Mask Set Errata for Mask 0N64Y

This report applies to mask 0N64Y for these products:

- MC56F83XXX
- MC56F83XXXA

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
ERR050307	ADC: ADC may malfunction in once sequential mode when sample 0~15 are disabled and sample 16 is configured as ADCB temperature sensor or ADCB analog input for on-chip generated signals
ERR050308	ADC: ADCB may malfunction when independent parallel mode is used and differential mode is enabled for any channel of ADCB
ERR050246	FlexCAN: Receive Message Buffers may have its Code Field corrupted if the Receive FIFO function is used
ERR050194	QTMR: Overflow flag and related interrupt cannot be generated when the timer is configured as upward count mode
ERR050309	ROM: Some registers are not restored to their reset values after ROM code execution.
ERR050274	USB: USB RAM reads immediately following writes may return incorrect data

Table 2. Revision History

Revision	Changes
0	Initial revision
1	Add MC56F83XXXA part. The following erratum was revised. • ERR050246
2	The following erratum was removed. • ERR050273

ERR050307: ADC: ADC may malfunction in once sequential mode when sample 0~15 are disabled and sample 16 is configured as ADCB temperature sensor or ADCB analog input for on-chip generated signals

Description: When once sequential mode is enable by setting CTRL1[SMODE] to 000b, sample 0~15 are disabled by setting SDIS to 0xFFFF, sample 16 is enabled in SDIS2 register, and CLIST5[SAMPLE16] is set to 10b or 11b, ADC may malfunction with no data converted. The ADC state machine may halt in this case.

Workaround: Set CLIST5[SAMPLE16] to 00b or 01b in this case.

ERR050308: ADC: ADCB may malfunction when independent parallel mode is used and differential mode is enabled for any channel of ADCB

Description: ADCB may malfunction with wrong conversion results when two conditions are met: 1. CTRL1[SMODE] is set to once parallel (001b), loop parallel (011b) or triggered parallel (101b) mode, and CTRL2[SIMULT] is set to zero, which means independent mode; 2. There is differential pair enabled among ANB0~ANB7 by setting bit3 or bit2 of CTRL1[CHNCFG_L] or CTRL2[CHNCFG_H].

Workaround: There are two methods to avoid this issue in independent parallel mode: 1). Make sure all the inputs of ADCB are configured as single ended inputs. 2). Make sure the following three conditions are met: a). Enable sample 0 of ADCA by clearing bit 0 of SDIS register. b). Disable the additional on-chip sample slots by setting 0xF to SDIS2. c). Enable full differential mode for sample 8~15 by setting bit2&3 of CTRL1[CHNCFG_L] and CTRL2[CHNCFG_H] while keeping bit2&3 of CTRL3[UPDEN_H] and CTRL3[UPDEN_L] cleared. Or enable unipolar differential mode for sample 8~15 by setting bit2&3 of CTRL1[CHNCFG_L] and CTRL2[CHNCFG_H] while keeping bit2&3 of CTRL3[UPDEN_H] and CTRL3[UPDEN_L] set.

ERR050246: FlexCAN: Receive Message Buffers may have its Code Field corrupted if the Receive FIFO function is used

Description: If the Code Field of a Receive Message Buffer is corrupted it may deactivate the Message Buffer, so it is unable to receive new messages. It may also turn a Receive Message Buffer into any type of Message Buffer as defined in the Message buffer structure section in the device documentation.

The Code Field of the FlexCAN Receive Message Buffers (MB) may get corrupted if the following sequence occurs.

1- A message is received and transferred to an MB (i.e. MBx)

2- MBx is locked by software for more than 20 CAN bit times (time determines the probability of erratum to manifest).

3- SMB0 (Serial Message Buffer 0) receives a message (i.e. message1) intended for MBx, but destination is locked by the software (as depicted in point 2 above) and therefore NOT transferred to MBx.

4- A subsequent incoming message (i.e. message2) is being loaded into SMB1 (as SMB0 is full) and is evaluated by the FlexCAN hardware as being for the FIFO.

5-During the message2, the MBx is unlocked. Then, the content of SMB0 is transferred to MBx and the CODE field is updated with an incorrect value.

In case a customer does use Rx FIFO only or dedicated MB only, (i.e. either Rx MB or Rx FIFO is used) the problem doesn't occur. In case a customer does use the Enhanced Rx FIFO and dedicated MB at the same application, the problem does not occur also. So bottom line the problem does only occur if the FlexCAN is programmed to receive in the Legacy FIFO and dedicated MB at the same application.

Workaround: This defect only applies if the Receive FIFO (Legacy Rx FIFO) is used. This feature is enabled by RFEN bit in the Module Control Register (MCR). If the Rx FIFO is not used, the Receive Message Buffer Code Field is not corrupted.

If available on the device, use the enhanced Rx FIFO feature instead of the Legacy Rx FIFO. The Enhanced Rx FIFO is enabled by the ERFEN bit in the Enhanced Rx FIFO Control Register (ERFCR).

The defect does not occur if the Receive Message Buffer lock time is less than or equal to the time equivalent to 20 x CAN bit time.

The recommended way for the CPU to service (read) the frame received in a mailbox is by the following procedure:

1. Read the Control and Status word of that mailbox.

2. Check if the BUSY bit is deasserted, indicating that the mailbox is not locked. Repeat step 1) while it is asserted.

3. Read the contents of the mailbox.

4. Clear the proper flag in the IFLAG register.

5. Read the Free Running Timer register (TIMER) to unlock the mailbox

In order to guarantee that this procedure occurs in less than 20 CAN bit times the MB receive handling process in software (step 1 to step 5 above) should be performed as a 'critical code section' (interrupts disabled before execution) and should ensure that the MB receive handling occurs in a deterministic number of cycles.

ERR050194: QTMR: Overflow flag and related interrupt cannot be generated when the timer is configured as upward count mode

Description: 1.Overflow flag and related interrupt cannot be generated successfully in upward count mode.

2.When TMR_CTRL[OUTMODE] is set to 110b, OFLAG output is not cleared on counter rollover when the timer counts upward.

Workaround: For item 1, using compare interrupt instead of overflow interrupt by setting compare value to 0xFFFF. The compare interrupt has the same timing effect as overflow interrupt in this way.

For item 2, there is no workaround.

ERR050309: ROM: Some registers are not restored to their reset values after ROM code execution.

Description: After ROM code execution, the program jumps to user's application, but some registers used in ROM are not restored to their reset values. These registers are TMRA_CSCTRL0, TMRA_CSCTRL1, CPU status register SR, I2C0_A1, I2C0_F, GPIOC_IENR, GPIOC_IPOLR and OCCS_CTRL.

Workaround: Set 0x0000 to TMRA_CSCTRL0, TMRA_CSCTRL1, I2C0_A1, I2C0_F, GPIOC_IENR and GPIOC_IPOLR, set 0x0300 to SR, set 0x0010 to OCCS_CTRL manually at the beginning of user's application. Add 6 NOPs after setting 0x0010 to OCCS_CTRL.

ERR050274: USB: USB RAM reads immediately following writes may return incorrect data

Description: When reading a USB RAM location immediately after a write to that location, incorrect results may be returned. This only occurs when 8-bit access to the USB RAM is performed.

Workaround: The workaround is to avoid using the USB RAM.

If the USB RAM has to be adopted, users must ensure one of the following prerequisites:

1) Do not perform 8-bit access to the USB RAM.

2) Insert a no-operation ("NOP") instruction (or other similar operation) between write to the location and read to the location. Note that it does not need to be (not recommended at all) coded at the assembly level, while it should be done at higher programming language levels (such as C, or C++).

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support