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. 7 and *SPC560Bx and SPC560Cx datasheet* rev. 7 (see *Section A.1: Reference documents*).

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

- MAJOR_MASK[3:0]: 1
- MINOR_MASK[3:0]: 2

Package device marking mask identifier: X.

Die mask ID: FB50X22X

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

Table 1. Device summary

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

Contents

1 Functional problems

1.1 ERR002970: 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

1.2 ERR003012: 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

1.3 ERR003023: Device may hang up due to functional reset while using debug handshaking mechanism

Description:

When the debug session is ongoing and the low power handshaking mode is enabled, a functional reset may hang up the device, waiting for the debugger acknowledgement.

Workaround:

NPC_PCR[LP_DBG_EN] bit must be clear to ensure the reset sequence complete successfully.

1.4 ERR003064: 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.

1.5 ERR003156: Incorrect watchdog period value on reset exit

Description:

Watchdog initial period may vary in the range from 7 ms 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.

1.6 ERR003176: MC_ME: STANDBY mode cannot be entered if the FlexCAN peripheral is active

Description:

If any FlexCAN module is enabled at the MC_ME (ME_RUN_PCx/ME_PCTLx) and enabled at the FlexCAN module, (MCR.B.MDIS = 0) STANDBY0 mode is not entered, and the device remains in DRUN or RUNx mode.

Workaround:

The FlexCAN module must be frozen (FLEXCANx_MCR[FRZ] = 1) in DRUN or RUNx mode before entering STANDBY0 mode.

1.7 ERR003198: F_SOFT bit set on STANDBY exit through external reset

Description:

When device is under standby and external reset is triggered, the device exit reset and RGM_FES[F_EXR] is correctly set. RGM_FES[F_SOFT] is instead incorrectly set even if no software reset was requested by application.

Workaround:

No workaround

1.8 ERR003203: NMI unavailable in standby mode when code flash is off on standby exit

Description:

NMI pin cannot be configured to generate non-maskable interrupt (NMI) event to the core (WKPU_NCR[NDSS] = “00”) if the standby mode is configured as follows:

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

The above stated configuration may result in the following scenario:

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

Workaround:

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

After standby exit, the core boots and configures the INTC_IVPR, it may then re-configure WKPU_NCR[NDSS] to the appropriate configuration for enabling NMI, Critical Interrupt or Machine Check.

1.9 ERR003211: PA[1] pull-up enabled when NMI activated

Description:

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

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

Workaround:

No workaround

1.10 ERR003240: PB[10] configuration during standby

Description:

The PB[10] pin includes GPIO, wakeup and analog functionalities. When used as a wake-up pin, it must be driven either high or low (possibly using the internal pull-up resistor) during standby.

In case the pin is connected to an external component providing an analog signal, it is important to check that this external analog signal is either less than 0.2 × VDD_HV or greater than 0.8 × VDD_HV to avoid additional power consumption.

Workaround:

No workaround

1.11 ERR003304: PIT events cannot be used to trigger ADC conversions in case BCTU runs on divided system clock

Description:

If the BCTU operates on a divided system clock (i.e, MC_CGM.CGM_SC_DC2.DIV0 NOT EQUAL 0x0), events from the PIT, which is always clocked with the system clock, cannot be used to trigger ADC conversions.

The BCTU passes the trigger event, but the ADC channel information gets corrupted.

Workaround:

To run the BCTU at system frequency it is mandatory to write “MC_CGM.CGM_SC_DC2.DIV0” to 0x0 address.

1.12 ERR002958: MC_RGM: Clearing a flag in RGM_DES or RGM_FES register may be prevented by a reset

Description:

Clearing a flag in the 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 ongoing, the reset may interrupt the clearing mechanism leaving the flag set.

Note that this unsuccessfull clearing operation has no impact on further flag clearing requests.

Workaround:

No workaround for all reset sources except for 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 the flag clearing has completed by reading the RGM_xES register before the SW reset is requested.

1.13 ERR002995: MC_ME: In RUNx mode after STOP0 exit system RAM can be accessed before it is ready

Description:

For the case when:

- 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.

1.14 ERR002999: 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 the low-power mode does not disable any peripheral which is mapped to MC_CGM or MC_PCU.

