SPC56xL70, SPC56xL64 Errata sheet

SPC56xL70, SPC56xL64 devices errata JTAG_ID = 0X0AEA_9041

Introduction

This errata sheet describes all the functional and electrical problems known in the cut 2 of the SPC56xL70, SPC56xL64 devices, identified with the JTAG_ID = 0X0AEA_9041.

All the topics covered in this document refer to RM0342 and SPC56xL70xx datasheet (see Appendix A: Additional information).

Device identification:

- JTAG ID = 0x0AEA_9041
- MCU ID Register 1 (MIDR1):
  - MAJOR MASK[3:0] = 0x1
  - MINOR MASK[3:0] = 0x0

Contents

List of tables

1 Functional problems

1.1 ERR000063: MC_ME: ME_PCTL18 not protected by the register protection

Description:

ME_PCTL18 is used to select the configuration of FlexCAN_2 during run and non-run modes. This register can't be protected by the register protection.

ME_PCTL16 and MC_PCTL17, which are related to FlexCAN_0 and FlexCAN_1, are not impacted by this issue.

Workaround:

The device embeds different mechanisms to protect resources against unwanted access, e.g. MPU and AIPS. Use these mechanisms to protect the ME_PCTL18 register against unexpected writing access.

1.2 ERR003320: Flash: single bit correction status is not available in the Error Correction Status Module (ECSM) and in the Fault Collection and Control Unit (FCCU)

Description:

A single bit error correction by the Flash is not passed to the Error Correction Status Module (ECSM) and to the Fault Collection and Control Unit (FCCU). The single bit error correction is only flagged by the SBC bit of the Flash Module Configuration Register (MCR).

Workaround:

Poll the SBC bit (Single Bit Correction Status) of the Flash Module Configuration Register (MCR) to detect a single bit error correction event.

1.3 ERR003511: FlexPWM : Incorrect PWM operation when IPOL is set.

Description:

When IPOL bit is set in complementary mode, the source of the PWMi waveform is supposed to be switched from the VAL2 and VAL3 registers to the VAL4 and VAL5 registers. This switch doesn’t happen.

Workaround:

Instead of setting IPOL bit, the application can swap the VAL2/3 values with the VAL4/5 values.

1.4 ERR003697: e200z: Exceptions generated on speculative prefetch

Description:

The e200z4 core can prefetch up to 2 cache lines (64 bytes total) beyond the current instruction execution point. If a bus error occurs when reading any of these prefetch locations, the machine check exception will be taken. For example, executing code within the last 64 bytes of a memory region such as internal SRAM, may cause a bus error when the core prefetches past the end of memory. An ECC exception can occur if the core prefetches locations that are valid, but not yet initialized for ECC.

Workaround:

Do not place code to be executed within the last 64 bytes of a memory region. When executing code from internal ECC SRAM, initialize memory beyond the end of the code until the next 32-byte aligned address and then an additional 64 bytes to ensure that prefetches cannot land in uninitialized SRAM.

1.5 ERR004016: ADC: Presampling on channels 9, 10, 15 leads to incorrect results

Description:

On ADC channels 9 (for factory test only), 10 (VREG_1.2V), 15 (TSENS) when performing presampling using VSS_HV_ADR (PREVAL0=01) and bypassing the sampling (PRECONV=1) results in an incorrect converted presampled value.

Workaround:

ADC Conversion Timing Register 1 (CTR1) and Presampling Control Register (PSCR), field PREVAL1(bits 27:28) can be programmed to select the conversion durations and reference voltages for ADC channels 9, 10, 15.

1.6 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,ch2,ch3,ch4) the Abort switch doesn't 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's possible to detect the unexpected behavior by using the CEOCFR0 register. The CEOCFR0.EOC_CHx field will not be set for a not restored or skipped channel, which indicates this issue has occurred. The CEOCFR0.EOC_CHx fields need to be checked before the next Normal chain execution (in scan mode). The CEOCFR0.EOC_CHx fields should be read by every ECH interrupt at the end of every chain execution.

1.7 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].

1.8 ERR004334: MC_RGM: Device stays in reset state on external reset assertion

Description:

On an external reset that is configured to be 'long' the device may remain in reset if the system clock is configured to be sourced by a clock source other than the 16 MHz Internal RC Oscillator (IRCOSC). Recovery from the reset in this case can only be achieved via a power-down and power-up cycle.

