

## Introduction

This errata sheet describes all the functional and electrical problems known in the cut 3.5 of the SPC560P50, SPC560P44 devices, identified with the JTAG\_ID = 0x6AE2\_1041.

All the topics covered in this document refer to RM0022 and SPC560P50, SPC560P44 datasheet (see [Section Appendix B: Additional information](#)).

Device identification:

- JTAG\_ID = 0x6AE2\_1041
- MCU ID Register 1 (MIDR1):
  - MAJOR\_MASK[3:0]: 4'b0001
  - MINOR\_MASK[3:0]: 4'b0110
- Package device marking mask identifier: BE
- Die Mask ID: FP50B

This errata sheet applies to SPC560P50, SPC560P44 devices in accordance with [Table 1](#).

**Table 1. Device summary**

Part number	Package
SPC560P44L3	LQFP100
SPC560P44L5	LQFP144
SPC560P50L3	LQFP100
SPC560P50L5	LQFP144

# Contents

<b>1</b>	<b>Functional problems</b>	<b>6</b>
1.1	ERR000575: DSPI: Changing CTARs between frames in continuous PCS mode causes error	6
1.2	ERR001082: DSPI: set up enough ASC time when MTFE=1 and CPHA=1	6
1.3	ERR001103: DSPI: PCS Continuous Selection Format limitation	6
1.4	ERR001388:FlexRay: Incomplete Transmission of Message Frame in Key Slot	7
1.5	ERR002161: NPC: Automatic clock gating does not work for MCKO_DIV = 8	7
1.6	ERR002302: FlexRay: Message Buffer cannot be disabled and not locked after CHI command FREEZE	8
1.7	ERR002813: MC_RGM: Reset source flag not set correctly after previous clear	8
1.8	ERR002883:FMPLL: FMPLL_CR[UNLOCK_ONCE] wrongly set	9
1.9	ERR002894: ADC: Minimum sampling time for ADC at 32MHz	9
1.10	ERR002897: ADC: Offset Cancellation Not Required	9
1.11	ERR002958: MC_RGM: Clearing a flag at RGM_DES or RGM_FES register may be prevented by a reset	9
1.12	ERR002963:CMU XOSC Monitoring cannot be guaranteed when RCDIV>0 and FOSC is less than $1.5 * Frc / 2^{rcdiv}$	10
1.13	ERR002964:Register Protection on full CMU_CSR	10
1.14	ERR002966: Serial Boot and Censorship: flash read access	10
1.15	ERR002971:ADC ABORT bit (single conversion)	11
1.16	ERR002972:ADC: Last conversion in chain not aborted	11
1.17	ERR002973: ADC ABORT CHAIN bit	11
1.18	ERR002977: MC_RGM: Long Reset Sequence Occurs on 'Short Functional' Reset Event	11
1.19	ERR002981:FMPLL: Do not poll flag FMPLL_CR[PLL_FAIL]	12
1.20	ERR002982: MCM: MRSR does not report Power On Reset event	12
1.21	ERR002997: ADC: Injected conversion not executed during scan mode.	12
1.22	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	12

1.23 ERR003001: MC\_ME: ME\_PsX registers may show '1' in a reserved bit field ..... 13

1.24 ERR003010: ADC: conversion chain failing after ABORT chain ..... 13

1.25 ERR003021: LINFlex: Unexpected LIN timeout in slave mode ..... 13

1.26 ERR003022: SWT: Watchdog is disabled during BAM execution ..... 13

1.27 ERR003028: LINFlex: BDRL/BDRM cannot be accessed as byte or half-word ..... 14

1.28 ERR003049:MC\_RGM: External Reset not asserted if Short Reset enabled ..... 14

1.29 ERR003060: MC\_RGM: SAFE mode exit may be possible even though condition causing the SAFE mode request has not been cleared ..... 14

1.30 ERR003069: ADC: ADC\_DMAE[DCLR] set to 1 clears the DMA request .. incorrectly 15

1.31 ERR003094: MC\_ME: Peripherals in unknown state after SAFE mode exit ..... 15

1.32 ERR003110: Debugging functionality could be lost when unsecuring a ... secured device. 15

1.33 ERR003114:FLASH: Erroneous update of the ADR register in case of multiple ECC errors ..... 16

1.34 ERR003164: FCU: Timeout feature doesn't work correctly. .... 16

1.35 ERR003165: BAM: Code download via FlexCAN not functioning in a CAN network ..... 16

1.36 ERR003219:MC\_CGM: System clock may stop for case when target clock source stops during clock switching transition ..... 17

1.37 ERR003248: ADC: Abort request during last sampling cycle corrupts the data register of next channel conversion ..... 17

1.38 ERR003256: eTimer: Configuring an eTimer counter channel in GATED-COUNT mode or in SIGNED-COUNT mode may cause a possible incorrect counting of 1 tick. .... 18

1.39 ERR003257: FlexPWM: Configuring the count value to set PWMA/PWMB first low and then high in the same cycle the output signals are low. .... 19

