STM32H7A3xI/G, STM32H7B0xB and STM32H7B3xI device errata

Applicability

This document applies to the part numbers of STM32H7A3xI/G, STM32H7B0xB and STM32H7B3xI devices and the device variants as stated in this page.

It gives a summary and a description of the device errata, with respect to the device datasheet and reference manual RM0455. Deviation of the real device behavior from the intended device behavior is considered to be a device limitation. Deviation of the description in the reference manual or the datasheet from the intended device behavior is considered to be a documentation erratum. The term “errata” applies both to limitations and documentation errata.

Table 1. Device summary

<table>
<thead>
<tr>
<th>Reference</th>
<th>Part numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32H7B0xB</td>
<td>STM32H7B0AB, STM32H7B0IB, STM32H7B0RB, STM32H7B0VB, STM32H7B0ZB</td>
</tr>
<tr>
<td>STM32H7B3xI</td>
<td>STM32H7B3RI, STM32H7B3VI, STM32H7B3QI, STM32H7B3ZI, STM32H7B3AI, STM32H7B3II, STM32H7B3NI, STM32H7B3LI</td>
</tr>
</tbody>
</table>

Table 2. Device variants

<table>
<thead>
<tr>
<th>Reference</th>
<th>Silicon revision codes</th>
<th>Device marking(1)</th>
<th>REV_ID(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>STM32H7A3xI</td>
<td></td>
<td>Z</td>
<td>0x1001</td>
</tr>
<tr>
<td>STM32H7A3xG</td>
<td></td>
<td>Z</td>
<td>0x1001</td>
</tr>
<tr>
<td>STM32H7B3xI</td>
<td></td>
<td>Z</td>
<td>0x1001</td>
</tr>
<tr>
<td>STM32H7B0xB</td>
<td></td>
<td>Z</td>
<td>0x1001</td>
</tr>
</tbody>
</table>

1. Refer to the device datasheet for how to identify this code on different types of package.
1 Summary of device errata

The following table gives a quick reference to the STM32H7A3xl/G, STM32H7B0xB and STM32H7B3xl device limitations and their status:

- **A** = workaround available
- **N** = no workaround available
- **P** = partial workaround available

Applicability of a workaround may depend on specific conditions of target application. Adoption of a workaround may cause restrictions to target application. Workaround for a limitation is deemed partial if it only reduces the rate of occurrence and/or consequences of the limitation, or if it is fully effective for only a subset of instances on the device or in only a subset of operating modes, of the function concerned.

