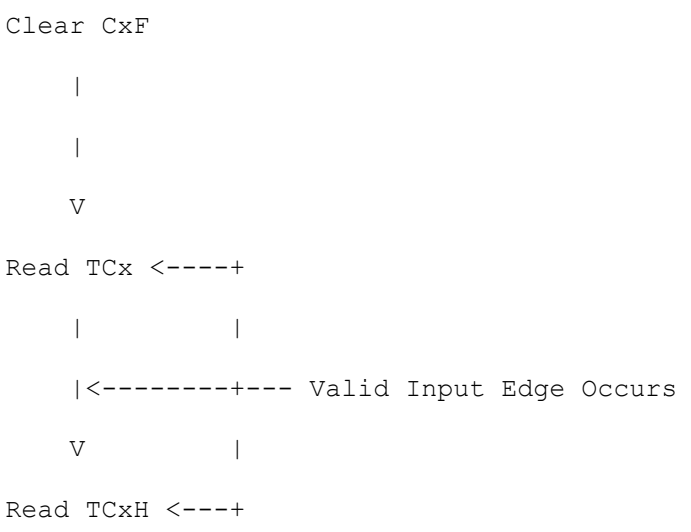
Details:

When any of the buffered input capture channels 0 - 3 are configured for buffered/queue mode (BUFEN=1, LATQ=0) each of the channel's input capture holding registers and each channel's associated pulse accumulator and its holding register are enabled. When the input capture channel is enabled by writing to a channel's EDGxB and EDGxA bits, both the input capture and input capture holding register are considered empty. The first valid edge received after enabling a channel will latch the ECT's free running counter into the input capture register (TCx) without setting the channel's associated CxF interrupt flag. The second valid edge received will transfer the value of the input capture register, TCx, into the channel's TCxH holding register, latch the current value of the free running timer into the input capture register and set the channel's associated CxF interrupt flag. In this condition, both the TCx and TCxH registers are considered 'full'.

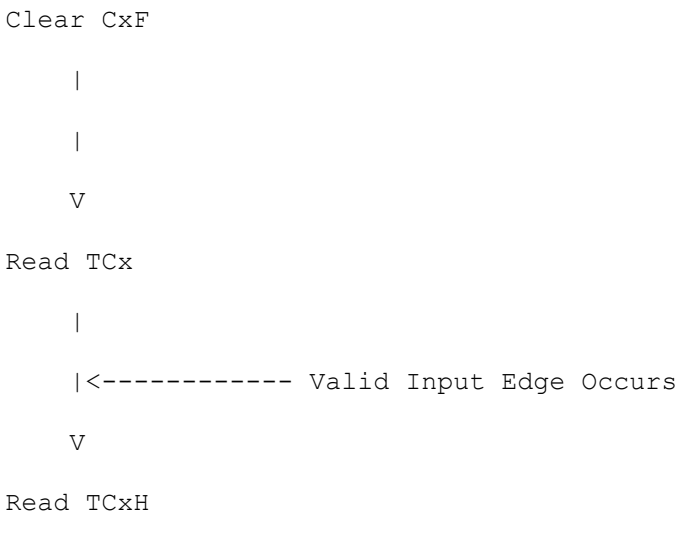
If a corresponding channel's NOVWx bit in the ICOVW register is set, the capture register or its holding register cannot be written by a valid edge at the input pin unless they are first emptied by reading the TCx and TCxH registers. The act of reading the TCx and TCxH registers and clearing the channel's associated CxF interrupt flag involves three separate operations. Two 16-bit read operations and an 8-
bit write operation.

If a channel's associated CxF interrupt flag is cleared before reading the TCx and TCxH registers and if a valid input edge occurs during or between the reading of the capture and holding register, a channel's

associated CxF interrupt flag will no longer be set as the result of valid input edges. For example:



If the TCx and TCxH registers are read before a channel’s associated CxF interrupt flag is cleared and if a valid input edge occurs between the reading of TCx/TCxH and the clearing of a channel’s associated CxF interrupt flag, a channel’s associated CxF interrupt flag will no longer be set as the result of valid input edges. For example:



Systems that service the interrupt request and read the TCx and TCxH registers before the next valid edge occurs at a channel's associated input pin will avoid the conditions under which the errata will occur.

Workaround

A simple workaround exists for this errata:

1. Clear the input capture channel's associated CxF bit.
2. Disable the input capture function by writing 0:0 to a channel's EDGxB and EDGxA bits.
3. Read TCx
4. Read TCxH
5. Re-enable the input capture function by writing to a channel's EDGxB and EDGxA bits.

Code Example:

```
unsigned char ICSave;
unsigned int TC0Val;
unsigned int TC0HVal;

ICSave = TCTL4 & 0x03;  /* save state of EDG0B and EDG0A */
TFLG1 = 0x01;           /* clear ECT Channel 0 flag */
TCTL4 &= 0xfc;          /* disable Channel 0 input capture function */
```

```
TC0Val = TC0;           /* Read value of TC0 */

TC0HVal = TC0H;         /* Read value of TC0H */

TCTL4 |= ICSave;        /* Restore Channel 0 input capture function */
```

PWM: Wrong output level after shutdown restart in 16bit concatenated channel mode — MUCts04135

Description

When the PWM is used in 16-bit (concatenation) channel and the emergency

shutdown feature is being used, after de-asserting PWM channel 7

(note:PWMRSTRT should be set) the PWM channels do

not show the state which is set by PWMLVL bit when the 16-bit counter is

non-zero.

Workaround

If emergency shutdown mode is required:

In 16-bit concatenation mode, user can disable the related PWM

channels and set the corresponding general-purpose IO to be the PWM

LVL value. After a intend period, restart the PWM channels.

PWM: Wrong output value after restart from stop or wait mode — MUCts04136

Description

In low power modes (P-STOP/STOP/WAIT mode) and during PWM7 de-assert and when PWM counter reaching 0, the PWM channel outputs cannot keep the state which is set by PWMLVL bit.

Workaround

Before entering low power modes, user can disable the related PWM channels and set the corresponding general-purpose IO to be the PWM LVL value. After a intend period, restart the PWM channels.

ECT_16B8C: Output compare pulse is inaccurate — MUCts04155

Description

The pulse width of an output compare (which resets the free running counter when TCRE = 1) will measure one more bus clock cycle than expected.

Workaround

The specification has been updated. Please refer to revision 02.05 (04 May 2010) or later.

In description of bitfield TCRE in register TSCR2,a note has been added:

TCRE=1 and TC7!=0, the TCNT cycle period will be TC7 x "prescaler counter width" + "1 Bus Clock". When TCRE is set and TC7 is not equal to 0, then TCNT will cycle from 0 to TC7. When TCNT reaches TC7 value, it will last only one bus cycle then reset to 0.

