SPC560B54x, SPC560B6x
Errata sheet

32-bit MCU family built on the Power Architecture® for automotive body electronics applications

Introduction

This errata sheet describes all the known functional and electrical limitations of the SPC560B54/6x microcontroller family.

All the topics covered in this document refer to *RM0037* Rev 9 and *SPC560B54x, SPC560B60x, SPC560B64x datasheet* Rev 8 (see *A.1: Reference document*).

The device identification can be read by software using the MIDR1 register:

Die mask ID: FB64X2

- MAJOR_MASK[3:0]: 1
- MINOR_MASK[3:0]: 4

This errata sheet applies to SPC560B54/6x devices in accordance with *Table 1*.

Table 1. Device summary

Package	Part number		
	768 KB Code Flash	1 MB Code Flash	1.5 MB Code Flash
LQFP176	—	SPC560B60L7	SPC560B64L7
LQFP144	SPC560B54L5	SPC560B60L5	SPC560B64L5
LQFP100	SPC560B54L3	SPC560B60L3	—

Contents

1 Functional problems

1.1 ERR002656: FlexCAN: Abort request blocks the CODE field

Description:

An Abort request to a transmit Message Buffer (TxMB) can block any write operation into its CODE field. Therefore, the TxMB cannot be aborted or deactivated until it completes a valid transmission (by winning the CAN bus arbitration and transmitting the contents of the TxMB).

Workaround:

Instead of aborting the transmission, use deactivation instead.

Note: There is a chance that the deactivated TxMB can be transmitted without setting IFLAG and updating the CODE field if it is deactivated.

1.2 ERR002883: FMPLL: FMPLL_CR[UNLOCK_ONCE] wrongly set

Description:

If the FMPLL is locked and a functional reset occurs, FMPLL_CR[UNLOCK_ONCE] is automatically set even when the FMPLL has not lost lock.

Workaround:

Do not use the FMPLL_CR[UNLOCK_ONCE] when a functional reset occurs.

1.3 ERR002958: MC_RGM: Clearing a flag at RGM_DES or RGM_FES register may be prevented by a reset

Description:

Clearing a flag at RGM_DES and RGM_FES registers requires two clock cycles because of a synchronization mechanism. As a consequence if a reset occurs while clearing is on-going the reset may interrupt the clearing mechanism leaving the flag set.

Note: This failed clearing has no impact on further flag clearing requests.

Workaround:

No workaround for all reset sources except SW reset.

Note: In case the application requests a SW reset immediately after clearing a flag in RGM_xES the same issue may occur. To avoid this effect the application must ensure that flag clearing has completed by reading the RGM_xES register before the SW reset is requested.

1.4 ERR002977: MC_RGM: Long Reset Sequence Occurs on 'Short Functional' Reset Event

Description:

If a 'functional' reset source is configured to initiate a short reset sequence via setting of the appropriate RGM_FESS bit and also configured to assert the external reset pad via setting of the appropriate RGM_FBRE bit, its assertion will not initiate a short reset starting with PHASE3 but rather a long reset sequence starting with PHASE1.

Workaround:

Do not configure 'functional' resets that are configured to initiate a short reset sequence to also assert the external reset pad. In other words, if RGM_FESS[i] is '1', RGM_FBRE[i] should be '0'.

1.5 ERR003021: LINFlex: Unexpected LIN timeout in slave mode

Description:

If the LINFlex is configured in LIN slave mode, an unexpected LIN timeout event (LINESR[OCF]) may occur during LIN Break reception.

Workaround:

It is recommended to disable this functionality during LINFlex initialization by clearing LINTCSR[IOT] and LINIER[OCIE] bits, and ignore timeout events.

1.6 ERR003049: MC_RGM: External Reset not asserted if Short Reset enabled

Description:

For the case when the External Reset is enabled for a specific reset source at RGM_FBRE and a short reset is requested for the same reset source at RGM_FESS the External Reset is not asserted.

Workaround:

None.

1.7 ERR003060: MC_RGM: SAFE mode exit may be possible even though condition causing the SAFE mode request has not been cleared

Description:

A SAFE mode exit should not be possible as long as any condition that caused a SAFE mode entry is still active. However, if the corresponding status flag in the RGM_FES register has been cleared, the SAFE mode exit may incorrectly occur even though the actual condition is still active.

Workaround:

Software must clear the SAFE mode request condition at the source before clearing the corresponding RGM_FES flag. This will ensure that the condition is no longer active when the RGM_FES flag is cleared and thus the SAFE mode exit can occur under the correct conditions.

1.8 ERR003080: ADC: CTUEN bit in ADC.MCR register cannot be reset if a BCTU channel is enabled

Description:

While any BCTU channels is enabled (CTU.EVTCFGRx.TM =1), the CTU will continuously send trigger requests to ADC. If CTUEN bit in MCR is reset while BCTU channels are enabled, the ADC DTU trigger state may become undefined and ADC module may not service trigger request from CTU anymore.

Workaround:

Ensure ADC.MCR.CTUEN is set before enabling any CTU channels (CTU.EVTCFGRx.TM =1). Ensure all CTU channels are disabled (CTU.EVTCFGRx.TM =0) before ADC.MCR.CTUEN is cleared.

1.9 ERR003119: SWT: SWT interrupt does not cause STOP0 mode exit

Description:

While in STOP0 mode, if the SWT is configured to generate an interrupt and the system clock is disabled (ME_STOP0_MC[SYSCLK] = 0xF), a SWT interrupt event will not trigger an exit from STOP0 mode.