The failure condition can only be seen with the following Reset Generation Module (MC_RGM) settings for Functional Event Short Sequence register, External Reset field (RGM_FESS[SS_EXR]) and Functional Bidirectional Reset Enable register, External Reset field (RGM_FBRE[BE_EXR]):

- RGM_FESS[SS_EXR] = 0b0 (long external reset)
- RGM_FBRE[BE_EXR] = 0b0 (asserted on external reset event)

Note: This condition can only occur if the cause of the device reset was the external reset assertion. It does not occur if, for example, the device reset was due to a power-on.

Note: RGM_FESS[SS_EXR] = 0b0 and RGM_FBRE[BE_EXR] = 0b0 are the default settings out of power-on reset (POR).

Workaround:

There are two possible workarounds. In both, the workaround takes effect only after software has reconfigured the MC_RGM. Therefore, in order to ensure that the issue cannot occur after POR exit and before the software has executed the workaround, the system clock must not be re-configured in the Mode Entry module (MC_ME) to be sourced by a clock source other than the IRCOSC until after the workaround has been executed.

1. Workaround #1:

Always configure the external reset event to prevent the external reset output to be driven by the MC_RGM by writing 0b1 to RGM_FBRE[BE_EXR].

If the external reset has been configured to be long (RGM_FESS[SS_EXR] = 0b0) and self testing has been enabled via the flash option, the external reset pin will still be asserted from the time of external assertion until reset sequence exit after start-up self test execution.

If the external reset has been configured to be long (RGM_FESS[SS_EXR] = 0b0) and self testing has been disabled via the flash option, the external reset pin will still be asserted from the time of external assertion until the chip configuration is loaded from the flash during reset PHASE3.

If the external reset has been configured to be short (RGM_FESS[SS_EXR] = 0b1), the external reset pin will still be released as soon as it is no longer asserted from off-chip.

2. Workaround #2:

Always configure the external reset as 'short' by writing 0b1 to RGM_FESS[SS_EXR]. In addition, use software to trigger a long 'functional' or 'destructive' reset via the Mode Entry module (MC_ME) if flash initialization or start-up self test is required.

The impact of this workaround is the additional time that the device is in reset (due to the short reset sequence triggered by the external reset) and the overhead required for software to check the reset status and request a software reset.

1.9 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). But the BOF bit will be cleared when the interrupt routine is entered, preventing the user to identify the source of error.

Workaround:

Buffer overrun errors in UART FIFO mode can be detected by enabling only this source in the LIN Error Combined interrupt.

1.10 ERR006481: NZ4C3/NZ7C3: Erroneous Resource Full Message under certain conditions

Description:

The e200zx core Nexus interface may transmit an erroneous Resource Full Message (RFM) following a Program Trace history message with a full history buffer. The History information for both of the messages are the same and the RFM should have not been transmitted. This occurs when the instruction following the indirect branch instruction (which triggers the Program Trace History message) would have incremented the History field. The instructions must be executed in back to back cycles for this problem to occur. This is the only case that cases this condition.

Workaround:

There are three possible workarounds for this issue:

1. Tools can check to see if the Program Trace History message and proceeding Resource Full Message (RFM) have the same history information. If the history is the same, then the RFM is extraneous and can be ignored.
2. Code can be rewritten to avoid the History Resource Full Messages at certain parts of the code. Insert 2 NOP instructions between problematic code. Or inserting an “isync” or an indirect branch somewhere in the code sequence to breakup/change the flow.
3. If possible, use Traditional Program Trace mode can be used to avoid the issue completely. However, depending on other conditions (Nexus port width, Nexus Port speed, and other enabled trace types), overflows of the port could occur

1.11 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 transition 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.

1.12 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.

1.13 ERR007120: NZxC3: DQTAG implemented as variable length field in DQM message

Description:

The e200zx core implements the Data Tag (DQTAG) field of the Nexus Data Acquisition Message (DQM) as a variable length packet instead of an 8-bit fixed length packet. This may result in an extra clock ("beat") in the DQM trace message depending on the Nexus port width selected for the device.

Workaround:

Tools should decode the DQTAG field as a variable length packet instead of a fixed length packet.

1.14 ERR007227: FCCU: FCCU Output Supervision Unit (FOSU) will not monitor faults enabled while already pending

Description:

The Fault Collection and Control Unit (FCCU) Output Supervision Unit (FOSU) will not monitor the FCCU reaction to fault inputs that are enabled with an already pending notification.