1.40 ERR003258:eTimer: Possible false DMA requests. .... 19

1.41 ERR003269: MC\_ME: Peripheral clocks may get incorrectly disabled or .. enabled after entering debug mode 19

1.42 ERR003310: FlexPWM: PWM signals are improperly synced when using Master Sync ..... 20

1.43 ERR003335: FlexPWM: Incorrect PWM operation when mixing DMA and non-DMA controlled channels ..... 20

---

1.44	ERR003407: FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1 . . . . .	20
1.45	ERR003442: CMU monitor: FXOSC/FIRC and FMPLL/FIRC relation . . .	21
1.46	ERR003511: FlexPWM : Incorrect PWM operation when IPOL is set. . . .	21
1.47	ERR003579: Nexus pins may drive an unknown value immediately after power up but before the 1st clock edge . . . . .	22
1.48	ERR003583: 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. . . . .	22
1.49	ERR003584: MC_ME: Possibility of Machine Check on Low-Power Mode Exit . . . . .	23
1.50	ERR003610: FlexCAN: Wrong data transmission exiting from STOP mode in case EXTAL frequency is greater than IRC . . . . .	24
1.51	ERR005113: ADC: triggering an ABORT or ABORTCHAIN before the conversion starts . . . . .	25
1.52	ERR005203: ADC: "Abort command" aborts the ongoing injected channel as well as the upcoming normal channel. . . . .	25
<b>Appendix A Defects across silicon version . . . . .</b>		<b>27</b>
<b>Appendix B Additional information. . . . .</b>		<b>30</b>
B.1	Reference documents . . . . .	30
B.2	Acronyms . . . . .	30
<b>Revision history . . . . .</b>		<b>32</b>



## List of tables

Table 1.	Device summary . . . . .	1
Table 2.	Defects across silicon version . . . . .	27
Table 3.	Acronyms . . . . .	30
Table 4.	Document revision history . . . . .	32

# 1 Functional problems

## 1.1 ERR000575: DSPI: Changing CTARs between frames in continuous PCS mode causes error

### Description:

Erroneous data could be transmitted if multiple Clock and Transfer Attribute Registers (CTAR) are used while using the Continuous Peripheral Chip Select mode (DSPIx\_PUSHR[CONT=1]). The conditions that can generate an error are:

1. If DSPIx\_CTARn[CPHA]=1 and DSPIx\_MCR[CONT\_SCKE = 0] and DSPIx\_CTARn[CPOL, CPHA, PCSSCK or PBR] change between frames.
2. If DSPIx\_CTARn[CPHA]=0 or DSPIx\_MCR[CONT\_SCKE = 1] and any bit field of DSPIx\_CTARn changes between frames except DSPIx\_CTARn[PBR].

### Workaround:

When generating DSPI bit frames in continuous PCS mode, adhere to the aforementioned conditions when changing DSPIx\_CTARn bit fields between frames.

## 1.2 ERR001082: DSPI: set up enough ASC time when MTFE=1 and CPHA=1

### Description:

When the DSPI is being used in the Modified Transfer Format mode (DSPI\_MCR[MTFE]=1) with the clock phase set for Data changing on the leading edge of the clock and captured on the following edge in the DSPI Clock and Transfer Attributes Register (DSPI\_CTARn[CPHA]=1), if the After SCK delay scaler (ASC) time is set to less than 1/2 SCK clock period the DSPI may not complete the transaction - the TCF flag will not be set, serial data will not receive, and last transmitted bit can be truncated.

### Workaround:

If the Modified Transfer Format mode is required DSPI\_MCR[MTFE]=1 with the clock phase set for serial data changing on the leading edge of the clock and captured on the following edge in the SCK clock (Transfer Attributes Register (DSPI\_CTARn[CPHA]=1) make sure that the ASC time is set to be longer than half SCK clock period.

## 1.3 ERR001103: DSPI: PCS Continuous Selection Format limitation

### Description:

When the DSPI module has more than one entry in the TX FIFO and only one entry is written and that entry has the CONT bit set, and continuous SCK clock selected the PCS levels may change between transfer complete and write of the next data to the DSPI\_PUSHR register.

For example:

If the CONT bit is set with the first PUSHR write, the PCS de-asserts after the transfer because the configuration data for the next frame has already been fetched from the next (empty) fifo entry. This behavior continues till the buffer is filled once and all CONT bits are one.

**Workaround:**

To insure PCS stability during data transmission in Continuous Selection Format and Continuous SCK clock enabled make sure that the data with reset CONT bit is written to DSPI\_PUSHR register before previous data sub-frame (with CONT bit set) transfer is over.

## 1.4 ERR001388: FlexRay: Incomplete Transmission of Message Frame in Key Slot

**Description:**

The FlexRay module will transmit an incomplete message in the key slot under the following circumstances:

1. The transmit message buffer n assigned to the key slot is located in the message buffer segment 2, i.e.  $FR\_MBSSUTR[MB\_LAST\_SEG1] < n$ , and
2. The data size of the message buffer segment 1 is smaller than the static payload length, i.e.  $FR\_MBDSR[MBSEG1DS] < PCR19[payload\_length\_static]$ .

In this case, the FlexRay module will transmit only  $FR\_MBDSR[MBSEG1DS]$  payload words from message buffer n. The remaining words are padded with 0's.

**Workaround:**

The transmit message buffer assigned to key slot must be located in message buffer segment 1.

## 1.5 ERR002161: 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) will be 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  $NPC\_PCR[MCKO\_DIV]=0b000, 0b001, \text{ or } 0b011$ ).

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  $NPC\_PCR[MCKO\_DIV]=0b111$ ).

## 1.6 **ERR002302: FlexRay: Message Buffer cannot be disabled and not locked after CHI command FREEZE**

### Description:

If a complete message was transmitted from a transmit message buffer or received into a message buffer and the controller host interface (CHI) command FREEZE is issued by the application before the end of the current slot, then this message buffer can not be disabled and locked until the module has entered the protocol state normal active.

