MAC71*x*1 Microcontroller Device Mask Set Errata

This document identifies implementation differences (summarized in Table 1) between specific MAC7100 family microcontroller mask sets and the functional descriptions contained in the *MAC7100 Microcontroller Family Reference Manual* (MAC7100RM). Refer to http://www.freescale.com for the latest updates.

1 Introduction

This errata provides information applicable to the following MCU mask set devices:

- 0L49P mask of MAC7101, MAC7111, MAC7121, MAC7131, MAC7141
- 1L49P mask of MAC7101, MAC7111, MAC7121, MAC7131, MAC7141
- 0L47W mask of MAC7101, MAC7111, MAC7121, MAC7131, MAC7141
- 1L47W mask of MAC7101, MAC7111, MAC7121, MAC7131, MAC7141

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

1.1 MCU Device Part Number Prefixes

All MAC7100 family devices are marked with a PAC, MAC or SAC prefix. These prefixes denote the following:

PAC	Devices that have been tested, but are not fully characterized or qualified over the full range of normal manufacturing process variations.
MAC	Fully characterized and qualified standard devices.
SAC	Fully characterized and qualified special or custom devices.

1.2 MCU Device Mask Set Identification

The mask set of a device is identified by a four-character code consisting of a letter, two numerical digits, and a letter, for example L49P. Slight variations to the mask set identification code may result in an optional numerical digit preceding the standard four-character code, for example 0L49P.

1.3 MCU Device Date Codes

In addition to the part number and mask set markings, each device is marked to indicate the week of manufacture. 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 example, the date code “0412” indicates that the device was manufactured during the 12th week of the year 2004.

1.4 Errata System Tracking Numbers

MUCts0*xxxx* is the tracking number for MAC7100 family device errata. An errata number can be used with the mask set and date code to identify a specific errata to a Motorola representative.

2 Errata Summary

Table 1. Summary of MAC71*x*1 Mask Set Errata

Errata Number	Brief Description	Module(s) Affected	Mask Set Affected			
			0L49P	1L49P	0L47W	1L47W
MUCts01045	RESET Timing Restricted	CRG	Yes	Yes	—	—
MUCts01057, MUCts01109	Read to CFM DACC Registers Returns Incorrect Value	CFM	Yes	Yes	—	—
MUCts01058	eDMA Bus Error On The Last Read/Write Is Not Recorded	eDMA	Yes	Yes	—	—
MUCts01059	FlexCAN May Fail To Reset The Error Status Bits After CPU Reads ESR	FlexCAN	Yes	Yes	—	—
MUCts01060	FlexCAN May Transmit a Malformed Frame After Bus Off	FlexCAN	Yes	Yes	—	—
MUCts01061	FlexCAN Requires Delay Before Attempting Consecutive Writes to Error Counter	FlexCAN	Yes	Yes	—	—
MUCts01062	FlexCAN Resets DOZE Bit On Wake-up From Stop Mode	FlexCAN	Yes	Yes	—	—
MUCts01063	FlexCAN Generates Wake-up Interrupt Before Finishing Exit From Doze Mode	FlexCAN	Yes	Yes	—	—
MUCts01065	Deactivation of a Tx Buffer During the Reception of Remote Request Frame	FlexCAN	Yes	Yes	—	—
MUCts01067, MUCts01110	Additional Block Programmed Due to Pipeline Issue / Protection Violation	CFM	Yes	Yes	—	—
MUCts01074	LIN Physical Bus Error Detection Fails	eSCI	Yes	Yes	—	—
MUCts01075	System Services Module WAKEUP Register Bit 1 Is Never Set	SSM	Yes	Yes	—	—
MUCts01076	eMIOS Wrong Internal Counter Enable	eMIOS	Yes	Yes	—	—
MUCts01091	Incorrect Nexus Messages When Exiting Debug Into Thumb State	CPU Core / Nexus	Yes	Yes	—	—
MUCts01107	Floating TA Signal On Some Package Variations Causes Increased Current	EIM	Yes	Yes	—	—
MUCts01108	Unable to Reset Debug Logic After Power On	JTAG / Nexus	Yes	Yes	—	—

Table 1. Summary of MAC71x1 Mask Set Errata (continued)

Errata Number	Brief Description	Module(s) Affected	Mask Set Affected			
			0L49P	1L49P	0L47W	1L47W
MUCts01114	Slave Timeout in LIN Mode Occurs Too Early At Lower Baud Rates	eSCI	Yes	Yes	—	—
MUCts01115	Nexus EVTI Debug Request Does Not Work	CPU Core / Nexus	Yes	Yes	—	—
MUCts01134	Cannot Read FlexCAN Timer and Error Counters Reliably Out of Freeze Mode	FlexCAN	Yes	Yes	—	—
MUCts01165	FlexCAN May Transmit The Wrong ID	FlexCAN	Yes	Yes	—	—
MUCts01195	FlexCAN Intermittent Tx/Rx Failures When Using the Oscillator Clock	FlexCAN	Yes	Yes	—	—
MUCts01247	No Wake-up From Doze Mode Using an Interrupt	INTC	Yes	Yes	—	—
MUCts01248	DMA Request Not De-asserted When Channel is De-activated	DMA Mux	Yes	Yes	—	—
MUCts01249	Incorrect External Bus Address Incrementing Using TA in Auto-Acknowledge Mode	EIM	Yes	Yes	—	—
MUCts01257	LIN Slave Timeout Flagged When Data Not Read	eSCI	Yes	Yes	—	—
MUCts01347, MUCts01374	MB Deactivation in Bus Off Holds Arbitration Until Reception	FlexCAN	Yes	Yes	Yes	Yes
MUCts01364	Link Register Not Properly Updated On Data Aborts in Thumb State	CPU Core	Yes	Yes	Yes	Yes
MUCts01460	Incorrect ADDR[1:0] and BS1 Outputs for Thumb Mode Instruction Fetches	EIM	Yes	Yes	Yes	—
MUCts01474, MUCts01476	Limitations of DSPI Continuous Chip Select Mode	DSPI	Yes	Yes	Yes	Yes
MUCts01515	Wake-up From Pseudo-Stop May Cause System Failure	VREG, CRG	—	—	Yes	—
MUCts01526	Writes to eMIOS B Register May Cause Match Ignore in OPWM Mode	eMIOS	—	—	Yes	Yes
MUCts01527	Flash Programming Overwrites PIM Registers	CFM, PIM	—	—	Yes	—
MUCts01593	Rx MB Receives Data and ID Transmitted By Tx MB	FlexCAN	Yes	Yes	—	—
MUCts01642 to MUCts01651	Writing Undefined Address Causes Bus Abort But Writes Register	ATD, CRG, DMA Mux, DSPI, eMIOS, eSCI, I^2C, PIT, VREG	—	—	Yes	Yes
MUCts01832, MUCts01833	Deactivating A Receive MB May Corrupt Another Active Receive MB	FlexCAN	Yes	Yes	Yes	Yes
MUCts01849, MUCts01855	Changing CTARs Between Frames in Continuous PCS Mode May Cause Error	DSPI	Yes	Yes	Yes	Yes
MUCts01888, MUCts01889	IPM/IPWM Modes, Value Read From UCAn May Be Incorrect	eMIOS	Yes	Yes	Yes	Yes
MUCts01916	VREG High Temperature Control Register (VREGHTCL) For Factory Use Only	VREG	—	—	Yes	Yes
MUCts01931, MUCts01932, MUCts01933	Clock Monitor Reset Causes System Lock-up	CRG	Yes	Yes	Yes	Yes
MUCts02084, MUCts02282	MCM Reset Status Register (MRSR) Always Reads 0x80	MCM	Yes	Yes	Yes	Yes
MUCts02092	FlexCAN Transmit Buffers May Freeze or Indicate Missing Frame	FlexCAN	—	—	Yes	Yes
MUCts02511	Debug Status Port Mode 2 Incorrect Signal Assignments in Documentation	SSM	—	—	Yes	Yes