FTX: Flash Command influenced by Backdoor Key write — MUCts04232

Description

When executing a flash erase verify (0x05) command sequence to a flash block different from the block where the backdoor keys are written to, both blocks will be erase verified. Any programmed location in either block will terminate the operation preventing the FSTAT.BLANK flag from setting.

When executing a flash data compress (0x06) command sequence to a flash

block different from the block where the backdoor keys are written to, a given number of words from both blocks will be compressed, this number will be equal to the value written at last key's address. The signature from the block containing the backdoor keys will affect the signature returned in the FDATA register.

When executing a flash program (0x20) command sequence to a flash block

different from the block where the backdoor keys are written to, both blocks will be programmed at the same relative address with the unselected block being programmed to a data value equal to the last key

written. Setting protection at the location in the block where the backdoor keys are written will not prevent the flash command from executing.

When executing a flash sector erase (0x40) command sequence to a flash block different from the block where the backdoor keys are written to, both blocks will receive the erase at the sector address provided in the flash sector erase command sequence. Setting protection in the location where the backdoor keys are written to will not prevent the flash command from executing.

When executing a flash mass erase (0x41) command sequence to a flash block different from the block where the backdoor keys are written to, both blocks will be erased. Setting protection in the block where the backdoor keys are written to will not prevent the flash command from

executing.

The flash sector erase abort (0x47) command is not impacted as all active sector erase operations will be terminated if successfully aborted.

Workaround

Write 0x30 to FSTAT register (ACCERR = 1, PVIOL = 1) prior to executing any flash command sequence when backdoor keys have been written. This step can be done in conjunction with or instead of checking the FSTAT register as shown in the flash command sequence flow in the reference manual.

MC9S12XDP512, Mask 1L15Y

Introduction

This errata sheet applies to the following devices:

MC9S12XDP512, MC9S12XDT512, MC9S12XA512, MC9S12XDT384, MC9S12XDQ526, MC9S12XDT256, MC9S12XD256, MC9S12XB256, MC9S12XA256, MC9S12XDG128, MC9S12XD128, MC9S12XA128, MC9S12XB128

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 1K79X. 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 "0201" indicates the first week of the year 2002.

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.

Errata System Tracking Numbers

MUCtsXXXXX is the tracking number for device errata. It can be used with the mask set and date code to identify a specific erratum.

Errata Summary

Errata Number	Module affected	Brief Description	Work-around
MUCts01812	s12x_cpu	False tagged breakpoint hits may be reported by the DBG module	YES
MUCts01816	s12x_dbg	Back to back accesses to DBG trace buffer may return incorrect data	YES
MUCts01818	s12x_bdm	BDM hardware read command may be corrupted by entering stop mode	NO
MUCts01974	pim_9xd	PIM: ECLK divider can be activated in emulation modes	YES
MUCts02366	s12x_mmc	Incorrect 23-Bit Program Counter Generated for DBG on global accesses	YES
MUCts02409	eetx	EEPROM: Protection Disabled on EPROT Read During Command Write Sequence	YES
MUCts02539	spi	SPI: Slave entering stop in wait mode, pending rx data not rejected	YES
MUCts02582	s12x_dbg	S12X_DBG: Indexed jump loop1 mode trace buffer entries may be missed	YES
MUCts02667	spi	SPI in slave mode (CPHA=0) and SS line not deasserted between transmissions may lead to corrupted rx data	YES
MUCts02988	xgate	XGATE: Carry-Flag may be falsely set by SSEM instruction	YES
MUCts03017	ect_16b8c	ECT: TCNT counter resets in Input Capture Mode	YES

MUCts03018	ect_16b8c	ECT: CxF flag clears following a read to TCx on an OC event with TFFCA=1	YES
MUCts03019	ect_16b8c	ECT: CxF flag clears following a wite to TCx on an IC event with TFFCA=1	YES
MUCts03037	ect_16b8c	ECT: Forced OC on PT7 occured even when TIOS7 = 0	NO
MUCts03039	ect_16b8c	ECT:Faulty OC event with OM/OL=0, OC7Mx=1, TIOSx=1	YES
MUCts03112	s12x_bdm	BDM: Incomplete Memory Access on misaligned access due to BDM features	YES
MUCts03332	ftx	FTX: Blind Spot in Data Compress Command Algorithm	YES
MUCts03390	atd_10b16c	ATD: Abort of an A/D conversion sequence with write to ATDxCTL0/1/2/3 may not work	YES
MUCts03391	atd_10b8c	ATD: Abort of an A/D conversion sequence with write to ATDxCTL0/1/2/3 may not work	YES
MUCts03453	mscan	MSCAN: Corrupt ID may be sent in early-SOF condition	YES
MUCts03617	s12x_dbg	DBG 'outside range' mode databus qualification ignored	YES
MUCts03620	s12x_dbg	DBG No address match if next transaction is misaligned word access	YES
MUCts03621	s12x_dbg	DBG Range Mode TAGB and TAGD influence in range modes	YES
MUCts03634	eetx	eetx4k An interrupt following immediately after execution of a STOP instruction may disable read access to the EEPROM array	YES
MUCts03646	vreg_3v3	vreg_3v3.05.00: Possible incorrect operation if device is wakened from stop mode within 4.7µs of stop mode entry	NO
MUCts03686	atd_10b8c	ADC: conversion does not start with 2 consecutive writes to ATDCTL5	YES
MUCts03689	atd_10b16c	ADC: conversion does not start with 2 consecutive writes to ATDCTL5	YES
MUCts03761	s12x_dbg	DBG: State flags and counter corrupted by simultaneous arm and disarm	YES
MUCts03870	s12x_cpu	CPU: Breakpoint missed at simultaneous taghits	YES
MUCts03977	pwm_8b8c	PWM: Emergency shutdown input can be overruled	YES
MUCts03996	ftx	Flash: Burst programming issue if bus clock frequency is higher than oscillator clock frequency	YES
MUCts03997	pim_9xd	PIM: Edge-sensitive mode of IRQ-pin may cause incorrect interrupt vector fetch	YES
MUCts04095	ect_16b8c	ECT: Channel 0 - 3 Input Capture interrupts inhibited when BUFEN=1, LATQ=0 and NOVWx=1	YES

MUCts04135	pwm_8b8c	PWM: Wrong output level after shutdown restart in 16bit concatenated channel mode	YES
MUCts04136	pwm_8b8c	PWM: Wrong output value after restart from stop or wait mode	YES
MUCts04155	ect_16b8c	ECT_16B8C: Output compare pulse is inaccurate	YES
MUCts04232	ftx	FTX: Flash Command influenced by Backdoor Key write	YES

False tagged breakpoint hits may be reported by the DBG module — MUCts01812

Description

If the device executes the BACKGROUND command (received over BDM) in the

cycle when it is about to execute a tagged instruction the DBG module

may indicate that a tagged breakpoint was hit even though the tagged

instruction itself was not executed yet.