The FOSU monitoring is triggered by an edge from a fault input. The edge detection will be blocked in following cases:

1. When a fault input is disabled in the FCCU and a fault occurs,
2. When a fault input is enabled in the FCCU and a fault occurs in the CONFIG state. FOSU edge detection remains blocked until it gets initialized by a FCCU reaction or a destructive reset

Workaround:

Apply the following procedure when enabling fault inputs in the FCCU in order to ensure correct monitoring by the FOSU:

1. Check for FCCU pending faults and clear them.
2. Configure the FCCU as desired. In addition enable fault input for interrupt reaction (software recovery mode) to an injected error on this input.
3. Immediately on exiting the CONFIG state, check for FCCU pending faults. If there is a fault status set then initiate a destructive reset.
4. Clear the FOSU status by injecting a fault on the FCCU fault input configured for software recovery mode. This will generate a FCCU reaction that will clear the FOSU edge detection logic.

Apply the following procedure when exiting FCCU CONFIG state in order to ensure correct monitoring by the FOSU:

1. Check for FCCU pending faults. If there is a fault status set then initiate a destructive reset

1.15 ERR007274: LINFlexD: Consecutive headers received by LIN Slave triggers the LIN FSM to an unexpected state

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 and the LIN Finite State Machine (FSM) should move to the protected identifier (PID) state. In the PID state, the LIN FSM waits for the detection of an eight bit frame identifier value.

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 and returns to IDLE state.

Workaround:

The following three steps should be followed:

1. Configure slave to Set the MODE bit in the LIN Time-Out Control Status Register (LINTCSR[MODE]) to '0'.
2. Configure slave to Set Idle on Timeout in the LINTCSR[IOT] register to '1'. This causes the LIN Slave to go to an IDLE state before the next header arrives, which will be accepted without any framing error.
3. Configure master to wait for Frame maximum time (TFrame_Maximum as per LIN specifications) before sending the next header.

Note: *THeader_Nominal = 34 * TBit*

*TResponse_Nominal = 10 * (NData + 1) * TBit*

*THeader_Maximum = 1.4 * THeader_Nominal*

*TResponse_Maximum = 1.4 * TResponse_Nominal*

TFrame_Maximum = THeader_Maximum + TResponse_Maximum

where TBit is the nominal time required to transmit a bit and NData is number of bits sent.

1.16 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).

1.17 ERR007352: DSPI: reserved bits in slave CTAR are writable

Description:

When the Deserial/Serial Peripheral Interface (DSPI) module is operating in slave mode (the Master [MSTR] bit of the DSPI Module Configuration Register [DSPIx_MCR] is cleared), bits 10 to 31 (31 = least significant bit) of the Clock and Transfer Attributes Registers (DSPIx_CTARx) should be read only (and always read 0). However, these bits are writable, but setting any of these bits to a 1 does not change the operation of the module.

Workaround:

There are two possible workarounds.
Workaround 1: Always write zeros to the reserved bits of the DSPIx_CTARn_SLAVE (when operating in slave mode).

Workaround 2: Mask the reserved bits of DSPIx_CTARn_SLAVE when reading the register in slave mode.

1.18 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[a]) 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

Workaround:

To ensure the application software returns to the run mode (RUN0/1/2/3) prior to the low power mode (HALT/STOP/STANDBY[a]) 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[a]) when desired. */
```

1.19 ERR007589: LINFlexD: Spurious timeout error when switching from UART to LIN mode or when resetting LINTCSR[MODE] bit in LIN mode

Description:

If 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).

If the LINFlexD module is enabled in LIN mode and the value of the MODE bit of the LIN Timeout Control Status register (LINTCSR) is changed from ‘1’ to ‘0’, then the old value of the Output Compare Value 1 (OC1) and Output Compare Value 2 (OC2) of the LIN Output Compare Register (LINOCR) is retained.

a. STANDBY mode is not available on all SPC56xx microcontrollers.

As a consequence, if the module is reconfigured from UART to Local Interconnect Network (LIN) mode, or LINTCSR MODE bit is changed from ‘1’ to ‘0’, an incorrect timeout exception is generated when the LIN communication starts.

Workaround:

If the LINFlexD module needs to be switched from UART mode to LIN mode, before writing UARTCR[UART] to 1, ensure that the LINTCSR[MODE] is first set to 1.

If the LINFlexD module is in LIN mode and LINTCSR[MODE] needs to be switched from 1 to 0 in between frames, the LINOCR must be set to 0xFFFF by software.

1.20 ERR007877: FlexPWM: do not enable the fault filter

Description:

Operation of the fault pin filter of the Flexible Pulse Width Modulation (FLEX_PWM) may be inconsistent if the Fault Filter is enabled, by setting the Filter Period greater than zero in the Fault Filter register (FFILT[FILT_PER] > 0). The fault filter flag may be set even though the pulse is shorter than the filter time.

Workaround:

Do not enable the PWM fault pin filters. Disable the fault pin filters by setting the Fault Filter Period to 0 in the Fault Filter Register (FFILT[FILT_PER] = 0).

1.21 ERR008070: SWG: GPIO[55] functionality is not available unless the SWG is powered down

Description:

The General Purpose Input/Output 55 (GPIO[55]) functionality on port D[7] is disabled if the Sine Wave Generator module (SWG) is not in power down mode. The SWG will not enter power down mode if the SWG clock input is disabled.

Workaround:

Ensure that the SWG clock input is enabled via the Aux Clock 0 Divider Configuration 1 register (CGM_AC0_DC1[DE1]=1) prior to putting the SWG in power down mode in the SWG control register (SWG_CTRL[PDS] = 1). This will allow GPIO[55] functionality on port D[7].

1.22 ERR008080: LINFlexD: TX pin gets set to High-Z when in IDLE state

Description:

LINFlex drives the buffer enable signal for its transmit pin output (TX) to be ‘0’ after transmitting the LIN frame. This causes the TX line to go to High-Z which will be an issue if the associated LIN transceiver has an internal “pull down”.

Issue will also occur when module is configured in UART mode with the TX output pin becoming High-Z when idle.

Workaround:

When operating in LIN mode, use a LIN transceiver with internal “pull up”. If the transceiver has an internal “pull down”, add an external “pull up”.

When operating in UART mode, the issue can be worked around by enabling the internal pull up on the TX pin using the corresponding SIU_MSCR register.

1.23 ERR008933: LINFlexD: Inconsistent sync field may cause an incorrect baud rate and Sync Field Error Flag may not be set

Description:

When the LINFlexD module is configured as follows:

1. LIN (Local Interconnect Network) slave mode is enabled by clearing the Master Mode Enable bit in the LIN Control Register 1 (LINCR1[MME] = 0b0), and
2. Auto synchronization is enabled by setting LIN Auto Synchronization Enable (LINCR1[LASE] = 0b1)

The LINFlexD module may automatically synchronize to an incorrect baud rate without setting the Sync Field Error Flag in the LIN Error Status register (LINESR[SFEF]) in case Sync Field value is not equal to 0x55, as per the Local Interconnect Network (LIN) specification.

The auto synchronization is only required when the baud-rate in the slave node cannot be programmed directly in software and the slave node must synchronize to the master node baud rate.

Workaround:

There are two possible workarounds.

Workaround 1:

When the LIN time-out counter is configured in LIN Mode by clearing the MODE bit of th LIN Time-Out Control Status register (LINTCSR[MODE]= 0x0]):

1. Set the LIN state Interrupt enable bit in the LIN Interrupt Enable register (LINIER[LSIE] = 0b1)
2. When the Data Reception Completed Flag is asserted in the LIN Status Register (LINSR[DRF] = 0b1) read the LIN State field (LINSR[LINS])