Other internal or external wakeup events (RTC, API, WKUP pins) are not affected and will trigger a STOP0 exit independent of the ME_STOP0_MC[SYSCLK] configuration.

Workaround:

If a SWT interrupt is to be used to wake the device during STOP0 mode, software may not disable the system clock (ME_STOP0_MC[SYSCLK] != 0xF).

1.10 ERR003190: MC_ME: Main VREG not disabled during STOP0 or HALT0 mode if RUN[0...3] mode selects FXOSC to be running and target mode selects FXOSC as system clock

Description:

If STOP0 or HALT0 is configured with ME_[mode]_MC.MVRON = '0', ME_[mode]_MC.FIRCON = '0' and ME_[mode]_MC.SYSCLK = '0010/0011' the Main VREG will nevertheless remain enabled during the STOP0 mode if the previous RUN[0...3] mode is configured with ME_RUN[0...3]_MC.FXOSCON = '1'. This will result in increased current consumption of 500 µA than expected.

Workaround:

Before entering STOP0 or HALT0 mode with the following configuration:

- ME_[mode]_MC.MVRON = '0', ME_[mode]_MC.FIRCON = '0' and ME_[mode]_MC.SYSCLK = '0010/0011'
- Ensure the RUN[0...3] mode switches off FXOSC
- ME_RUN[0...3]_MC.FXOSCON = '0' before attempting to low power mode transition

1.11 ERR003195: LINFlex: Limitations for DMA access to LINFlex

Description:

The DMA handshaking to the LINFlex can fail when the LINFlex operates on a divided peripheral clock.

Workaround:

Don't divide the LINFlex peripheral clock if DMA access is required.

1.12 ERR003202: MC_ME: Invalid Configuration not flagged if PLL is on while OSC is off

Description:

PLL clock generation requires oscillator to be on. Mode configuration in which PLLON bit is "1" and OSCON bit is "0" is an invalid mode configuration. When ME_XXX_MC registers are attempted with such an invalid configuration, ME_IS.I_ICONF is not getting set which is wrong. Eventually the mode transition did not complete and system hangs.

Workaround:

Always program Oscillator to be on when PLL is required.

1.13 ERR003209: NMI pin configuration limitation in standby mode

Description:

NMI pin cannot be configured to generate Non Maskable Interrupt event to the core (WKPU_NCR[NDSS] = "00") if the following standby mode is to be used:

- NMI pin enabled for wake-up event,
- Standby exit sequence boot from RAM,
- Code flash module power-down on standby exit sequence.

With following configuration following scenario may happen:

1. System is in standby
2. NMI event is triggered on PA[1]
3. System wakeup z0 core power domain.
4. z0 core reset is released and NMI event is sampled by core on first clock-edge.
5. z0 core attempt to fetch code at 0x10 address (IVPR is not yet initialized by application) and receive an exception since flash is not available
6. z0 core enter machine check and execution is stalled.

Workaround:

If NMI is configured as wake-up source, WKPU_NCR[NDSS] must be configured as "11". This will ensure no NMI event is triggered on the core but ensure system wakeup is triggered.

After standby exit, core will boot and configure its IVOR/IVPR, it may then re-configure WKPU_NCR:DSS to the appropriate configuration for enabling NMI/CI/MCP.

1.14 ERR003210: PA[1] pull-up enabled when NMI activated

Description:

When NMI is enabled (either WKPU_NCR[NREE] or WKPU_NCR[NFEE] set to '1'), PA[1] pull-up is automatically activated independently from SIUL_PCR1[WPS] and SIUL_PCR1[WPE].

This has no effect during STANDBY mode. PA[1] pull-up is then correctly configured through WKPU_WIPUER[IPUE[2]].

Workaround:

None.

1.15 ERR003219: MC_CGM: System clock may stop for case when target clock source stops during clock switching transition

Description:

The clock switching is a two step process. The availability of the target clock is first verified. Then the system clock is switched to the new target clock source within two target clock cycles.

For the case when the FXOSC stops during the required two cycles, the switching process may not complete, causing the system clock to stop and prevent further clock switching. This may happen if one of the following cases occurs while the system clock source is switching to FXOSC:

- FXOSC oscillator failure
- SAFE mode request occurs, as this mode will immediately switch OFF the FXOSC (refer to ME_SAFE_MC register configuration)

Workaround:

The device is able to recover through any reset event ('functional', 'destructive', internal or external), so typically either the SWT (internal watchdog) will generate a reset or, in case it is used in the application, the external watchdog will generate an external reset. In all cases the devices will restart properly after reset.

To reduce the probability that this issue occurs in the application, disable SAFE mode transitions when the device is executing a mode transition with the FXOSC as the system clock source in the target mode.

1.16 ERR003242: PB[10],PD[0:1] pins configuration during standby

Description:

PB[10], PD[0:1] are the pins having both wake-up functionality and analog functionality.

As for all wake-up pins, they must be driven either high level or low level (possibly using the internal pull- up) during standby.

In case the pins are connected to external component providing analog signal, it is important to check that this external analog signal is either lower than 0.2*VDD_HV or higher than 0.8*VDD_HV not to incur extra consumption.

Workaround:

None.

1.17 ERR003247: MC_ME: A mode transition will not complete if the FlexCAN is disabled for target mode at MC_ME and is enabled at the FlexCAN peripheral

Description:

If a FlexCAN module is enabled for the current mode at MC_ME using the ME_RUN_PCx/ME_PCTLx registers and also enabled at the FlexCAN Module Configuration Register, for the case when the target mode (run or low power) disables the FlexCAN module, this transition will only complete if the FlexCAN is disabled at the FlexCAN peripheral prior to the target mode transition.