Workaround

If the DBG module indicates a tagged breakpoint hit after the BACKGROUND

command was issued, verify contents of PC to determine whether the

tagged breakpoint was hit or not.

Back to back accesses to DBG trace buffer may return incorrect data — MUCts01816

Description

When the trace buffer is unlocked for reading and data is read from the

trace buffer register in the next bus cycle (back-to-back accesses),

wrong data may be read out.

Workaround

When using the core or Xgate to read the contents of the trace buffer,

do not use back-to-back bus accesses.

OR

Only use the BDM to access contents of the trace buffer.

BDM hardware read command may be corrupted by entering stop mode — MUCts01818

Description

The BDM can generate invalid data when the BDM is processing a hardware

(HW) read command with the handshake feature of the BDM enabled and the

CPU enters STOP mode with the device not reaching system STOP mode

because of XGATE activity or set-up (as described below).

The BDM read data is cleared in the BDM shift register because the CPU

enters STOP mode but the corresponding ACK pulse of the HW read command

is not prevented. The occuring ACK pulse indicates to the host that

valid data is ready, which is not the case. Instead of valid data a data

value of $FF will be received.

This behaviour occurs only if the CPU enters into STOP mode after a

BDM

read access has occured on the internal bus and before the corresponding

ACK pulse of the HW read command is sent.

It only occurs when the CPU enters into STOP mode while XGATE is not

idle or the XGFACT bit is set. In this case only CPU clocks are stopped.

This behaviour does not occur when a HW read command is sent while the

CPU is already in STOP mode.

It does not occur if the device reaches system STOP mode (all clocks

stopped), which happens when the CPU enters STOP mode and XGATE is in

the idle state and the XGFACT bit is cleared.

This is a debug only issue with a low probability of occurance because:

1) The timing window during which the CPU transition (from run mode to

STOP mode) must occur is very short. (Timing window means the time from

the BDM read access on the internal bus until the BDM internal ACK pulse

control signal is set - in the case of no clock switching this is two

bus clock cycles).

2) Only a transition of the CPU from run mode to STOP mode during the

time window will cause the issue.

3) The issue occurs only if system STOP is not reached.

Workaround

None

PIM: ECLK divider can be activated in emulation modes MUCts01974

Description

The ECLK does not run at bus clock rate in emulation modes if an EDIV value greater than zero is selected. In this case the ECLK will be divided as specified or stopped if NECLK is set. This may affect emulation systems which rely on constant ECLK rate.

Workaround

Always keep the NECLK and EDIV value at zero in case constant ECLK rate

is required.

In order to pass the required divider settings to the emulation system,

the EDIV settings should alternatively be mapped to the reserved bits ECLKCTL[3:2]. The emulation system should utilize these bits instead of

EDIV to control the ECLK generator for the target application. This software modification needs to be undone when the application runs without the emulator. The emulator can monitor values written into ECLKCTL[1:0] and notify the user when non-zero value can cause the emulator to function incorrectly.

Note:

Please note that this erratum only concerns the ECLK signal. Signal ECLKX2 is not affected by EDIV values when operating in emulation

modes.

Incorrect 23-Bit Program Counter Generated for DBG on global accesses — MUCts02366

Description

An incorrect trace buffer entry occurs when the DBG module attempts to trace a Change Of Flow (COF) instruction that follows a global access. This only happens for COF instructions where the source address should be stored to the trace buffer.

This errata only affects the functionality of the DBG module, causing incorrect DBG trace buffer entries.

During global instruction accesses, the program counter bits [22:16] are derived from the global address bus bits [22:16] which, at that moment, contain the address of the global access.

The program counter should always provide the global opcode address and should be independent of addresses associated with memory accesses. In the errata case, the page part of the opcode address contains the data access page address.

The program counter bits [15:0] are not affected by this problem.

Workaround

In debug mode, put a NOP before the global instruction.

EEPROM: Protection Disabled on EPROT Read During Command Write Sequence — MUCts02409

Description

If EPROT EOPEN bit is set with EPROT EPDIS bit clear (i.e. protection is

on) and EPROT is read during a command write sequence then protection is

disabled until the next reset or write of the EPROT register.

Workaround

Do not read EPROT register during a command write sequence.

SPI: Slave entering stop in wait mode, pending rx data not rejected — MUCts02539

Description

In slave mode, pending data in the receive shift register is not

rejected when entering Wait mode with the SPISWAI bit set.

When the SPI stops on entering Wait mode (executing WAI with SPISWAI bit

set), data pending in the receive shift register should be rejected

(lost). This is to prevent pending data in the receive shift register

from being corrupted by SCK cycles while halted in Wait mode.

Workaround

Workaround #1: Ensure that the receive buffer queue is empty before entering Wait mode.

Workaround #2: Make sure SCK does not toggle while in Wait mode.

Workaround #3: Do not use the 'stop in Wait mode' feature.

S12X_DBG: Indexed jump loop1 mode trace buffer entries may be missed — MUCts02582

Description

Loop1 Mode inhibits consecutive duplicate source address entries that would typically be stored in most tight looping constructs. It does not inhibit repeated entries of destination addresses. However, the logic prevents the storage of valid indexed jump destination addresses if consecutive indexed jumps have the same opcode address.

Considering the code below; the first occurence of JMP 0,X stores MARK5

to the trace buffer. Since the code returns to MARK4 with an unconditional branch, the next trace buffer entry should be the next destination address of JMP 0,X (MARK6 the second time around).

This entry is missed because further COFs from MARK4 are masked out.

```
        LDX     #MARK5
MARK4   JMP     0,X
        NOP
MARK5   LDX     #MARK6
        BRA     MARK4
MARK6   NOP
```

This does not affect XGATE loop mode tracing

Workaround

If code contains consecutive indexed jumps from an idential opcode address, then normal mode tracing can be temporarily used to confirm the incidence of consecutive indexed jumps from the same address. Subsequently loop1 mode can be enabled to continue further debugging.

SPI in slave mode (CPHA=0) and SS line not deasserted between transmissions may lead to corrupted rx data — MUCts02667

Description

When the SPI is in slave mode with both the SPI data register (SPIDR) and the receive shift register containing pending data, on reception of further data the contents of the shift register should be discarded and the new data byte shifted in. Reading the SPIDR should not cause the shift register contents to be transferred to the SPIDR until the latest data byte is completely received.

In the erratum condition, the shift register pending state is not discarded and reading the SPIDR during the reception of a new data byte will cause the shift register contents to be immediately transferred to

the SPIDR register causing data corruption.

This condition only occurs in slave mode with CPHA = 0 and if SS is not

de-asserted between transmissions.