3. If LINSR[LINS]= 0b0101, read the Counter Value field of the LIN Time-Out Control Status register (LINTCSR[CNT]), otherwise repeat step 2
4. If LINTCSR[CNT] is greater than 0xA, discard the frame.

When the LIN Time-out counter is configured in Output Compare Mode by setting the LINTCSR[MODE] bit:

1. Set the LIN State Interrupt Enable bit in the LIN Interrupt Enable register (LINIER[LSIE])
2. When the Data Reception Completed flag bit is asserted in the LIN Status Register (LINSR[DRF] = 0b1), read the LINSR[LINS] field
3. If LINSR[LINS]= 0b0101, store LINTCSR[CNT] value in a variable (ValueA), otherwise repeat step 2
4. Clear LINSR[DRF] flag by writing LINSR[LINS] field with 0xF
5. Wait for LINSR[DRF] to become asserted again and read LINSR[LINS] field
6. If LINSR[LINS] = 0b0101, store LINTCSR[CNT] value in a variable (ValueB), else repeat step 4
7. If ValueB - ValueA is greater than 0xA, discard the frame.

Workaround 2:

Do not use the auto synchronization feature (disable with LINCR1[LASE] = 0b0) in LIN slave mode.

1.24 ERR008970: LINFlexD: Spurious bit error in extended frame mode may cause an incorrect Idle State

Description:

The LINFlexD module may set a spurious Bit Error Flag (BEF) in the LIN Error Status Register (LINESR), when the LINFlexD module is configured as follows:

- Data Size greater than eight data bytes (extended frames) by configuring the Data Field Length (DFL) bitfield in the Buffer Identifier Register (BIDR) with a value greater than seven (eight data bytes)
- Bit error is able to reset the LIN state machine by setting Idle on Bit Error (IOBE) bit in the LIN Control Register 2 (LINCR2)

As consequence, the state machine may go to the Idle State when the LINFlexD module tries the transmission of the next eight bytes, after the first ones have been successfully transmitted and Data Buffer Empty Flag (DBEF) was set in the LIN Status Register (LINSR).

Workaround:

Do not use the extended frame mode by configuring Data Field Length (DFL) bit-field with a value less than eight in the Buffer Identifier Register (BIDR) (BIDR[DFL] < 8).

1.25 ERR009682: SPI: Inconsistent loading of shift register data into the receive FIFO following an overflow event

Description:

In the Serial Peripheral Interface (SPI) module, when both the receive FIFO and shift register are full (Receive FIFO Overflow Flag bit in Status Register is set (SR [RFOF] = 0b1)) and then the Clear Rx FIFO bit in Module Configuration Register (MCR [CLR_RXF]) is asserted to clear the receive FIFO, shift register data is loaded into the receive FIFO after the clear operation completes.

Avoid a receive FIFO overflow condition (ie SR[RFOF] should never be 0b1). To do this, monitor the RX FIFO Counter field of the Status Register (SR[RXCTR]) which indicates the number of entries in receive FIFO and clear before the counter equals the FIFO depth.

Workaround:

Alternatively, after every receive FIFO clear operation (MCR[CLR_RXF] = 0b1) following a receive FIFO overflow (SR[RFOF] = 0b1) scenario, perform a single read from receive FIFO and discard the read data.

1.26 ERR009849: BAM: Serial boot not supported when flash is programmed with Power Architecture instruction set software

Description:

The Boot Assist Module (BAM) doesn't support serial boot when the flash memory is programmed with Power Architecture instruction set (Book E) software. The functionality of the BAM depends on the value of the Variable Length Encoding (VLE) bit in the Reset Configuration Half Word (RCHW) of the bootable flash sector. If the VLE bit = 0, indicating that Book E software is programmed in flash memory, the BAM doesn't function correctly and the serial boot is not supported.

Workaround:

Erase flash memory or program flash memory with VLE software prior to executing serial boot with Book E software.

1.27 ERR009928: FlexPWM: Half cycle automatic fault clearing does not work in PWM submodule 0 under some conditions

Description:

When

1. the EXT_SYNC signal is selected to cause initialization by setting the Submodule 0 Control 2 Register FlexPWM_SUB0_CTRL2[INIT_SEL] = 11, and
2. a specific FAULTx input is associated with the submodule 0 outputs using the Submodule 0 Fault Disable Mapping Register (FlexPWM_SUB0_DISMAP), and
3. the respective bit for that FAULTx is 0 in the FFULL bit field of the Fault Status Register (FlexPWM_FSTS), and
4. the respective bit for that FAULTx is 1 in the FAUTO bit field of the Fault Control Register (FlexPWM_FCTRL),

then the PWM outputs of submodule 0 is re-enabled at the cycle boundary (full cycle) and is not reenabled at the cycle midpoint (half cycle).

Workaround:

When the EXT_SYNC signal is used to cause initialization in submodule 0 and the submodule 0 PWM outputs are disabled by a specific FAULTx input, use full cycle automatic fault clearing for the specific FAULTx input by setting the corresponding bit of the Fault Status Register FlexPWM_FSTS[FFULL] to 1.

1.28 ERR009976: DSPI: Incorrect data received by master with Modified transfer format enabled when using Continuous serial communication clock mode

Description:

When the Deserial Serial Peripheral Interface (DSPI) module is configured as follows:

1. Master mode is enabled (Master/Slave Mode Select bit in Module Configuration Register is set (DSPI_MCR [MSTR] = 0b1))
2. Modified transfer format is enabled (Modified Transfer Format Enable bit in Module Configuration Register is set (DSPI_MCR [MTFE] = 0b1))
3. Continuous serial communication clock mode is enabled (Continuous SCK Enable bit in Module Configuration Register is set (DSPI_MCR [CONT_SCKE] = 0b1))

In this configuration if the frame size of the current frame is greater than the frame size of the next received frame, corrupt frames are received in two scenarios:

1. Continuous Peripheral Chip Select Enable bit in PUSH TX FIFO Register is set (DSPI_PUSHR [CONT] = 0b1)
2. DSPI_PUSHR [CONT] = 0b0 and lower significant bit of the frame is transferred first (LSB first bit in Clock and Transfer Attributes Register is set (DSPI_CTAR [LSBFE] =0b1))

Workaround:

To receive correct frames:

1. When DSPI_PUSHR [CONT] = 0b1, configure the frame size of the current frame less than or equal to the frame size of the next frame (for all frames).
2. When DSPI_PUSHR [CONT] = 0b0, configure DSPI_CTAR [LSBFE] = 0b0. Alternatively, configure the frame size of the current frame less than or equal to the frame size of the next frame (for all frames).

Make sure that for all received frames, the bits are read equal to their respective frame sizes and any extra bits during POP operation are masked.

1.29 DAN-0046129 STCU: Pattern to run the MBIST-only wrong

Description:

The reference manual in section "Self-test bypass and MBIST-only mode" states the pattern to run the configuration MBIST-only. The user must copy this patter in a specific location of the shadow sector of the flash.

The second word of 128-bit shows:

`7F140000_00080388_FFFFFFFF_FFFFFFFF.`

This value is wrong. The correct value is:

`7F140000_000803A0_FFFFFFFF_FFFFFFFF.`

Workaround:

The user must not consider the correct value and not use the wrong one.

Appendix A Additional information

A.1 Reference document

- *SPC56XL70xx 32-bit MCU family built on the embedded Power Architecture®* (RM0342, Doc ID 023986).
- *32-bit Power Architecture® microcontroller for automotive SIL3/ASILD chassis and safety applications* (SPC56xL70, SPC56xL64 datasheet, Doc ID 027962).

A.2 Acronyms

Table 1. Acronyms

Acronym	Name
ADC	Analog-to-digital converter
APC	Analog pad control
BAM	Boot assist module
CMU	Clock monitor unit
CPU	Control processing unit
CHIERFR	Controller host interface error flag register
CTU	Cross trigger unit
DMA	Direct memory access
DPM	Decoupled parallel mode
ECSM	Error correction status module
FCCU	Fault collection and control unit
FIFO	First in first out
ILSA_EF	Illegal system bus address error flag
LSM	Lock step mode
MB	Message buffer
MCU	Microcontroller unit
PCR	Pad configuration register
POC	Protocol operation control
RCCU	Redundancy control checker unit
RWE	Read-while-write event error
RWW	Read while write
RXGMASK	RX global mask
RXIMR	RX individual mask register
SBC	Single bit correction-status
SLL	Secondary low/mid address space block lock register

Table 1. Acronyms (continued)

Acronym	Name
SWG	Sine-wave generator
XOSC	External oscillator
UART	Universal Asynchronous Receiver/Transmitter
LINTCSR	LIN Timeout Control Status Register
OC2	Output Compare Value 2
LINOCR	LIN Output Compare Register
MME	Master Mode Enable
LASE	LIN Auto Synchronization Enable
SFEF	Sync Field Error Flag
LINESR	LIN Error Status Register
BIDR	Buffer Identifier Register
DBEF	Data Buffer Empty Flag

Revision history

Table 2. Document revision history

Date	Revision	Changes
05-Dec-2012	1	Initial release.
17-Sep-2013	2	Updated Disclaimer.
08-Jul-2014	3	Add the following errata: – e6726 – e7120 – e7274 – e7322 – e7352 – e7394
07-Oct-2014	4	Add the following errata: – ERR007877 – ERR008070
19-Nov-2015	5	Added the functional problems: – ERR007589 – ERR008080 – ERR008933 – ERR008970 Updated the functional problems: – ERR004186 – ERR007274
20-Mar-2018	6	Added the functional problems: – ERR003511 – ERR006481 – ERR007227 – ERR009682 – ERR009849 – ERR009928 – ERR009976 – DAN-0046129 Updated the functional problems: – ERR008933