3 Errata Details

This section provides a detailed description of each errata and a description of a possible work-around, where appropriate.

3.1 MUCts01045 — $\overline{\text{RESET}}$ Timing Restricted

Description

If the external $\overline{\text{RESET}}$ line is held low, and released within 64 clock cycles after the CRG stops driving it, the JTAG/Nexus modules will remain in reset for a further 64 clock cycles.

Work-Around

The problem can be avoided by these methods:

1. External resets are applied to the $\overline{\text{RESET}}$ pin for less than 256 clock cycles
2. External resets are applied to the $\overline{\text{RESET}}$ pin for more than 320 clock cycles
3. No restriction on asserting the $\overline{\text{RESET}}$ pin, but JTAG or Nexus commands will not be applied before 321 clock cycles since assertion of the $\overline{\text{RESET}}$ pin have passed.

3.2 MUCts01057, MUCts01109 — Read to CFM DACC Registers Returns Incorrect Value

Description

The DACC registers for the Instruction and Data Flash (CFMDACC and CFMDFDACC) return incorrect values when read after writing. The value written into the registers will be correctly written. However, the value read back will be the bitwise inversion of the value written. The description of the bits is as follows:

- DACC[M] = 0: Program Flash logical sector M is placed in data address space.
- DACC[M] = 1: Program Flash logical sector M is placed in data and instruction address space.

In summary, the values read will be:

1. If the Flash is erased (i.e., programmed to all 1’s), the corresponding DACC register will contain all 1’s, meaning that all logical sectors are placed in both data and instruction address space.
2. If a ‘1’ is written to DACC[M], the value read back for that bit will be ‘0’. The corresponding logical sector will be placed in both data and instruction address space.
3. If a ‘0’ is written to DACC[M], the value read back for that bit will be ‘1’. The corresponding logical sector will be placed only in data address space.

Work-Around

When reading the DACC register for either the Instruction or Data Flash, always perform a bitwise inversion of the value read in order to determine the value actually stored in the register. For future compatibility, this inversion should be conditional. Future members of the MAC7100 family will return correct values when the DACC registers are read.

3.3 MUCts01058 — eDMA Bus Error On The Last Read/ Write Is Not Recorded

Description

The eDMA records a bus error in the programmer's model via the DMAES and DMAERR registers. The DMAES register records the source of the error and the DMAERR indicates an error status on a per channel basis. The DMAERR has the ability to generate an error interrupt if properly enabled. A bus error on the last read/write before a channel is retired or preempted is not captured in the DMAES or DMAERR registers. The dma_engine recognizes the error and cancels any remaining transfers but the error status registers are not updated in the programmer's model. The TCD.done status is correct. The TCD.done bit is not asserted if the major loop is exhausted and a bus error occurs during the last read/write. Typically, a bus error occurs during application software development in response to an access to a reserved or illegal location. For example, an address that accesses an unused portion of the memory map generates a bus error.

Work-Around

1. Verify the eDMA does not use any illegal addresses, or
2. Configure the channel's TCD.biter = TCD citer = 1 and verify the TCD.done bit is set after the channel completes.

3.4 MUCts01059 — FlexCAN May Fail To Reset The Error Status Bits After CPU Reads ESR

Description

When the CPU reads the Error and Status Register (ESR), FlexCAN should reset the error status bits (BIT1_ERR, BIT0_ERR, ACK_ERR, CRC_ERR, FRM_ERR), but it may fail to do so if the read operation is done shortly after the time in which the error was detected. To be able to reset the error bits, FlexCAN needs one CAN domain clock cycle before attempting to read the ESR Register.

Work-Around

A general solution is to repeat the read operation until ESR is reset. If errors are handled by servicing the error interrupt, then the time it takes from the error detection until the CPU is able to read ESR should be large enough, unless the difference between the CAN and bus clock frequencies is high. The required time between error interrupt and ESR reading is one CAN domain clock cycle (normally the crystal oscillator clock period).

3.5 MUCts01060 — FlexCAN May Transmit a Malformed Frame After Bus Off

Description

In Bus Off state, the Error Counter counts sequences of 11 recessive bits. During Bus Off, if Freeze Mode is entered when the Error Counter is at 127, or is written to 127 when already in Freeze Mode, and if there was a pending frame to be transmitted, then upon leaving Bus Off FlexCAN will transmit the pending frame with a wrong DLC.

Work-Around

If Freeze Mode is entered during Bus Off, read the value of the Error Counter. If it is 127, something else should be written to it, for example 126. Care should be taken when writing to the Error Counter, as it should only be decremented. If the Error Counter is incremented, then FlexCAN will break the CAN protocol by sending a frame before 128 sequences of 11 recessive bits have passed, as required by the CAN standard.

3.6 MUCts01061 — FlexCAN Requires Delay Before Attempting Consecutive Writes to Error Counter

Description

After writing to the Error Counter in Freeze Mode, the CPU should keep reading the Error Counter to discover when the write operation finished. After reading the Error Counter and confirming that the write operation has finished, if the CPU attempts to write again immediately to the Error Counter, the new write operation may fail because FlexCAN did not finish yet processing the previous write operation.

Work-Around

After confirming the first write operation by successfully reading the Error Counter, if it is desired to write the Error Counter again, a delay should be inserted. The required delay is 3 CAN clock cycles plus 3 bus clock cycles. The CAN clock is the clock applied to the CAN engine. The bus clock is the clock applied to the peripheral bus.

3.7 MUCts01062 — FlexCAN Resets DOZE Bit On Wake-up From Stop Mode

Description

When FlexCAN wakes-up from Stop Mode due to CAN bus activity, the DOZE bit in the MCR Register is automatically reset. This should only happen upon wake-up from Doze Mode.