Workaround

Workaround #1: Deassert SS for minimum idle time (0.5*Tsck) between

transmissions.

Workaround #2: Ensure that the receive buffer queue is empty before

starting a new transmission.

XGATE: Carry-Flag may be falsely set by SSEM instruction MUCts02988

Description

If the S12X_CPU and the XGATE attempt to lock a semaphore at the same

time the S12X_CPU will obtain the semaphore, but the Carry-Flag will be

set for the XGATE (falsely indicating that the XGATE has locked the

semaphore).

Workaround

Execute two consecutive "SSEM" instructions to set a semaphore. Ignore

result of the first "SSEM" instruction. Carry-Flag will indicate

correctly whether XGATE has allocated the semaphore after the second

"SSEM" instruction is executed.

ECT: TCNT counter resets in Input Capture Mode MUCts03017

Description

Normal Operation:

Timer Free Running Counter, also called as Timer Counter resets to 0x0000 on a Channel 7 Output Compare event when Timer Counter Reset Enable (TCRE) bit of TSCR2 registers is set to 1. This assumes that TIOS7 bit of TIOS register is set to a 1 (Output Compare mode).

Issue: Erroneously this behaviour was found to occur even when TIOS7 bit of TIOS register was set to a "0" (Input Capture mode)

Workaround

Reseting Free Running Counter alias Timer Counter can be avoided by setting Timer Counter Reset Enable (TCRE) bit of TSCR2 register to a "0".

ECT: CxF flag clears following a read to TCx on an OC event with TFFCA=1 MUCts03018

Description

Problem:

Normal Operation:

With TIOS = 1 (Output Compare mode) and TFFCA = 1 (Fast Flag Clearing mechanism) a write to TCx register following an output compare event clears the flag.

Erroneous Operation:

A read to TCx register following an Output Compare event clears Cxf flag

in TFLG1 register.

Workaround

Customer should avoid reading TCx register following an Ouput Compare event.

ECT: CxF flag clears following a wite to TCx on an IC event with TFFCA=1 — MUCts03019

Description

Normal Operation:

With TIOS = 0 (Input Capture (IC) mode) and TFFCA = 1 (Fast Flag

Clearing mechanism) a read to TCx register following an Input Capture

(IC) event clears CxF (flag) bit of TFLG1 register.

Erroneous Operation:

A write to TCx register following an Input Capture (IC) event clears Cxf

(flag) bit of TFLG1 register.

Workaround

Customer should avoid writing to TCx register following an Input Capture

(IC) event.

ECT: Forced OC on PT7 occured even when TIOS7 = 0 — MUCts03037

Description

Correct Operation:

Forced Output Compare (OC) operation on channel 7 requires TIOS7 bit be

set to a “1” for a successful Ouput Compare (OC) event.

Faulty Operation:

Forced Ouput Compare (OC) action was noticed on Channel 7 inspite of

TIOS7 being set to a “0”.

Workaround

None.

ECT:Faulty OC event with OM/OL=0, OC7Mx=1, TIOSx=1 — MUCts03039

Description

Correct Operation:

When timer is enabled (TEN bit of TSCR1 register is set to 1) and Output

Compare functionality is selected (TIOSx bit of TIOS register is set) on

a match (TCNT = Tcx) an output compare event(dictated by OM/OL bits of

TCTL1/TCTL2 register) is generated. Please not that the Output Compare

is registered internally.

Faulty Operation:

Consider a case, when an normal Output Compare event (driving port to 1)

was performed. You intend to dis-connect output compare logic from pin

logic. Hence you configure (OM/OL bits to a “0”), but your TIOSx bit is

still set to a “1”. Also, you intend to use masking feature of output

compare so you set TIOS7=1 and you have OC7Mx bit set to a “1”. If the

next output compare match(TCNT = Tcx) happens to be normal output

compare event, the default state of internal register (0) was noticed on

the channel.

In reality the output port should not have toggled since OM/OL were set

to 0.

This is an erroneous behaviour.

Workaround

Make sure that OC7Mx bit is set to "0" if not using OC7M feature for

that channel. If using masking feature, please remember to set OC7Mx bit

to "0" as soon as you have a channel 7 Compare event.

BDM: Incomplete Memory Access on misaligned access due to BDM features MUCts03112

Description

If a misaligned word write access is directly followed by an attempted entry into active BDM, then the second byte access may be performed on a different target due to the memory map switching for BDM active.

Depending on the type of access this can lead to the following situations...

1. When writing in the address range from $7F_FEFF to $7F_FFFD in the external space the MMC splits the accesses into two 8-bit accesses. It is able to complete the first access, but just before the second access the BDM firmware becomes mapped into this address location and the second byte access goes to the firmware.

This can only occur if the last cycle of the access instruction is a word write to an odd address AND the instruction is followed by BGND or a tagged breakpoint.

2. On a misaligned word write with a global store (GSTD,GSTS,GSTX,GSTY) in the address range $XX_0FFF to $XX_3FFD, the first split access is performed correctly in the external space, but in the second byte cycle the MMC is instructed to interpret the address coming from the CPU as a local address, which is the internal RAM.

This can only occur if the CPU executes a global store instruction which writes to an odd address in the last instruction cycle AND the instruction is followed by BGND or a tagged breakpoint.

3. Generally on a misaligned global store (GSTD,GSTS,GSTX,GSTY) the lower 16-bits of the global address are interpreted as a local address for the second byte access. This can result in unintended accesses to registers or external RAM.

Workaround

To prevent scenarios (2) and (3) of the errata occurring, avoid setting

breakpoints after a GSTD.

When not debugging, remove all instances of BGND from code.

FTX: Blind Spot in Data Compress Command Algorithm — MUCts03332

Description

If the range of Flash addresses to be compressed is 32K or greater, the

data at one of the addresses will be effectively ignored. The address affected is 32K from the upper address read in the data compress algorithm, e.g., for an address range of 32K, the first data read in the algorithm will not affect the final signature provided by the algorithm.

Workaround

Limit range of addresses to be compressed to less than 32K addresses.

Execute multiple data compress commands to compress larger Flash address ranges.

ATD: Abort of an A/D conversion sequence with write to ATDxCTL0/1/2/3 may not work — MUCts03390

Description

Starting a conversion with a write to ATDxCTL5 or on an external trigger event, and aborting immediately afterwards with a write to ATDxCTL0, ATDCTL1, ATDxCTL2 or ATDxCTL3 can fail to stop the conversion process.

Workaround

Only write to ATDxCTL4 to abort an ongoing conversion sequence.

Use the recommended start and abort procedures from the Block Guide.

Section : Initialization/Application Information

Subsection: Setting up and starting an A/D conversion

Subsection: Aborting an A/D conversion

ATD: Abort of an A/D conversion sequence with write to ATDxCTL0/1/2/3 may not work — MUCts03391

Description

Starting a conversion with a write to ATDxCTL5 or on an external trigger event, and aborting immediately afterwards with a write to ATDxCTL0, ATDCTL1, ATDxCTL2 or ATDxCTL3 can fail to stop the conversion process.

