SPC570Sxx
Errata sheet

SPC570Sx device family errata JTAG_ID = 0x110A2041

Introduction

This errata sheet refers to SPC570Sx device family, cut2.1 silicon revision identified with the JTAG_ID = 0x110A2041 and describes the functional and electrical deviations to device documentation (*RM0349 rev 6* and *SPC570S40E1, SPC570S40E3, SPC570S50E1, SPC570S50E3 datasheet* rev 6) (see *Section A.1: Reference document*).

Device identification:

- Package device marking mask identifier: BB
- JTAG_ID = 0x110A2041
- MCU ID Register 1 (MIDR1):
  - MAJOR_MASK[3:0]: 0x1
  - MINOR_MASK[3:0]: 0x1

This errata sheet applies to SPC570Sx family devices in accordance with *Table 1*.

Table 1. Device summary

Part number	Package
SPC570S50E1	eTQFP64
SPC570S50E3	eTQFP100
SPC570S40E1	eTQFP64
SPC570S40E3	eTQFP100

Contents

1 Functional problems

1.1 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").

1.2 ERR004136: XOSC and IRCOSC: Bus access errors are generated in only half of non-implemented address space of XOSC and IRCOSC

Description:

Bus access errors are generated in only half of the non-implemented address space of Oscillator External Interface (40MHz XOSC) and IRCOSC Digital Interface (16MHz Internal RC oscillators [IRC]).

Workaround:

Do not access unimplemented address space for XOSC and IRCOSC register areas OR write software that is not dependent on receiving an error when access to unimplemented XOSC and IRCOSC space occurs.

1.3 ERR006350: LINFlexD: WLS feature cannot be used in buffered mode.

Description:

The LINFlex module may not operate correctly if the Special Word Length (WLS) for enabling 12-bit data length is selected in the Universal Asynchronous Receiver/Transmitter (UART) Mode Control Register (UARTCR) and configured in the transmit buffered mode.

Workaround:

When WLS mode is required, always use the First In, First Out (FIFO_ mode of the UART LINFLEX module by setting the Transmit FIFO/Buffer mode bit of the UARTCR (UARTCR[TFBM]=1).

1.4 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.

1.5 ERR006994: XBIC: XBIC may trigger false FCCU alarm.

Description:

The Crossbar Integrity Checker (XBIC) will incorrectly signal a fault alarm when a system bus request results in a bus error termination from a crossbar client. The Fault Correction and Collection Unit (FCCU) alarm number 40 (for XBIC_0) or number 35 for (XBIC_1) will be signaled.

Workaround:

Software should handle faults on FCCU alarm #35 and alarm #40 in case of a system bus error.

1.6 ERR007061: SPU: Reserved location can be written.

Description:

Read accesses to reserved locations of the Sequence Processing Unit (SPU) do not return 0. Instead the last written value will be read. Note: the SPU registers are accessible only via JTAG and are used by debugging tools to create triggers combining internal events of the device.

Workaround:

Do not access to reserved locations of the SPU. Alternatively, always write 0 to the reserved locations.

1.7 ERR007103: MC_CGM: Incorrect cause for the latest clock source switch may be reported by the CGM if a safe mode request arrives when the system clock is the IRC

Description:

If the current system clock source is the Internal RC oscillator (IRC) as reported in the Clock Generation Module System Clock Select Status Register System Clock Source Selection field (CGM_SC_SS.SELSTAT = 0b0000) and the Clock Generation Module System Clock Select Status Register Switch Trigger Cause shows the cause for the latest clock switch as MC_ME succeeded (CGM_SC_SS.SWTRG = 0b001) indicating that a successful Mode Entry mode change was the cause of the last clock change then the CGM_SC_SS.SWTRG will incorrectly continue to show the cause for the latest clock switch as MC_ME succeeded after a safe mode request is generated. If a subsequent safe mode request is generated CGM_SC_SS.SWTRG switches to report the correct status value of 0b100 (switch to system clock source 0 due to SAFE mode request or reset succeeded).

Workaround:

If the CGM_SC_SS.SELSTAT shows the system clock as IRC (0b0000), then software should check the Mode Entry Global Status register Current Mode field (ME_GS.CURRENT_MODE) and the Mode Entry Interrupt Status Register Safe mode Interrupt (ME_IS.I_SAFE) to establish the cause of the switch.