<table>
<thead>
<tr>
<th>Function</th>
<th>Section</th>
<th>Limitation</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>2.1.1</td>
<td>CPUID register returns r1p1 instead of r1p2</td>
<td>N</td>
</tr>
<tr>
<td>System</td>
<td>2.2.2</td>
<td>Performing a system reset during Flash memory program or erase operation is not supported</td>
<td>N</td>
</tr>
<tr>
<td>BDMA</td>
<td>2.3.1</td>
<td>BDMA disable failure and error flag omission upon simultaneous transfer error and global flag clear</td>
<td>A</td>
</tr>
<tr>
<td>DMA</td>
<td>2.4.1</td>
<td>DMA stream locked when transferring data to/from USART/UART</td>
<td>A</td>
</tr>
<tr>
<td>DMAMUX</td>
<td>2.5.1</td>
<td>SOFx not asserted when writing into DMAMUX_CCFR register</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>2.5.2</td>
<td>OFx not asserted for trigger event coinciding with last DMAMUX request</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>2.5.3</td>
<td>OFx not asserted when writing into DMAMUX_RGCFR register</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>2.5.4</td>
<td>Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event</td>
<td>A</td>
</tr>
<tr>
<td>FMC</td>
<td>2.6.1</td>
<td>Dummy read cycles inserted when reading synchronous memories</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>2.6.2</td>
<td>Wrong data read from a busy NAND memory</td>
<td>A</td>
</tr>
<tr>
<td>OCTOSPI</td>
<td>2.7.1</td>
<td>Indirect read and auto-polling transfers without address phase not starting</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>2.7.2</td>
<td>Maxtran period not respected in specific condition</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>2.7.3</td>
<td>Octal DDR indirect read data corrupted if last two bytes are read at a specific condition</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>2.7.4</td>
<td>Spurious interrupt in AND-match polling mode with full data masking</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>2.7.5</td>
<td>Hybrid wrap data transfer corruption upon an internal event</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>2.7.6</td>
<td>Hybrid wrap registers not functional</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>2.7.7</td>
<td>Odd address alignment and odd byte number not supported at specific conditions</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>2.7.8</td>
<td>OCTOSPI DDR mode not supported with DQS disabled</td>
<td>N</td>
</tr>
<tr>
<td>DAC</td>
<td>2.8.1</td>
<td>Invalid DAC channel analog output if the DAC channel MODE bitfield is programmed before DAC initialization</td>
<td>A</td>
</tr>
<tr>
<td>VREFBUF</td>
<td>2.9.1</td>
<td>Overshoot on VREFBUF output</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>2.9.2</td>
<td>VREFBUF Hold mode cannot be used</td>
<td>N</td>
</tr>
<tr>
<td>PSSI</td>
<td>2.10.1</td>
<td>Bus error if FIFO overrun occurs during master access to the FIFO</td>
<td>A</td>
</tr>
<tr>
<td>TIM</td>
<td>2.11.1</td>
<td>One-pulse mode trigger not detected in master-slave reset + trigger configuration</td>
<td>P</td>
</tr>
<tr>
<td>LPTIM</td>
<td>2.12.1</td>
<td>MCU may remain stuck in LPTIM interrupt when entering Stop mode</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>2.12.2</td>
<td>MCU may remain stuck in LPTIM interrupt when clearing event flag</td>
<td>P</td>
</tr>
<tr>
<td>Function</td>
<td>Section</td>
<td>Limitation</td>
<td>Status</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>RTC and TAMP</td>
<td>2.13.1</td>
<td>Calendar initialization may fail in case of consecutive INIT mode entry</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>2.13.2</td>
<td>Alarm flag may be repeatedly set when the core is stopped in debug</td>
<td>N</td>
</tr>
<tr>
<td>I2C</td>
<td>2.14.1</td>
<td>Wrong data sampling when data setup time (TSU;DAT) is shorter than one I2C kernel clock period</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>2.14.2</td>
<td>Spurious bus error detection in master mode</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>2.14.3</td>
<td>Spurious master transfer upon own slave address match</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>2.14.4</td>
<td>START bit is cleared upon setting ADDRCF, not upon address match</td>
<td>D</td>
</tr>
<tr>
<td>USART</td>
<td>2.15.1</td>
<td>Receiver timeout counter wrong start in two-stop-bit configuration</td>
<td>A</td>
</tr>
<tr>
<td>SPI2S</td>
<td>2.16.1</td>
<td>Master data transfer stall at system clock much faster than SCK</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>2.16.2</td>
<td>Corrupted CRC return at non-zero UDRDET setting</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>2.16.3</td>
<td>TXP interrupt occurring while SPI disabled</td>
<td>A</td>
</tr>
</tbody>
</table>
2 Description of device errata

The following sections describe limitations of the applicable devices with Arm® core and provide workarounds if available. They are grouped by device functions.

Note: Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

2.1 Core

Reference manual and errata notice for the Arm® Cortex®-M7 core revision r1p2 is available from http://infocenter.arm.com.

2.1.1 CPUID register returns r1p1 instead of r1p2

Description

The devices are based on Arm® Cortex®-M7 core revision r1p2. However, when reading the CPUID register at address 0xE000 ED00, r1p1 is returned instead of r1p2.

Workaround

None.

2.2 System

2.2.1 Unstable LSI when it clocks RTC or CSS on LSE

Description

The LSI clock can become unstable (duty cycle different from 50 %) and its maximum frequency can become significantly higher than 32 kHz, when:

• LSI clocks the RTC, or it clocks the clock security system (CSS) on LSE (which holds when the LSECSSON bit set), and
• the V_DD power domain is reset while the backup domain is not reset, which happens:
  – upon exiting Shutdown mode
  – if V_BAT is separate from V_DD and V_DD goes off then on
  – if V_BAT is tied to V_DD (internally in the package for products not featuring the VBAT pin, or externally) and a short (< 1 ms) V_DD drop under V_DD(min) occurs

Workaround

Apply one of the following measures:

• Clock the RTC with LSE or HSE/32, without using the CSS on LSE
• If LSI clocks the RTC or when the LSECSSON bit is set, reset the backup domain upon each V_DD power up (when the BORRSTF flag is set). If V_BAT is separate from V_DD, also restore the RTC configuration, backup registers and anti-tampering configuration.