Consequently, this message buffer can not be disabled and locked by the application in the protocol config state, which prevents the application from clearing the commit bit CMT and the module from clearing the status bits. The configuration bits in the Message Buffer Configuration, Control, Status Registers (MBCCSRn) and the message buffer configuration registers MBCCFRn, MBFIDRn, and MBIDXRn are not affected.

At most one message buffer per channel is affected.

### Workaround:

There are two types of workaround.

The application should not send the CHI command FREEZE and use the CHI command HALT instead.

Before sending the CHI command FREEZE the application should repeatedly try to disable all message buffers until all message buffers are disabled. This maximum duration of this task is three static or three dynamic slots.

## 1.7 **ERR002813: MC\_RGM: Reset source flag not set correctly after previous clear**

### Description:

Bits in the RGM\_FES and RGM\_DES registers may not show the correct status if a reset event occurs after its corresponding flag has previously been cleared. This error occurs only if no other MC\_RGM register access has taken place after clearing the status flag and before the reset event.

### Workaround:

Perform a MC\_RGM register access (e.g. a 'dummy' read) directly after each write access of the RGM\_FES and RGM\_DES registers.



## 1.8 **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.

## 1.9 **ERR002894: ADC: Minimum sampling time for ADC at 32MHz**

### Description:

When ADC is running at 32MHz the minimum sampling time of 135ns specified is not met.

### Workaround:

At 32MHZ the minimum sampling time must be at least 180ns.

## 1.10 **ERR002897: ADC: Offset Cancellation Not Required**

### Description:

The offset cancellation mechanism does not improve the ADC performance as the intrinsic ADC precision is better than the offset cancellation resolution.

### Workaround:

Do not use the offset cancellation feature as it will not improve ADC precision.

## 1.11 **ERR002958: MC\_RGM: Clearing a flag at RGM\_DES or RGM\_FES register may be prevented by a reset**

### Description:

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

Note 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.

## 1.12 ERR002963:CMU XOSC Monitoring cannot be guaranteed when RCDIV>0 and FOSC is less than $1.5 \cdot Frc / 2^{rcdiv}$

### Description:

A False OLRI (Oscillator frequency less than RC frequency) event may be generated when FOSC is less than  $1.5 \cdot (FRC / 2^{RCDIV})$  and  $RCDIV > 0$ , and not  $FRC / 2^{RCDIV}$  as described in the Reference Manual).

Correct crystal clock monitoring is guaranteed when FOSC is strictly above  $1.5 \cdot (FRC / 2^{RCDIV})$ .

### Workaround:

There are 2 workarounds available :

1. Keep  $RCDIV = 0$
2. When  $RCDIV > 0$ , ensure FOSC is greater than  $FRC / 2^{(RCDIV - 1)}$  to avoid false OLRI (CMU external oscillator failure) event.

## 1.13 ERR002964:Register Protection on full CMU\_CSR

### Description:

The register protection on CMU\_CSR of CMU0 works only on the full 32 bit, while it should protect only the bits 24-31.

As a consequence, when register protection is active on CMU\_CSR the frequency meter cannot be used anymore.

This issue is also present on CMU1.

### Workaround:

In order to perform a frequency meter operation, the register protection of the relevant CMU must be disabled first; this workaround would work only when soft lock is active.

## 1.14 ERR002966: Serial Boot and Censorship: flash read access

### Description:

In a secured device, starting with a serial boot, it is possible to read the content of the four flash locations where the RCHW is stored.

For example if the RCHW is stored at address 0x00000000, the reads at address 0x00000000, 0x00000004, 0x00000008 and 0x0000000C will return a correct value. Any other flash address is not readable.

### Workaround:

None

## 1.15 ERR002971:ADC ABORT bit (single conversion)

### Description:

If the User starts one single ADC conversion and, after, starts a chain of conversions, when the User tries to abort one of the chained conversions, the abort command aborts the chain instead of a single conversion of the chain.

### Workaround:

- A minimum of two conversions must be programmed.
- A dummy conversion must be started before or after a single conversion.

## 1.16 ERR002972:ADC: Last conversion in chain not aborted

### Description:

If the user aborts the last ADC conversion of a chain the conversion appears to be aborted but the relevant data register is updated with the conversion data.

### Workaround:

None

## 1.17 ERR002973: ADC ABORT CHAIN bit

### Description:

If user aborts a chain of ADC conversions, the current conversion appears as aborted but the relative data register is however updated.

### Workaround:

None

## 1.18 ERR002977: MC\_RGM: Long Reset Sequence Occurs on 'Short Functional' Reset Event

### Description:

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

### Workaround:

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

### 1.19 **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.

### 1.20 **ERR002982: MCM: MRSR does not report Power On Reset event**

**Description:**

The flag POR of MRSR register stays low after Power On Reset event on the device.

**Workaround:**

Do not use MRSR[POR] to determine Power on reset cause. Use RGM\_DES instead.

### 1.21 **ERR002997: ADC: Injected conversion not executed during scan mode.**

**Description:**

When ADC is converting a chain in scan mode -configured using NSTART bit in non-CTU mode operation; and a injected conversion arrives -triggered by software with JSTART bit or by hardware from eTimer\_1 channel 5 (internal connection); the ADC gets stuck in the sampling phase (the triggered conversion is not executed and the chain is not restarted).