1.15 ERR003001: 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.

1.16 ERR003049: MC_RGM: External reset not asserted if short reset enabled

Description:

For the case when the RGM_FBRE register enables the external reset for a specific reset source and the RGM_FESS register requests a short reset for the same reset source, the external reset is not asserted.

Workaround:

No workaround

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

Description:

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

Workaround:

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

1.18 ERR003086: 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 register 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 when the system releases the external reset before the end of the reset sequence PHASE1.

Workaround:

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

1.19 ERR003119: SWT interrupt does not cause STOP0 mode exit

Description:

While in STOP0 mode, if the SWT is configured to generate an interrupt and the system clock is disabled (SYSCLK field of ME_STOP0_MC register set to “1111”), an SWT interrupt event does not trigger an exit from STOP0 mode.

Other internal or external wake-up events (RTC, API, etc.) are not affected and trigger a STOP0 exit independently of the ME_STOP0_MC[SYSCLK] configuration.

Workaround:

If an SWT interrupt wakes up the device during STOP0 mode, the software may not disable the system clock (ME_STOP0_MC[SYSCLK] != 0xF).

1.20 ERR002161: NPC: Automatic clock gating does not work for MCKO_DIV = 8

Description:

If the Nexus clock divider (NPC_PCR[MCKO_DIV]) is set to divide-by-8 and the Nexus clock gating control (NPC_PCR[MCKO_GT]) is enabled, the Nexus clock (MCKO) is disabled prior to the completion of the Nexus message data transmission.

Workaround:

Do not enable the automatic clock gating mode when the Nexus clock divider is set to divide-by-8. If Nexus clock gating is required, use a divide-by-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 datasheet. If the 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).

1.21 ERR003010: 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.

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

Description:

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

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

- FXOSC oscillator failure
- SAFE mode request occurs as this mode switches off immediately the FXOSC (refer to ME_SAFE_MC register configuration in *RM0017, see Section A.1: Reference documents*)

Workaround:

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

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

1.23 ERR003066: ADC: ADC_MCR[CTUEN] cannot be reset if a CTU channel is enabled

Description:

While any CTU channels is enabled (CTU.EVTCFGRx.TM = 1), the CTU continously sends trigger requests to ADC.

If ADC_MCR[CTUEN] is reset, then sometime after setting the bit again, while a CTU channel has remained enabled, the ADC module may not service the trigger request from the CTU.

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.

1.24 ERR002992: CAN sampler: Second Frame mode not operable

Description:

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

Workaround:

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

1. First frame mode (CANSAMPLER_CR[Mode = 1]) selected and FIRC 'ON' in STANDBY
2. First frame mode (CANSAMPLER_CR[Mode = 1]) selected and FIRC 'OFF' in STANDBY with CAN baud rate less than 75 Kbps

1.25 ERR003114: Flash: Erroneous update of the ADR register in case of multiple ECC errors

Description:

An erroneous update of the address register (ADR) occurs whenever there is a sequence of three or more events affecting the 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.
- Although the last event does not update the ADR, it sets the read while write event error (RWE) or the ECC data correction (EDC) in the module configuration register (MCR).

For this case the ADR is wrongly updated with the address related to one of the intervening events.

For example, if a sequence of 2 double-bit ECC errors is followed by a single-bit correction without clearing the ECC event error flag (EER) in the MCR, 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.).

1.26 ERR003021: LINFlex: Unexpected LIN timeout in slave mode

Description:

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

Workaround:

No workaround. It is recommended to disable this functionality during LINFlex initialization by clearing LINTCSR[IOT] and LINIER[OCIE] bits, and to ignore timeout events.

1.27 ERR002883: FMPLL: FMPLL_CR[UNLOCK_ONCE] wrongly set

Description:

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

Workaround:

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

1.28 ERR002981: 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.

1.29 ERR003269: MC_ME: Peripheral clocks may get incorrectly disabled or enabled after entering debug mode

Description:

If ME_RUN_PCx, ME_LP_PCx, ME_PCTLx registers are changed to enable or disable a peripheral, and the device enters debug mode before a subsequent mode transition, the peripheral clock gets enabled or disabled according to the new configuration programmed. Also ME_PSx registers report incorrect status as the peripheral clock status is not expected to change on debug mode entry.