1.8 ERR007115: DSPI: Mixing 16 and 32 bits frame size in XSPI Mode can cause incorrect data to be transmitted.

Description:

The Deserial Serial Peripheral Interface (DSPI) features an Extended SPI mode (XSPI) supporting frames of up to 32 bits.When the XSPI Mode is enabled, transferring a mixture of frames having a size up to 16 bits and those having size above 16 bits can cause an incorrect data transmission to occur. This happens when the First In/First Out (FIFO) queue read pointers roll-over and a frame needs to be extracted from both the bottom of the FIFO and the top of the FIFO when the Frame Size is greater than 16 bits.

Workaround:

Even number of Transmit FIFO Register (TXFR) registers: Do not mix frames that have data sizes of less than 16 bits with those having a size more than 16 bits in XSPI Mode.Odd number of TXFR registers: Do not mix frames that have data sizes of less than 16 bits with those having a size more than 16 bits in XSPI Mode.If the frame size is greater than 16, initially send a dummy frame (a frame with no chip select, but containing data) of less than or equal to 16 bits. Continue sending a dummy frame after each (number of TXFR Registers - 1) / 2 frames.

1.9 ERR007246: SARADC: First conversion after exit from stop mode may be corrupted

Description:

In the Successive Approximation Analog to Digital Converter (SARADC), if a chain conversion is on going and a transition to stop mode request is done, the result of the first conversion after exit of the stop mode (in other words, the conversion that was interrupted when going into stop mode) will be corrupted in the following cases:

Case A: the peripheral bridge clock (PBRIDGEx_CLK) becomes lower than the SARADC clock (SAR_CLK)

Note: This might be the case if the input of the PBRDIGEx_CLK divider is changed during the mode transitions (from the output of the PLL to the internal RC Oscillator for example)

Case B: the PBRIDGEx_CLK is resumed after the SAR_CLK, with a delay greater than 10 cycles of SAR_CLK.

Workaround:

The following workarounds are possible:

1. Disable PBRIDGEx_CLK during stop mode and enable it only with a configuration such that it is greater than SAR_CLK.OR
2. Verify that no analog conversion is ongoing before issuing a stop mode request by reading the ADC status field of the Main Status Register (SARADC_x.MSR[ADCSTATUS] = 0b000, also known as IDLE).OR
3. Ignore the result of the first conversion after stop mode exit, considering it as corrupted.

1.10 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).

1.11 ERR007339: STCU2: STCU2 fault injected by FCCU is self clearing.

Description:

In the Self-Test Control Unit (STCU2), a fault can be injected by the Fault Collection and Control Unit (FCCU) in order to verify the correct behavior of the interface (fake fault).The STCU_LMBIST_USR_ERR signal, which is connected to the FCCU input #8, generates only a pulse when an error is injected to this signal by the FCCU.This is different to other signals from STCU2, where injected faults remain asserted until explicitly cleared.

Workaround:

Use a software recoverable fault (select-able with FCCU_RF_CFG) for FCCU input #8, when a fault is injected into the STCU2.

1.12 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.

1.13 ERR007433: JTAGM: Nexus error bit is cleared by successful RWA

Description:

The JTAG Master module status register includes a Nexus error status bit (JTAGM_SR[Nexus_err]) that indicates the status of the last Nexus Read/Write Access (RWA) command. Once this information is latched, it can only be cleared by performing a successful RWA transaction via the same core that caused the error. In addition, if a RWA transaction is performed by a different core, the error bit will not be cleared and it is not possible to determine if the access by the second core RWA was successful or generated another error.

In general, this bit should only be set when the Nexus RWA accesses non-existent or protected memory spaces.

Workaround:

If the status information is required from a specific core, the user software or tool should read the error bit (ERR) of the e200zx core's Nexus Read/Write Access Control/Status register. To avoid setting the error bit, do not perform illegal memory accesses.

1.14 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.

1.15 ERR007788: SIUL2: A transfer error is not generated for 8-bit accesses to non-existent MSCRs.

Description:

An 8-bit access attempt to non-existent MSCRs (Multiplexed Signal Configuration Registers) in the SIUL2 (System Integration Unit Light 2) address space does not generate a transfer error. 16-bit or 32-bit accesses to non-existent MSCRs will generate a transfer error.

