SPC560B4x/50 - SPC560C4x/50 Errata sheet

32-bit MCU family built on the Power Architecture™ embedded category for automotive body electronics applications

Introduction

This errata sheet describes all the known functional and electrical limitations of the SPC560B4x/50 - SPC560C4x/50 microcontroller family.

All the topics covered in this document refer to *RM0017* rev 3 and *SPC560Bx and SPC560Cx datasheet* rev 4 (see *Section A.1: Reference document*).

The device identification can be read by software using the MIDR registers:

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

Package device marking mask identifier: B.

Die mask ID: FB50X20B

This errata sheet applies to SPC560B4x/50 - SPC560C4x/50 devices in accordance with *Table 1*.

Table 1. Device summary

256 Kbyte Flash		384 Kbyte Flash		512 Kbyte Flash	
SPC560B40L5	-	SPC560B44L5	-	SPC560B50L5	-
SPC560B40L3	SPC560C40L3	SPC560B44L3	SPC560C44L3	SPC560B50L3	SPC560C50L3
-	-	-	-	SPC560B50B2	-

Contents

1 Functional problems

1.1 e3269 PS: Serial boot and censorship - RCHW read access

Description:

In a secured device starting in serial boot mode, it is possible to read the content of the four Flash locations where the RCHW (reset configuration half-word) can be stored.
For example, if the RCHW is stored at address 0x00000000, the reads at address 0x00000000, 0x00000004, 0x00000008 and 0x0000000C returns a correct value. Any other Flash address cannot be accessed.

Workaround:

No workaround

1.2 e3270 PS: Serial boot in uncensored device - SRAM access allowed

Description:

In uncensored devices, it is possible to download code via LINFlex or FlexCAN (serial boot mode) into internal SRAM even if the 64-bit private password stored in the Flash and provided during the boot sequence is an illegal password.

Workaround:

No workaround

1.3 e3320 PS: VREG register is mirrored at consecutive addresses

Description:

The VREG control register (VREG_CTL) is mapped at address C3FE8080. It is also mirrored at the following addresses: C3FE8080 C3FE80A0 C3FE80C0 C3FE80E0 access to any of the above addresses is considered a valid access and no transfer error is generated.

Workaround:

No workaround

1.4 e3498 PS: STANDBY exit time above specification

Description:

When boot from RAM at STANDBY exit is selected the latency between the STANDBY wake-up event and the release of the device reset is 80 µs. In the data sheet this time is specified at 50 µs.

Workaround:

No workaround

1.5 e4019 PS: Wake-up interrupt may be generated without any recessive to dominant transition on FlexCAN pad

Description:

Wake-up interrupt may be generated without any recessive to dominant transition on FlexCAN pad in case following conditions occur:

- FlexCAN is configured for communication.
- WAK_MSK as well as SLF_WAK bits of MCR register are set.
- Apply this write access sequence to MDIS bit: set, then reset. In this configuration, a wake-up interrupt is wrongly generated when MDIS bit is clear.

Workaround:

Always program SLF_WAK and WAK_MSK bits to ‘0’.

1.6 e4264 PS: Low power transition - debugger handshake mechanism failure

Description:

The handshake mechanism used by the device to communicate entry/exit of low-power mode with the debugger is leading to device stall. This is only affecting low-power mode transition while in debug mode. In order to recover it is necessary to generate a destructive reset.

Workaround:

Do not use low power transition handshake mechanism with the debugger.

1.7 e4554 PS: RGM: Register RGM_FES bit PLL_FAIL is set in case of LVD2.7

Description:

When the PLL is used and a low voltage event is detected on LVD2.7 at the register RGM_FES the bit PLL_FAIL may be set.

Workaround:

No workaround

1.8 e4727 PS: STANDBY entry prevented in debug mode

Description:

While running the application with the debugger it is not possible to enter STANDBY.

Workaround:

No workaround

1.9 e5363 PS: RAM: No data abort exception generated above 0x4000BFFF

Description:

System RAM reserved space extends from 0x40000000 to 0x400FFFFF. On this device, memory is implemented from 0x40000000 to 0x4000BFFF. No data abort error is generated when accessing 0x4000C000-0x400FFFFF address range.

Workaround:

Use memory protection unit (MPU) when specific control is to be implemented for out of memory accesses.

1.10 e5911 PS: TDO pin floating in STANDBY mode

Description:

The TDO pad has been moved into the STANDBY domain in order to allow low-power debug handshaking in STANDBY mode. However, no pull-resistor is active on the TDO pad while in STANDBY mode. At this time the pad is configured as an input. When no debugger is connected the TDO pad is floating causing additional current consumption.