2.2.2 Performing a system reset during Flash memory program or erase operation is not supported

Description

The microcontroller may be stalled if a system reset occurs during a Flash memory program or erase operation. A power-on reset must be performed to restart the system.
2.3 BDMA

2.3.1 BDMA disable failure and error flag omission upon simultaneous transfer error and global flag clear

**Description**

Upon a data transfer error in a BDMA channel x, both the specific TEIFx and the global GIFx flags are raised and the channel x is normally automatically disabled. However, if in the same clock cycle the software clears the GIFx flag (by setting the CGIFx bit of the BDMA_IFCR register), the automatic channel disable fails and the TEIFx flag is not raised.

This issue does not occur with ST’s HAL software that does not use and clear the GIFx flag, but uses and clears the HTIFx, TCIFx, and TEIFx specific event flags instead.

**Workaround**

Do not clear GIFx flags. Instead, use HTIFx, TCIFx, and TEIFx specific event flags and their corresponding clear bits.

2.4 DMA

2.4.1 DMA stream locked when transferring data to/from USART/UART

**Description**

When a USART/UART is issuing a DMA request to transfer data, if a concurrent transfer occurs, the requested transfer may not be served and the DMA stream may stay locked.

**Workaround**

Use the alternative peripheral DMA channel protocol by setting bit 20 of the DMA_SxCR register. This bit is reserved in the documentation and must be used only on the stream that manages data transfers for USART/UART peripherals.

2.5 DMAMUX

2.5.1 SOFx not asserted when writing into DMAMUX_CCFR register

**Description**

The SOFx flag of the DMAMUX_CSR status register is not asserted if overrun from another DMAMUX channel occurs when the software writes into the DMAMUX_CCFR register.

This can happen when multiple DMA channels operate in synchronization mode, and when overrun can occur from more than one channel. As the SOFx flag clear requires a write into the DMAMUX_CCFR register (to set the corresponding CSOFx bit), overrun occurring from another DMAMUX channel operating during that write operation fails to raise its corresponding SOFx flag.

**Workaround**

None. Avoid the use of synchronization mode for concurrent DMAMUX channels, if at least two of them potentially generate synchronization overrun.
2.5.2 OFx not asserted for trigger event coinciding with last DMAMUX request

Description
In the DMAMUX request generator, a trigger event detected in a critical instant of the last-generated DMAMUX request being served by the DMA controller does not assert the corresponding trigger overrun flag OFx. The critical instant is the clock cycle at the very end of the trigger overrun condition.

Additionally, upon the following trigger event, one single DMA request is issued by the DMAMUX request generator, regardless of the programmed number of DMA requests to generate.

The failure only occurs if the number of requests to generate is set to more than two (GNBREQ[4:0] > 00001).

Workaround
Make the trigger period longer than the duration required for serving the programmed number of DMA requests, so as to avoid the trigger overrun condition from occurring on the very last DMA data transfer.

2.5.3 OFx not asserted when writing into DMAMUX_RGCFR register

Description
The OFx flag of the DMAMUX_RGSR status register is not asserted if an overrun from another DMAMUX request generator channel occurs when the software writes into the DMAMUX_RGCFR register. This can happen when multiple DMA channels operate with the DMAMUX request generator, and when an overrun can occur from more than one request generator channel. As the OFx flag clear requires a write into the DMAMUX_RGCFR register (to set the corresponding COFx bit), an overrun occurring in another DMAMUX channel operating with another request generator channel during that write operation fails to raise the corresponding OFx flag.

Workaround
None. Avoid the use of request generator mode for concurrent DMAMUX channels, if at least two channels are potentially generating a request generator overrun.

2.5.4 Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event

Description
If a write access into the DMAMUX_CxCR register having the SE bit at zero and SPOL[1:0] bitfield at a value other than 00:
• sets the SE bit (enables synchronization),
• modifies the values of the DMAREQ_ID[5:0] and SYNC_ID[4:0] bitfields, and
• does not modify the SPOL[1:0] bitfield,
and if a synchronization event occurs on the previously selected synchronization input exactly two AHB clock cycles before this DMAMUX_CxCR write, then the input DMA request selected by the DMAREQ_ID[5:0] value before that write is routed.

Workaround
Ensure that the SPOL[1:0] bitfield is at 00 whenever the SE bit is 0. When enabling synchronization by setting the SE bit, always set the SPOL[1:0] bitfield to a value other than 00 with the same write operation into the DMAMUX_CxCR register.