Workaround:

Do not expect transfer errors on 8-bit accesses to non-existent MSCRs in the SIUL2 address space.

1.16 ERR007869: FCCU: FOSU monitoring of a fault is blocked for second or later occurrence of the same fault.

Description:

The Fault Collection and Control Unit (FCCU) Output Supervision Unit (FOSU) will not monitor the FCCU for the second or later occurrence of a given fault in the following cases:

1. Reset is programmed as the only reaction for the fault.
2. Assertion of the fault coincides with the long/short functional reset reaction to a fault previously asserted.

Workaround:

There are two possible workarounds. Either one can be used with the same effectiveness.

1. In addition to the reset reaction, enable either the interrupt (IRQ) or Non-maskable Interrupt (NMI) or error out signaling reaction for the faults that have a reset reaction enabled.
2. Apply the following procedure during the FCCU configuration after a reset and in the fault service routine while clearing the fault status inside the FCCU.
   a) Check for FCCU pending faults and clear them.
   b) Configure the FCCU as desired.
   c) Enable a fault as software recoverable by setting its corresponding bit in the NCF Configuration Register (FCCU_NCF_CFGn)
   d) Inject a fake fault to the fault set up in step “c” by writing the corresponding code into the NCF Fake Register (FCCU_NCFF)
   e) Check that there are no pending faults else clear the pending faults and repeat steps “d” and “e”
   f) Reconfigure the fault that was configured for software recovery mode.

1.17 ERR007904: PASS: Programming Group Lock bit (PGL) can be de-asserted by multiple masters writing the correct password sections to the CINn registers.

Description:

The eight Challenge Input Registers (CINn) in the Password and Device Security Module (PASS) where the 256-bit unlock lock password (8 x 32-bit registers) is provided, can be written by multiple masters. If the written password is correct even though it has been provided from different masters, the password Group Lock (PASS PGL) in the Password Group n Lock 3 Status register (PASS_LOCK3_PGn) is de-asserted and UnLockMaster (MSTR) is set to 0xF.

Therefore, internal registers would not be writable by any of the master other than master whose ID is 0xF if the Master Only (MO) bit is set PASS_LOCK3_PGn.

If a Master wants to update internal registers, it needs to unlock the PASS by writing into all the 8 Password registers.

Workaround:

Set the master only bit inside the PASS (LOCK3_PGn.MSTR) to block other master accesses to the unlocked registers. If the written password has been provided from different masters, a single master should perform the unlock operation again by writing into all the 8 password registers.

1.18 ERR007905: PIT: Accessing the PIT by peripheral interface may fail immediately after enabling the PIT peripheral clock.

Description:

If a write to the Periodic Interrupt Timer (PIT) module enable bit (PIT_MCR[MDIS]) occurs within two bus clock cycles of enabling the PIT clock gate in the MC_CGM (Clock Generation Module) register, the write will be ignored and the PIT will not be enabled.

Workaround:

After enabling the PIT clock in the MC_CGM, insert a read of the PIT_MCR register before writing to the PIT_MCR register. This guarantees a minimum delay of two bus clocks to guarantee the write is not ignored.

1.19 ERR007947: XOSC: Incorrect external oscillator status flag after CMU event clear.

Description:

If an external oscillator (XOSC) is enabled and it becomes unstable (or the crystal fails), the Oscillator Lost Reference status flag in the Clock Monitor Unit Interrupt Status register (CMU0.CMU_ISR[OLRI]) will be set. In addition, the Crystal Oscillator Status flag in the Mode Entry module Global Status Register (MC_ME_GS.S_XOSC) will be cleared (1 = stable clock, 0 = no valid clock). However, if the CMU_ISR[OLRI] is cleared while the oscillator is still in a failing condition, the MC_ME_GS.S_XOSC will incorrectly be set, indicating a valid crystal oscillator.

Workaround:

Monitor the XOSC external oscillator status using the MC_ME_GS.S_XOSC before the CMU0.CMU_ISR.OLRI flag is set. After the CMU0.CMU_ISR.OLRI flag has been set, the MC_ME_GS.S_XOSC flag is valid only after a functional reset. Alternately, the response to the OLRI flag after loss of XOSC clock, can be set in the FCCU to cause a functional reset to clear the MC_ME_GS.S_XOSC flag.