Work-Around

Re-program the DOZE bit after wake-up from Stop Mode due to CAN bus activity.

3.8 MUCts01063 — FlexCAN Generates Wake-up Interrupt Before Finishing Exit From Doze Mode

Description

Upon wake-up from Doze Mode, an interrupt is generated as soon as activity is detected on the CAN bus. When this happens FlexCAN resets the DOZE bit in MCR and exits Doze Mode. The wake-up interrupt is generated before FlexCAN completes the exit from Doze Mode procedure. If the CPU clears the interrupt flag before FlexCAN exits Doze Mode, the interrupt flag may be set again by activity on the CAN bus.

Work-Around

Upon receiving a wake-up from Doze Mode interrupt, make sure to clear the interrupt flag only after FlexCAN exits Doze Mode. This information can be obtained by reading the LPM_ACK bit in the MCR Register.

3.9 MUCts01065 — Deactivation of a Tx Buffer During the Reception of Remote Request Frame

Description

When FlexCAN receives a remote request frame, the code field and RTR bit of matched message buffer are changed (at 6th bit of EOF) to make it a Tx MB and allow it to participate in the next arbitration process. If the user writes to the CODE field of the highest priority Tx MB during the reception of a remote request frame, the RTR bit may not be reset, therefore this MB may be transmitted as remote frame and not data frame.

Work-Around

Do not write to the CODE field of active Tx MBs (transmission pending) when not in Freeze Mode. If it is necessary to abort the transmission of an MB, go to Freeze Mode first and then write to the CODE field. Note that there is no restriction to writing to inactive MBs to initiate a new Tx or Rx.

3.10 MUCts01067, MUCts01110 — Additional Block Programmed Due to Pipeline Issue / Protection Violation

Description

After pipelining a CFM command (interlocked or not), if another command to a different block follows, the command will be executed on both the new and the previously pipelined block(s). Generating a protection violation on one block, clearing the violation and writing to a different block will result on both blocks having the current command executed on them. This is due to an internal register not being cleared during the pipelined operation or after a protection violation.

Work-Around

To get around this issue, an access error must be generated after pipelining or a protection violation must be generated, and then the access error may immediately be cleared. This forces a clear of the internal register. The recommended way to generate an access error is by initiating a command sequence which writes to the same block twice (no command needs to be written). The access error and protection flags must be cleared separately.

3.11 MUCts01074 — LIN Physical Bus Error Detection Fails

Description

For LIN operation, the eSCI may incorrectly flag a physical bus error. For baud rates with a prescaler greater than one, it will flag physical bus errors despite the LIN bus behaving correctly. This error causes a reset to the LIN FSM, which aborts the current frame.

Work-Around

If the LDBG bit is set, detected errors will not cause the LIN FSM to reset. This should work fine for normal operation, and LIN will work as intended for all baud rates. In addition the physical bus error flag must be masked out.

Genuine physical bus errors must be detected by software. In some cases a physical bus error will cause large numbers of bit errors and can thus be detected. If the bus is stuck at a constant value, however the transmitted byte can not be read back, and thus bit errors will not be flagged.

For TX frames this can be solved using an external reference (for example, a timer or periodic calls to the driver routines). If after the normal timeout period the frame has not completed (FRC not set) then this indicates a physical bus error.

For RX frames an external reference is also required. If after the timeout period the RX header has not been transmitted (i.e. TXRDY flag is not set - see MUCts01114 – Slave-Timeout in LIN mode occurs too early at lower baud rates), then this indicates a physical bus error. If the header has been transmitted, but the frame is not completed, then a slave timeout error has occurred.

3.12 MUCts01075 — System Services Module WAKEUP Register Bit 1 Is Never Set

Description

In the WAKEUP register in the System Services Module, bit [1] corresponds to a wake-up from the interrupt controller. This bit will never get set, even if the interrupt controller causes the wake-up event. Although this bit is never set, the wake-up functionality is unaffected, and the system will wake-up as specified when an interrupt source is used.

Work-Around

The source(s) for a system wake-up event can not be determined solely by reading the WAKEUP register in the SSM. In order to determine all sources for a wake-up from either STOP or DOZE, the following steps must be taken:

1. Read the WAKEUP register in the SSM. If no bits are set, then the wake-up source must have come from the interrupt controller. In either case, proceed to step 2.
2. Read the IPRL/IPRH registers in the INTC to determine interrupt sources that may have caused a wakeup event.

Note that there may be multiple wake-up sources active at any one time.

If any interrupt source is pending when the system wakes up, then that interrupt will be taken immediately. Therefore, it is also possible to determine whether an interrupt source was active at wake-up from STOP mode by examining the flow of code execution following the write to the SDMCTL register. If a service routine associated with an interrupt wake-up source is entered immediately after the write to the STOP command, then at least one interrupt source caused the wake-up. The following steps illustrates this:

1. Clear a software flag ‘INT_WAKEUP’ (for example)
2. Write the SDMCTL[STOP] bit (enter STOP/pseudo-STOP mode)
3. Execute a NOP command (ensures next command will not be executed before the interrupt is serviced)
4. Is ‘INT_WAKEUP’ set? Yes: interrupt must have been serviced. No: no interrupt wake-up sources were pending after wake-up.
5. In the ISR(s) for any interrupt sources that are used as wake-ups, set the ‘INT_WAKEUP’ flag

Note that this procedure will not work for DOZE mode, as the core continues to execute commands after the SDMCTL register is written.

3.13 MUCts01076 — eMIOS Wrong Internal Counter Enable

Description

The Unified Channel ignores the prescaler divide ratio when running the WPTA mode. The increment ratio of the internal counter is always 1 when running this mode, regardless of the prescaler divide ratio programmed by GPRE and UCPRE values. When running the QDEC mode, if the divide ratio is not set

to “divide by 1” (GPRE and UCPRE cleared) then the Unified Channel operating in QDEC mode may lose input events because the channel counter is operating at the prescaler rate (with less resolution).

Work-Around

When running the WPTA mode, reduce the time window programmed by MTSA and MTSB registers, in order to avoid a premature internal counter overflow. Prior to running the QDEC mode, write 0 to GPRE field (MTSMCR register) and 0 to UCPRE (MTSC register of the Unified Channel running QDEC). Note that GPRE = 0 will affect other channel that may have been configured assuming a global prescaler divide ratio greater than 1.

3.14 MUCts01091 — Incorrect Nexus Messages When Exiting Debug Into Thumb State

Description

If a program is running in 0-wait-state and debug mode is exited to Thumb state and Nexus program trace is enabled, the fields of the first Nexus messages will contain the following information which deviates from Nexus Spec:

- The first message since returning from Debug which has a history field will have one extra history bit in the most significant bit following the stop bit (this message should be either a HIST/s or a RFM)
- The first instruction count of the first message, since returning from Debug, will have 1 less count if there was no direct branch preceding this message (In other words if the first message is a history message, the history field must be 0x3 or 0x2 (1 bit extra because of the error in the first history field)) (this first message should be either an IBM/s, DBM/s or HIST/s).