**Workaround:**

None

### 1.22 **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 read or write accesses to this peripheral will be ignored without producing a data storage exception.

**Workaround:**

For any mode other than a low-power mode do not disable any peripheral that is mapped to MC\_CGM or MC\_PCU.

## 1.23 ERR003001: MC\_ME: ME\_P<sub>S</sub>x registers may show '1' in a reserved bit field

### Description:

Some bits in the ME\_P<sub>S</sub>x registers that 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.

## 1.24 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 will be 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.

## 1.25 ERR003021: LINFlex: Unexpected LIN timeout in slave mode

### Description:

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

### Workaround:

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

## 1.26 ERR003022: 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 will be necessary to generate an external reset to recover.

### Workaround:

None

## 1.27 **ERR003028: LINFlex: BDRL/BDRM cannot be accessed as byte or half-word**

### Description:

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

### Workaround:

Access BDRL/BDRM registers as word only.

## 1.28 **ERR003049:MC\_RGM: External Reset not asserted if Short Reset enabled**

### Description:

For the case when the External Reset is enabled for a specific reset source at RGM\_FBRE and a short reset is requested for the same reset source at RGM\_FESS the External Reset is not asserted.

### Workaround:

None

## 1.29 **ERR003060: MC\_RGM: SAFE mode exit may be possible even though condition causing the SAFE mode request has not been cleared**

### Description:

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

### Workaround:

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

### 1.30 **ERR003069: ADC: ADC\_DMAE[DCLR] set to 1 clears the DMA request incorrectly**

**Description:**

When ADC\_DMAE[DCLR] is set the DMA request should be cleared only after the data registers are read. However for this case the DMA request is automatically cleared and will not be recognised by the eDMA.

**Workaround:**

None

### 1.31 **ERR003094: MC\_ME: Peripherals in unknown state after SAFE mode exit**

**Description:**

Peripherals that reside in auxiliary clock domains may be in an unknown state after exiting the SAFE mode to enter the DRUN mode.

**Workaround:**

Execute a software reset while in the SAFE mode if one or more peripherals present in auxiliary clock domains is required for further operation.

### 1.32 **ERR003110: 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.

### 1.33 **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 3 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.

The last event although it does not update ADR 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.

Example - If a sequence of two 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, and so on).

### 1.34 **ERR003164: FCU: Timeout feature doesn't work correctly.**

**Description:**

A fault occurs, timeout for this fault is enabled and the fault is not recovered automatically before the timeout :- In this case device will go to ALARM state and waits for timeout, after timeout has elapsed it enters to FAULT state. However, if another fault occurs with timeout enabled the FCU will go directly to FAULT state without waiting for timeout to be elapsed.

If it is desired that FCU will go again to ALARM state for the second fault, a Watchdog reset needs to be asserted after first fault is detected.

**Workaround:**

None

### 1.35 **ERR003165: BAM: Code download via FlexCAN not functioning in a CAN network**

**Description:**

When the serial download via FlexCAN is selected setting the FAB (Force Alternate Boot) pin, and ABS (Alternate Boot Selector) pins (ABS0 = 1 and ABS1 = 0) and the micro is part of a CAN network, the serial download protocol may unexpectedly stop in case of CAN traffic.

After the code has been downloaded, the BAM tries to disable the FLeXCAN module writing the MCR (Module Configuration Register) without waiting for the acknowledge bit



LPM\_ACK (Low Power Mode Acknowledge) to be set. The FlexCAN cannot enter the low power mode until all current transmissions or receptions have finished. Further writings into any FlexCAN register may cause the low power mode not to be entered and, as consequence, the BAM to stop.

**Workaround:**

Since the higher the traffic, the higher the chance for the BAM to try to disable the FlexCAN module during a CAN frame reception, make sure that no other CAN frame is sent until the code download protocol has been completed.

### 1.36 **ERR003219:MC\_CGM: System clock may stop for case when target clock source stops during clock switching transition**

**Description:**

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

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

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

**Workaround:**

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

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

### 1.37 **ERR003248: ADC: Abort request during last sampling cycle corrupts the data register of next channel conversion**

**Description:**

If the abort pulse is valid in the last cycle of the SAMPLE phase, the current channel is correctly aborted but the data register (CDR[0..15]) of the next channel conversion shows an invalid value.

**Workaround:**

None

## 1.38 ERR003256: eTimer: Configuring an eTimer counter channel in GATED-COUNT mode or in SIGNED-COUNT mode may cause a possible incorrect counting of 1 tick.

### Description:

This bug affects eTimer in the following counting modes:

- GATED-COUNT mode, the CNTMODE field is '011' (count rising edges of primary source while secondary input is high);
- SIGNED-COUNT mode, the CNTMODE field is '101' (count primary source rising edges, secondary source specifies direction (up/down)).

Delays in the edge detection circuitry lead to a bug where the rising edge on the primary source is compared to the secondary source value one clock later in time. This means that if there is a rising edge on the primary source followed immediately by the secondary source going high, the eTimer logic could see this as a rising primary edge while the secondary is high even though the secondary input was low at the time of the rising primary edge.

Note: The counter will also increment if the primary source is already high when the secondary source goes high.

The inputs are sampled by MOTC\_CLK and a transition detector finds the rising edge. It is not an asynchronous counter. Also, transitions are limited to one rising or falling edge per clock period. The primary and secondary inputs come into the Timer and are sampled. The primary input is checked for a rising edge which takes another clock period. This extra cycle of delay on only the primary input is the problem.

