Mask Set Errata 1

MMC2107 32-Bit Microcontroller Unit

INTRODUCTION

This mask set errata provides information pertaining to numerous topics applicable to this MMC2107 MCU mask set device:

- K61K

MCU DEVICE MASK SET IDENTIFICATION

The mask set is identified by a 5-character code consisting of a letter, two numerical digits, and a letter, for example 3J74Y. Slight variations to the mask set identification code may result in an altered version number, for example 4J74Y.

MCU DEVICE DATE CODES

Device markings indicate the week of manufacture and the mask set used. The data 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 “9915” indicates the 15th week of the year 1999.

MCU DEVICE PART NUMBER PREFIXES

Some MCU samples and devices are marked with an SC or XC prefix. An SC prefix denotes special/custom device. 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 prefix.

When contacting a Motorola representative for assistance, please have the MCU device mask set and date code information available.

MSE Published Date: 6/28/01

MOTOROLA

Two problems appear when the source of the counter clock is not the system clock with a prescaler of 1 (the default). This includes all other possible sources of the counter clock: system (with prescaler other than 1), PACLK, PACLK/256, and PACLK/65536.

Problem 1:

When the TCRE (timer counter reset enable) bit is set, the counter only stays at the channel 3 compare value for one timer clock cycle and then it resets to 0x0000. The 0x0000 count is then one timer counter clock cycle short. (for example, three clocks for div 4). The picture shows what happens for div 4, TCRE enabled and the timer channel 3 register set to 0x0002. See **Figure 1**.

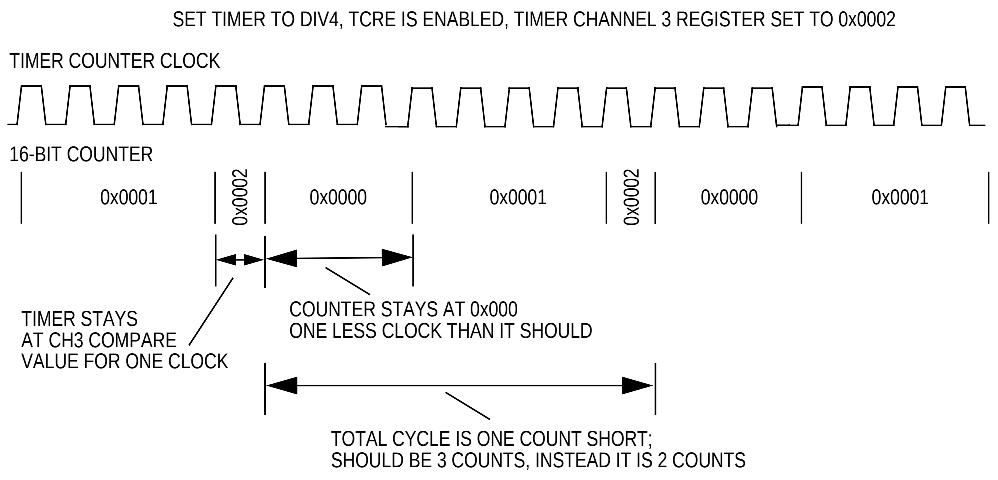
SET TIMER TO DIV4, TCRE IS ENABLED, TIMER CHANNEL 3 REGISTER SET TO 0x0002
TIMER COUNTER CLOCK
16-BIT COUNTER
0x0001
0x0002
0x0000
0x0001
0x0002
0x0000
0x0001
TIMER STAYS AT CH3 COMPARE VALUE FOR ONE CLOCK
COUNTER STAYS AT 0x000 ONE LESS CLOCK THAN IT SHOULD
TOTAL CYCLE IS ONE COUNT SHORT; SHOULD BE 3 COUNTS, INSTEAD IT IS 2 COUNTS

Figure 1. Timer

The net result is that you lose a count in the whole repeating loop. The repeating loop is only two timer counter counts long instead of three.

Workaround:

Make the channel 3 compare value one larger to account for the missing count.