2.6 FMC

2.6.1 Dummy read cycles inserted when reading synchronous memories

Description
When performing a burst read access from a synchronous memory, two dummy read accesses are performed at the end of the burst cycle whatever the type of burst access.
The extra data values read are not used by the FMC and there is no functional failure.

**Workaround**  
None.

### 2.6.2 Wrong data read from a busy NAND memory

**Description**  
When a read command is issued to the NAND memory, the R/B signal gets activated upon the de-assertion of the chip select. If a read transaction is pending, the NAND controller might not detect the R/B signal (connected to NWAIT) previously asserted and sample a wrong data. This problem occurs only when the MEMSET timing is configured to 0x00 or when ATTHOLD timing is configured to 0x00 or 0x01.

**Workaround**  
Either configure MEMSET timing to a value greater than 0x00 or ATTHOLD timing to a value greater than 0x01.

### 2.7 OCTOSPI

#### 2.7.1 Indirect read and auto-polling transfers without address phase not starting

**Description**  
Indirect read and auto-polling transfers, configured through the CCR register to contain command and SDR or DDR octal data phases but no address phase, do not start.

**Workaround**  
Configure the transfer to contain address phase and no command phase, then send the command through the address register.

#### 2.7.2 Maxtran period not respected in specific condition

**Description**  
Under the following condition:  
- arbitration activated  
- memory-mapped write initiated on one OCTOSPI instance, with a data byte number corresponding to less than two OCTOSPI clock cycles in the data phase  
- another OCTOSPI instance requests the I/O port,
  
the I/O port is not granted even if the Maxtran period expired, unless another mechanism finishes the transaction of the first OCTOSPI instance, such as timeout, new memory request, or the arrival of additional bytes to write. For example, in octal DDR, the minimum byte number for two clock cycles in the data phase is four. A memory-mapped write of less than four bytes by the OCTOSPI1 instance that holds the I/O port prevents the OCTOSPI2 instance to take it over when requested.

**Workaround**  
Activate the timeout feature to trigger arbitration and select a medium timeout value. A too small value would lead to excessive chip select activity and increase power consumption, and a too big value would lead to excessive arbitration delay and inappropriate system latency.
2.7.3 Octal DDR indirect read data corrupted if last two bytes are read at a specific condition

Description
Indirect read from an octal DDR memory may lead to data corruption upon the following condition:
• Number of bytes to read, defined in OCTOSPI_DLR register, is a multiple of 32 plus two, for example 34, 66, 98, and so on.
• The last two bytes are read with different requests.
• The second-last request read size is different from one byte.

Workaround
Apply one of the following measures:
• Read the last two bytes of a transfer with the same request.
• Read the last two bytes each with transfer size of one byte.

2.7.4 Spurious interrupt in AND-match polling mode with full data masking

Description
In AND-match polling mode with the MASK[31:0] bitfield set to 0x0000 0000 (all bits masked), a spurious interrupt may occur.

Workaround
Avoid setting the MASK[31:0] bitfield to 0x0000 0000.

2.7.5 Hybrid wrap data transfer corruption upon an internal event

Description
An internal event pertaining to TIMEOUT[15:0], CSBOUND[4:0], MAXTRAN[7:0], or REFRESH[31:0] bitfields may disturb any ongoing hybrid wrap transaction and result in corruption of the remaining data to transfer.

Workaround
Manage the TIMEOUT[15:0], CSBOUND[4:0], MAXTRAN[7:0], and REFRESH[31:0] bitfields such as to avoid any related internal event during hybrid wrap transactions.

2.7.6 Hybrid wrap registers not functional

Description
OCTOSPI_WPABR and OCTOSPI_WPTCR registers are not functional. As a consequence, external memory devices that require the setting of OCTOSPI_WPABR and OCTOSPI_WPTCR registers for the hybrid wrap because it is different from the settings of OCTOSPI_ABR and OCTOSPI_TCR registers used for the read, are not supported.

Note: Most memory devices allow the same settings for the hybrid wrap and the read.

Workaround
Only use memory devices allowing the same settings for the hybrid wrap and the read.
2.7.7 Odd address alignment and odd byte number not supported at specific conditions

Description
Odd address alignment and odd transaction byte number is not supported for some combinations of memory access mode, access type, and other settings. The following table summarizes the supported combinations, and provides information on consequences of accessing an illegal address and/or of setting an illegal number of bytes in a transaction.