Workaround:

To avoid the extra consumption TDO must be connected. An external pull-up resistor in the range of 47 to 100 kΩ should be added between the TDO pin and VDD. Only in case the TDO pin is used as application pin and a pull-up cannot be used then a pull-down resistor with the same value should be used between TDO pin and GND instead.

1.11 e6943 PS: Incorrect watchdog period value on reset exit

Description:

Watchdog initial period may vary in the range of 7 to 13 ms with respect to the typical 10 ms value.

Workaround:

Ensure to reload watchdog and update watchdog counter value at the beginning of application, latest 7 ms after the internal reset has been released.

1.12 e4783 PS: SWT: Watchdog is disabled during BAM execution

Description:

The watchdog is disabled at the start of BAM execution. In the case of an unexpected issue during BAM execution the CPU may be stalled and it is necessary to generate an external reset to recover.

Workaround:

No workaround

1.13 e14593 IPG: FlexCAN: Global Masks misalignment

Description:

Convention: MSB = 0.

During CAN messages reception by FlexCAN, the RXGMASK (RX global mask) is used as acceptance mask for most of the RX message buffers (MB). When the FIFO enable bit in the FlexCAN module configuration register (CANx_MCR[FEN], bit 2) is set, the RXGMASK also applies to most of the elements of the ID filter table. However there is a misalignment between the position of the ID field in the RX MB and in RXIDA, RXIDB and RXIDC fields of the ID tables. In fact RXIDA filter in the ID tables is shifted one bit to the left from RX MBs ID position as shown below:

- RX MB ID = bits 3 to 31 of ID word corresponding to message ID bits 0 to 28
- RXIDA = bits 2 to 30 of ID table corresponding to message ID bits 0 to 28.

Note that the mask bits one-to-one correspondence occurs with the filters bits, not with the incoming message ID bits.
This leads the RXGMASK to affect RX MB and RX FIFO filtering in different ways.

For example, if the user intends to mask out the bit 24 of the ID filter of message buffers then the RXGMASK is configured as 0xffff_ffef. As result, bit 24 of the ID field of the incoming message is ignored during filtering process for message buffers. This very same configuration of RXGMASK would lead bit 24 of RXIDA to be "do not care" and thus bit 25 of the ID field of the incoming message would be ignored during filtering process for RX FIFO.

Similarly, both RXIDB and RXIDC filters have multiple misalignments with regards to position of ID field in RX MBs, which can lead to erroneous masking during filtering process for either RX FIFO or MBs.

RX14MASK (RX 14 mask) and RX15MASK (RX 15 mask) have the same structure as the RXGMASK. This includes the misalignment problem between the position of the ID field in the RX MBs and in RXIDA, RXIDB and RXIDC fields of the ID tables.

Workaround:

Therefore, it is recommended that one of the following actions be taken to avoid problems:

- Do not enable the RXFIFO. If CANx_MCR[FEN] = 0 then the RX FIFO is disabled and thus the masks RXGMASK, RX14MASK and RX15MASK do not affect it.
- Enable rx individual mask registers. If the backwards compatibility configuration bit in the FlexCAN module configuration register (CANx_MCR[BCC], bit 15) is set then the RX individual mask registers (RXIMR[0:63]) are enabled and thus the masks RXGMASK, RX14MASK and RX15MASK are not used.
- Do not use masks RXGMASK, RX14MASK and RX15MASK (that is, let them in reset value which is 0xffff_ffff) when CANx_MCR[FEN] = 1 and CANx_MCR[BCC] = 0. In this case, filtering processes for both RX MBs and RX FIFO are not affected by those masks.
- Do not configure any MB as RX (that is, let all MBs as either TX or inactive) when CANx_MCR[FEN] = 1 and CANx_MCR[BCC] = 0. In this case, the masks RXGMASK, RX14MASK and RX15MASK can be used to affect ID tables without affecting filtering process for RX MBs.

1.14 e2835 PS: 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 that this failed clearing has no impact on further flag clearing requests.

Workaround:

No workaround for all reset sources except SW reset. Note that 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.

1.15 e3953 PS: MC_ME: In RUNx mode after STOP0 exit system RAM can be accessed before it is ready

Description:

In the case in which:

- ME_STOP0_MC[FXOSC] is enabled, ME_STOP0_MC[FIRC] is disabled
- ME_RUNx_MC[FIRC] is enabled, ME_RUNx_MC[SYSCLK] = FXOSC or FXOSC_DIV.