Workaround:

Before initiating the target mode change at the MC_ME the FlexCAN Module Configuration Register should be configured to set Freeze Enable, Halt and Module Disable (FLEXCAN_MCR) i.e.

FLEXCAN_MCR[FRZ] = FLEXCAN_MCR[HALT] = FLEXCAN_MCR[MDIS] = 1.

1.18 ERR003270: Risk of increased Analog to Digital converter pad leakage under extreme current injection conditions

Description:

Post AEC latch-up stressing (100 mA, 125 °C), there is a risk of leakage drift on the following pads:

176LQFP:

PA(3)/Physical pin 114

PA(7)/Physical pin 128

PA(10)/Physical pin 131

PA(11)/Physical pin 132

PE(12)/Physical pin 133

144LQFP:

PA(3)/Physical pin 90

PA(7)/Physical pin 104

PA(10)/Physical pin 107

PA(11)/Physical pin 108

PE(12)/Physical pin 109

100LQFP:

PA(3)/Physical pin 68

PA(7)/Physical pin 71

PA(10)/Physical pin 74

PA(11)/Physical pin 75

PE(12)/Physical pin 76

Increased leakage on these pads has been observed under AEC latch-up conditions, which are significantly above the +/- 5 mA current injection spec. Leakage drift observed ranges from low nA to <10 µA, depending on latch-up pulse width and wafer characteristics.

This leakage results from Current Injection stress, rather than a latch-up event. No increased leakage has been observed under spec current injection conditions.

Workaround:

Adhere to current injection spec of +/- 5 mA per pin. If performing AEC latch-up trials, be aware of risk of increased leakage on these pins.

1.19 ERR003407: FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1

Description:

FlexCAN does not transmit an expected message when the same node detects an incoming Remote Request message asking for any remote answer.

The issue happens when two specific conditions occur:

1. The Message Buffer (MB) configured for remote answer (with code "a") is the last MB. The last MB is specified by Maximum MB field in the Module Configuration Register (MCR[MAXMB]).
2. The incoming Remote Request message does not match its ID against the last MB ID.

While an incoming Remote Request message is being received, the FlexCAN also scans the transmit (Tx) MBs to select the one with the higher priority for the next bus arbitration. It is expected that by the Intermission field it ends up with a selected candidate (winner). The coincidence of conditions (1) and (2) above creates an internal corner case that cancels the Tx winner and therefore no message will be selected for transmission in the next frame.

This gives the appearance that the FlexCAN transmitter is stalled or "stops transmitting".

The problem can be detectable only if the message traffic ceases and the CAN bus enters into Idle state after the described sequence of events.

There is NO ISSUE if any of the conditions below holds:

a) The incoming message matches the remote answer MB with code "a".

b) The MB configured as remote answer with code "a" is not the last one.

c) Any MB (despite of being Tx or Rx) is reconfigured (by writing its CS field) just after the Intermission field.

d) A new incoming message sent by any external node starts just after the Intermission field.

Workaround:

Do not configure the last MB as a Remote Answer (with code "a").

1.20 ERR003442: CMU monitor: FXOSC/FIRC and FMPLL/FIRC relation

Description:

Functional CMU monitoring can only be guaranteed when the following conditions are met:

- FXOSC frequency must be greater than (FIRC / 2^RCDIV) + 0.5 MHz in order to guarantee correct FXOSC monitoring
- FMPLL frequency must be greater than (FIRC / 4) + 0.5 MHz in order to guarantee correct FMPLL monitoring

Workaround:

Refer to description.

1.21 ERR003446: CTU: The CTU (Cross Trigger Unit) CLR_FLAG in EVTCFGR register does not function as expected

Description:

If the CTU CLR_FLG is set and the CTU is idle, a PIT triggered request to the CTU does not result in the correct ADC channel number being latched. The previous ADC channel number is latched instead of the requested channel number.

Workaround:

There is no software workaround to allow the CLR_FLAG functionality to operate correctly. Do not program the CLR_FLAG bit to '1'.

1.22 ERR003449: DEBUG: Device may hang due to external or 'functional' reset while using debug handshaking mechanism

Description:

If the low-power mode debug handshake has been enabled and an external reset or a 'functional' reset occurs while the device is in a low-power mode, the device will not exit reset.

Workaround:

The NPC_PCR[LP_DBG_EN] bit must be cleared to ensure the correct reset sequence.

1.23 ERR003466: LINFlexD: Register bus aborts are not generated on illegal accesses to reserved addresses within the register address space of LINFlexD

Description:

Register bus aborts are not generated on illegal accesses to reserved addresses within the register address space of LINFlexD. This is applicable to LINFlex modules supporting master-only mode.

Workaround:

None.

1.24 ERR003512: ECSM: ECSM_PFEDR displays incorrect endianness

Description:

The ECSM_PFEDR register reports ECC data using incorrect endiannes. For example, a flash location that contains the data 0xAABBCCDD would be reported as 0xDDCCBBAA at ECSM_PFEDR.

This 32-bit register contains the data associated with the faulting access of the last, properly-enabled flash ECC event. The register contains the data value taken directly from the data bus.

Workaround:

Software must correct endianness.