Problem 2:

The TOV (toggle-on-overflow) feature does not work when the timer's counter clock is not the system clock with a prescaler of 1 (the default). When using the system clock with a divisor other than 1, the output compare value does in fact toggle, but toggles for the same number of times as the divisor (for example, toggles twice for div2 and four times for div4, etc.). Also, while using the PACLK, PACLK/256, or PACLK/65536, the port will toggle multiple times during each overflow state. When not using the system clock with a prescaler of 1, the TOV should always be disabled.

Workaround:

Either do not use the TOV feature for prescalers larger than 1, or use the channel 3 masking feature to simulate the effects of the toggle-on-overflow.For example, set the OC3 timer channel register to 0xFFFF and have the channel 3 masks clear or set the other OC channels as necessary.

The most common use for this feature is to generate a waveform with a specific duty cycle. Implementing this using the channel 3 masking feature allows 16-bit resolution for both the period and the duty.

This is demonstrated in the following code.

```
#include "mmc2107.h"


void main()
{
 INT32U period;        // period (using 32 bit value for period so that it doesn't
                              // overflow in the for loop, causing an infinite loop)
 INT16U edge;                 // position of edge
 INT16U duty_cycle;           // in percent * 100 (ie. 25% = 2500)
 INT16U i;                    // counter

 //setup channel 3 and 0 to be used for duty cycle
 reg_TIM1IOS.reg = 0x09;      //set chan 0 and 3 to output compare
 reg_TIM1CTL1.reg = 0x01;     //set chan 0 to toggle on compare,
                              //leave chan 3 disconnected from output logic
 //setup channel 3 masking
 reg_TIM1OC3M.reg = 0x01;           //enable channel 3 masking for channel 0
 reg_TIM1OC3D.reg = 0x01;           //set channel 0 to 1 on channel 3 compare;

 reg_TIM1SCR1.reg = 0x80;           //enable timer

 while (1)
 {
  //sweep through full range of periods
  duty_cycle = 2500;                //set duty cylce
  reg_TIM1SCR2.reg = 0x0A;          //set TCRE so we can change the period
                                    //and set prescalar to div 4

  for (period=0; period < 0xFFFF; period+=0x1E)
  {
   edge = (period * duty_cycle)/10000;  //get edge

   reg_TIM1C0 = edge;         //set to edge
   reg_TIM1C3 = period + 1;   //set to period + 1 (see TCRE timer errata)
                              //wait for channel 3 compare (one period)
   reg_TIM1FLG1.reg = 0x08;   //clear chan3 compare flag
   while (!(reg_TIM1FLG1.reg & 0x08)); //wait for chan3 compare
  }
  //sweep through full range of duty cycle

  reg_TIM1SCR2.reg = 0x0A;    //disable TCRE so we can use the full range of the period
                              //keep prescalar at div 4
  reg_TIM1C3 = 0xFFFF;        //set ch3 compare to full range
                       //don't have to add one because bug is only in TCRE function
  period = 0xFFFF;
  for (duty_cycle=0; duty_cycle<10000; duty_cycle+=5)
  {
   edge = (period * duty_cycle)/10000; //get edge
   reg_TIM1C0 = edge;         //set to edge

   //wait for channel 3 compare (one period)
   //could have also waited for the overflow flag
   reg_TIM1FLG1.reg = 0x08;   //clear chan3 compare flag
   while (!(reg_TIM1FLG1.reg & 0x08)); //wait for chan3 compare
  }
 } //end of while(1)

return;
}
```

FLASH ENABLE

The FLASH enable pin is not masked by the RCON bit. To have the FLASH work, the D28/PA4 pin must be pulled high during reset.

Workaround:

This is only a problem if attempting to use the part in its default state (single-chip mode, internal boot, etc.) and RCON is not asserted. Even if the RCON bit is not asserted, the FLASH enable pin D28/PA4 must be asserted. This is also true for the 100-pin package. This can be as simple as tying the D28/PA4 pin to V_{DD} through a resistor.