1.20 ERR008042: FCCU: EOUT signals are active, even when error out signaling is disabled.

Description:

Every time the Fault Collection and Control Unit (FCCU) moves into fault state caused by an input fault for which the error out reaction is disabled (FCCU_EOUT_SIG_ENn[EOUTENx]=0), the Error Out 1 and 2 (EOUT[0] and EOUT[1]) will become active for a duration of 250 us plus the value programmed into the FCCU Delta Time register (FCCU_DELTA_T[DELTA_T]). EOUT is not affected if the FCCU moves into the alarm state that generates an interrupt (IRQ), if the Fault is cleared before the alarm timeout.

This erratum does not affect the outputs of other pins (for example, for communication modules like CAN/FlexRay) which are configured by setting the Safe MODE Control bit in the System Integration Unit Lite 2 Multiplexed Signal Configuration Register (SIUL2_MSCR_IO_n [SMC]) to be disabled when the FCCU moves into a fault condition.

Workaround:

There are three possible workarounds:

1. Enable EOUT signaling for all enabled error sources.
2. In case external device (which evaluates EOUT) can communicate with the MCU, the following procedure could be used:
   a) Program any duration of EOUT as per application needs (FCCU_DELTA_T[DELTA_T])
   b) For faults requiring error out reaction, the software shall validate EOUT via separate communication channel (like I2C) while EOUT is asserted.
   c) External device shall implement a timeout mechanism to monitor EOUT validation by separate channel.
   d) Following scenarios shall be considered as valid EOUT reactions:
      Validation is performed while EOUT is asserted
      Timeout occurs but no validation and EOUT is still asserted.
3. In case external device (which evaluates EOUT) cannot communicate with the MCU, the following procedure could be used:
   a) Program the error out duration to a duration x (FCCU_DELTA_T[DELTA_T]).
   b) For faults requiring error out reaction, clear the fault after the pin has continued to be asserted for a longer duration (e.g. 2*duration x). This will artificially create a long pulse on EOUT.
   c) For faults which do not require error out reaction, clear the fault within duration x. This will artificially create a short pulse on EOUT.
   d) External device should ignore short pulse of duration x while recognizing longer pulses as valid reaction.
   e) While clearing the fault, the associated software shall check the pending faults.

1.21 ERR008054:PIT: DMA request stays asserted when initiated by PIT trigger, until PIT is reset

Description:

When a Periodic Interrupt Timer 0 (PIT0) channel trigger is used to initiate a Direct Memory Access (DMA) transfer, the DMA request does not negate at the end of the DMA transfer. The result is that if that DMA channel is re-enabled, a subsequent PIT-triggered DMA transfer will be initiated

Workaround:

Either do not use the PIT0 to initiate DMA transfers, or write software such that anytime a PIT0 channel trigger is used to initiate a DMA transfer, the PIT0 module is then reset after that DMA transfer completes, prior to re-enabling the DMA channel that was used for that transfer. The PIT0 module should be reset by setting the PIT_RTC_0 reset bit in the Peripheral Reset Register 0 in the Reset Generation Module.

Other timer systems, such as the System Timer Module (STM), can be used instead of the PIT to trigger the DMA.

1.22 ERR008056: LBIST: Flash must be idle during LBIST.

Description:

Logic Built-In Self-Test (LBIST) should only be performed on the flash partition while the flash is in the idle state. LBIST operations must not be initiated for the LBIST partition including flash while flash program or erase operations are in progress. This information was not included in the user documentation.

Workaround:

Flash must be in an idle state for LBIST. Ensure that flash is not performing program, erase, or flash user accessible test mode operations when LBIST is initiated.

1.23 ERR008526: LINFlexD: LIN or UART state may be incorrectly indicated by LINSR[LINS] bitfield.

Description:

The Local Interconnect Network (LIN) or Universal Asynchronous Receiver/Transmitter (UART) state is shown in the read only LIN state bits in the LIN Status Register (LINSR[LINS]). Whenever the LinFLEXD (in either LIN or UART mode) updates these state bits, there is a possibility that the wrong LIN or UART state is indicated for one Peripheral Bridge Clock (PBRIDGEx_CLK) cycle. If software is asynchronously polling the LIN or UART state bits, the state read by the software may be incorrect.