1.25 ERR003556: DMA_MUX: Low Power Entry may not be completed when peripherals run on divided clock with DMA enabled mode

Description:

System may not enter into Low Power Mode (HALT/STOP/STANDBY) when all the below conditions are true simultaneously:

1. A Peripheral with DMA capability is programmed to work on divided clock.
2. Above peripheral is programmed to be stopped in Low Power Mode and active in RUN Mode.
3. Above Peripheral is active with DMA transfer while Software requests change to Low Power mode.

Workaround:

Software should ensure that all the DMA enabled peripherals have completed their transfer before requesting Low Power mode Entry

1.26 ERR003570: MC_ME: Possibility of Machine Check on Low-Power Mode Exit

Description:

When executing from the flash and entering a Low-Power Mode (LPM) where the flash is in low- power or power-down mode, 2-4 clock cycles exist at the beginning of the RUNx to LPM transition during which a wakeup or interrupt will generate a checkstop due to the flash not being available on RUNx mode re-entry. This will cause either a checkstop reset or machine check interrupt.

Workaround:

If the application must avoid the reset, two workarounds are suggested:

1. Configure the application to handle the machine check interrupt in RAM dealing with the problem only if it occurs
2. Configure the MCU to avoid the machine check interrupt, executing the transition into low power modes in RAM.

There is no absolute requirement to work around the possibility of a checkstop reset if the application can accept the reset, and associated delays, and continue. In this event, the WKPU.WISR will not indicate the channel that triggered the wakeup though the F_CHKSTOP flag will indicate that the reset has occurred. The F_CHKSTOP flag could still be caused by other error conditions so the startup strategy from this condition should be considered alongside any pre-existing strategy for recovering from an F_CHKSTOP condition.

1.27 ERR004146: When an ADC conversion is injected, the aborted channel is not restored under certain conditions

Description:

When triggered conversions interrupt the ADC, it is possible that the aborted conversion does not get restored to the ADC and is not converted during the chain. Vulnerable configurations are:

- Injected chain over a normal chain
- CTU trigger over a normal chain
- CTU trigger over an injected chain

When any of these triggers arrive whilst the ADC is in the conversion stage of the sample and conversion, the sample is discarded and is not restored. This means that the channel data register will not show the channel as being valid and the CEOCFRx field will not indicate a pending conversion. The sample that was aborted is lost.

When the trigger arrives during the final channel in a normal or injected chain, the same failure mode can cause two ECH/JECH interrupts to be raised.

If the trigger arrives during the sampling phase of the last channel in the chain, an ECH is triggered immediately, the trigger is processed and the channel is restored and after sampling/conversion, a second ECH interrupt occurs.

In scan mode, the second ECH does not occur if the trigger arrives during the conversion phase. In one-shot mode, the trigger arriving during the conversion phase of the last

channel restarts the whole conversion chain and the next ECH occurs at completion of that chain.

Workaround:

It is suggested that the application check for valid data using the CDR status bits or the CEOCFRx registers to ensure all expected channels have converted. This can be tested by running a bitwise AND and an XOR with either the JCMRx or NCMRx registers and the CEOCFRx registers during the ECH of JECH handler. Any non-zero value for (xCMRx & (xCMRx CEOCFRx)) indicates that a channel has been missed and conversion should be requested again.

Spurious ECH/JECH interrupts can be detected by checking the NSTART/JSTART flags in the ADC Module Status Registers – if the flag remains set during an ECH/JECH interrupt then another interrupt will follow after the restored channel or chain has been sampled and converted.

The spurious ECH/JECH workaround above applies to single-shot conversions. In single-shot mode, NSTART changes from 1 to 0. Therefore, the user can rely on checking the NSTART bit to confirm if a spurious ECH has occurred. However, for scan mode, the NSTART bit will remain set during normal operation, so it cannot be relied upon to check for the spurious ECH/JECH issue.

Consequently, if CTU is being used in trigger mode, the conversions must be single-shot and not scan mode.

1.28 ERR004168: ADC: Abort switch aborts the ongoing injected channel as well as the upcoming normal channel

Description:

If an Injected chain (jch1,jch2,jch3) is injected over a Normal chain (nch1,nch2,nch3,nch4) the Abort switch does not behave as expected.

Expected behavior:

Correct Case (without SW Abort on jch3):
Nch1- Nch2(aborted) -Jch1 - Jch2 - Jch3 - Nch2(restored)- Nch3 - Nch4

Correct Case (with SW Abort on jch3):
Nch1 - Nch2(aborted) -Jch1 - Jch2 - Jch3(aborted) - Nch2(restored) - Nch3 - Nch4

Observed unexpected behavior:

Fault1 (without SW abort on jch3):
Nch1 - Nch2(aborted) - Jch1 - Jch2 - Jch3 - Nch3 - Nch4 (Nch2 not restored)

Fault2 (with SW abort on jch3):
Nch1- Nch2 (aborted) - Jch1 - Jch2 - Jch3(aborted) - Nch4 (Nch2 not restored & Nch3 conversion skipped)

Workaround:

It is possible to detect the unexpected behavior by using the CEOCFRx register. The CEOCFRx fields will not be set for a not restored or skipped channel, which indicates this issue has occurred. The CEOCFRx fields need to be checked before the next Normal chain execution (in scan mode). The CEOCFRx fields should be read by every ECH interrupt at the end of every chain execution.

1.29 ERR004186: ADC: Do not trigger ABORT or ABORTCHAIN prior to the start of CTU triggered ADC conversions and do not trigger ABORTCHAIN prior to the start of INJECTED triggered ADC conversions.