FLASH

The first page (64 bytes) of the shadow row may not be blank from the factory because it is used for factory tracking purposes. It can be erased with no ill effects.

INCORRECT READ DATA FROM PORTS IN EMULATION MODE

In emulation mode, if an external port replacement unit (PRU) is mapped to the same address space as the internal PORTS (0x00c0_0000-0x00cf_ffff), then port register reads will return all 1s. PRU writes function correctly.

Workaround:

Software should map the port registers for the PRU to an increment of the internal PORTS addresses. Thus, instead of the base address for the port registers residing at 0x00c0_0000, the base address should be mapped to 0x01c0_0000, or 0x02c0_0000 . . . 0x7fc0_0000.

Note that accesses to the new locations will not be terminated internal to Sika. Thus, the external TA input will need to be used to terminate the port access.

CONCURRENT ASSERTION OF BREAKPOINT FINT, DBEV CAUSES WRONG EXCEPTION

With concurrent assertion of a breakpoint, fast interrupt (FINT), and debug request, the FPC is loaded with the base address of the FINT. The hardware breakpoint should be taken instead, since it has a higher priority.

Workaround:

None

NOTE: *This situation would occur only if a breakpoint from the OnCE port occurred without the FDB bit being set, a debug request was asynchronously asserted from the external DBEV pin, and an FINT was recognized on the same internal clock. In this case, the breakpoint would be ignored, and the FPC would be incorrectly loaded, potentially causing the FINT exception handler to execute incorrectly. This is an unlikely scenario that only affects debug (breakpoint) operation.*

LD. (B,H) FOLLOWED BY JMP MAY NOT ZERO EXTEND

LD. (B,H) followed by a JMP with a data dependency will not zero extend the jump address correctly.

Workaround:

Put a sync instruction between the LD. (B,H) and the JMP.

AFTER MULT OR DIV, DEBUG REQUEST MAY CAUSE INADVERTENT INCREMENT OF THE PC

If the M•CORE is at the end of a multiply or a divide instruction, still fetching the buffer, and debug request is asserted with debug enabled, then an inadvertent increment of the PC occurs. This may be dependent on wait states changing during the access.

Workaround:

None

AFTER MULTI OR DIV, EXITING DEBUG AND EXECUTES JMP/BR RESULTS PC –2

If the M•CORE is at the end of a multiply or a divide instruction, using wait-stated memory and executing a change of flow (JMP/BR) with a debug request, then the M•CORE will decode an inadvertent instruction upon exiting debug mode and the PC points to change of destination – 2.

Workaround:

Use sync instruction after multiply or divide instructions.

PSR FE BIT MAY NOT CLEAR ON FAST INTERRUPT

If an execute exception occurs and the processor status register (PSR) exception enable (EE) and fast interrupt enable (FE) control bits are both set, and a breakpoint on an instruction access (instruction fetch) and a fast interrupt both occur, the PSR FE bit may not be cleared when the fast interrupt is taken. This is only apparent in debug mode.

Workaround:

Don’t set breakpoints on instruction access or disable fast interrupts in debug mode. This is not a problem if using SDS tools with an EBDI or HP probe.

BREAKPOINT INCORRECTLY TAKEN ON LOAD/STORE MULTIPLE INSTRUCTIONS

If a data breakpoint on a load/store multiple (LDM/STM/LDQ/STQ) and a fast interrupt both occur with the PSR FE and IC bits set and EE bit clear, the breakpoint is recognized instead of the unrecoverable. This is only apparent in debug mode.

Workaround:

To limit exposure to this situation, save the EPC/EPSR and set the PSR EE bit as soon as possible after the EE bit is cleared. This is not a problem if using the SDS tools with an EBDI or HP probe.