Workaround:

After modifying any of the ME_RUN_PCx, ME_LP_PCx, ME_PCTLx registers, request a mode change and wait for the mode change to be completed before entering debug mode in order to have consistent behaviour on peripheral clock control process and clock status reporting in the ME_PSx registers.

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

Description:

If STOP0 or HALT0 is configured with ME_[mode]_MC.MVRON = ‘0’, ME_[mode]_MC.FIRCON = ‘0’ and ME_[mode]_MC.SYSCLK = ‘0010/0011’ the Main VREG remains nevertheless enabled during the STOP0 mode if the previous RUN[0..3] mode is

configured with ME_RUN[0..3]_MC.FXOSCON = ‘1’. This results in increased current consumption of 500 µA than expected.

Workaround:

Before entering STOP0 or HALT0 mode with the following configuration - ME_[mode]_MC.MVRON = ‘0’, ME_[mode]_MC.FIRCON = ‘0’ and ME_[mode]_MC.SYSCLK = ‘0010/0011’ - ensure the RUN[0..3] mode switches off FXOSC - ME_RUN[0..3]_MC.FXOSCON = ‘0’ before attempting to low power mode transition.

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

Description:

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

Workaround:

Always program Oscillator to be on when PLL is required.

1.32 ERR002656: FlexCAN: Abort request blocks the CODE field

Description:

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

Workaround:

Instead of aborting the transmission, use deactivation.

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

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

Description:

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

Workaround:

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

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

1.34 ERR003574: MC_RGM: A non-monotonic ramp on the VDD_HV_REG supply can cause the RGM module to clear all flags in the DES register

Description:

During power up, if there is non linearity in power supply lines > 100mV (Hysteresis on LVD) due to external factors, such as battery cranking or weak board regulators, the SoC may show a no flag condition (F_POR == LVD12 == LVD27 == 0) under corner conditions.

Under these situation, it is recommended that customers to use the workaround below to detect a power on condition.

In all cases, the initialization of the device is completed normally.

Workaround:

The software workaround need only be applied when neither of the F_POR, LVD27 and LVD12 flags are set and involves checking SRAM contents and monitoring for ECC errors during this process. In all cases, an ECC error is assumed to signify a power-on reset (POR).

Two suggestions are made for software workarounds. In both cases, if POR is detected all RAM should be initialized otherwise no power-on condition is detected and it is possible to initialize only needed parts of RAM while preserving required information.

Software workaround 1:

An area of RAM can be reserved by the compiler into which a KEY, such as 0x3EC1_9678, is written. This area can be checked at boot and if the KEY is incorrect or an ECC error occurs, POR can be assumed and the KEY should be set. Use of a KEY increases detection rate to 31 bits (≤10e-9) or 23bits (≤5.10e-6) instead of 7-bit linked to ECC (≤10e-2)

Software workaround 2:

When runtime data should be retained and RAM only fully re-initialized in the case of POR, a checksum should be calculated on the runtime data area after each data write. In the event of a reset where no flags are set, the checksum should be read and compared with one calculated across the data area. If reading the checksum and the runtime data area succeeds without an ECC error, and the checksums match, it is assumed than no POR occurred. The checksum could be a CRC, a CMAC or any other suitable hash.

Appendix A Further information

A.1 Reference documents

- *SPC560B4x, SPC560B50, SPC560C4x, SPC560C50 32-bit MCU family built on the embedded Power Architecture®* (RM0017, Doc ID 14629).
- *32-bit MCU family built on the Power Architecture® embedded category for automotive body electronics applications* (SPC560B40x, SPC560B44x, SPC560B50x , SPC560C40x, SPC560C44x, SPC560C50x datasheet, Doc ID 14619).