If the primary source transitions high while the secondary is low, then no counting should occur. If the secondary source transitions high in the clock cycle after the primary source transition, then the extra delay to find the rising edge will confuse the counter and it will think, that the primary rising edge occurred while the secondary was high and will increment the counter.

If the transitions of the primary and secondary sources are more than one clock period apart, then there will be no incorrect counting.

Possibly, the user will accept that if the edges occur this close together, then its okay to consider it as a countable occurrence, this is application dependant and the responsibility of the user.

### Workaround:

The source selected as the secondary input to the eTimer channel needs to have an additional clock cycle of delay added to it. This can be achieved by using the input filters.

For the primary source set `FILT_PER==1` and `FILT_CNT==0`. For the secondary source set `FILT_PER==1` and `FILT_CNT==1`.

This will introduce a 5 clock cycle latency on the primary source and a 6 cycle latency on the secondary source which will properly align the two signals for Gated and Signed count modes.

Note: Ensure that the primary source is low before the secondary source goes high to avoid a false count.

Note: This work-around is only valid when the sources are external and pass through the input filter, internal signals have no work-around.

### 1.39 **ERR003257: FlexPWM: Configuring the count value to set PWMA/PWMB first low and then high in the same cycle the output signals are low.**

**Description:**

The VAL2 and VAL4 registers define the turn-on edge and the VAL3 and VAL5 registers define the turn off edge of the PWMA/PWMB signals respectively. VAL3 cannot be less than VAL2 and VAL5 cannot be less than VAL4. Doing so will cause the PWM signal to turn off at the correct time (VAL3 or VAL5), but it will not turn on at the time defined by VAL2 or VAL4.

This can be an issue during the generation of phase delayed pulses where the PWM signal goes high late in PWM cycle N and remains high across the cycle boundary before going low early in cycle N+1 and goes high again in PWM cycle N+1. This errata will allow that to happen.

VAL3 register must be "greater than or equal to" VAL2 register and VAL5 must be "greater than or equal to" VAL4.

**Workaround:**

None

### 1.40 **ERR003258:eTimer: Possible false DMA requests.**

**Description:**

The sources of the eTimer's DMA requests are controlled by the DREQ[x] registers. If any of these registers have the same value, then multiple DMA requests can be generated in response to a single flag/condition.

**Workaround:**

Ensure that each of the DREQ[x] registers have a unique value prior to enabling DMA requests.

### 1.41 **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\_PSt registers will 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\_PSt registers.

## 1.42 **ERR003310: FlexPWM: PWM signals are improperly synced when using Master Sync**

### Description:

If Master Sync signal, originated as the Local Sync from sub-module 0, is selected as the counter initialization signal for sub-modules 1-2-3 (slaves), with a prescaler PRSC < 0x2 on PWM clock, the slave sub-module PWM outputs are delayed approx 2 IP clock against sub-module0.

For PRSC > 0x1 the delay on slave sub-modules disappears.

### Workaround:

If Master Sync signal is requested, use a prescaler value PRSC  $\geq 0x2$  to synchronize PWM outputs of Sub-module 0 to slave sub-modules PWM outputs, or don't use the sub-module 0 channel A and B.

## 1.43 **ERR003335: FlexPWM: Incorrect PWM operation when mixing DMA and non-DMA controlled channels**

### Description:

When some submodules use DMA to load their VALx registers and other submodules use non-DMA means that means direct writes from the CPU, the LDOK bits for the non-DMA submodules can be incorrectly cleared at the completion of the DMA controlled load cycle. This leads to the non-DMA channels not being properly updated.

Submodules that use DMA to read the input capture registers do not cause a problem for non-DMA submodules.

### Workaround:

Set the DMA enable bit to 1 also for non-DMA submodules, according to this the DMA will not incorrectly clear the LDOK bit for non-DMA submodules but they will be set to 1 at the end of each DMA cycle. When the CPU has to update the VALx registers of non-DMA submodules, first clear LDOK bit for non-DMA submodules.

## 1.44 **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").

## 1.45 **ERR003442: CMU monitor: FXOSC/FIRC and FMPLL/FIRC relation**

**Description:**

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

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

**Workaround:**

Refer to description.

## 1.46 **ERR003511: FlexPWM : Incorrect PWM operation when IPOL is set.**

**Description:**

When IPOL bit is set in complementary mode, the source of the PWM<sub>i</sub> waveform is supposed to be switched from the VAL2 and VAL3 registers to the VAL4 and VAL5 registers. This switch is not happening.

**Workaround:**

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

## 1.47 **ERR003579:Nexus pins may drive an unknown value immediately after power up but before the 1st clock edge**

### Description:

The Nexus Output pins (Message Data outputs 0:3 [MDO] and Message Start/End outputs 0:1 [MSEO]) may drive an unknown value (high or low) immediately after power up but before the 1st clock edge propagates through the device (instead of being weakly pulled low). This may cause high currents if the pins are tied directly to a supply/ground or any low resistance driver (when used as a general purpose input [GPI] in the application).

### Workaround:

1. Do not tie the Nexus output pins directly to ground or a power supply.
2. If these pins are used as GPI, limit the current to the ability of the regulator supply to guarantee correct start up of the power supply. Each pin may draw upto few hundred mA current.  
If not used, the pins may be left unconnected.