Work-Around

One of the following:

1. Program the memory to response with at least one wait state, if possible
2. Avoid exiting into Thumb state from debug.
3. Ignore the first history field and the first count field. Use the sync address for reconstruction
4. After message Debug exit, take out 1 bit following the stop bit in the first HIST/s or RFM message. Also add 1 to the instruction count if the first message is an IBM/s or a DBM/s or a HIST/s with a history of 0x3 or 0x2.

Note: These work-arounds only apply to exiting debug mode into Thumb state when executing from 0-wait-state memory.

3.15 MUCts01107 — Floating $\overline{TA}$ Signal On Some Package Variations Causes Increased Current

Description

On mask set L49P MAC7101, MAC7121 and MAC7141 devices, where the $\overline{TA}$ signal is not bonded out, this signal is left internally floating, which can cause increased current drain on the device.

Work-Around

There is no work-around available on mask set L49P devices. Mask set L47W and later devices implement programmable pull-up/down devices that may be enabled to prevent signal float.

3.16 MUCts01108 — Unable to Reset Debug Logic After Power On

Description

Once the device has been powered on, it is not possible to reset the EICE and Nexus, including all programmable registers.

Work-Around

The work-around is to write the reset value of each EICE and Nexus register into the registers and put the TAP controller into the Test-Logic Reset state (by holding TMS high for the required number of cycles).

3.17 MUCts01114 — Slave Timeout in LIN Mode Occurs Too Early At Lower Baud Rates

Description

For baud rates with a prescaler greater than one, the eSCI slave timeout detection will flag timeout errors even if a slave responds within the required time frame. This causes LIN RX frames to be aborted after the header has been transmitted.

Work-Around

In the LINCTRL register, set STIE to 0, which prevents an STO timeout interrupt from being generated automatically by the LIN protocol controller. With this interrupt disabled, slave timeouts must be detected in a different manner. This can be done by using an external time reference (for example, a timer or periodic calls to the driver routines).

Although the timeout interrupt will not be generated and the header for RX frames will be transmitted normally, the frame will be aborted immediately after the header transmission. The end of the header transmission is indicated by a raised TXRDY flag, however, any byte arriving from a LIN slave node will

be judged to be an unrequested byte because the LIN protocol controller believes the timeout period has expired. This caused a Bit Error and hence raises a BERR flag as well as an RXRDY flag. BERR flags which are header related occur before TXRDY is set, the work-around must therefore ignore BERR flags after the header transmission.

Checksum and CRC checking must be handled in software for RX frames.

3.18 MUCts01115 — Nexus $\overline{\text{EVTI}}$ Debug Request Does Not Work

Description

Using the $\overline{\text{EVTI}}$ pin on the Nexus interface to trigger an external debug request to the ARM7™ core does not work. All alternate $\overline{\text{EVTI}}$ functionality works as specified.

Work-Around

There is no work-around available.

3.19 MUCts01134 — Cannot Read FlexCAN Timer and Error Counters Reliably Out of Freeze Mode

Description

If the CLK_SRC bit is negated, meaning that CPU and protocol engine will have different clock domains, it is not possible to read the Timer and the Error Counter Register reliably when not in Freeze Mode. If the read operation catches the counter value when it is transitioning, inconsistent data may be obtained because of different delays among individual counter bits.

Work-Around

Only read the Error Counter Register when the module is in Freeze Mode. When reading the Timer for the purpose of unlocking Message Buffers, the actual timer contents must be ignored.

3.20 MUCts01165 — FlexCAN May Transmit The Wrong ID

Description

When a Tx Message Buffer is activated for transmission when the CAN bus is idle, FlexCAN may sporadically transmit a frame with the wrong ID. This problem can only happen when:

- Bus clock is applied to the CAN engine (CTRL[CLK_SRC]=1) and PRESDIV is programmed with a value < 2 (which results in a CPI-to-S clock ratio < 3).
- Oscillator clock is applied to the CAN engine (CTRL[CLK_SRC]=0) and the CAN time quanta frequency is ≤ 4 times the bus frequency (which is the CPU frequency ÷ 2).

Work-Around

Method 1 — When setting the CAN timing parameters, the following restrictions must be observed:

- Set the CTRL[CLK_SRC] bit (select bus clock).
- Set the CTRL[PRESDIV] field to a value ≥ 2 (which results in a CPI-to-S clock ratio ≥ 3).

With the above settings, the module will be fully functional but will have the following limitations:

- To operate at a 1 MHz CAN frequency, it is necessary to run the CPU at 48 MHz.
- The jitter performance of the CAN bus will be limited by the jitter performance of the PLL.

Method 2 — Before activating a Tx MB for transmission (by writing to C/S word), follow this procedure (may be used regardless of the CTRL[CLK_SRC] setting (oscillator or bus clock)):

1. Read the IDLE bit in ESR
2. If IDLE is set, read the timer once, then keep reading the timer until it changes
3. Write to the C/S word of the MB to activate the transmission

3.21 MUCts01195 — FlexCAN Intermittent Tx/Rx Failures When Using the Oscillator Clock

Description

When the CAN engine is programmed to operate with the oscillator clock (CLK_SRC=1), there is a race condition between clock domains that may cause two sporadic anomalous behaviors:

1. Tx messages may get retransmitted, and
2. Rx messages may not be stored in a message buffer that has a valid matching ID.

In both cases, a successful transmission or reception is not properly communicated by the CAN engine to the message buffer control logic, so the corresponding interrupt flag will not be generated, and false error flags may be set in the Error and Status Register. In case of a Tx frame, when a new arbitration is initiated (new transmission or reception in another MB), then the frame is retransmitted because the code field indicates a transmission is still pending.

Work-Around

Use the bus clock to feed the CAN engine by setting the MCR[CLK_SRC] bit.

3.22 MUCts01247 — No Wake-up From Doze Mode Using an Interrupt

Description

When the system is put into Doze mode, only the following wake-up sources will cause a wake-up:

- PIT (RTI)
- FlexCan A/B/C/D

All other wake-up sources, including all other peripherals and external interrupts will not cause the system to leave Doze mode.

Work-Around

When the system is in Doze mode, the CPU and interrupt controller are still running. Therefore, the system will still correctly service any interrupts. In order to emulate the correct behavior, all Interrupt Service Routines (ISRs) should check whether the system is in Doze mode, and if desired, wake up the system by clearing the DOZE bit in the CRG.

3.23 MUCts01248 — DMA Request Not De-asserted When Channel is De-activated

Description

If a dynamic channel switch is attempted in the DMA Channel Mux while the eDMA is servicing requests on the channel, the DMA request may not get properly de-asserted. This may result in an errant request for the new channel.