Table 4. Summary of supported combinations

<table>
<thead>
<tr>
<th>Memory access mode / other settings</th>
<th>Access type</th>
<th>Address allowed</th>
<th>Consequence of illegal address access</th>
<th>Byte number allowed</th>
<th>Consequence of illegal byte number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-SPI, Dual-SPI, Quad-SPI, RAM / DQM = 0 or Octo-SPI / SDR mode</td>
<td>ind read</td>
<td>any</td>
<td>N/A</td>
<td>any</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>mm read</td>
<td>any</td>
<td>N/A</td>
<td>any</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>ind write</td>
<td>any</td>
<td>N/A</td>
<td>any</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>mm write</td>
<td>any</td>
<td>N/A</td>
<td>any</td>
<td>N/A</td>
</tr>
<tr>
<td>Single-SPI, Dual-SPI, Quad-SPI, RAM / DQM = 1 or Octo-SPI, RAM / DDR mode, no RDS, no WDM</td>
<td>ind read</td>
<td>even</td>
<td>ADDR[0] cleared</td>
<td>even</td>
<td>DLR[0] cleared</td>
</tr>
<tr>
<td></td>
<td>mm read</td>
<td>any</td>
<td>N/A</td>
<td>any</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>ind write</td>
<td>even</td>
<td>ADDR[0] cleared</td>
<td>even</td>
<td>DLR[0] cleared</td>
</tr>
<tr>
<td></td>
<td>mm write</td>
<td>even</td>
<td>slave error</td>
<td>even</td>
<td>last byte lost</td>
</tr>
<tr>
<td>Octo-SPI, RAM / DDR mode, with RDS or WDM or HyperBus™</td>
<td>ind read</td>
<td>even</td>
<td>ADDR[0] cleared</td>
<td>even</td>
<td>DLR[0] cleared</td>
</tr>
<tr>
<td></td>
<td>mm read</td>
<td>any</td>
<td>N/A</td>
<td>any</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>ind write</td>
<td>any</td>
<td>N/A</td>
<td>any</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>mm write</td>
<td>any</td>
<td>N/A</td>
<td>any</td>
<td>N/A</td>
</tr>
</tbody>
</table>

1. “RDS” = read data strobe, “WDM” = write data mask
2. “ind read” = indirect read, “mm read” = memory-mapped read, “ind write” = indirect write, “mm write” = memory-mapped write
3. “N/A” = not applicable

Workaround
Avoid illegal address accesses and illegal byte numbers in transactions.

2.7.8 OCTOSPI DDR mode not supported with DQS disabled

Description
The Octo-SPI interface does not support DDR mode when DQS is disabled. This is true for all non-Hyperbus protocols and all data modes: octal, dual-quad, quad, dual and single.

Workaround
None.

2.8 DAC
2.8.1 Invalid DAC channel analog output if the DAC channel MODE bitfield is programmed before DAC initialization

Description
When the DAC operates in Normal mode and the DAC enable bit is cleared, writing a value different from 000 to the DAC channel MODE bitfield of the DAC_MCR register before performing data initialization causes the corresponding DAC channel analog output to be invalid.

Workaround
Apply the following sequence:
1. Perform one write access to any data register.
2. Program the MODE bitfield of the DAC_MCR register.

2.9 VREFBUF

2.9.1 Overshoot on VREFBUF output

Description
An overshoot might occur on VREFBUF output if VREF+ pin has residual voltage when VREFBUF is enabled (ENVR set to 1 in VREFBUF_CSR register).

Workaround
Discharge VREF+ pin below $V_{REFBUF\_OUT}$ voltage -1 V. This can be achieved by switching VREFBUF buffer OFF (ENVR=0 and HIZ=0 in VREFBUF_CSR register) during sufficient time to discharge the output capacitor through VREFBUF pull-down resistor.

2.9.2 VREFBUF Hold mode cannot be used

Description
When VREFBUF operates in Hold mode (ENVR=1 and HIZ=1 in VREFBUF_CSR register), the output voltage, $V_{REFBUF\_OUT}$, is kept on the external capacitor with a reduced VREFBUF current consumption.
When exiting Hold mode, enabling the VREFBUF might result in an overshoot on the $V_{REFBUF\_OUT}$ if VREF+ pin has residual voltage.

Workaround
None.

2.10 PSSI

2.10.1 Bus error if FIFO overrun occurs during master access to the FIFO

Description
If a PSSI FIFO overrun occurs while a master is accessing the FIFO, a bus error is reported to the master.