## 1.48 **ERR003583: 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:

At system power-up a non-monotonic voltage ramp-up or a very slow voltage ramp-up (also known as 'soft start-up') can cause incorrect flag setting in the RGM\_DES register. During monotonic power-up, F\_POR flag is set when the high voltage regulator supply (VDD\_HV\_REG) goes above LVD27\_VREG low voltage detector threshold and the 1.2V supply (VDD\_LV\_REGCOR) goes above LVD12\_PD0 low voltage detector threshold. Expected behavior POR =1 , LVD27 =0, LVD12=0

During a non-monotonic power-up the VDD\_HV\_REG may show a non-linearity in the ramp up. When the VDD\_HV\_REG supply dips below LVD27\_VREG threshold, LVD27\_VREG low voltage detector is re-fired. If VDD\_LV\_REGCOR is already above LVD12\_PD0 low voltage detector threshold, F\_POR flag is reset and F\_LVD27\_VREG is set. Expected behavior POR=0 , LVD27 =1, LVD12=0

This errata reports behavior when the non-linearity on VDD\_HV\_REG coincides with the ramp-up of VDD\_LV\_REGCOR completion and LVD27\_VREG is re-fired just after the LVD12\_PD0 is released. In this case, neither F\_POR flag nor F\_LVD27\_VREG flag will be set. In this case, application code cannot use the flags to tell if a power-on reset has occurred.

This errata only affects the flag circuit and not a the device initialization. Device initializes correctly under all conditions.

### Workaround:

Hardware Workaround

Ensure that non-linearity on VDD\_HV\_REG is < 100mV. This is the hysteresis limit of the device. Board regulator should be chosen accordingly.

Software Workaround

The software workaround need only be applied when neither the F\_POR, LVD27 or 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 re-initialized in the case of POR, the CRC module should be used to calculate and store a CRC signature when writing data that can be checked at boot time. If CRC signature is incorrect, POR can be assumed.

## 1.49 ERR003584:MC\_ME: Possibility of Machine Check on Low-Power Mode Exit

### Description:

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

### Workaround:

This issue can be handled in one of the following ways. Workaround #1 configures the application to handle the machine check interrupt in RAM dealing with the problem if it occurs. Workaround #2 configures the MCU to avoid the machine check interrupt.

#### Workaround #1

The application can be configured to handle the machine check interrupt in RAM; when this occurs, the Mode Entry module can be used to bring up the flash normally and resume execution. Before stop mode entry, ensure the following:

1. Enable the Machine Check interrupt at the core MSR[ME] – this prevents a machine check reset occurring
2. Copy IVOR vector table to RAM
3. Point IVPR to vector table in RAM
4. Implement Machine Check interrupt handler in RAM to power-cycle flash to synchronise status of flash between Mode Entry and flash module

The interrupt handler should perform the following steps:

1. Test Machine Check at LPM exit due to wakeup/interrupt event
2. ME\_RUNx\_MC[CFLAON] = 0b01 (power-down)
3. Re-enter mode RUNx (x = 0,1,2,3) to power down flash
4. Wait for transition to RUNx mode to complete (ME\_GS[S\_MTRANS] = 1)
5. ME\_RUNx\_MC[CFLAON] = 0b11 (normal)
6. Re-enter mode RUNx (x = previous x) to power up flash
7. Wait for transition to RUNx mode to complete (ME\_GS[S\_MTRANS] = 1)
8. On completion, code execution will return to flash (via se\_rfc)

#### Workaround #2

The application can be configured to avoid the machine check interrupt; low-power mode can be entered from a RAM function and Mode Entry configured to have flash off on return to the current RUNx mode. Flash can then be re-enabled by Mode Entry within the RAM function before returning to execution from flash.

1. Prior to LPM mode entry request branch to code execution in RAM while flash is still in normal mode
2. Set ME\_RUNx\_MC[CFLAON] = 0b01 (power-down) or 0b10 (low-power) for STOP0/HALT0
3. Set ME\_STOP0/HALT0\_MC[CFLAON] = ME\_RUNx\_MC[CFLAON]
4. Enter STOP0/HALT0 mode
5. At wakeup or interrupt from STOP0/HALT0, MCU enters RUNx mode executing from RAM with flash in low-power or power-down as per the ME\_RUNx\_MC configuration from step 2.
6. After the STOP0/HALT0 request, set ME\_RUNx\_MC[CFLAON] = 0b11 (normal)
7. Enter RUNx mode
8. Wait for transition to RUNx mode to complete (ME\_GS[S\_MTRANS] = 0)
9. Return to code execution in flash

## 1.50 ERR003610:FlexCAN: Wrong data transmission exiting from STOP mode in case EXTAL frequency is greater than IRC

### Description:

The FlexCAN module has got a programmable clock source that can be either the system clock (SYS\_CLK) or oscillator clock (XOSC\_CLK) selected by CLK\_SRC bit in the FlexCAN\_CTRL register.

In case FlexCAN module has selected the oscillator clock as clock source and XOSC\_CLK is bigger than IRC frequency @16MHz and the system clock is PLL\_CLK if the device enters STOP mode and the FlexCAN module is in transmission then when device exits from STOP mode the FlexCAN module can transmit wrong data.

This behavior happens because during STOP mode exit, SYS\_CLK will be IRC @16MHz till PLL gets locked and if a frame transmission happens during this time itself then there will be a CAN Spec violation.

The FlexCAN module clock source should not be faster than SYS\_CLK.



Just before entering/requesting the STOP mode, set the "FRZ" and "HALT" bit of CAN\_MCR register to '1' to request for freeze mode.