Description:

When ADC_MCR[ABORT] or ADC_MCR[ABORTCHAIN] is set prior to the ADC receiving a CTU trigger, the next CTU triggered ADC conversion will not be performed and further CTU triggered ADC conversions will be blocked.

When ADC_MCR[ABORTCHAIN] is set prior to the ADC receiving an INJECTED trigger, the next INJECTED ADC conversion will not be performed. Following the ABORTCHAIN command the MCU behaviour does not meet the specification as ADC_ISR[JECH] is not set and ADC_MCR[ABORTCHAIN] is not cleared.

Workaround:

Do not program ADC_MCR[ABORT] or ADC_MCR[ABORTCHAIN] before the start of ADC conversions.

The case when CTU triggered ADC conversions are blocked should be avoided however it is possible to reactivate CTU conversions by clearing and setting ADC_MCR[CTUEN].

1.30 ERR004340: LINFlexD: Buffer overrun can not be detected in UART Rx FIFO mode

Description:

When the LINFlexD is configured in UART Receive (Rx) FIFO mode, the Buffer Overrun Flag (BOF) bit of the UART Mode Status Register (UARTSR) register is cleared in the subsequent clock cycle after being asserted.

User software can not poll the BOF to detect an overflow.

The LINFlexD Error Combined Interrupt can still be triggered by the buffer overrun. This interrupt is enabled by setting the Buffer Overrun Error Interrupt Enable (BOIE) bit in the LIN Interrupt Enable Register (LINIER). However, the BOF bit will be cleared when the interrupt routine is entered, preventing the user from identifying the source of error.

Workaround:

Buffer overrun errors in UART FIFO mode can be detected by enabling only the Buffer Overrun Interrupt Enable (BOIE) in the LIN Interrupt Enable Register (LINIER).

1.31 ERR004405: SR bit of LINFlexD GCR register is not cleared automatically by hardware

Description:

After setting the SR bit of GCR (Global Control Register) to reset the LINFlexD controller, this bit is not cleared automatically by the hardware, keeping the peripheral in reset state

Workaround:

This bit should be cleared by software to perform further operations

1.32 ERR005569: ADC: The channel sequence order will be corrupted when a new normal conversion chain is started prior to completion of a pending normal conversion chain

Description:

If One shot mode is configured in the Main Configuration Register (MCR[MODE] = 0) the chained channels are automatically enabled in the Normal Conversion Mask Register 0 (NCMR0). If the programmer initiates a new chain normal conversion, by setting MCR[NSTART] = 0x1, before the previous chain conversion finishes, the new chained normal conversion will not follow the requested sequence of converted channels.

For example, if a chained normal conversion sequence includes three channels in following sequence: channel0, channel1 and channel2, the conversion sequence is started by MCR[NSTART] = 0x1. The software re-starts the next conversion sequence when MCR[NSTART] is set to 0x1 just before the current conversion sequence finishes.

The conversion sequence should be: channel0, channel1, channel2, channel0, channel1, channel2.

However, the conversion sequence observed will be:

channel0, channel1, channel2, channel1, channel1, channel2.

Channel0 is replaced by channel1 in the second chain conversion and channel1 is converted twice.

Workaround:

Ensure a new conversion sequence is not started when a current conversion is ongoing. This can be ensured by issuing the new conversion setting MCR[NSTART] only when MSR[NSTART] = 0.

Note: MSR[NSTART] indicates the present status of conversion. MSR[NSTART] = 1 means that a conversion is ongoing and MSR[NSTART] = 0 means that the previous conversion is finished.

1.33 ERR006026: DSPI: Incorrect SPI Frame Generated in Combined Serial Interface Configuration

Description:

In the Combined Serial Interface (CSI) configuration of the Deserial Serial Peripheral Interface (DSPI) where data frames are periodically being sent (Deserial Serial Interface, DSI), a Serial Peripheral Interface (SPI) frame may be transmitted with incorrect framing.

The incorrect frame may occur in this configuration if the user application writes SPI data to the DSPI Push TX FIFO Register (DSPI_PUSHR) during the last two peripheral clock cycles of the Delay-after- Transfer (DT) phase. In this case, the SPI frame is corrupted.

Workaround:

Workaround 1: Perform SPI FIFO writes after halting the DSPI.

To prevent writing to the FIFO during the last two clock cycles of DT, perform the following steps every time a SPI frame is required to be transmitted:

Step 1: Halt the DSPI by setting the HALT control bit in the Module Configuration Register (DSPI_MCR[HALT]).

Step 2: Poll the Status Register's Transmit and Receive Status bit (DSPI_SR[TXRXS]) to ensure the DSPI has entered the HALT state and completed any in-progress transmission. Alternatively, if continuous polling is undesirable in the application, wait for a fixed time interval such as 35 baud clocks to ensure completion of any in-progress transmission and then check once for DSPI_SR[TXRXS].

Step 3: Perform the write to DSPI_PUSHR for the SPI frame.

Step 4: Clear bit DSPI_MCR[HALT] to bring the DSPI out of the HALT state and return to normal operation.

Workaround 2: Do not use the CSI configuration. Use the DSPI in either DSI-only mode or SPI-only mode.

Workaround 3: Use the DSPI's Transfer Complete Flag (TCF) interrupt to reduce worst-case wait time of Workaround 1.