Workaround:

The incorrect state in the LINSR[LINS] bit-field is auto-corrected in the next PBRIDGEx_CLK cycle. Therefore, any software polling the LINSR[LINS] bit-field should read the bit-field twice and confirm that the back to back reads are the same value before taking further action based on the state.

1.24 ERR008561: LINFlexD: Corruption of Tx data in LIN mode with DMA feature enabled.

Description:

The LINFlexD module is driven by two different clocks. The transmit/reception logic is controlled by the module clock (LIN_CLK) and register accesses are controlled by the peripheral bus clock (PBRIDGEx_CLK). In LIN mode, the re-synchronization of the "Idle on bit error" between the two clocks may cause the Direct Memory Access (DMA) Finite State Machine inside the LINFlexD module to move to the idle state while a transmission is in process. This unwanted idle state transition could lead trigger a new DMA request, potentially overwriting the Buffer Identifier Register (BIDR) and the Buffer Data Registers (BDRL and BDRM).

Workaround:

Do not enable the "Idle on bit error" of LIN Control Register 2 (lLINCR2[IOBE] = 0). Instead of using the “Idle on bit error", use the bit error interrupt of LIN Interrupt Enable Register (LINIER[BEIE] = 1) to trigger an Interrupt service routine and force the LIN into idle mode through software if needed.

1.25 ERR008573: LINFlexD: Pre-mature header/response timeout in LIN mode.

Description:

The LINFlexD instance is driven by two different clocks. The transmit/reception logic is controlled by the module clock (LIN_CLK) and register accesses are controlled by the peripheral bus clock (PBRIDGEx_CLK). The PBRIDGEx_CLK is used to control timer logic during header/response reception, while the LIN_CLK is used to control the header/response reception. In LIN mode, the resynchronization between the two clocks may cause the pre-mature setting of the OCF bit (Output Compare Flag) of LIN Error Status Register (LINESR).

Depending on LIN Timeout, Control and Status Register, IOT = Idle On TimeOut (LINTCSR[IOT]) settings, effects of the pre-mature setting may be:

- If LINTCSR[IOT] is set , termination of the data reception
- If LINTCSR[IOT] is not set, generation of spurious timeout event, though the reception will continue.

Workaround:

Always configure LINTCSR[IOT] as ‘0'. In case of a time out event, the spurious event can be detected by comparing the LINOCR[OC1] and LINOCR[OC2] values with the timer value in LINTCSR[CNT]. The difference should not be more than one bit time period.

1.26 ERR008602: LINFlexD: Tx through DMA can be re-triggered after abort in LIN/UART modes or can prematurely end on the event of bit error with LINCR2[IOBE] bit being set in LIN mode.

Description:

The LINFlexD module is driven by two different clocks. The transmit/reception logic is controlled by the module clock (LIN_CLK) and register accesses are controlled by the peripheral bus clock (PBRIDGEx_CLK). Due to possible synchronization issue between the two clock domains, there is a possibility that DMA transmission get stuck due to DMA Finite State Machine doesn't go into idle. This may occur in one of the following conditions:

- If an abort request is triggered (LINCR2[ABRQ]=1) in LIN or UART modes
- If idle on bit error feature is enabled (LINCR2[IOBE]=1) in LIN mode, and a bit error occurs.

DMA state machine will not generate any transaction, waiting for data transmission flag LINSR[DTF] to be set which will never occur.

Workaround:

If DMA is used:

- Bit error interrupt should be enabled through LINEIER[BEIE]. When a bit error interrupt is triggered, the interrupt service routine must either reset the DMA Tx channel enable (DMATXE) or the DMA Rx channel enable (DMARXE) registers
- If an abort is requested (LINCR2[ABRQ]=1) in LIN/UART mode, either reset DMATXE/DMARXE of LINFlexD after writing LINCR2 [ABRQ].

1.27 ERR008627: LINFlexD: LINFLEX clocks frequencies should verify relation 2/3*LINCLK≥PBRIDGEx_CLK>1/3*LINCLK.

Description:

The LINFlexD module is driven by two different clocks. The transmit/reception logic is controlled by the module clock (LIN_CLK) and register accesses are controlled by the peripheral bus clock (PBRIDGEx_CLK).