Work-Around

In order to dynamically switch DMA channels in the DMA Channel Mux, perform the following series of steps:

1. Ensure all pending requests are finished by reading the appropriate register(s) in the eDMA
2. Disable the DMA channel in the eDMA
3. Switch the channel in the DMA Channel Mux
4. Re-enable the DMA channel in the eDMA

3.24 MUCts01249 — Incorrect External Bus Address Incrementing Using $\overline{TA}$ in Auto-Acknowledge Mode

Description

If the external bus is used and set to auto-acknowledge mode (by driving pin PA15 high during reset), then the address presented on the external bus may not be correct in the following situation:

1. External bus set to Auto Ack mode (by driving pin PA15 high during Reset)
2. External $\overline{TA}$ signal asserted (for “early” termination of the Auto Ack)
3. Decomposed transfer (in other words, a 32-bit access with a 16- or 8-bit port size, a 16-bit access with an 8-bit port size)

If all of the above conditions are met, then the address presented on the external bus will not be incremented during the course of a single transfer. For example, on a 32-bit access to address 0x00_0000, with burst inhibit set and a 16-bit port size, the address on the external bus should be presented as 0x00_000 for the first acknowledge, and as 0x00_0002 for the second acknowledge. If the above conditions are met, then only address 0x00_0000 will be presented for both acknowledges.

Work-Around

There is no work-around available.

3.25 MUCts01257 — LIN Slave Timeout Flagged When Data Not Read

Description

The eSCI will raise a slave-timeout flag when a LIN frame is not completely received by the specified timeout period. When the last byte of the frame is received but not read by the CPU the slave-timeout counter will continue counting. This will cause a slave-timeout error if the data is not read before the counter expires. Since the slave already sent the data, flagging a slave-timeout is not correct.

Work-Around

The LIN driver must be written so that the last data byte of a frame can be read from the eSCI before the timeout period expires. If necessary, the timeout period can be increased to make this possible.

3.26 MUCts01347, MUCts01374 — MB Deactivation in Bus Off Holds Arbitration Until Reception

Description

If all Tx Message Buffers (MB) are deactivated while the module is in Bus Off state, and then upon exiting Bus Off one or more Tx MBs are activated while the bus is Idle, the frames will not be transmitted until FlexCAN detects a frame on the bus. In other words, after Bus Off it will not take the initiative to be the first to transmit, unless at least one Tx MB was activated during Bus Off, or already programmed Tx MBs were not deactivated during Bus Off.

Work-Around

If FlexCAN is required to take the initiative to be the first to transmit after Bus Off, the following alternatives (one of them) must be considered:

1. Do not deactivate all Tx MBs during Bus Off
2. Activate the Tx MBs before exiting Bus Off
3. After Bus Off, put the module in Freeze Mode before programming the Tx MBs

3.27 MUCts01364 — Link Register Not Properly Updated On Data Aborts in Thumb State

Description

The following information is from the *ARM7TDMI-S Errata List* (document FR002-PRDC-002719 3.0):

Summary
If the processor is in Thumb state and executing the code sequence STR, STMIA or PUSH followed by a PCrelative load, and the STR, STMIA or PUSH is aborted, the PC is saved to the abort link register in only word resolution, instead of half-word resolution.

Conditions
The processor must be in Thumb state, and the following sequence must occur:

```
<any instruction>
<STR, STMIA, PUSH> <---- data abort on this instruction
LDR rn, [pc,#offset]
```

In this case the PC is saved to the link register R14_abt in only word resolution, not half-word resolution. The effect is that the link register holds an address that could be #2 less than it should be, so any abort handler could return to one instruction earlier than intended.

Implications
- In a system that does not use Thumb state, there will be no problem.
- In a system that uses Thumb state but does not use data aborts, or does not try to use data aborts in a recoverable manner, there will be no problem.
- In a system that uses Thumb state, and uses data aborts in a recoverable manner, such as in a demand paging environment, where the STR, STMIA or PUSH aborts for paging reasons, the code will patch up the MMU and reexecute the STR, STMIA or PUSH. What matters in this case is the instruction preceding the STR etc, and whether this can be re-executed harmlessly. As this cannot be predicted with certainty, it is likely to be a problem.

Impacted revisions
This erratum impacts all revisions of ARM7TDMI-S up to and including r4p2.

Work-Around

The work-around is to ensure that a STR, STMIA or PUSH cannot precede a PC-relative load. One method for this is to add a NOP before any PC-relative load instruction. However this is not something currently supported by ARM™ software tools, and would have to be done manually.

This erratum will be fixed in r4p3.

3.28 MUCts01460 — Incorrect ADDR[1:0] and $\overline{\text{BS1}}$ Outputs for Thumb Mode Instruction Fetches

Description

If the ARM7 core is executing code in Thumb mode, and this code is stored in an external memory, the instructions fetches to that memory will have an incorrect address and possible incorrect byte strobes. In particular, the expected value of ADDR[1:0] and $\overline{\text{BS}}$[1:0] should be as follows:

Address Ends With	ADDR[1:0]	8-bit Access		16-bit Access	
		$\overline{BS1}$	$\overline{BS0}$	$\overline{BS1}$	$\overline{BS0}$
0, 4, 8, C	00	0	1	0	0
	01	0	1	—	—
2, 6, A, E	10	0	1	0	0
	11	0	1	—	—

However, due to the fact that the ARM7 core sets bit 0 of its internal address bus to 1 for Thumb mode instruction fetches, the actual address and byte strobes that will appear on the external bus is:

Address Ends With	ADDR[1:0]	8-bit Access		16-bit Access	
		$\overline{BS1}$	$\overline{BS0}$	$\overline{BS1}$	$\overline{BS0}$
0, 4, 8, C	01	0	1	0	0
	10	0	1	—	—
2, 6, A, E	11	0	1	1	0
	00	0	1	—	—

Note that the $\overline{BS}$[1:0] signals are correct for all cases except a 16-bit port size to addresses ending with 2, 6, A, E. All data reads and data writes work as specified; only Thumb mode instructions fetches are affected.

Work-Around

If a 16-bit port size is used, it may safely connected to a 16-bit program memory if it does not require the ADDR0 address signal or $\overline{BS1}$ read strobe. If the $\overline{BS1}$ read strobe is required, then it must be qualified such that it is always driven low (asserted) when the R/$\overline{W}$ output is negated (read).

$\overline{BS1}$' = ~R/$\overline{W}$ && $\overline{BS1}$

This will allow the use of any external memory with a 16-bit interface to a 16-bit port size.

Because it is impossible to distinguish between an instruction fetch and a data read on the external bus, it is not possible to interface program memories of any width using an 8-bit port size. Only instruction fetches in Thumb mode with an 8-bit port size are incorrect. All other combinations, including the following examples, work as expected:

- External memories for any data reads/writes with any port size.
- External memories for instruction fetches in non-Thumb mode with any port size.
- External memories for instruction fetches in any mode with 16-bit port size (if the ADDR0 output is unused and the $\overline{BS1}$ output is either unused or qualified as described above).
- Any internal memory for instruction fetches in any mode.