Workaround

Only write to ATDxCTL4 to abort an ongoing conversion sequence.

Use the recommended start and abort procedures from the Block Guide.

Section : Initialization/Application Information

Subsection: Setting up and starting an A/D conversion

Subsection: Aborting an A/D conversion

MSCAN: Corrupt ID may be sent in early-SOF condition — MUCts03453

Description

The initial eight ID bits will be corrupted if a message is set up for transmission during the third bit of INTERMISSION and a dominant bit is sampled leading to an early-SOF*.

The CRC is calculated from the resulting bit stream so that the receiving nodes will still validate the message.

An early-SOF condition may only occur if the oscillators in the network operate at a tolerance range which could lead to a cumulated phase error after 11 bit times larger than phase segment 2.

In case arbitration is lost during transmission of the corrupt identifier, a non-corrupted ID will be sent with the next attempt if the transmit request remains active.

*The CAN protocol condition referred to as 'early-SOF' in this erratum is detailed in "Bosch CAN Specification Version 2.0" Part A, section 9, and a Note to section 3.2.5 INTERFRAME SPACING – INTERMISSION in Part B.

Workaround

Due to increased oscillator tolerance a transmission start in the third bit of intermission is possible and allowed. The errata can be avoided when calculating the maximum oscillator tolerance of the overall CAN

system. The phase error after 11 bit times due to the oscillator tolerance should be smaller than phase segment 2.

If an early-SOF cannot be avoided the following methods will provide prevention:

- Assigning the same value to all upper eight ID bits in the network
- Allocating dedicated data length codes (DLC) to every identifier used in the network and checking for correspondence after reception
- Assigning only IDs (x) which do not consist of a combination of other assigned IDs (y,z) and using the acceptance filters to reject erroneous messages, i.e.
  - for standard frames: IDx[11:0] != {IDy[11:3], IDz[2:0]}
  - for extended frames: IDx[28:21] != {IDy[28:21],IDz[20:0]}

DBG 'outside range' mode databus qualification ignored — MUCts03617

Description

When using a comparator pair for a range comparison, the databus can also be used for qualification by using the comparator A/C data and data mask registers. This does not work correctly.

Scenario:

With CompA/CompB in 'outside range' mode an access to a memory location

with an address above the high boundary of CompB always generates a comparator match. The data qualification is not carried out.

If the accessed memory is located below the address defined in CompA, the behavior is correct.

CompC/CompD behavior is the same.

Workaround

Using 2 comparator pairs configured for inside range mode an outside range match with databus qualification can be generated.

```
MAP
0x000000 |COMPA|
         |     | Inside Range A/B with DB qualification
ADDRESSX |COMPB|
         |     |
ADDRESSY |COMPC|
         |     | Inside Range C/D with DB qualification
0x7FFFFF |COMPD|
```

DBG No address match if next transaction is misaligned word access — MUCts03620

Description

Memory accesses in successive bus cycles must both be able to generate forced triggers if both accesses match.

If accesses occur in successive cycles whereby the second access is a misaligned word access, then a comparator match is lost. This could cause a forced breakpoint to be missed if the state sequencer is dependent on both the successive matches to reach final state.

Example...

```
        LDX    #WORD_MISALIGNED
        STD    0,X              ; First access M0->State2
        LDD    WORD_MISALIGNED  ; Second access M0->Final State
```

If the STD last bus cycle is a write and the LDD first bus cycle is a read then only one state sequencer transition occurs.

In range modes, this can occur when 2 different addresses within/outside

the specified range are accessed in successive bus cycles, otherwise it

can only happen if the same address is written and then read in successive bus cycles.

Workaround

Insert a NOP instruction before misaligned word accesses if they can follow accesses to the predefined comparator range.

```
        LDX    #WORD_MISALIGNED-1
        STD    0,X              ; First access in range M0->State2

        NOP                     ; NOP insertion
```

```
        LDD    WORD_MISALIGNED   ; Second access in range M0->Final
State
```

DBG Range Mode TAGB and TAGD influence in range modes — MUCts03621

Description

The comparator A and C TAG bits are used to tag range comparisons for the AB and CD ranges respectively. The comparator B and D TAG bits should have no effect in range modes. However the TAGB/TAGD bits do have an effect.

If the A/C TAG bit is 0 but the paired B/D TAG bit is set, a valid match may be missed.

Workaround

Clear TAGB when configured for forced range mode comparisons using CompAB

Clear TAGD when configured for forced range mode comparisons using CompCD

---

eetx4k An interrupt following immediately after execution of a STOP instruction may disable read access to the EEPROM array — MUCts03634

Description

An interrupt request immediately following execution of a STOP instruction may cause the EEPROM array to return incorrect data when read. The problem will only occur if all of the following conditions are
met:

1) S bit in the CCR register is cleared (stop mode enabled)
2) I bit in the CCR register is cleared (interrupts enabled)
3) A STOP instruction is executed by the microcontroller
4) An interrupt becomes pending between 0 and 0.5 bus clock cycles after beginning of the STOP instruction execution.

In this case the microcontroller will wake-up correctly from the stop mode, but the EEPROM array will return incorrect data when a read operation is performed.

Access (read or write) to any of the EEPROM registers or write into the
EEPROM array (to start a command sequence) will restore the read access. The EEPROM will return the correct data afterwards.

Workaround

If the STOP instruction is executed while interrupts are enabled, read or write any of the EEPROM registers after wake-up from stop mode to ensure correct operation. Alternatively a write into the EEPROM array (as part of a command sequence) can be performed instead of the register access.

vreg_3v3.05.00: Possible incorrect operation if device is wakened from stop mode within 4.7µs of stop mode entry

MUCts03646

Description

It is possible that after the device enters Stop or Pseudo-Stop mode it

may reset rather than wake up normally upon reception of the wake-up

signal.

CONDITIONS:

This event will only happen provided ALL of the following conditions are

met:

1) Device is powered by the on-chip voltage regulator.

2) Device enters stop or pseudo-stop mode (see Stop mode entry

description below)

3) The wake-up signal is activated within a specific and very short

window (typically 11ns long, not longer than 20ns). The position of the

window varies between different devices, however it never starts sooner

than 1.6µs and never ends later than 4.7µs after the stop mode entry.

This narrow width of the susceptible window makes the erratum unlikely

to ever show in the applications life.

Stop or Pseudo-Stop mode entry are:

1) Execution of STOP instruction by the CPU (provided the S-bit in

CCR is cleared)

NOTE: The part enters stop mode either after 12 oscillator clock cycles

with the PLL disengaged or 3 PLL clock cycles and 8 oscillator clock

cycles with the PLL engaged after the STOP command is executed.