Reference manual is wrongly reporting the following relation between the two clocks frequencies:

LINCLK>PBRIDGEx_CLK≥1/3*LINCLK

whereas correct relation is as follow:

2/3*LINCLK≥PBRIDGEx_CLK>1/3*LINCLK

Workaround:

Use the correct relation between clocks frequencies when defining LINFlexD module clocks:

2/3*LINCLK≥PBRIDGEx_CLK>1/3*LINCLK

1.28 ERR008731: LINFlexD: Corruption of Received Rx data in UART mode.

Description:

The LINFlexD module is driven by two different clocks. The transmit/reception logic is controlled by the module clock (LIN_CLK) and register accesses are controlled by the peripheral bus clock (PBRIDGEx_CLK).

In the Universal Asynchronous Receive/Transmit (UART) Mode, the resynchronization of the access of the Buffer Data (BDRL and BDRM) registers may reset the internal counter which generates the baud sampling signal to receive the incoming bits. This will occur only if LIN Integer Baud Rate (LINIBRR) register is greater than 1. The data being received may not be correctly sampled:

- Sampling point will be anticipated by maximum of 1 bit period
- The final data may be shifted by one bit in the data BDRM register.

Workaround:

When using the LINFlex module in UART mode:

- If possible, LINIBRR should be configured with value ‘1’. This is possible in cases where the baud rate has ratio of 4,5,6,8,16 with respect to clock LIN_CLK. For example with LIN_CLK at 80MHz this allows 20M baud down to 5M baud reception rate.
- If the baud rate is not compatible with a LINIBRR value of ‘1’
  - In UART full duplex mode, application software must manage systematic detection of the failure and on detection, request that the receive data be resent. See possible implementation below
  - In UART half-duplex mode (receive mode only), systematic detection can be implemented as above. Alternatively, software or DMA should ensure that BDRM is read before the start of the next frame reception. Reception information is available through interrupt, DMA request, or the UART status register (UARTSR). This information is available 5 times the LINIBRR[IBR] period of the LIN_CLK before the completion of the STOP bit reception. Further delay may be available depending on delay in between frame reception.

Systematic detection can be performed by using a '0' logic parity bit and enable the generation of the Framing Error Interrupt. This is done by programming UARTCR[PCE] = ‘1’, UARTCR[PC1] = ‘1’, UARTCR[PC0] = ‘0’, and LINIER[FEIE] = ‘1’.

Depending on the timing of the possible glitch occurrence, detection will be indicated by:

- Either the setting of UARTSR[FE] bit, in other words, a Framing Error which generates a Framing Error Interrupt,
- Or the setting of the noise flag (UARTSR[NF]). Note: No interrupt is generated in this case.

1.29 ERR008915: SARADC: wrong behavior when aborting the conversion of a chain.

Description:

An ongoing conversion of the Successive Approximation Analog to Digital Converter (SARADC) can be aborted by setting the Abort Conversion (ABORT) bit of the Main Configuration Register (MCR).

During a conversion of chain, if the ABORT bit is set during last cycles of the conversion evaluation phase or first cycles of the sampling phase, the Internal Channel Data Register (ICDRn) of the next channel may be corrupted and the corresponding end of conversion interrupt pending bit (EOC_CHn) of the Internal Channel Interrupt Pending Register (ICIPR) might be wrongly set.

For instance, assuming a normal chain conversion of two channels: channel x (CHx) and channel y (CHy), if the ABORT bit is set during the last cycle of the evaluation phase of CHx, then data of aborted conversion CHx may be written in the ICDR register corresponding to CHy and ICIPR[EOC_CH] bit corresponding to CHy might be wrongly set.

Workaround:

When abort of an SARADC conversion is required, the software must:

1. Poll the SARADC status (ADCSTATUS) bit field of the Main Status Register (MSR), until it is in sample phase (0b100);
2. Wait for SARADC to start the conversion phase (ADCSTATUS=0b110);
3. Issue an abort command by setting the MCR[ABORT] bit.

1.30 e1040PS: ADCSAR: Wrong ADC data values after ADC comes out of powerdown exit.