3.29 MUCts01474, MUCts01476 — Limitations of DSPI Continuous Chip Select Mode

Description

If transmit FIFO fill DMA requests are enabled (DSPIx_RSER[TFFF_RE, TFFF_DIRS] = 0b11) and the eDMA is busy servicing other DMA requests, a scenario may occur where the TX FIFO becomes empty. In this circumstance, the associated chip select state will be negated. The effect of negating PCS*n_x* between words in a multi-word continuous transfer will depend on the external serial device; it can cause the second word to overwrite the previous word transfered.

Work-Around

There is no workaround available

3.30 MUCts01515 — Wake-up From Pseudo-Stop May Cause System Failure

Description

After waking up from pseudo-stop mode, the CRG releases all enabled modules to resume execution before the VREG has returned to full-performance mode. Thus, the system may draw too much power before the regulator is able to provide it, and a system failure may result.

Work-Around

No work-around available, do not use pseudo-stop mode.

3.31 MUCts01526 — Writes to eMIOS B Register May Cause Match Ignore in OPWM Mode

Description

When using the double-buffered capabilities of an eMIOS unified channel in OPWM mode, the output of the OPWM channel may remain asserted when it is expected to be negated, depending on the timing of the write to the B register. The problem sequence is:

1. OPWM output is asserted when A match occurs.
2. The A match triggers a software event (normally via an interrupt) to update the B register. B is double buffered in this mode, such that while the host writes a new compare value to B, the OPWM output negates when the timer matches the original value of B, then the new value is copied to the comparator to be used in the next PWM cycle.
3. If the write to B occurs on the same system clock that B match occurs, the match will not be recognized.
4. The OPWM output remains asserted until the next B match following the next A match.

Work-Around

Follow one of the following procedures:

1. Read the UCCNT*n* register before performing the B register write. If the counter value is “just below” the previous B value, then the B register update should be delayed.
2. Write to the B register and then check if the value of the the UCCNT*n* register is bigger than the old B value. If so, the pin value must be forced to the correct state via the UCCR*n*[FORCMB] bit.
3. When using an interrupt service routine to update the B register, verify that the PWM pulse width is larger than the interrupt latency.
4. Keep the B value constant and only perform writes to the A register to alter the pulse width. In this case, the value of A is updated after an interrupt from a previous channel A match. For this to work the pulse period must be greater than the interrupt latency, so that the new A value is written before the next A channel compare is enabled (note that the A register is not double buffered).

3.32 MUCts01527 — Flash Programming Overwrites PIM Registers

Description

Because the PIM module does not correctly generate the an internal enable signal, all IPS writes to a specific address range will result in writes to global control registers in the PIM (refer to Table 18-3 of the *MAC7100 Microcontroller Family Reference Manual*). Specifically, all IPS writes to the following address ranges will map into writes to the global control registers:

```
0xxxxx_03C0 - 0xxxxx_03CF        0xxxxx_83C0 - 0xxxxx_83CF
0xxxxx_43C0 - 0xxxxx_43CF        0xxxxx_C3C0 - 0xxxxx_C3CF
```

As an example, writing to the Flash programming interface at any of the following addresses will write both the Flash (as intended) and the PIM CONFIG_TA register (not intended) with the value intended for the Flash:

```
0xFC10_03CC        0xFC11_03CC
0xFC10_43CC        0xFC11_43CC
0xFC10_83CC        0xFC11_83CC
0xFC10_C3CC        0xFC11_C3CC
```

The full range of affected addresses are:

Program Flash

0xFC10_03C2 - 0xFC10_03CD	0xFC12_03C2 - 0xFC12_03CD
0xFC14_03C2 - 0xFC14_03CD	0xFC16_03C2 - 0xFC16_03CD
0xFC10_43C2 - 0xFC10_43CD	0xFC12_43C2 - 0xFC12_43CD
0xFC14_43C2 - 0xFC14_43CD	0xFC16_43C2 - 0xFC16_43CD
0xFC10_83C2 - 0xFC10_83CD	0xFC12_83C2 - 0xFC12_83CD
0xFC14_83C2 - 0xFC14_83CD	0xFC16_83C2 - 0xFC16_83CD
0xFC10_C3C2 - 0xFC10_C3CD	0xFC12_C3C2 - 0xFC12_C3CD
0xFC14_C3C2 - 0xFC14_C3CD	0xFC16_C3C2 - 0xFC16_C3CD
0xFC11_03C2 - 0xFC11_03CD	0xFC13_03C2 - 0xFC13_03CD
0xFC15_03C2 - 0xFC15_03CD	0xFC17_03C2 - 0xFC17_03CD
0xFC11_43C2 - 0xFC11_43CD	0xFC13_43C2 - 0xFC13_43CD
0xFC15_43C2 - 0xFC15_43CD	0xFC17_43C2 - 0xFC17_43CD
0xFC11_83C2 - 0xFC11_83CD	0xFC13_83C2 - 0xFC13_83CD
0xFC15_83C2 - 0xFC15_83CD	0xFC17_83C2 - 0xFC17_83CD
0xFC11_C3C2 - 0xFC11_C3CD	0xFC13_C3C2 - 0xFC13_C3CD
0xFC15_C3C2 - 0xFC15_C3CD	0xFC17_C3C2 - 0xFC17_C3CD

Data Flash

0xFE00_03C2 - 0xFE00_03CD	0xFE00_43C2 - 0xFE00_43CD

Note that this is not a symmetric situation (i.e., writing the PIM registers will not write the Flash contents).

Work-Around

In the Flash programming routine, perform the following steps when programming a word into the program or data Flash

1. Logical AND the 32-bit address and `0x0000_3FFF`
2. Compare new address to `0x0000_03C0 - 0x0000_03CF`
3. If a hit occurs within the above range, save the values of the PIM registers (starting at address `0xFC0E_83C0`) before performing the programming routine, execute the programming, and then write the previous values of the global control registers back into the PIM.

3.33 MUCts01593 — Rx MB Receives Data and ID Transmitted By Tx MB

Description

When all of the following conditions are present on the FlexCAN module:

- There are Rx and Tx buffers (at least one each),
- Two clock domains (CTRL[CLK_SRC] = 1) or one clock domain with $f_{IPS} < 20$ MHz,
- The application activates a Tx MB to transmit a frame (writing to the C/S word of the MB) near the start bit of an incoming frame.

Under these conditions, there is a probability that the Rx MB receives the frame stored on the Tx MB, even if the ID does not match, and the Tx MB transmits only its ID in the data field.

Work-Around

Program the module to operate with one clock domain and set $f_{IPS} \geq 20$ MHz ($f_{SYS} \geq 40$ MHz).