2) End of XGATE thread (providing the CPU is in stop or pseudo-stop

mode)

The incorrect behavior will never occur if ANY of the wake-up conditions

are met at the time when the stop mode entry is attempted (an enabled

interrupt is pending).

EFFECT:

If this incorrect behavior occurs, the device will Reset and indicate a

Low Voltage Reset (LVR) as the reset source.

The device will operate normally after the reset.

Workaround

None.

--

Asynchronous Low Voltage Resets are possible in any microcontroller

application (due to power supply drops) and the integrated LVR and LVI

features and dedicated LVR reset vector are provided to manage this fact

cleanly. For best practice, the application's software should be written

to recover from a Low Voltage Reset in a controlled manner.

An application software written to deal with valid Low Voltage Resets

should correctly manage erroneous LVR events.

It can also be possible to avoid erroneous Low Voltage Resets from

synchronous wake-up events by configuring the application software to

ensure that the entry into stop occurs at such a time, in relation to

the wake-up event timer, that a wake-up event does not occur within

1.6µs to 4.7µs after Stop/Pseudo-Stop entry.

ADC: conversion does not start with 2 consecutive writes to ATDCTL5 — MUCts03686

Description

When the ATD is started with write to ATDCTL5

and, which is very unusual and not necessary,

within a certain period again started with write

to ATDCTL5. The conversion will not start at all.

This does only happen if the two consecutive writes to ATDCTL5 occur

within one "ATD clock period". An ATD clock period is defined by a full

rollover of the ATD clock prescaler. That is for example PRS[4:0] = 2 -

> (2+1)*2 = within 6 bus cycles.

Workaround

Only write once to ATDCTL5 when starting a conversion.

Use the recommended start and abort procedures from the Block Guide.

Section : Initialization/Application Information Subsection: Setting up

and starting an A/D conversion Subsection: Aborting an A/D conversion

---

ADC: conversion does not start with 2 consecutive writes to ATDCTL5 — MUCts03689

Description

When the ATD is started with write to ATDCTL5

and, which is very unusual and not necessary,

within a certain period again started with write

to ATDCTL5. The conversion will not start at all.

This does only happen if the two consecutive writes to ATDCTL5 occur

within one "ATD clock period". An ATD clock period is defined by a full

rollover of the ATD clock prescaler. That is for example PRS[4:0] = 2 -

> (2+1)*2 = within 6 bus cycles.

Workaround

Only write once to ATDCTL5 when starting a conversion.

Use the recommended start and abort procedures from the Block Guide.

Section : Initialization/Application Information Subsection: Setting up

and starting an A/D conversion Subsection: Aborting an A/D conversion

DBG: State flags and counter corrupted by simultaneous arm and disarm — MUCts03761

Description

Simultaneous disarming (hardware) and arming (software) results in status and counter register (DBGSR, DBGCNT) corruption.

Hardware disarming initiated by an internal trigger or by tracing completion takes 2 clock cycles. If a write to DBGC1 with the ARM bit set occurs in the final clock cycle of this disarming process, arming is suppressed but the DBGSR register is initialized to state1 and DBGCNT is initialized to zero.

The result is that the DBG module is disarmed by hardware but DBGSR indicates state1.

NOTE: DBGC1 is typically only written to whilst armed to set the TRIG bit or update the COMRV bits to map different registers to the address map window.

Generally during debugging, after arming the DBG module and returning to application code a further write access of DBGC1 over the BDM interface requires considerable time relative to application code execution, therefore in many cases the breakpoint may be reached before a DBGC1 update is attempted. Furthermore the probability of hitting the same cycle is seen to be very low.

Workaround

If the fault condition is caused by writing to DBGC1 to set the TRIG

bit (to request an immediate breakpoint), then the application code may

be rerun again without the attempted setting of TRIG. The software trigger (TRIG) is unnecessary in any case at the same point point in time as an internal hardware trigger occurs.

Development tool vendors should avoid COMRV updates while the DBG is armed.

Users that observe the problem due to development tool COMRV updates can add a NOP to code and rerun to shift the disarm cycle, thereby preventing a collision with the COMRV updates.

CPU: Breakpoint missed at simultaneous taghits MUCts03870

Description

The CPU execution priority encoder evaluates taghits and then generates a breakpoint if a taghit must lead to an immediate breakpoint as determined by the DBG module.

If the DBG module indicates that this taghit leads to an immediate breakpoint then the CPU loads the execution stage with SWI or BGND thus

generating the breakpoint.

At simultaneous taghits the lowest channel has priority.
If taghits on 2 channels simultaneously, whereby the lower channel tag must be ignored, but the higher channel tag must cause a breakpoint, then the breakpoint request is erroneously missed.
Thus if channel[1:0] taghits occur simultaneously whereby channel[0] must be ignored but channel[1] must cause a breakpoint, then the breakpoint request on channel[1] is erroneously missed and no breakpoint generated.

The DBG module recognises the taghit and the state sequencer transitions accordingly. Furthermore the taghit causes the DBG module to request a forced breakpoint, which means that a late CPU breakpoint occurs. If the tagged instruction is a single cycle instruction, the breakpoint occurs at the second instruction following the tagged instruction.
Otherwise the breakpoint occurs at the instruction immediately following the tagged instruction.

This bug requires that separate tags are placed on the same instruction. This is not a typical case when using exact tag addresses.
It is more relevant in debugging environments using range modes, where tags may cover a whole range. In this case a tagged range may cover a tag or another tagged range, making simultaneous taghits possible.

This is a debugging issue only.

Workaround

Do not attach multiple tags to the same exact address.

When attaching multiple tags to the same address by overlapping tag ranges in range modes, or covering a tag with a tag range in range modes, then the missed tag can be avoided by mapping the final state change to the lower channel number.

For example the case

Channel[0] tags a range; channel[3] tags an instruction within that range.

From DBG State1 Taghit[0] leads to State2. (DBGSCR1=$6)

From DBG State2 Taghit[3] leads to FinalState.(DBGSCR2=$5)

In State2 a simultaneous Taghit[3,0] scenario would miss the breakpoint.

This can be avoided by using the alternative DBG configuration

Channel[2] tags a range; channel[0] tags an instruction within that range.

From DBG State1 Taghit[2] leads to State2. (DBGSCR1=$3)

From DBG State2 Taghit[0] leads to FinalState.(DBGSCR2=$7)

PWM: Emergency shutdown input can be overruled MUCts03977

Description

If the PWM emergency shutdown feature is enabled (PWM7ENA=1) and PWM

channel 7 is disabled (PWME7=0) another lower priority function available on the related pin can take control over the data direction.

This does not lead to a problem if input mode is maintained. If the alternative function switches to output mode the shutdown function may unintentionally be triggered by the output data.