Description:

SARADC gives wrong values if any ADC channel is converted just after powerdown exit.

Workaround:

A delay is needed to wait that the SARADC is in idle mode when its powerdown bit is negated before starting any ADC channel conversion.

1.31 e1045PS: PMC_DIG: Interrupt reaction to temperature flag does not work properly.

Description:

The reaction to temperature sensor flags is either a reset event (destructive or functional), an interrupt or a FCCU event.

All reactions work fine except the interrupt reaction which is triggered on falling edge of flag in place of the rising edge.

Workaround:

Instead of using the interrupt directly from PMC_Dig, it is possible to configure the FCCU event and program the reaction as an interrupt.

1.32 e1439PS: eDMA: Transfer corruption possible when trace is enabled.

Description:

When trace is enabled, DMA transfer can get corrupted in the following cases:

- DMA operating in burst mode
- DMA operating in scatter-gather mode with BWC bit=0

It also results in incorrect trace information.

Workaround:

Trace should not be enabled with DMA, when scatter-gather mode or burst mode (BWC > 0) is used.

1.33 e1440PS: FCCU: Spurious fault set when debugger connected in lockstep mode.

Description:

When the debugger is connected to device, lockstep RCCUs compares some debug related signals which result in setting the following FCCU faults:

- Channel 10: CPU_NON_AHB
- Channel 11: CPU_AHB
- Channel 51: LOCKSTEP Disable

Workaround:

Ignore these faults when operating in lockstep during debug mode.

1.34 e1566PS: LBIST: ADC is not converting correctly after LBIST RUN.

Description:

When BIST is run reset value of some TEST registers are getting changed. Due to this ADC values (for both ADC_0 and ADC_B) are lower than the expected value.

Workaround:

After LBIST run and before ADC initialization, restore RESET value of non-user TEST registers as default by SW.

Below the listing of the SW statements to execute to restore default configuration:

```
* ((vuint32_t *) 0xFFE38000) = 0x03020100;
* ((vuint32_t *) 0xFFE38004) = 0x07060504;
* ((vuint32_t *) 0xFFE38008) = 0x0B0A0908;
* ((vuint32_t *) 0xFFE3800C) = 0x0F0E0D0C;
* ((vuint32_t *) 0xFFE38010) = 0x13121110;
* ((vuint32_t *) 0xFFE38014) = 0x17161514;
* ((vuint32_t *) 0xFFE38018) = 0x1B1A1918;
* ((vuint32_t *) 0xFFE3801C) = 0x1F1E1D1C;
* ((vuint32_t *) 0xFFE38020) = 0x23222120;
* ((vuint32_t *) 0xFFE38024) = 0x27262524;
* ((vuint32_t *) 0xFFE38028) = 0x2B2A2928;
* ((vuint32_t *) 0xFFE3802C) = 0x2F2E2D2C;
* ((vuint32_t *) 0xFFE38030) = 0x33323130;
* ((vuint32_t *) 0xFFE38034) = 0x37363534;
* ((vuint32_t *) 0xFFE38038) = 0x3B3A3938;
* ((vuint32_t *) 0xFFE3803C) = 0x3F3E3D3C;

* ((vuint32_t *) 0xFFE04000) = 0x03020100;
* ((vuint32_t *) 0xFFE04004) = 0x07060504;
* ((vuint32_t *) 0xFFE04008) = 0x0B0A0908;
* ((vuint32_t *) 0xFFE0400C) = 0x0F0E0D0C;
* ((vuint32_t *) 0xFFE04010) = 0x13121110;
* ((vuint32_t *) 0xFFE04014) = 0x17161514;
* ((vuint32_t *) 0xFFE04018) = 0x1B1A1918;
* ((vuint32_t *) 0xFFE0401C) = 0x1F1E1D1C;
* ((vuint32_t *) 0xFFE04020) = 0x23222120;
* ((vuint32_t *) 0xFFE04024) = 0x27262524;
* ((vuint32_t *) 0xFFE04028) = 0x2B2A2928;
* ((vuint32_t *) 0xFFE0402C) = 0x2F2E2D2C;
* ((vuint32_t *) 0xFFE04030) = 0x33323130;
* ((vuint32_t *) 0xFFE04034) = 0x37363534;
```

```
* ((vuint32_t *) 0xFFE04038) = 0x3B3A3938;
* ((vuint32_t *) 0xFFE0403C) = 0x3F3E3D3C;
```

1.35 e1681PS: eDMA: Unexpected Non-correctable EDC error is reported to FCCU on DMA transfers of less than 32-bit destination size.