3.34 MUCts01642 to MUCts01651 — Writing Undefined Address Causes Bus Abort But Writes Register

Description

When writing to registers above the defined register space, but within the 16 Kbyte address range of a peripheral module, a bus abort will be produced but a write to one of the valid registers may still occur. For example, writing to offset `0xFC09_0001`, `0xFC09_1001`, `0xFC09_2001` and `0xFC09_3001` are all decoded to the same register within the VREG module. If the bus abort error is enabled, writing to `0xFC09_1001`, `0xFC09_2001` or `0xFC09_3001` will trigger a bus abort, but the register at offset `0xFC09_0001` will be written.

The following modules are affected by this errata:

Errata Number	Module	Module Address Range	Undefined Register Space Offsets
MUCts01645	DMA Mux	`0xFC08_4000` to `0xFC08_7FFF`	`0x0010` to `0x3FFF`
MUCts01647, MUCts01648	CRG	`0xFC08_8000` to `0xFC08_BFFF`	`0x0009` to `0x3FFF`
MUCts01646	PIT	`0xFC08_C000` to `0xFC08_FFFF`	`0x0114` to `0x3FFF`
MUCts01642	VREG	`0xFC09_0000` to `0xFC09_3FFF`	`0x0004` to `0x3FFF`
MUCts01649	I^2C	`0xFC0A_C000` to `0xFC0A_FFFF`	`0x0005` to `0x3FFF`
MUCts01650	DSPI	`0xFC0B_4000` to `0xFC0B_7FFF` `0xFC0B_8000` to `0xFC0B_BFFF`	`0x008C` to `0x3FFF`

MUCts01644	eSCI	`0xFC0C_4000` to `0xFC0C_7FFF`
		`0xFC0C_8000` to `0xFC0C_BFFF`
		`0xFC0C_C000` to `0xFC0C_FFFF`
		`0xFC0D_0000` to `0xFC0D_3FFF`
			`0x001A` to `0x3FFF`

MUCts01651	eMIOS	`0xFC0D_C000` to `0xFC0D_FFFF`	`0x0220` to `0x3FFF`
MUCts01643	ATD	`0xFC0E_0000` to `0xFC0E_3FFF` `0xFC0E_4000` to `0xFC0E_7FFF`	`0x0018` to `0x3FFF`

Work-Around

There is no work-around available.

NOTE

Previous versions of this document identified this errata as MUCts01366, which did not fully define the error conditions.

3.35 MUCts01832, MUCts01833 — Deactivating A Receive MB May Corrupt Another Active Receive MB

Description

Deactivating a FlexCAN receive message buffer (MB) may cause corruption of another active receive MB if the following sequence occurs:

1. A receive MB is locked via reading the Control/Status word, and has a pending message in the temporary receive serial message buffer (SMB).
2. A message is received that matches a second receive MB, and is queued in the second SMB.
3. The first MB is unlocked during the time between the CRC field and the 6th bit of EOF.
4. The second MB is deactivated within 9 f_{IPS} clock cycles of the first MB being unlocked, resulting in corruption of the first MB.

Work-Around

Do not write to the Control/Status word after initializing a receive MB, and use the IFLAG status bit to determine reception of a new frame, as the Control/Status field will always indicate FULL or OVERRUN after receiving the first frame.

If a write (deactivation) is required to the Control/Status field of an active receive MB, a delay of 25 CAN bit times plus 9 f_{IPS} clock cycles between unlocking one MB and deactivating another MB will avoid corruption, however frames may still be lost.

3.36 MUCts01849, MUCts01855 — Changing CTARs Between Frames in Continuous PCS Mode May Cause Error

Description

Under some conditions in continuous operation mode (CONT = 1), the command word associated with the data frame is not always executed properly. An incorrect transfer may occur when multiple frames are transferred in continuous PCS mode and the frames use different CTAR registers. For example, if an application tries to transmit a 12-bit frame and a 16-bit frame without negating PCS, two 12-bit frames are transferred. This has been observed in simulations where CPHA = 0. The two frames are transmitted correctly if CPHA = 1.

Work-Around

When CPHA = 0 and continuous PCS mode is used, extended length (> 16 bits) frames may be created only by using two frames of equal size. This means that certain frame sizes cannot be constructed (prime numbers > 16).

3.37 MUCts01888, MUCts01889 — IPM/IPWM Modes, Value Read From UCA*n* May Be Incorrect

Description

When reading the UCA*n* register in Input Pulse Width Measurement (IPWM) or Input Period Measurement (IPM) modes, if the IPS bus cycle starts on the same clock cycle as an A2 capture, the data read will not be coherent with the one at the next UCB*n* read.

- In IPWM mode, data read from UCB*n* will be greater than UCA*n* (UCB*n* minus UCA*n* will be the pulse width measurement of the polarity opposite that defined by EDPOL).
- In IPM mode, data read from UCA*n* and UCB*n* will be the same.

The expected scenario is that UCA*n* will be greater than UCB*n* for both modes. Note that coherency is guaranteed in a sequence of several measurements only if the combined UCA*n* / UCB*n* reads for each new measurement are performed after the correspondent new flag event.

Work-Around

After reading UCA*n* and UCB*n*, if UCA*n* is not greater than UCB*n*, discard this pulse measurement and read both registers again in the usual order: first read UCA*n*, then read UCB*n*.

3.38 MUCts01916 — VREG High Temperature Control Register (VREGHTCL) For Factory Use Only

Description

Although information describing the VREGHTCL register is included in the *MAC7100 Microcontroller Family Reference Manual* (MAC7100RM) revisions 0.6, 0.6.1 and 1.0, the functions provided by this register are not fully characterized for customer use. Thus, VREGHTCL is reserved for factory testing during manufacturing processes, and is not suitable for application use.

Work-Around

There is no work-around available, nor are there plans to characterize this circuitry for customer use. Future versions of the MAC7100RM will remove the descriptions of the high temperature functions.

3.39 MUCts01931, MUCts01932, MUCts01933 — Clock Monitor Reset Causes System Lock-up

Description

If the clock monitor reset function is enabled and a clock monitor time-out occurs, the chip will be placed into a reset state, but it will never exit that state. In order to trigger this errata all of the following conditions must be true:

- Clock monitor is enabled (CRG PLLCTL[CME] = 0b1)
- Loss of clock is detected
- Self-clock mode is disabled (CRG PLLCTL[SCME] = 0b0)

The only way to recover from this error is via an external or low-voltage reset sequence.

Work-Around

The default values of the CRG PLLCTL[CME] and PLLCTL[SCME] bits are 0b1, and thus the error is avoided if the clock monitor reset mode is never enabled. In order to prevent inadvertent enabling of the clock monitor reset function, PLLCTL[SCME] should be written with 0b1 during the first write to the PLLCTL register following a reset. Since the PLLCTL[SCME] bit is write once, this will prevent the enabling of clock monitor reset.