Workaround
Use the PSSI with the DMA and manage PSSI FIFO overrun through the DMA transfer error interrupt.

2.11 TIM
2.11.1 One-pulse mode trigger not detected in master-slave reset + trigger configuration

Description

The failure occurs when several timers configured in one-pulse mode are cascaded, and the master timer is configured in combined reset + trigger mode with the MSM bit set:

- OPM = 1 in TIMx_CR1,
- SMS[3:0] = 1000,
- MSM = 1 in TIMx_SMCR.

The MSM delays the reaction of the master timer to the trigger event, so as to have the slave timers cycle-accurately synchronized.

If the trigger arrives when the counter value is equal to the period value set in the TIMx_ARR register, the one-pulse mode of the master timer does not work and no pulse is generated on the output.

Workaround

None. However, unless a cycle-level synchronization is mandatory, it is advised to keep the MSM bit reset, in which case the problem is not present. The MSM = 0 configuration also allows decreasing the timer latency to external trigger events.

2.12 LPTIM

2.12.1 MCU may remain stuck in LPTIM interrupt when entering Stop mode

Description

This limitation occurs when disabling the low-power timer (LPTIM).

When the user application clears the ENABLE bit in the LPTIM_CR register within a small time window around one LPTIM interrupt occurrence, then the LPTIM interrupt signal used to wake up the MCU from Stop mode may be frozen in active state. Consequently, when trying to enter Stop mode, this limitation prevents the MCU from entering low-power mode and the firmware remains stuck in the LPTIM interrupt routine.

This limitation applies to all Stop modes and to all instances of the LPTIM. Note that the occurrence of this issue is very low.

Workaround

In order to disable a low power timer (LPTIMx) peripheral, do not clear its ENABLE bit in its respective LPTIM_CR register. Instead, reset the whole LPTIMx peripheral via the RCC controller by setting and resetting its respective LPTIMxRST bit in RCC_APBxRSTRz register.

2.12.2 MCU may remain stuck in LPTIM interrupt when clearing event flag

Description

This limitation occurs when the LPTIM is configured in interrupt mode (at least one interrupt is enabled) and the software clears any flag by writing the LPTIM_ICR bit in the LPTIM_ISR register. If the interrupt status flag corresponding to a disabled interrupt is cleared simultaneously with a new event detection, the set and clear commands might reach the APB domain at the same time, leading to an asynchronous interrupt signal permanently stuck high.

This issue can occur either during an interrupt subroutine execution (where the flag clearing is usually done), or outside an interrupt subroutine. Consequently, the firmware remains stuck in the LPTIM interrupt routine, and the MCU cannot enter Stop mode.

Workaround

To avoid this issue, it is strongly advised to follow the recommendations listed below:

- Clear the flag only when its corresponding interrupt is enabled in the interrupt enable register.
- If for specific reasons, it is required to clear some flags that have corresponding interrupt lines disabled in the interrupt enable register, it is recommended to clear them during the current subroutine prior to those which have corresponding interrupt line enabled in the interrupt enable register.
- Flags must not be cleared outside the interrupt subroutine.
2.13 RTC and TAMP

2.13.1 Calendar initialization may fail in case of consecutive INIT mode entry

Description

If the INIT bit of the RTC_ICSR register is set between one and two RTCCLOCK cycles after being cleared, the INITF flag is set immediately instead of waiting for synchronization delay (which should be between one and two RTCCLOCK cycles), and the initialization of registers may fail. Depending on the INIT bit clearing and setting instants versus the RTCCLOCK edges, it can happen that, after being immediately set, the INITF flag is cleared during one RTCCLOCK period then set again. As writes to calendar registers are ignored when INITF is low, a write occurring during this critical period might result in the corruption of one or more calendar registers.

Workaround

After existing the initialization mode, clear the BYPSHAD bit (if set) then wait for RSF to rise, before entering the initialization mode again.

Note: It is recommended to write all registers in a single initialization session to avoid accumulating synchronization delays.

2.13.2 Alarm flag may be repeatedly set when the core is stopped in debug

Description

When the core is stopped in debug mode, the clock is supplied to subsecond RTC alarm downcounter even though the device is configured to stop the RTC in debug. As a consequence, when the subsecond counter is used for alarm condition (the MASKSS[3:0] bitfield of the RTC_ALRMASSR and/or RTC_ALRMBSSR register set to a non-zero value) and the alarm condition is met just before entering a breakpoint or printf, the ALRAF and/or ALRBF flag of the RTC_SR register is repeatedly set by hardware during the breakpoint or printf, which makes any tentative to clear the flag(s) ineffective.