Step 1: When a SPI frame is required to be sent, halt the DSPI as in Step 1 of Workaround 1 above. Step 2: Enable the TCF interrupt by setting the DSPI DMA/Interrupt Request Select and Enable Register's Transmission Complete Request Enable bit (DSPI_RSER[TCF_RE])

Step 3: In the TCF interrupt service routine, clear the interrupt status (DSPI_SR[TCF]) and the interrupt request enable (DSPI_RSER[TCF_RE]). Confirm that DSPI is halted by checking DSPI_SR[TXRXS] and then write data to DSPI_PUSHR for the SPI frame. Finally, clear bit DSPI_MCR[HALT] to bring the DSPI out of the HALT state and return to normal operation.

1.34 ERR006082: LINFlexD: LINS bits in LIN Status Register(LINSR) are not usable in UART mode.

Description:

When the LINFlexD module is used in the Universal Asynchronous Receiver/Transmitter (UART) mode, the LIN state bits (LINS3:0]) in LIN Status Register (LINSR) always indicate the value zero. Therefore, these bits cannot be used to monitor the UART state.

Workaround:

LINS bits should be used only in LIN mode.

1.35 ERR006620: FLASH: ECC error reporting is disabled for Address Pipelining Control (APC) field greater than Read Wait-State Control (RWSC) field.

Description:

The reference manual states the following at the Platform flash memory controller Access pipelining functional description.

“The platform flash memory controller does not support access pipelining since this capability is not supported by the flash memory array. As a result, the APC (Address Pipelining Control) field should typically be the same value as the RWSC (Read Wait-State Control) field for best performance, that is, BKn_APC = BKn_RWSC. It cannot be less than the RWSC.”

The reference manual advises that the user must not configure APC to be less than RWSC and typically APC should equal RWSC. However the documentation does not prohibit the configuration of APC greater than RWSC and for this configuration ECC error reporting will be disabled. Flash ECC error reporting will only be enabled for APC = RWSC.

For the case when flash ECC is disabled and data is read from a corrupt location the data will be transferred via the system bus however a bus error will not be asserted and neither a core exception nor an ECSM interrupt will be triggered. For the case of a single-bit ECC error the data will be corrected but for a double-bit error the data will be corrupt.

Note: Both CFlash & DFlash are affected by this issue.

For single-bit and double-bit Flash errors neither a core exception nor an ECSM interrupt will be triggered unless APC=RWSC.

The Flash Array Integrity Check feature is not affected by this issue and will successfully detect an ECC error for all configurations of APC >= RWSC.

For the APC > RWSC configuration other than flash ECC error reporting there will be no other unpredictable behaviour from the flash.

The write wait-state control setting at PFCRx[BKn_WWSC] has no affect on the flash. It is recommended to set WWSC = RWSC = APC.

Workaround:

PFCRx[BKy_APC] must equal PFCRx PFCRx[BKy_RWSC]. See datasheet for correct setting of RWSC.

1.36 ERR006726: NPC: MCKO clock may be gated one clock period early when MCKO frequency is programmed as SYS_CLK/8.and gating is enabled

Description:

The Nexus auxiliary message clock (MCKO) may be gated one clock period early when the MCKO frequency is programmed as SYS_CLK/8 in the Nexus Port Controller Port Configuration Register (NPC_PCR[MCKO_DIV]=111) and the MCKO gating function is enabled (NPC_PCR[MCKO_GT]=1). In this case, the last MCKO received by the tool prior to the gating will correspond to the END_MESSAGE state. The tool will not receive an MCKO to indicate the transition to the IDLE state, even though the NPC will transitate to the IDLE state internally. Upon re-enabling of MCKO, the first MCKO edge will drive the Message Start/End Output (MSEO=11) and move the tool's state to IDLE.

Workaround:

Expect to receive the MCKO edge corresponding to the IDLE state upon re-enabling of MCKO after MCKO has been gated.

1.37 ERR006967: eDMA: Possible misbehavior of a preempted channel when using continuous link mode

Description:

When using Direct Memory Access (DMA) continuous link mode Control Register Continuous Link Mode (DMA_CR[CLM]) = 1) with a high priority channel linking to itself, if the high priority channel preempts a lower priority channel on the cycle before its last read/write sequence, the counters for the preempted channel (the lower priority channel) are corrupted. When the preempted channel is restored, it continues to transfer data past its "done" point (that is the byte transfer counter wraps past zero and it transfers more data than indicated by the byte transfer count (NBYTES)) instead of performing a single read/write sequence and retiring.

The preempting channel (the higher priority channel) will execute as expected.

Workaround:

Disable continuous link mode (DMA_CR[CLM]=0) if a high priority channel is using minor loop channel linking to itself and preemption is enabled. The second activation of the preempting channel will experience the normal startup latency (one read/write sequence + startup) instead of the shortened latency (startup only) provided by continuous link mode.

1.38 ERR006976: MC_ME: SAFE mode not entered immediately on hardware-triggered SAFE mode request during STOP0 mode

Description:

If a SAFE mode request is generated by the Reset Generation Module (MC_RGM) while the chip is in STOP0 mode, the chip does not immediately enter SAFE mode if STOP0 is configured as follows in the STOP0 Mode Configuration register (ME_STOP0_MC):

- The system clock is disabled (ME_STOP0_MC[SYSCLK] = 0b1111)
- The internal RC oscillator is enabled (ME_STOP0_MC[IRCON] = 0b1)

In this case, the chip will remain in STOP0 mode until an interrupt request or wakeup event occurs, causing the chip to return to its previous RUNx mode, after which the still pending SAFE mode request will cause the chip to enter SAFE mode.

Workaround:

There are two possibilities.