There is no plan to correct this function, as will be reflected in future revisions of the *MAC7100 Microcontroller Family Reference Manual* (MAC7100RM).

3.40 MUCts02084, MUCts02282 — MCM Reset Status Register (MRSR) Always Reads 0x80

Description

The reset status register (MRSR) in the MCM module, which is intended to provide a read-only status of the last reset event, returns an incorrect status. In particular, the functionality of the MRSR reporting is limited such that all reset events are reported as power-on resets, regardless of the actual source of the reset. As a result, all reads of the MRSR return a data value of 0x80.

Work-Around

There is no work-around available.

3.41 MUCts02092 — FlexCAN Transmit Buffers May Freeze or Indicate Missing Frame

Description

If a received frame is serviced during reception of a second frame identified for that same MB (message buffer) and a new Tx frame is also initiated during this time, the Tx MB can become frozen and will not transmit while the bus is idle. The MB remains frozen until a new frame appears on the bus.

If the new frame is a received frame, the frozen MB is released and will arbitrate for external transmission. If the new frame is a transmitted frame from another Tx MB, the frozen MB changes its C/S (control status word) and IFLAG to indicate that transmission has occurred, although no frame was actually transmitted.

The frozen MB occurs if lock, unlock and initiate Tx events all occur at specific times during reception of two frames. The timing of the lock event affects the timing window of the unlock event as follows:

Situation A: Rx MB is locked during the second frame.

A frozen Tx MB occurs if:

1. Both of these events occur in either a-then-b or b-then-a order:
   a) A new transmission is initiated by writing its C/S sometime between CRC3 (third bit of CRC field) and EOF7 (seventh bit of end of frame) of the second frame.
   b) The Rx MB is locked by reading its C/S word sometime after EOF6 of first frame and before EOF6 of second frame.
2. The Rx MB is unlocked between EOF7 and intermission at end of the second frame.

Notice in this situation that if the lock / unlock combination happens close together, the lock must have been just before EOF6 of the second frame, and therefore the system is very close to having an overrun condition due to the delayed handling of received frames.

Situation B: Rx MB was locked before EOF6 of the first frame; in other words, before its IFLAG is set.

This is a less likely situation but provides a larger window for the unlock event. A frozen Tx MB occurs if:

1. The Rx MB is locked by reading its C/S word before EOF6 of the first frame.
2. Both of these events occur in either a-then-b or b-then-a order:
   a) A new transmission is initiated by writing its C/S word sometime between CRC3 and EOF7 of the second frame.
   b) The Rx MB is unlocked between CRC3 and intermission at end of the second frame.

Notice in this situation that if the unlock occurs after EOF6, the first frame would be lost and the second frame would be moved to the Rx MB due to the delayed handling of received frames.

Situation C: Rx unlocked during bus idle.

A frozen/missing Tx occurs if:

1. An Rx MB is locked before EOF6 of an incoming frame with matching ID and remains locked at least until intermission. This situation would usually occur only if the received frame was serviced after reception of a second frame.
2. An internal arbitration period is triggered by writing a C/S field of an MB.
3. The locked Rx MB is unlocked within two internal arbitration periods (defined below) of step 2.
4. 0xC is written to the C/S field of a Tx MB within these same two arbitration periods. This step is optional if a 0xC was written in step 2 above.

Two internal arbitration periods are calculated as:

$$\frac{(2 \times \text{number of MBs}) + 16}{f_{IPS}} = t_{ARB} \qquad \textit{Eqn. 1}$$

The number of MBs can be reduced by writing to the FlexCAN MCR[MAXMB] field. The f_{IPS} clock frequency is used in this calculation regardless of the CTRL[CLK_SRC] setting.

Additional Notes:

- The received frames can be transmitted from the same node, but they must be received into an Rx MB.
- When the frozen Tx MB's IFLAG becomes set, an interrupt will occur if enabled.

- The timestamp of the missing Tx will be set to the same timestamp value as the last reception before it was frozen.
- If the user software locks the Rx MB before a frame is received, situation A can occur with a single received frame.
- The issue does not occur if there were any additional pending Tx MBs before CRC3.
- If multiple Tx MBs are initiated within the CRC3/EOF7 window (situation A and B) or two internal arbitration windows (situation C), they all become frozen.

Work-Around

If received frames can be handled (lock/unlocked) before EOF6 of the next frame, situations A and C are avoided. If they are handled before CRC3, or lock times are below 23 CAN bit times, situation B is avoided.

If these conditions cannot be guaranteed by the existing system design, situations A and B are avoided by inserting a delay of at least 28 CAN bit times between initiating a transmission and unlocking an Rx MB and vice versa. Typically a system will use a mechanism to selectively add the necessary delay. For example, software might use a global variable to record an external timer value (the FlexCAN timer can't be used, as that would unlock) when initiating a new Tx or unlocking an Rx, and then add the required delay before performing the second action.

Situation C can also be avoided by inserting a delay of at least two internal arbitration periods between writing 0xC and unlocking the locked Rx MB.

3.42 MUCts02511 — Debug Status Port Mode 2 Incorrect Signal Assignments in Documentation

Description

The signal assignments listed for Debug Status Port Mode 2 in Chapter 26, "System Services Module (SSM)," of the *MAC7100 Microcontroller Family Reference Manual* (MAC7100RM) v 1.0 are incorrect. The table below shows the incorrect and correct signal assignments.

THIS PAGE INTENTIONALLY LEFT BLANK

Table 2. Port F Debug Status Mode 2 Correct Signal Assignments

Port F Pin	Mode 2 Function	
	Incorrect	Correct
PF0	System is entering STOP mode	System is entering STOP mode
PF1	Platform has entered STOP mode	Platform has entered STOP mode
PF2	ATD A has entered STOP mode	ATD B has entered STOP mode
PF3	ATD B has entered STOP mode	ATD A has entered STOP mode
PF4	SCI A has entered STOP mode	PIT has entered STOP mode
PF5	SCI B has entered STOP mode	I^2C has entered STOP mode
PF6	SCI C has entered STOP mode	CAN D has entered STOP mode
PF7	SCI D has entered STOP mode	CAN C has entered STOP mode
PF8	CAN A has entered STOP mode	CAN B has entered STOP mode
PF9	CAN B has entered STOP mode	CAN A has entered STOP mode
PF10	CAN C has entered STOP mode	SPI B has entered STOP mode
PF11	CAN D has entered STOP mode	SPI A has entered STOP mode
PF12	PIT has entered STOP mode	SCI D has entered STOP mode
PF13	I^2C has entered STOP mode	SCI C has entered STOP mode
PF14	SPI A has entered STOP mode	SCI B has entered STOP mode
PF15	SPI B has entered STOP mode	SCI A has entered STOP mode

Work-Around

Use the correct signals as shown above.

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

MAC7101MSE1
Rev. 7
04/2005