PC CORRUPTED IF BKPT ON MULTICYCLE INSTRUCTION WITH ZERO WAIT STATED MEMORY

If the instruction buffer is full (user is running out of zero wait-state instruction memory), on a data breakpoint (during a data fetch) followed by a multicycle instruction, the breakpoint will not be taken, causing the PC to be corrupted. This is only apparent in debug mode.

Workaround:

Don’t allow data side breakpoints (during data fetch) around multicycle instructions (mul/div) when executing from zero wait-state memory. This is not a problem if using SDS tools with an EBDI or HP probe.

PSR FE BIT NOT CLEARED AS SPECIFIED

If an execute exception occurs with the processor status register (PSR) fast interrupt enable (FE) bit set and the exception enable (EE) bit is cleared, and a fast interrupt occurs, an unrecoverable exception is taken, which is the correct behavior. But the PSR FE bit is cleared.

Workaround:

To limit exposure to this situation, save the EPC/EPSR and set the PSR EE bit as soon as possible after the EE bit is cleared.

ACCESS ERROR WITH BKPTS ON JSRI OR JMPI TABLE ACCESSES

If there is a data breakpoint on the table access of a JSRI or JMPI instruction, and an access error (TEA_B) occurs on the destination fetch, an access error exception is taken instead of recognizing the breakpoint. This is only apparent in debug mode.

Workaround:

If possible, alter code so that JMPI/JSRI instructions do not cause an access error. Otherwise, don’t allow data breakpoints on JSRI/JMPI table accesses. This is not a problem if using SDS tools with an EBDI or HP probe.

NON-ATOMIC OPERATION OF MTCR FOLLOWED BY LOW-POWER INSTRUCTION

If a low-power instruction follows an MTCR instruction, the two are not treated as an atomic operation if the instructions are executed out of wait-stated memory. In this case, if the MTCR instruction enables interrupts, an interrupt may be taken before the low-power instruction can be executed.

Workaround:

Run the MTCR/low-power mode sequence in zero wait-stated memory to make this an atomic operation.

PST PINTS MAY NOT REFLECT STATE OF PROCESSOR

Processor status pins (PST) are for debug purposes only and may not truly reflect the state of the machine.

Workaround:

None

USER MODE EXCEPTION ERROR WITH HANDLER IN SUPERVISOR SPACE

If the processor status register (PSR) supervisor/user (SU) bit is cleared, and the processor is executing from memory with two or more wait states, and there is an exception and the handler memory is in supervisor space, the first instruction fetch of the exception handler occurs while the processor is still in user mode. This causes an unrecoverable error.

Workaround:

Run in supervisor mode or place the exception handler in a memory location where user mode accesses are permitted.

UNNECESSARY TRACE EXCEPTIONS OCCUR AROUND RTE, RFI, MTCR, AND MFCR INSTRUCTIONS

If the processor is executing from wait-stated memory and the processor status register (PSR) trace mode (TM) bit is set, unnecessary trace exceptions occur around RTE, RFI, MTCR, and MFCR instructions, which could potentially lock up the machine.

Workaround:

Execute from zero-wait stated memory when debugging code with trace mode enabled. Other choices are don't use trace mode or use the SDS tool with an EBDI or HP probe.

CORRUPTED FPC WITH CONCURRENT DIVIDE BY 0, BKPT, AND FAST INTERRUPT

In debug mode, if a divide by 0, a breakpoint, and a fast interrupt all occur at the same time, the fast interrupt shadow program counter (FPC) may become corrupted.

Workaround:

Don't divide by 0.

CANNOT WRITE OTC, MBCA, AND MBCB REGISTERS DURING RESET

Three JTAG registers (OTC, MBCA, MBCB) are not writable during reset. These are the trace and breakpoint a and breakpoint b counters.

Workaround:

Enter debug mode out of reset, then write these registers. The single-step debugger takes care of this problem. This is not a problem if using SDS tools with an EBDI or HP probe.

IMPROPER SETTING OR CLEARING OF SWO BIT ON ENTERING/EXITING DEBUG MODE

The OnCE status register (OSR) SWO bit may be inadvertently set when debug mode is entered. This bit also may not be cleared when exiting debug mode.