Workaround

When using the PWM emergency shutdown feature the GPIO function on the pin associated with PWM channel 7 should be selected as an input.

In the case that this pin is selected as an output or where an alternative function is enabled which could drive it as an output, enable PWM channel 7 by setting the PWME7 bit. This prevents an active shutdown level driven on the (output) pin from resulting in an emergency shutdown of the enabled PWM channels.

Flash: Burst programming issue if bus clock frequency is higher than oscillator clock frequency — MUCts03996

Description

If S12X is running at a bus clock frequency higher than the oscillator clock frequency (Fbus > Fosc), flash words may remain not programmed after a program burst sequence. Software methods can be used to avoid this problem. If burst programming is not used, no errors occur.

The root cause of this issue is related to flash internal state

machine

which needs about 2 to 3 oscillator clock periods to be ready to accept

a new flash command after the assertion of the CBEIF flag.

Note that this latency of flash state machine is related to the

assertion of CBEIF and the start of a new command sequence (i.e. a write

to a flash word). There is no latency after the assertion of the CCIF flag.

Workaround

Adding a time delay between the check of CBEIF flag and the start of the

next flash program command sequence will ensure that all words will be

programmed. Add a delay of (3 * (Fbus/Fosc)) Bus clock cycles after

CBEIF is set, that can be achieved by the addition of NOPs (one NOP

instruction takes one bus cycle to execute).

Note that since a flash word program operation takes much longer than 3

oscillator clock periods, there is no impact in the total programming

time of a long program burst sequence by adding a delay of 3 clocks.

The example below illustrates the proposed workaround:

Code below executes a program burst sequence by launching a new flash

program command right after the assertion of the CBEIF flag.

```
1    LDX     #(PGM_ADDR_START+PGM_SIZE) ;Load X with last addr+1

2    STX     TMP_VAR                    ;Store last programmed addr+1
at

tmp_w1 var

3    LDX     #PGM_ADDR_START            ;Load X with start addr

4

5  LOOPGM:                              ;Loop Program

6    BRCLR   FSTAT, #$80, *             ;Wait for buffer empty (CBEIF
= 1)

7

8    ; -> Time delay of 3 Osc clock periods must be inserted at this
point.

9


10   MOVW    #DATA      2,X+            ;Write DATA to address pointed
by

index X

11   MOVB    #FCMD_PGM  FCMD            ;Write PGM command code to FCMD

register

12   MOVB    #$80       FSTAT           ;Launch command

13   CPX     TMP_VAR

14   BLO     LOOPGM
```

The minimum time delay for the code above can be found by the equation

below:

Time delay (in Bus clock cycles) = 3*(Fbus/Fosc) - (bus clock cycles

need by BRCLR at line 6) - (bus clock cycles needed by MOVW at line 10)

Considering that the BRCLR instruction at line 6 takes 3 bus clock cycles after the assertion of the CBEIF and that the MOVW takes 3 bus clock cycles to be executed, the equation above can be written as:

Time delay (in Bus clock cycles) = 3*(Fbus/Fosc) - 6

For the case of Fosc=4MHz and Fbus=12MHz, a time delay of 3*(12MHz/4MHz) - 6 = 3 bus clock periods is needed. So, at least 3 NOP instructions must be inserted at line 8 for proper operation, in this example.

Depending on the configuration of Bus clock frequency and oscillator clock frequency, and due to the actual code used in the application, the required time delay of 3 oscillator clock cycles may be already spent inside of the programming routine and the problem will not be detected.

Under certain conditions described by the equations and explanation above, adding NOPs may not be required.

PIM: Edge-sensitive mode of IRQ-pin may cause incorrect interrupt vector fetch — MUCts03997

Description

Where the IRQ interrupt is being used in edge-sensitive mode and a lower priority interrupt is already pending when an IRQ edge event occurs, if the IRQ edge event occurs inside a very narrow (<3ns) window

just as the pending interrupt vector is being fetched, then a different

vector other than that relating to either the pending interrupt or IRQ will be taken.

In the case that a programmed interrupt vector is fetched both the originally pending interrupt and the IRQ interrupt request will remain pending and will be serviced once the erroneously called service routine has completed (and RTI has been executed).

In the case that the incorrect vector fetch is from an unprogrammed vector table entry (i.e. erased state = 0xFFFF) then erroneous execution from the start of the register space will occur most often resulting in an illegal memory access reset, COP reset or Unimplemented

Instruction Trap occurring.

In the less likely case that one of the three reset vectors is incorrectly fetched then execution will jump to the appropriate reset code.

The following vectors are not affected will not cause erroneous behavior if pending:

$F0 - RTI

$F6 - SWI

$E2 – ECT channel 6

$B2 – CAN0 Rx

$72 – XGATE software trigger 0

This issue is limited to the edge-sensitive mode of the IRQ input only -

applications not using IRQ interrupts or configured for level-sensitive IRQ input are not affected.

There is no issue where a pending interrupt has higher priority than the IRQ request.

Workaround

Where using IRQ in edge-sensitive mode then configure the interrupt priority levels of all interrupts being used to ensure that the IRQ request always has the lowest priority.

For new designs, where possible use the IRQ input in level-sensitive mode or alternatively use a key-interrupt port.

There are a number of ‘best practices’ and features of the S12X which

can help minimize the impact of this errata in the case of it occurring:

1) As ‘best practice’ initialize all unused and unimplemented/reserved interrupt vector table locations to point to a dummy interrupt service routine (terminated with an RTI).

2) Where possible, check for appropriate asserted flags in interrupt service routines and return / flag a system error if no request flag is set.

3) Support is provided on the MCU for managing the following system conditions:

* COP watchdog reset

* Illegal access reset

* Unimplemented instruction trap

For ‘best practice’ the application's software should be written to recover from any of these conditions in a controlled manner.

4) In the case of erroneous code execution jumping to unused Flash the typical practice of filling all unused Flash and RAM space with the op-code for the SWI instruction will help manage this. SWI exception routine should be written in this case to manage this event.

ECT: Channel 0 - 3 Input Capture interrupts inhibited when BUFEN=1, LATQ=0 and NOVWx=1 — MUCts04095

Description

Channel 0 – 3 Input Capture interrupts are inhibited when BUFEN=1, LATQ=0 and NOVWx=1 if an Input Capture edge occurs during or between a read of TCx and TCxH or between a read of TCx/TCxH and clearing of CxF.