At the transition of STOP0 to RUNx, the RUNx mode can be entered before the system RAM is ready. If the application software accesses the RAM during this time the RAM value can not be guaranteed.

Workaround:

There are two possible workarounds:

1. Do not disable the IRC if the system clock source is not disabled. The XOSC draws a lot more current than the IRC, so there should be no noticeable increase in the STOP0 mode power consumption.
2. Have the software check that the mode transition has completed via the ME_GS register before accessing the system RAM.

1.16 e4107 PS: MC_CGM and MC_PCU: A data storage exception is not generated on an access to MC_CGM or MC_PCU when the respective peripheral is disabled at MC_ME

Description:

If a peripheral with registers mapped to MC_CGM or MC_PCU address spaces is disabled via the MC_ME any write accesses to this peripheral is ignored without producing a data storage exception.

Workaround:

For any mode other than a low-power mode do not disable any peripheral which is mapped to MC_CGM or MC_PCU.

1.17 e4163 PS: MC_ME: ME_PSx registers may show ‘1’ in a reserved bit field

Description:

Some bits in the ME_PSx registers which are defined to always return the value '0' may instead return the value ‘1’.

Workaround:

It is recommended as a general rule that the user should always ignore the read value of reserved bit fields.

1.18 e4565 PS : MC_RGM: MCU enters BAM static state if reset during STANDBY0 mode

Description:

If the device is in STANDBY0 mode, and it is configured to re-boot from RAM (RGM_STDBY[BOOT_FROM_BKP_RAM] = 1) while keeping the Code Flash in power-down or low-power mode (ME_DRUN_MC[CFLAON] != “11"), an external reset causes the device to enter the BAM static state where it remains inoperable until another reset occurs.

Workaround:

Generate the external reset by asserting the RESETB pin twice. The two pulses must be separated by at least 200 µs.

1.19 e5095 PS: 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:

No workaround.

1.20 e5955 PS: MC_RGM: External reset not asserted if STANDBY0 is exited due to external reset event

Description:

If the device is in STANDBY0 mode and an external reset occurs, the MC_RGM may not assert the external reset for the duration of the reset sequence even if RGM_FBRE.BE_EXR = '0'. This incorrect behavior occurs only if the system releases the external reset before the end of reset sequence PHASE1.

Workaround:

Ensure that the system asserts the external reset for at least the maximum duration of reset sequence PHASE1.

1.21 e7216 IPG: NPC: MCKO_DIV can be set to 0x0 (1X MCKO)

Description:

The Nexus port controller port configuration register MCKO divider bits (NPC_PCR[MCKO_DIV]) can be set to 0b000 to select a 1X clock rate as the Nexus auxiliary output port frequency for the MCKO and MDO pins.

Note: Depending on the system frequency, this may force the MCKO and MDO pins to switch at a frequency higher than can be supported by the pins. This maximum frequency is specified in the device electrical specification of the Nexus MCKO and MDO pins.

Workaround:

Insure that the maximum operating frequency of the MDO and MCKO pins is not violated when setting the NPC_PCR[MCKO_DIV] values.

Note: Tools may not support 1X mode. Check with your tool vendor.

1.22 e7810 IPG: NPC: Automatic clock gating does not work for MCKO_DIV = 8

Description:

If the Nexus clock divider (NPC_PCR[MCKO_DIV]) in the Nexus port controller port configuration register is set to 8 and the Nexus clock gating control (NPC_PCR[MCKO_GT]) is enabled, the Nexus clock (MCKO) is disabled prior to the completion of transmission of the Nexus message data.

Workaround:

Do not enable the automatic clock gating mode when the Nexus clock divider is set to 8. If Nexus clock gating is required, use a divide value of 1, 2 or 4 (set NPC_PCR[MCKO_GT] = 0b1 and NPC_PCR[MCKO_DIV] = 0b000, 0b001, or 0b011). However, MCKO must be kept below the maximum Nexus clock rate as defined in the device data sheet. If divide by 8 clock divider is required, then do not enable clock gating (set NPC_PCR[MCKO_GT] = 0b0 and NPC_PCR[MCKO_DIV] = 0b111).

1.23 e3399 PS: Flash: RWW-error during stall-while-write

Description:

If stall/abort-while-write is enabled and an erase operation is started on one sector while fetching code from another then the following sequence is executed:

- CPU is stalled when Flash is unavailable
- PEG flag set (stall case) or reset (abort case)
- Interrupt triggered if enabled. However, in all cases also the RWE flag is set, indicating a RWW error.