1. Configure the internal RC oscillator to be disabled during STOP0 mode (ME_STOP0_MC[IRCON] = 0b0) if the device supports it.
2. Prior to entering STOP0 mode, configure all hardware-triggered SAFE mode requests that need to cause an immediate transition from STOP0 to SAFE mode to be interrupt requests. This is done in the MC_RGM's 'Functional' Event Alternate Request register (RGM_FEAR).

1.39 ERR007274: LINFlexD: Consecutive headers received by LIN Slave triggers error interrupt

Description:

As per the Local Interconnect Network (LIN) specification, the processing of one frame should be aborted by the detection of a new header sequence.

In LINFlexD, if the LIN Slave receives a new header instead of data response corresponding to a previous header received, it triggers a framing error during the new header's reception. The LIN Slave still waiting for the data response corresponding to the first header received.

Workaround:

The following three steps should be followed :

1. Set the MODE bit in the LIN Time-Out Control Status Register (LINTCSR[MODE]) to '0'.
2. Set Idle on Timeout in the LINTCSR[IOT] register to '1'.
3. Configure master to wait until the occurrence of the Output Compare flag in LIN Error Status Register (LINESR[OCF]) before sending the next header. This flag causes the LIN Slave to go to an IDLE state before the next header arrives, which will be accepted without any framing error.

1.40 ERR007322: FlexCAN: Bus Off Interrupt bit is erroneously asserted when soft reset is performed while FlexCAN is in Bus Off state

Description:

Under normal operation, when FlexCAN enters in Bus Off state, a Bus Off Interrupt is issued to the CPU if the Bus Off Mask bit (CTRL[BOFF_MSK]) in the Control Register is set. In consequence, the CPU services the interrupt and clears the ESR[BOFF_INT] flag in the Error and Status Register to turn off the Bus Off Interrupt.

In continuation, if the CPU performs a soft reset after servicing the bus off interrupt request, by either requesting a global soft reset or by asserting the MCR[SOFT_RST] bit in the Module Configuration Register, once MCR[SOFT_RST] bit transitions from 1 to 0 to acknowledge the soft reset completion, the ESR[BOFF_INT] flag (and therefore the Bus Off Interrupt) is re-asserted.

The defect under consideration is the erroneous value of Bus Off flag after soft reset under the scenario described in the previous paragraph.

The Fault Confinement State (ESR[FLT_CONF] bit field in the Error and Status Register) changes from 0b11 to 0b00 by the soft reset, but gets back to 0b11 again for a short period, resuming after certain time to the expected Error Active state (0b00). However, this late correct state does not reflect the correct ESR[BOFF_INT] flag which stays in a wrong value and in consequence may trigger a new interrupt service.

Workaround:

To prevent the occurrence of the erroneous Bus Off flag (and eventual Bus Off Interrupt) the following soft reset procedure must be used:

1. Clear CTRL[BOFF_MSK] bit in the Control Register (optional step in case the Bus Off Interrupt is enabled).
2. Set MCR[SOFT_RST] bit in the Module Configuration Register.
3. Poll MCR[SOFT_RST] bit in the Module Configuration Register until this bit is cleared.
4. Wait for 4 peripheral clocks.
5. Poll ESR[FLTCONF] bit in the Error and Status Register until this field is equal to 0b00.
6. Write "1" to clear the ESR[BOFF_INT] bit in the Error and Status Register.
7. Set CTRL[BOFF_MSK] bit in the Control Register (optional step in case the Bus Off Interrupt is enabled).

1.41 ERR007394: MC_ME: Incorrect mode may be entered on low-power mode exit.

Description:

For the case when the Mode Entry (MC_ME) module is transitioning from a run mode (RUN0/1/2/3) to a low power mode (HALT/STOP/STANDBY*) if a wake-up or interrupt is detected one clock cycle after the second write to the Mode Control (ME_MCTL) register, the MC_ME will exit to the mode previous to the run mode that initiated the low power mode transition.

Example correct operation DRUN->RUN1-> RUN3->STOP->RUN3

Example failing operation DRUN->RUN1-> RUN3->STOP->RUN1

Note: STANDBY mode is not available on all MPC56xx microcontrollers

Workaround:

To ensure the application software returns to the run mode (RUN0/1/2/3) prior to the low power mode (HALT/STOP/STANDBY*) it is required that the RUNx mode prior to the low power mode is entered twice.

The following example code shows RUN3 mode entry prior to a low power mode transition:

```
ME.MCTL.R = 0x70005AF0;          /* Enter RUN3 Mode & Key */
ME.MCTL.R = 0x7000A50F;          /*Enter RUN3 Mode & Inverted Key */
while (ME.GS.B.S_MTRANS) {}      /*Wait for RUN3 mode transition to complete*/
ME.MCTL.R = 0x70005AF0           /*Enter RUN3 Mode & Key*/
ME.MCTL.R = 0x7000A50F           /*Enter RUN3 Mode & Inverted Key */
while (ME.GS.B.S_MTRANS) {}      /*Wait for RUN3 mode transition to complete*/
/* Now that run mode has been entered twice can enter low power mode  */
/* (HALT/STOP/STANDBY*) when desired.*/
```

1.42 ERR007589: LINFlexD: Erroneous timeout error when switching from UART to LIN mode

Description:

When the LINFlexD module is enabled in Universal Asynchronous Receiver/Transmitter (UART) mode and the value of the MODE bit of the LIN Timeout Control Status register (LINTCSR) is 0 (default value after reset), any activity on the transmit or receive pins will cause an unwanted change in the value of the 8-bit field Output Compare Value 2 (OC2) of the LIN Output Compare register (LINOCR).