Workaround

None.

2.14 I2C

2.14.1 Wrong data sampling when data setup time (t_{SU;DAT}) is shorter than one I2C kernel clock period

Description

The I2C-bus specification and user manual specify a minimum data setup time (t_{SU;DAT}) as:

- 250 ns in Standard mode
- 100 ns in Fast mode
- 50 ns in Fast mode Plus

The MCU does not correctly sample the I2C-bus SDA line when t_{SU;DAT} is smaller than one I2C kernel clock (I2C-bus peripheral clock) period: the previous SDA value is sampled instead of the current one. This can result in a wrong receipt of slave address, data byte, or acknowledge bit.
Workaround
Increase the I2C kernel clock frequency to get I2C kernel clock period within the transmitter minimum data setup time. Alternatively, increase transmitter’s minimum data setup time. If the transmitter setup time minimum value corresponds to the minimum value provided in the I2C-bus standard, the minimum I2CCLK frequencies are as follows:
• In Standard mode, if the transmitter minimum setup time is 250 ns, the I2CCLK frequency must be at least 4 MHz.
• In Fast mode, if the transmitter minimum setup time is 100 ns, the I2CCLK frequency must be at least 10 MHz.
• In Fast-mode Plus, if the transmitter minimum setup time is 50 ns, the I2CCLK frequency must be at least 20 MHz.

2.14.2 Spurious bus error detection in master mode
Description
In master mode, a bus error can be detected spuriously, with the consequence of setting the BERR flag of the I2C_SR register and generating bus error interrupt if such interrupt is enabled. Detection of bus error has no effect on the I2C-bus transfer in master mode and any such transfer continues normally.

Workaround
If a bus error interrupt is generated in master mode, the BERR flag must be cleared by software. No other action is required and the ongoing transfer can be handled normally.

2.14.3 Spurious master transfer upon own slave address match
Description
When the device is configured to operate at the same time as master and slave (in a multi-master I2C-bus application), a spurious master transfer may occur under the following condition:
• Another master on the bus is in process of sending the slave address of the device (the bus is busy).
• The device initiates a master transfer by bit set before the slave address match event (the ADDR flag set in the I2C_ISR register) occurs.
• After the ADDR flag is set:
  – the device does not write I2C_CR2 before clearing the ADDR flag, or
  – the device writes I2C_CR2 earlier than three I2C kernel clock cycles before clearing the ADDR flag

In these circumstances, even though the START bit is automatically cleared by the circuitry handling the ADDR flag, the device spuriously proceeds to the master transfer as soon as the bus becomes free. The transfer configuration depends on the content of the I2C_CR2 register when the master transfer starts. Moreover, if the I2C_CR2 is written less than three kernel clocks before the ADDR flag is cleared, the I2C peripheral may fall into an unpredictable state.

Workaround
Upon the address match event (ADDR flag set), apply the following sequence.
Normal mode (SBC = 0):
1. Set the ADDRCF bit.
2. Before Stop condition occurs on the bus, write I2C_CR2 with the START bit low.
Slave byte control mode (SBC = 1):
1. Write I2C_CR2 with the slave transfer configuration and the START bit low.
2. Wait for longer than three I2C kernel clock cycles.
3. Set the ADDRCF bit.
4. Before Stop condition occurs on the bus, write I2C_CR2 again with its current value.

The time for the software application to write the I2C_CR2 register before the Stop condition is limited, as the clock stretching (if enabled), is aborted when clearing the ADDR flag.
Polling the BUSY flag before requesting the master transfer is not a reliable workaround as the bus may become busy between the BUSY flag check and the write into the I2C_CR2 register with the START bit set.

2.14.4 START bit is cleared upon setting ADDRCF, not upon address match

Description
Some reference manual revisions may state that the START bit of the I2C_CR2 register is cleared upon slave address match event. Instead, the START bit is cleared upon setting, by software, the ADDRCF bit of the I2C_ICR register, which does not guarantee the abort of master transfer request when the device is being addressed as slave. This product limitation and its workaround are the subject of a separate erratum.

Workaround
No application workaround is required for this description inaccuracy issue.

2.15 USART

2.15.1 Receiver timeout counter wrong start in two-stop-bit configuration

Description
In two-stop-bit configuration, the receiver timeout counter starts counting from the end of the second stop bit of the last character instead of starting from the end of the first stop bit.