During the STOP mode exit, check for the mode transition completion. As mode transition will be over, only when all clock sources are on and the PLL is selected as system clock, unfreeze the CAN by resetting the "FRZ" or "HALT" bit.

## 1.51 ERR005113: ADC: triggering an ABORT or ABORTCHAIN before the conversion starts

### Description:

When ABORTCHAIN is programmed (ADC\_MCR[ABORTCHAIN] = 1) and an injected chain conversion is programmed afterwards, the injected chain is aborted, but neither ADC\_ISR[JECH] is set, nor ADC\_MCR [ABORTCHAIN] is reset.

When ABORT is programmed (ADC\_MCR[ABORT] = 1) and normal/injected chain conversion comes afterwards, the ABORT bit is reset and chain conversion runs without a channel abort.

If ABORT or ABORTCHAIN feature is programmed after the start of the chain conversion, it works properly.

### Workaround:

Do not program ADC\_MCR[ABORT]/ ADC\_MCR[ABORTCHAIN] before starting the execution of the chain conversion.

## 1.52 ERR005203:ADC: "Abort command" aborts the ongoing injected channel as well as the upcoming normal channel.

### Description:

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

Expected behavior:

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

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

Observed unexpected behavior:

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

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

### Workaround:

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

## Appendix A Defects across silicon version

Table 2. Defects across silicon version

Defect ID	Title	CUT3.4	CUT3.5	Workaround
ERR000575	DSPI: Changing CTARs between frames in continuous PCS mode causes error	x	x	Available
ERR000817	JTAGC: EVTI and RDY require TCK to toggle	x	(1)	Available
ERR001082	DSPI: set up enough ASC time when MTFE=1 and CPHA=1	x	x	Available
ERR001103	DSPI: PCS Continuous Selection Format limitation	x	x	Available
ERR001388	FlexRay: Incomplete transmission of message frame in key slot	x	x	Available
ERR002161	NPC: Automatic clock gating does not work for MCKO_DIV = 8	x	x	Available
ERR002302	FlexRay: Message Buffer can not be disabled and not locked after CHI command FREEZE	x	x	Available
ERR002360	FlexCAN: Global Masks misalignment	x	(1)	Available
ERR002421	FLEXRAY: Message Buffer can not be disabled in POC state INTEGRATION_LISTEN	x	(1)	Available
ERR002423	FlexRay: Transmission in a slot n of Channel A in dynamic segment may be corrupted or duplicated on both the channels	x	(1)	Available
ERR002656	FlexCAN: Abort request blocks the CODE field	x	(1)	Available
ERR002685	FlexCAN: Module Disable Mode functionality not described correctly	x	(1)	Available
ERR002813	MC_RGM: Reset source flag not set correctly after previous clear	x	x	Available
ERR002883	FMPLL: FMPLL_CR[UNLOCK_ONCE] wrongly set	x	x	No workaround
ERR002894	ADC: Minimum sampling time for ADC at 32MHz	x	x	Available
ERR002897	ADC: Offset Cancellation Not Required	x	x	Available
ERR002958	MC_RGM: Clearing a flag at RGM_DES or RGM_FES register may be prevented by a reset	x	x	Available
ERR002963	CMU XOSC Monitoring cannot be guaranteed when RCDIV>0 and FOSC is less than $1.5 \cdot Frc / 2^{rcdiv}$	x	x	Available
ERR002964	Register Protection on full CMU_CSR	x	x	Available
ERR002966	Serial Boot and Censorship: Flash read access	x	x	No workaround
ERR002971	ADC ABORT bit (single conversion)	x	x	Available
ERR002972	ADC: Last conversion in chain not aborted	x	x	No workaround
ERR002973	ADC ABORT CHAIN bit	x	x	No workaround
ERR002977	MC_RGM: Long Reset Sequence Occurs on 'Short Functional' Reset	x	x	Available

Table 2. Defects across silicon version (continued)

Defect ID	Title	CUT3.4	CUT3.5	Workaround
ERR002981	FMPLL: Do not poll flag FMPLL_CR[PLL_FAIL]	x	x	Available
ERR002982	MCM: MRSR does not report Power On Reset event	x	x	Available
ERR002997	ADC: Injected conversion not executed during scan mode	x	x	No workaround
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. DRAFT	x	x	Available
ERR003001	MC_ME: ME_Px registers may show '1' in a reserved bit field	x	x	Available
ERR003010	ADC: conversion chain failing after ABORT chain	x	x	Available
ERR003021	LINFlex: Unexpected LIN timeout in slave mode	x	x	No workaround
ERR003022	SWT: Watchdog is disabled during BAM execution	x	x	No workaround
ERR003028	LINFlex: BDRL/BDRM cannot be accessed as byte or half-word	x	x	Available
ERR003049	MC_RGM: External Reset not asserted if Short Reset enabled	x	x	No workaround
ERR003060	MC_RGM: SAFE mode exit may be possible even though condition causing the SAFE mode request has not been cleared	x	x	Available
ERR003069	ADC: ADC_DMAE[DCLR] set to 1 clears the DMA request incorrectly	x	x	No workaround
ERR003094	MC_ME: Peripherals in unknown state after SAFE mode exit	x	x	Available
ERR003110	Debugging functionality could be lost when unsecuring a secured device.	x	x	Available
ERR003114	FLASH: Erroneous update of the ADR register in case of multiple ECC errors	x	x	Available
ERR003164	FCU timeout feature does not work correctly	x	x	No workaround
ERR003165	BAM: Code download via FlexCAN not functioning in a CAN network	x	x	Available
ERR003219	MC_CGM: System clock may stop for case when target clock source stops during clock switching transition	x	x	Available
ERR003248	ADC: Abort request during last sampling cycle corrupts the data register of next channel conversion	x	x	No workaround
ERR003256	eTimer: Configuring an eTimer counter channel in mode may cause a possible incorrect counting of 1 tick.	x	x	Available
ERR003257	FlexPWM: Configuring the count value to set PWMA/PWMB first low and then high in the same cycle	x	x	No workaround
ERR003258	eTimer: Possible false DMA requests	x	x	Available
ERR003269	MC_ME: Peripheral clocks may get incorrectly disabled or enabled after entering debug mode	x	x	Available