Workaround:

If stall/abort-while-write is used then application software should ignore the setting of the RWE flag. The RWE flag should be cleared after each HV operation. If stall/abort-while-write is not used the application software should handle RWE error.

1.24 e6309 PS : Debugging functionality could be lost when unsecuring a secured device.

Description:

Providing the backdoor password via JTAG or via serial boot would unsecure the device, but on some devices may leave the Nexus interface and potentially the CPUs in an undetermined state. Normal operation without a debugger is unaffected, debugging unsecured devices is also unaffected.

Workaround:

- A second connection attempt may be successful.
- Boot in serial mode (using the Flash password), then execute code which unsecures the device (the JTAG interface needs to be inactive while the unsecure happens).
- Implement a separate backdoor in application software. Once the software detects the custom backdoor sequence it can unlock the device via Flash write.
- Leave device unsecured for debugging.

1.25 e4455 PS: ADC: conversion chain failing after abort chain

Description:

During a chain conversion while the ADC is in scan mode when ADC_MCR[ABORTCHAIN] is asserted the current chain is aborted as expected. However, in the next scan the first conversion of the chain is performed twice and the last conversion of the chain is not performed.

Workaround:

When aborting a chain conversion enable ADC_MCR[ABORTCHAIN] and disable ADC_MCR[START]. ADC_MCR[START] can be enabled when the abort is complete.

1.26 e5402 PS: 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 continously sends 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.

1.27 e3826 PS: CAN sampler: CAN_RX metastability issue

Description:

CAN Sampler operation cannot be guaranteed in second frame mode (CR.Mode = 0).

Workaround:

CAN sampler operates reliably in first frame mode (CR.Mode = 1) for the following 2 configurations:

1. First frame mode (CR.Mode = 1) selected and IRC ‘ON’ in STANDBY.
2. First frame mode (CR.Mode = 1) selected and IRC ‘OFF’ in STANDBY with CAN baud rate less than 75 kbps.

1.28 e5007 PS: Flash - Programming a value over another value may not indicate an error if no bits are being programmed (1 --> 0)

Description:

The Flash programming/erase good status flag (PEG) in the Flash module configuration register (Flash_MCR) may not indicate a failure to program if an attempt is made to program bits high (0b1) that are currently programmed low (0b0) if no bits in the 64-bit word or the 8- bit error correction code (ECC) are being programmed low (to a 0b0).

For example, PEG doesn’t indicate a failure (FLASH_MCR[PEG] = 1) if an attempt is made to program the value 0x5555_FFFF over a Flash location that already was programmed to 0x5555_5555. The Flash contents are not changed (stays 0x5555_5555) even though it should fail since there were no bits actually selected to be programmed (cleared).

However, PEG correctly indicates a failure (FLASH_MCR[PEG] = 0) if an attempt is made to program the value of 0x5555_00FF over a Flash location that was already programmed to 0x5555_5555. The resulting Flash contents are 0x5555_0055.

Note: Reprogramming bits from a programmed state (0b0) to an erased state (0b1) is not supported on Flash *memory technology. A bit can only be set (0b1) by erasing the full* Flash *block.*

Workaround:

Do not expect an error to be flagged if there are no bits being cleared (0b0) in the word being programmed when programming a new value into a location in the Flash that has already been programmed with a previous (not still erased) value.

1.29 e5008 PS: Flash - Array integrity check start at address 0x00000010

Description:

Array integrity check skips the first 2 double words (addresses 0x000000 and 0x000008), starting from address 0x00000010

Workaround:

The customer application should take care of checking the first two double words.

1.30 e5034 PS: Flash - SLL and HBL not properly initialized

Description:

During boot, the SLL register is not loaded with NVSLL value. The HBL register is not loaded with NVHBL value.

Workaround:

The application must initialize SLL and HBL with the desired values.

1.31 e5060 PS : Flash - Check of ECC errors of NVBIU2 missing

Description:

The check of eventual double ECC errors present in NVBIU2 location is missing. In case of ECC double error the volatile register BIU2 is anyway loaded with the data read from NVBIU2 that can therefore contain invalid bits.

Workaround:

None

1.32 e5673 PS: Flash - Margin mode usage forbidden

Description:

The margin mode operation does not work correctly. The usage of this operation is forbidden.

Workaround:

Verification of the Flash programming can be replaced by the usage of the array integrity.