A.2 Acronyms

Table 2. Acronyms

Acronym	Name
ADC	Analog-to-digital converter
BAM	Boot assist mode
ECC	Error correction code
DSDR	Decode signals delay register
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
10-Nov-2009	1	Initial release
19-Feb-2010	2	Deleted e4264 Deleted e4727 Deleted e7216 Added *Section 1.4: e4839PS: Device may hang up due to functional reset while using debug handshaking mechanism* Added *Section 1.8: e8270PS: F_SOFT bit set on STANDBY exit through external reset* Added *Section 1.9: e8378PS: NMI unavailable in standby mode when code flash is off on standby exit* Added *Section 1.10: e8504PS: PA[1] pull-up enabled when NMI activated* Added *Section 1.20: e6440PS: SWT interrupt does not cause STOP0 mode exit* Added *Section 1.25: e7199PS: When AD_clk = ipg_clk/2, ADC DSD delay odd value is not supported* Added *Section 1.29: e7587PS: Flash: Erase suspend latency is out of specification in Flash except Code Flash 0* Added *Section 1.33: e4781PS: LINFlex: Unexpected LIN timeout in slave mode*
19-Apr-2010	3	Updated *Section 1.4: e4839PS: Device may hang up due to functional reset while using debug handshaking mechanism* Added *Section 1.7: e7316PS: MC_ME: STANDBY mode cannot be entered if the FlexCAN peripheral is active* Added *Section 1.23: e8761PS: MC_CGM: System clock may stop for case when target clock source stops during clock switching transition* Updated *Section 1.24: e5402PS: ADC: ADC_MCR[CTUEN] cannot be reset if a CTU channel is enabled* Updated *Section 1.25: e3826PS: CAN sampler: Second Frame mode not operable* Updated *Section 1.26: e5008PS: Flash: Array integrity check does not check the first two double words* Updated *Section 1.27: e5034PS: Flash: SLL and HBL not properly initialized* Added *Section 1.28: e5060PS: Flash: Double ECC errors of default non volatile image of BIU2 (PFCR2/PFAPR) are not checked* Updated *Section 1.30: e5673PS: Flash: Margin mode is not supported* Updated *Section 1.32: e6384PS: Flash: Erroneous update of the ADR register in case of multiple ECC errors*

Table 3. Document revision history (continued)

Date	Revision	Changes
03-Nov-2010	4	Deleted the following sections: – “e3269PS: Serial boot and censorship - RCHW read access” section – “e3270PS: Serial boot in uncensored device - SRAM access allowed” section – “e3498PS: STANDBY exit time above specification” section – “e3399PS: Flash: RWW-error during stall-while-write” section – “e5911PS: TDO pin floating in Standby mode” section – “e4783PS: SWT: Watchdog is disabled during BAM execution” section – “e14593IPG: FlexCAN: Global Masks misalignment” section – “e7199PS: When AD_clk = ipg_clk/2, ADC DSD delay odd value is not supported” section Added the following sections: – “e9033PS: PB[10] configuration during standby” section – “e9978PS: PIT events cannot be used to trigger ADC conversions in case BCTU runs on divided system clock” section – “e5301PS: MC_RGM: SAFE mode exit may be possible even though condition causing the SAFE mode request has not been cleared” section
17-May-2011	5	Added following functional problem: – ERR003472 Changed following functional problems defect ID: – From e3320PS to ERR002970 – From e4019PS to ERR002998 – From e4554PS to ERR003012 – From e4839PS to ERR003023 – From e5363PS to ERR003064 – From e6943PS to ERR003156 – From e7316PS to ERR003176 – From e8270PS to ERR003198 – From e8378PS to ERR003203 – From e8504PS to ERR003211 – From e9033PS to ERR003240 – From e9978PS to ERR003304 – From e2835PS to ERR002958 – From e3953PS to ERR002995 – From e4107PS to ERR002999 – From e4163PS to ERR003001 – From e5095PS to ERR003049 – From e5301PS to ERR003060 – From e5955PS to ERR003086 – From e6440PS to ERR003119 – From e7810IPG to ERR002161 – From e4455PS to ERR003010 – From e8761PS to ERR003219

Table 3. Document revision history (continued)

Date	Revision	Changes
17-May-2011	5 (continued)	Changed following functional problems defect ID: – From e5402PS to ERR003066 – From e3826PS to ERR002992 – From e5673PS to ERR003073 – From e6384PS to ERR003114 – From e4781PS to ERR003021 – From e4897PS to ERR003028 – From e1685PS to ERR002883 – From e3630PS to ERR002981
28-Jul-2011	6	Removed following functional problems: – ERR002998 – ERR003028 – ERR003073 – ERR003472 – e5007PS – e5008PS – e5034PS – e5060PS – e7587PS Added following functional problems: – ERR003269 – ERR003190 – ERR003202 – ERR002656 – ERR003570 – ERR003574