Table 2. Defects across silicon version (continued)

Defect ID	Title	CUT3.4	CUT3.5	Workaround
ERR003310	FlexPWM: PWM signals are improperly synced when using Master Sync	x	x	Available
ERR003324	FIRC - FIRC_CTL[TRIM] does not display correct trim value after reset	x	(1)	Available
ERR003335	FlexPWM: Incorrect PWM operation when mixing DMA and non-DMA controlled channels	x	x	Available
ERR003407	FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1	x	x	Available
ERR003442	CMU monitor: FXOSC/FIRC and FMPLL/FIRC relation.	x	x	Available
ERR003511	FlexPWM: Incorrect PWM operation when IPOL bit is set.	x	x	Available
ERR003579	Pad Ring:Nexus pins may drive an unknown value immediately after power up but before the 1st clock edge	x	x	Available
ERR003583	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	x	x	Available
ERR003584	MC_ME: Possibility of Machine Check on Low-Power Mode Exit	x	x	Available
ERR003610	FlexCAN: Wrong data transmission exiting from STOP mode in case EXTAL frequency is greater than IRC	x	x	Available
ERR005113	ADC: triggering an ABORT or ABORTCHAIN before the conversion starts	x	x	Available
ERR005203	ADC: "Abort command" aborts the ongoing injected channel as well as the upcoming normal channel.	x	x	Available

1. Described in the RM0022 rev 5(see [Section B.1: Reference documents](#)).

## Appendix B Additional information

### B.1 Reference documents

- 32-bit MCU family built on the Power Architecture™ embedded category for automotive chassis and safety electronics applications (RM0022, Doc ID14891)
- l 32-bit Power Architecture™ based MCU with 576 KB Flash memory and 40 KB RAM for automotive chassis and safety applications (SPC560P44L3, SPC560P44L5, SPC560P50L3, SPC560P50L5 datasheet, Doc ID14723)

### B.2 Acronyms

Table 3. Acronyms

Acronym	Name
ABS	Alternate boot selector
ADC	Analog-to-digital converter
BAM	Boot assist module
BDRL	Buffer data register least significant
BDRM	Buffer data register most significant
CHI	Controller host interface
CMU	Clock monitor unit
DMA	Direct memory access
ECC	Error correction code
EDC	ECC data correction
FAB	Force alternate boot
FCU	Fault collection unit
FIFO	First in first out
HBL	High address space block locking register
ISR	Interrupt service routine
LPM_ACK	Low power mode acknowledge
MB	Message buffer
MBCCSR	Message buffer configuration, control, status register
MCKO	Nexus clock
MCR	Module configuration register
MCU	Microcontroller unit
PCR	Pad configuration register
POC	Protocol operation control
RWE	Read-while-write event error

**Table 3. Acronyms (continued)**

Acronym	Name
RXGMASK	RX global mask
RXIMR	RX individual mask register
SLL	Secondary low/mid address space block lock register

## Revision history

**Table 4. Document revision history**

<b>Date</b>	<b>Revision</b>	<b>Changes</b>
22-May-2013	1	Initial release
17-Sep-2013	2	Updated Disclaimer.
26-Sep-2013	3	Changed the package device marking mask identifier from BD in BE.



**Please Read Carefully:**

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

**UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.**

**ST PRODUCTS ARE NOT DESIGNED OR AUTHORIZED FOR USE IN: (A) SAFETY CRITICAL APPLICATIONS SUCH AS LIFE SUPPORTING, ACTIVE IMPLANTED DEVICES OR SYSTEMS WITH PRODUCT FUNCTIONAL SAFETY REQUIREMENTS; (B) AERONAUTIC APPLICATIONS; (C) AUTOMOTIVE APPLICATIONS OR ENVIRONMENTS, AND/OR (D) AEROSPACE APPLICATIONS OR ENVIRONMENTS. WHERE ST PRODUCTS ARE NOT DESIGNED FOR SUCH USE, THE PURCHASER SHALL USE PRODUCTS AT PURCHASER'S SOLE RISK, EVEN IF ST HAS BEEN INFORMED IN WRITING OF SUCH USAGE, UNLESS A PRODUCT IS EXPRESSLY DESIGNATED BY ST AS BEING INTENDED FOR "AUTOMOTIVE, AUTOMOTIVE SAFETY OR MEDICAL" INDUSTRY DOMAINS ACCORDING TO ST PRODUCT DESIGN SPECIFICATIONS. PRODUCTS FORMALLY ESCC, QML OR JAN QUALIFIED ARE DEEMED SUITABLE FOR USE IN AEROSPACE BY THE CORRESPONDING GOVERNMENTAL AGENCY.**

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2013 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

[www.st.com](http://www.st.com)