As a consequence, if the module is reconfigured from UART to Local Interconnect Network (LIN) mode, an incorrect timeout exception is generated when a LIN communication starts.

Workaround:

Before enabling UART communication, set to 1 the MODE bit of the LIN Timeout Control Status register (LINTCSR) (selecting the output compare mode). This is preventing the LINOCR.OC2 field from being updated during UART communications.

Then, after reconfiguring the LINFlexD to LIN mode, reset the LINRCSR.MODE bit (selecting the LIN mode) before starting LIN communications

1.43 ERR007688: RTC: An API interrupt may be triggered prematurely after programming the API timeout value

Description:

When the API is enabled (RTCC[APIEN]), the API interrupt flag is enabled (RTCC[APIIE]) and the API timeout value (RTCC[APIVAL]) is programmed the next API interrupt may be triggered before the programmed API timeout value. Successive API Interrupts will be triggered at the correct time interval.

Workaround:

The user must not use the first API interrupt for critical timing tasks.

1.44 ERR007938: ADC: Possibility of missing CTU conversions

Description:

The CTU prioritizes and schedules trigger sources so that the ADC will receive only one CTU trigger at a time. However, whilst a Normal or Injected ADC conversion is ongoing as the ADC moves state from IDLE-to-SAMPLE, SAMPLE-to-WAIT, WAIT-to-SAMPLE and WAIT-to-IDLE there are 2 clock cycles at the state transition that a CTU trigger may be missed by the ADC.

Workaround:

To ensure all CTU triggers are received at the ADC Normal and Injected modes must be disabled.

1.45 ERR007953: ME: All peripherals that will be disabled in the target mode must have their interrupt flags cleared prior to target mode entry

Description:

Before entering the target mode, software must ensure that all interrupt flags are cleared for those peripheral that are programmed to be disabled in the target mode. A pending interrupt from these peripherals at target mode entry will block the mode transition or possibly lead to unspecified behaviour.

Workaround:

For those peripherals that are to be disabled in the target mode the user has 2 options:

1. Mask those peripheral interrupts and clear the peripheral interrupt flags prior to the target mode request.
2. Through the target mode request ensure that all those peripheral interrupts can be serviced by the core.

1.46 ERR008227: CGM & ME: The peripheral set clock must be active during a peripheral clock enable or disable request

Description:

An individual peripheral clock can be enabled or disabled for a target mode via the Mode Entry Peripheral Control register (ME_PCTL) and the Mode Entry RUN/Low Power Peripheral Configuration register (ME_RUN_PC & ME_LP_PC). For this process to complete the user must ensure that the peripheral set clock relative to the specific peripheral is enabled for the duration of the current-mode-to-target-mode transition. The peripheral set clock is configured at the Clock Generation Module System Clock Divider Configuration Register (CGM_SC_DC).

A caveat for FlexCAN is for the case when the FXOSC is selected for the CAN Engine Clock Source (FLEXCAN_CTRL[CLK_SRC]). In this instance to enable or disable the FlexCAN peripheral clock the user must ensure FXOSC is enabled through the target mode transition i.e. FXOSC must be enabled for the target mode.

Workaround:

To enable a peripheral clock:

1. Enable the peripheral set clock at CGM_SC_DC.
2. Enable the peripheral clock for the target mode at ME_PCTL & ME_RUN_PC/ ME_LP_PC.
3. Note steps 1 & 2 are interchangeable.
4. Transition to the target mode to enable the peripheral clock.

To disable a peripheral clock:

1. Disable the peripheral clock for the target mode at ME_PCTL & ME_RUN_PC/ ME_LP_PC.
2. Transition to the target mode to disable the peripheral clock.
3. Optionally disable peripheral set clock at CGM_SC_DC. Note to check other peripherals in this peripheral set are not required.

Appendix A Further information

A.1 Reference document

- *Supports microcontrollers SPC560B54x, SPC560B60x and SPC560B64x* (RM0037, Doc ID 15700)
- *32-bit MCU family built on the Power Architecture® for automotive body electronics applications* (SPC560B54x, SPC560B60x, SPC560B64x datasheet, Doc ID 15131)

A.2 Acronyms

Table 2. Acronyms

Acronym	Name
TxMB	Transmit Message Buffer
MC_RGM	Reset Generation Module
MC_ME	Mode Entry
NMI	Non-Maskable Interrupt
LINFlex	Flexible Local Interconnect Network
SWT	Software Watchdog Timer
ADC	Analog-to-Digital Converter
FMPLL	Frequency Modulated Phase-Locked Loop
MB	Message Buffer
MCR	Module Configuration Register
CTU	Cross Trigger Unit
ECSM	Error Correction Status Module
DMA	Direct Memory Access
BOF	Buffer Overrun Flag
BOIE	Buffer Overrun Error Interrupt Enable
LINIER	LIN Interrupt Enable Register
GCR	Global Control Register
CSI	Combined Serial Interface
SPI	Serial Peripheral Interface
DSPI	Deserial Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter
LINSR	LIN Status Register
APC	Address Pipelining Control
RWSC	Read Wait-State Control
LIN	Local Interconnect Network

Table 2. Acronyms (continued)

Acronym	Name
LINTCSR	LIN Timeout Control Status register
LINOCR	LIN Output Compare register
OC2	Output Compare Value 2

Revision history

Table 3. Document revision history

Date	Revision	Changes
22-Sep-2015	1	Initial release.