SPC564Bxx, SPC56ECxx Errata sheet

SPC564Bxx, SPC56ECxx devices errata
JTAG_ID = 0x0AE49041

Introduction

This errata sheet describes all the functional and electrical problems known in the revision of SPC564Bxx, SPC56ECxx cut 2.0.

Device identification:

- JTAG_ID = 0x0AE49041
- MCU ID Register 1
  - MAJOR_MASK : 01
  - MINOR_MASK: 00
- Package device marking mask identifier: BA
- Die Mask ID: VG121929

This errata sheet applies to SPC564Bxx, SPC56ECxx devices in accordance with *Table 1*.

Table 1. Device summary

Package	Part number		
	1.5 MByte	**2 MByte**	**3 MByte**
LQFP176	SPC564B64L7 SPC56EC64L7	SPC564B70L7 SPC56EC70L7	SPC564B74L7 SPC56EC74L7
LQFP208	SPC564B64L8 SPC56EC64L8	SPC564B70L8 SPC56EC70L8	SPC564B74L8 SPC56EC74L8
BGA256	SPC56EC64B3	SPC56EC70B3	SPC56EC74B3

Contents

1 Functional problems

1.1 e2656: 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 instead.

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

1.2 e3407: 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").

1.3 e3436: Nexus Debug : MSEO[1] functionality is not supported on PL[11]

Description:

Only MSEO[0] on pin (PL[9]) is implemented.

Workaround:

The debugger should be configured to only use MSEO[0].

1.4 e3442: 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) + 0.5MHz in order to guarantee correct FXOSC monitoring
- FMPLL frequency must be greater than (FIRC / 4) + 0.5MHz in order to guarantee correct FMPLL monitoring .

Workaround:

Refer to description.

1.5 e3476: 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 ongoing the reset may interrupt the clearing mechanism leaving the flag set.

Note: Note that this failed clearing has no impact on further flag clearing requests.

Workaround:

No workaround for all reset sources except SW reset.

Note: 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.

1.6 e3512: ECSM: ECSM_PFEDR displays incorrect endianness

Description:

The ECSM_PFEDR register reports ECC data using incorrect endiannes. For example, a flash location that contains the data 0xAABBCCDD would be reported as 0xDDCCBBAA at ECSM_PFEDR. This 32-bit register contains the data associated with the faulting access of the last, properly-enabled flash ECC event. The register contains the data value taken directly from the data bus.

Workaround:

Software must correct endianness.

1.7 e3556: DMA_MUX : Low Power Entry may not be completed when peripherals run on divided clock with DMA enabled mode

Description:

System may not enter into Low Power Mode (HALT/STOP/STANDBY) when all the below conditions are true simultaneously:

1. A Peripheral with DMA capability is programmed to work on divided clock.
2. Above peripheral is programmed to be stopped in Low Power Mode and active in RUN Mode.
3. Above Peripheral is active with DMA transfer while Software requests change to Low Power mode.

Workaround:

Software should ensure that all the DMA enabled peripherals have completed their transfer before requesting Low Power mode Entry.

1.8 e3570: 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 checkstop due to the Flash not being available on RUNx mode re-entry. This will cause either a checkstop reset or machine check interrupt.

Workaround:

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

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

There is no absolute requirement to work around the possibility of a checkstop reset if the application can accept the reset, and associated delays, and continue. In this event, the WKPU.WISR will not indicate the channel that triggered the wakeup though the F_CHKSTOP flag will indicate that the reset has occurred. The F_CHKSTOP flag could still be caused by other error conditions so the startup strategy from this condition should be considered alongside any pre-existing strategy for recovering from an F_CHKSTOP condition.

1.9 e3697: e200z: Exceptions generated on speculative prefetch

Description:

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

Workaround:

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

2 Revision history

Table 2. Document revision history

Date	Revision	Changes
24-Jan-2012	1	Initial release.