1.33 e6384 PS: Flash ECC: Erroneous update of the address in case of multiple errors

Description:

Whenever there is a sequence of 3 or more events affecting ADR (ECC single or double bit errors or RWW error) and both the following conditions apply:

- The priorities are ordered in such a way that only the first event should update ADR.
- The last event although it does not update ADR sets MCR[RWE] or MCR[SBC]. For this case the ADR is wrongly updated with the address related to one of the intervening events. For example, If a sequence of two double-bit ECC errors is followed by a single-bit correction without clearing MCR[EER], then the value found in ADR after the single-bit correction event is the one related to the second double-bit error (instead of the first one, as specified).

Workaround:

Always process Flash ECC errors as soon as they are detected. Clear MCR[RWE] at the end of each Flash operation (program, erase, array integrity check, etc.).

1.34 e6583 PS : Diode path between VDD_LV and VDD_HV

Description:

The design implements diodes between VDD_LV and VDD_HV. This leads to current flowing from VDD_LV to VDD_HV if power-up and power-down sequences are not handled correctly. VDD_LV to VDD_HV current may stress the device and should be avoided when possible.

Workaround:

Insert a schottky diode between VDD_LV and VDD_HV.

1.35 e4781 PS: LINFlex: Unexpected LIN timeout in slave mode

Description:

If the LINFlex is configured in LIN slave mode an unexpected LIN time-out event (LINESR[OCF]) may occur during LIN break reception.

Workaround:

No workaround.

1.36 e4897 PS: LINFlex: BDRL/BDRM cannot be accessed as byte or half-word

Description:

LINFlex data buffers (BDRL/BDRM) cannot be accessed as byte or half-word. Accessing BDRL/BDRM in byte/half-word mode leads to incorrect data writing/reading.

Workaround:

Access BDRL/BDRM registers as word only.

1.37 e1685 PS: 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.

1.38 e3630 PS: FMPLL: Do not poll flag FMPLL_CR[PLL_FAIL]

Description:

For the case when the FMPLL is indicating loss of lock the flag FMPLL_CR[PLL_FAIL] is unpredictable.

Workaround:

To avoid reading an incorrect value of FMPLL_CR[PLL_FAIL] only read this flag inside the FMPLL interrupt service routine (ISR). The ISR indicates the flag has been set correctly and at this point the flag can be cleared. Do not poll flag FMPLL_CR[PLL_FAIL] at any other point in the application software.

Appendix A Further information

A.1 Reference document

1. *SPC560B4x, SPC560B5x, SPC560C4x, SPC560C5x 32-bit MCU family built on the embedded Power Architecture™* (RM0017, Doc ID 14629).
2. *32-bit MCU family built on the* Power Architecture™ embedded category for automotive body electronics applications (SPC560B4x, SPC560B5x, SPC560C4x, SPC560C5x datasheet, Doc ID 14619).

A.2 Acronyms

Table 2. Acronyms

Acronym	Name
ADC	Analog-to-digital converter
BAM	Boot assist mode
FIFO	First in first out
ISR	Interrupt service routine
LVD	Low voltage detector
MB	Message buffer
MPU	Memory protection unit
RCHW	Reset configuration half-word
RXGMASK	RX global mask
RXIMR	RX individual mask register
TDO	Test data output

Revision history

Table 3. Document revision history

Date	Revision	Changes
09-Jul-2009	1	Initial release.
10-Nov-2009	2	Deleted e2348 Deleted e4991 Deleted e5006 Added *Section 1.5: e4019 PS: Wake-up interrupt may be generated without any recessive to dominant transition on FlexCAN pad* Added *Section 1.9: e5363 PS: RAM: No data abort exception generated above 0x4000BFFF* Added *Section 1.10: e5911 PS: TDO pin floating in STANDBY mode* Added *Section 1.11: e6943 PS: Incorrect watchdog period value on reset exit* Added *Section 1.20: e5955 PS: MC_RGM: External reset not asserted if STANDBY0 is exited due to external reset event* Added *Section 1.21: e7216 IPG: NPC: MCKO_DIV can be set to 0x0 (1X MCKO)* Added *Section 1.24: e6309 PS : Debugging functionality could be lost when unsecuring a secured device.* Added *Section 1.28: e5007 PS: Flash - Programming a value over another value may not indicate an error if no bits are being programmed (1 --> 0)* Added *Section 1.31: e5060 PS : Flash - Check of ECC errors of NVBIU2 missing* Added *Section 1.32: e5673 PS: Flash - Margin mode usage forbidden* Added *Section 1.33: e6384 PS: Flash ECC: Erroneous update of the address in case of multiple errors* Added *Section 1.34: e6583 PS : Diode path between VDD_LV and VDD_HV* Added *Section 1.37: e1685 PS: FMPLL: FMPLL_CR[UNLOCK_ONCE] wrongly set*