Details:

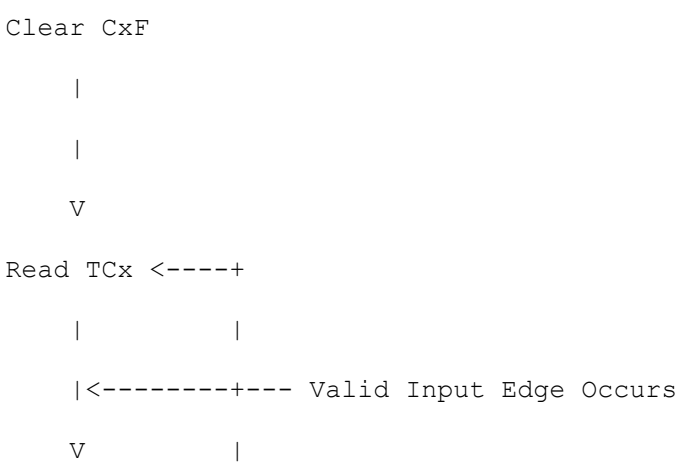
When any of the buffered input capture channels 0 - 3 are configured for buffered/queue mode (BUFEN=1, LATQ=0) each of the channel's input capture holding registers and each channel's associated pulse accumulator and its holding register are enabled. When the input

capture channel is enabled by writing to a channel's EDGxB and EDGxA bits, both the input capture and input capture holding register are considered empty. The first valid edge received after enabling a channel will latch the ECT's free running counter into the input capture register (TCx) without setting the channel's associated CxF interrupt flag. The second valid edge received will transfer the value of the input capture register, TCx, into the channel's TCxH holding register, latch the current value of the free running timer into the input capture register and set the channel's associated CxF interrupt flag. In this condition, both the TCx and TCxH registers are considered 'full'.

If a corresponding channel's NOVWx bit in the ICOVW register is set, the capture register or its holding register cannot be written by a valid edge at the input pin unless they are first emptied by reading the TCx and TCxH registers. The act of reading the TCx and TCxH registers and clearing the channel's associated CxF interrupt flag involves three separate operations. Two 16-bit read operations and an 8-

bit write operation.

If a channel’s associated CxF interrupt flag is cleared before reading the TCx and TCxH registers and if a valid input edge occurs during or between the reading of the capture and holding register, a channel’s associated CxF interrupt flag will no longer be set as the result of valid input edges. For example:



If the TCx and TCxH registers are read before a channel’s associated CxF interrupt flag is cleared and if a valid input edge occurs between the reading of TCx/TCxH and the clearing of a channel’s associated CxF interrupt flag, a channel’s associated CxF interrupt flag will no longer be set as the result of valid input edges. For example:

```
Read TCx
    |
    |<------------ Valid Input Edge Occurs
    V
Read TCxH
```

Systems that service the interrupt request and read the TCx and TCxH registers before the next valid edge occurs at a channel's associated input pin will avoid the conditions under which the errata will occur.

Workaround

A simple workaround exists for this errata:

1. Clear the input capture channel's associated CxF bit.
2. Disable the input capture function by writing 0:0 to a channel's EDGxB and EDGxA bits.
3. Read TCx
4. Read TCxH
5. Re-enable the input capture function by writing to a channel's EDGxB and EDGxA bits.

Code Example:

```
unsigned char ICSave;
```

```
unsigned int TC0Val;

unsigned int TC0HVal;


ICSave = TCTL4 & 0x03;  /* save state of EDG0B and EDG0A */

TFLG1 = 0x01;           /* clear ECT Channel 0 flag */

TCTL4 &= 0xfc;          /* disable Channel 0 input capture function */

TC0Val = TC0;           /* Read value of TC0 */

TC0HVal = TC0H;         /* Read value of TC0H */

TCTL4 |= ICSave;        /* Restore Channel 0 input capture function */
```

PWM: Wrong output level after shutdown restart in 16bit concatenated channel mode — MUCts04135

Description

When the PWM is used in 16-bit (concatenation) channel and the emergency

shutdown feature is being used, after de-asserting PWM channel 7

(note:PWMRSTRT should be set) the PWM channels do

not show the state which is set by PWMLVL bit when the 16-bit counter is

non-zero.

Workaround

If emergency shutdown mode is required:

In 16-bit concatenation mode, user can disable the related PWM

channels and set the corresponding general-purpose IO to be the PWM LVL value. After a intend period, restart the PWM channels.

PWM: Wrong output value after restart from stop or wait mode — MUCts04136

Description

In low power modes (P-STOP/STOP/WAIT mode) and during PWM7 de-assert and when PWM counter reaching 0, the PWM channel outputs cannot keep the state which is set by PWMLVL bit.

Workaround

Before entering low power modes, user can disable the related PWM channels and set the corresponding general-purpose IO to be the PWM LVL value. After a intend period, restart the PWM channels.

ECT_16B8C: Output compare pulse is inaccurate MUCts04155

Description

The pulse width of an output compare (which resets the free running counter when TCRE = 1) will measure one more bus clock cycle than expected.

Workaround

The specification has been updated. Please refer to revision 02.05 (04 May 2010) or later.

In description of bitfield TCRE in register TSCR2,a note has been added:

TCRE=1 and TC7!=0, the TCNT cycle period will be TC7 x "prescaler counter width" + "1 Bus Clock". When TCRE is set and TC7 is not equal to 0, then TCNT will cycle from 0 to TC7. When TCNT reaches TC7 value, it will last only one bus cycle then reset to 0.

FTX: Flash Command influenced by Backdoor Key write MUCts04232

Description

When executing a flash erase verify (0x05) command sequence to a flash block different from the block where the backdoor keys are written to, both blocks will be erase verified. Any programmed location in either block will terminate the operation preventing the FSTAT.BLANK flag from setting.

When executing a flash data compress (0x06) command sequence to a flash block different from the block where the backdoor keys are written to, a given number of words from both blocks will be compressed, this number will be equal to the value written at last key's address. The signature from the block containing the backdoor keys will affect the signature returned in the FDATA register.

When executing a flash program (0x20) command sequence to a flash block different from the block where the backdoor keys are written to, both blocks will be programmed at the same relative address with the unselected block being programmed to a data value equal to the last key written. Setting protection at the location in the block where the backdoor keys are written will not prevent the flash command from executing.

When executing a flash sector erase (0x40) command sequence to a flash block different from the block where the backdoor keys are written to, both blocks will receive the erase at the sector address provided in the flash sector erase command sequence. Setting protection in the

location where the backdoor keys are written to will not prevent the flash command from executing.

When executing a flash mass erase (0x41) command sequence to a flash block different from the block where the backdoor keys are written to, both blocks will be erased. Setting protection in the block where the backdoor keys are written to will not prevent the flash command from executing.

The flash sector erase abort (0x47) command is not impacted as all active sector erase operations will be terminated if successfully aborted.

Workaround

Write 0x30 to FSTAT register (ACCERR = 1, PVIOL = 1) prior to executing any flash command sequence when backdoor keys have been written. This step can be done in conjunction with or instead of checking the FSTAT register as shown in the flash command sequence flow in the reference manual.