Workaround:

Do not rely solely on the SWO bit to determine what caused entry into debug mode. The single-step debugger takes care of this. This is not a problem if using SDS tools with an EBDI or HP probe.

LSRL MODE DOES NOT FUNCTION PROPERLY

LSRL mode does not function properly.

Workaround:

Do not shift regsel = LSRL into the JTAG IR. This is used in factory test only.

BREAKPOINT WITH OCR FDB BIT SET CAN CAUSE ACCESS ERROR EXCEPTION

When the OncE control register (OCR) FDB bit is set (occurs when using the single-step debugger), if there is a data breakpoint on the first fetch of a load multiple (LDM/LEQ) and an access error occurs (TEA_B asserted) on the second fetch of the load multiple, an access error exception is taken instead of entering debug mode.

Workaround:

If possible, alter code so that load multiple accesses do not cause access errors. Otherwise, don’t set data breakpoints on load multiple data accesses. This is not a problem if using SDS tools with an EBDI or HP probe.

INCORRECT PC INCREMENT ON LOAD/STORE INSTRUCTIONS WITH WAIT-STATED MEMORY

When the OnCE control register (OCR) FDB is set (occurs when using the single-step debugger), if a data breakpoint occurs in conjunction with an access error (TEA_B asserted) on a load/store type (LDQ/STQ/LDM/STM/LD/ST) instruction to/from wait-stated memory, the PC is not incremented correctly. This causes problems when emulator software scans the CPUSCR and subtracts 4 from the PC, because the PC correction flag (PCOK) bit is set in the OnCE TCL register. When debug mode is exited, the PC incorrectly points to the instruction prior to the load/store instruction.

Workaround:

If possible, alter code so that load/store type instructions do not cause access errors. Otherwise, avoid setting data breakpoints on wait-stated memory. This is not a problem if using SDS tools with an EBDI or HP probe.

INSTRUCTION TRACE MODE ERROR WHEN ENTERING DEBUG AFTER RTE OR RFI

In instruction trace mode (PSR TM bits set to 01), if debug mode is entered immediately after an RTE or RFI instruction, the RTE or RFI is traced after exiting debug mode.

Workaround:

Do not use instruction trace mode when debug mode may be entered. This is not a problem if using SDS tools with an EBDI or HP probe.

CORRUPTED OnCE WRITE-BACK BUS REGISTER

In a OnCE single-step of an LD.W instruction, the OnCE write-back bus register (WBBR) may be corrupted.

Workaround:

Follow OnCE single-step of an LD.W to the Rx register with MOV Rx,Rx to correct the value in the WBBR. The single step debugger takes care of this. This is not a problem if using SDS tools with an EBDI or HP probe.

SEQUENTIAL BREAKPOINTS DO NOT FUNCTION AS SPECIFIED

When the OnCE control register (OCR) sequential control field (SQC[1:0]) has a value of 10 or 11, sequential breakpoints do not function as specified. In addition, the OnCE status register (OSR) sequential breakpoint (SQB,SQA) bits are not updated properly in all sequential breakpoint modes.

Workaround:

Don’t use sequential breakpoints. Development tools take care of this in another way. This is not a problem if using SDS tools with an EBDI or HP probe.

INADVERTENT DATA FETCHES

When OnCE control register (OCR) FDB bit is set (occurs when using the single-step debugger) on a data breakpoint with the buffer full (occurs with zero wait-stated instruction memory) a subsequent load or store instruction causes inadvertent data fetches before the processor enters debug mode. The address of the inadvertent data fetches is equal to the data from the previous load.

Workaround:

These extraneous data fetches should not cause problems. If that is not the case, avoid data breakpoints on instructions followed by multicycle instructions. This is not a problem if using SDS tools with an EBDI or HP probe.

Additional mask set erratas can be found at http://www.motorola.com/semiconductors/ on the World Wide Web.

MOTOROLA