Description:

Direct Memory Access (DMA) transfers, where the transfer destination is mapped to peripheral space, and the destination transfer size is less than 32-bits, may result in the reporting of a non-correctable Error Detection (EDC) event to the FCCU.

Workaround:

When transferring data to the peripheral memory space, program the transfer control descriptors of the DMA to set the destination size to 32-bit.

1.36 DAN-0043391: PLL configuration needs to be changed for MBIST run.

Description:

- In device's RM, STCU_PLL_CFG (PLL configuration) value is provided for configuring PLL at 73.3MHz for Offline MBIST run.
- A too high PLL frequency might lead to the corruption of the logic of Peripheral Clock Divider (AIPS0), which wakes up – after Self Test sequence completion – with a wrong setting (dividing AIPS0 clock by 1 instead of 2).
- Recommended value for PLL frequency for Offline MBIST run is 40MHz: maximum allowed value is 60MHz.

Workaround:

- STCU_PLL_CFG configuration has to be changed from 0x06020038 00080010 to 0x0C02003C 00080010 to configure PLL at 40MHz or lower.
- The duration of MBIST run increases consequently, according to the selected algorithm. Total Offline Self Test sequence contains anyway also LBIST, which is not affected by PLL frequency, since it is run using on-chip RC oscillator.

Note: Running Offline MBIST at a lower speed with respect to the maximum speed does not affect ASIL D targets for the hardware architectural metrics of the device. From Safety Analysis (FMEDA) point of view, Error Correction Code (ECC) is the primary Safety Mechanism against memory faults: on-field MBIST (Offline or Online) must be run only to prevent accumulation of faults, which would negatively impact diagnostic coverage of the primary safety mechanism (ECC). ECC does not cover multiple bit faults, it just corrects single bit errors and detects double bit errors.

Appendix A Further information

A.1 Reference document

1. *SPC570Sx 32-bit Power Architecture® microcontroller for automotive ASILD applications* (RM0349, Doc ID 024507)
2. *32-bit Power Architecture® microcontroller for automotive ASILD applications* (SPC570S40x, SPC570S50x datasheet, Doc ID 024492)

A.2 Acronyms

Table 2. Acronyms

Acronym	Name
MC_ME	Mode Entry Module
MC_CGM	Clock Generation Module
DR	Data Register
FCCU	Fault Collection and Control Unit
MCR	Module Configuration Register
SSCM	System Status and Configuration Module
SARADAC	Successive Approximation Register Analog to Digital Converter
SPU	Sequence Processing Unit
NAR	Nexus Aurora Router
IRC	Internal RC oscillator
MEMU	Memory Error Management Unit
NMI	Non-maskable Interrupt
PGL	Programming Group Lock
PIT	Periodic Interrupt Timer
XOSC	External oscillator
DMA	Direct Memory Access
LBIST	Logic Built-In Self-Test
LIN	Local Interconnect Network
UART	Universal Asynchronous Receiver/Transmitter
DMA	Direct Memory Access
BIDR	Buffer Identifier Register
EDC	Error Detection
ICIPR	Internal Channel Interrupt Pending Register

Revision history

Table 3. Document revision history

Date	Revision	Changes
06-Aug-2014	1	Initial release.
16-Dec-2014	2	Added following functional problems: – ERR008526 – ERR008561 – ERR008573 – ERR008602 – ERR008627 – ERR008731
30-Mar-2015	3	Added following functional problems: – e1244PS – e1254PS Removed the following functional problems: – ERR006850 – ERR007126 – ERR006904
29-May-2015	4	Added following functional problems: – e1040PS – e1045PS – e1071PS – e1439PS – e1440PS – e1566PS – e1681PS – ERR008915
24-Sep-2015	5	Removed the following functional problems: – e1071PS – e1244PS – e1254PS Removed Appendix B: Defects across silicon version.
10-May-2017	6	Added: – DAN-0043391
10-May-2017	7	Typo error