Workaround
Subtract one bit duration from the value in the RTO bitfield of the USARTx_RTOR register.

2.16 SPI2S

2.16.1 Master data transfer stall at system clock much faster than SCK

Description
With the system clock (spi_pclk) substantially faster than SCK (spi_ker_ck divided by a prescaler), SPI master data transfer can stall upon setting the CSTART bit within one SCK cycle after the EOT event (EOT flag raise) signaling the end of the previous transfer.

Workaround
Apply one of the following measures:
• Disable then enable SPI after each EOT event.
• Upon EOT event, wait for at least one SCK cycle before setting CSTART.
• Prevent EOT events from occurring, by setting transfer size to undefined (TSIZE = 0) and by triggering transmission exclusively by TXFIFO writes.

2.16.2 Corrupted CRC return at non-zero UDRDET setting

Description
With non-zero setting of UDRDET[1:0] bitfield, the SPI slave can transmit the first bit of CRC pattern corrupted, coming wrongly from the UDRCFG register instead of SPI_TXCRC. All other CRC bits come from the SPI_TXCRC register, as expected.

Workaround
Keep TXFIFO non-empty at the end of transfer.
2.16.3 TXP interrupt occurring while SPI disabled

Description
SPI2S peripheral is set to its default state when disabled (SPE = 0). This flushes the FIFO buffers and resets their occupancy flags. TXP and TXC flags become set (the latter if the TSIZE field contains zero value), triggering interrupt if enabled with TXPIE or EOTIE bit, respectively. The resulting interrupt service can be spurious if it tries to write data into TXFIFO to clear the TXP and TXC flags, while both FIFO buffers are inaccessible (as the peripheral is disabled).

Workaround
Keep TXP and TXC (the latter if the TSIZE field contains zero value) interrupt disabled whenever the SPI2S peripheral is disabled.
Revision history

Table 5. Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>29-Jan-2020</td>
<td>7</td>
<td>First public release.</td>
</tr>
</tbody>
</table>
Contents

1 Summary of device errata .......................................................... 2

2 Description of device errata ....................................................... 4

2.1 Core .......................................................................................... 4

2.1.1 CPUID register returns r1p1 instead of r1p2 ......................... 4

2.2 System ....................................................................................... 4

2.2.1 Unstable LSI when it clocks RTC or CSS on LSE ..................... 4

2.2.2 Performing a system reset during Flash memory program or erase operation is not supported ................................................. 4

2.3 BDMA ....................................................................................... 5

2.3.1 BDMA disable failure and error flag omission upon simultaneous transfer error and global flag clear ........................................ 5

2.4 DMA ....................................................................................... 5

2.4.1 DMA stream locked when transferring data to/from USART/UART .......................................................... 5

2.5 DMAMUX ............................................................................... 5

2.5.1 SOFx not asserted when writing into DMAMUX_CCFR register . 5

2.5.2 OFx not asserted for trigger event coinciding with last DMAMUX request .......................................................... 5

2.5.3 OFx not asserted when writing into DMAMUX_RGCFR register .......................................................... 5

2.5.4 Wrong input DMA request routed upon specific DMAMUX_CxCR register write coinciding with synchronization event ................................................. 6

2.6 FMC ....................................................................................... 6

2.6.1 Dummy read cycles inserted when reading synchronous memories .......................................................... 6

2.6.2 Wrong data read from a busy NAND memory .......................... 7

2.7 OCTOSPI ............................................................................... 7

2.7.1 Indirect read and auto-polling transfers without address phase not starting .......................................................... 7

2.7.2 Maxtran period not respected in specific condition .................. 7

2.7.3 Octal DDR indirect read data corrupted if last two bytes are read at a specific condition .......................................................... 7

2.7.4 Spurious interrupt in AND-match polling mode with full data masking .......................................................... 7

2.7.5 Hybrid wrap data transfer corruption upon an internal event ........................................................................... 7

2.7.6 Hybrid wrap registers not functional ....................................... 7

2.7.7 Odd address alignment and odd byte number not supported at specific conditions .......................................................... 8

2.7.8 OCTOSPI DDR mode not supported with DQS disabled ............. 8
## DAC
Invalid DAC channel analog output if the DAC channel MODE bitfield is programmed before DAC initialization.

## VREFBUF
- Overshoot on VREFBUF output
- VREFBUF Hold mode cannot be used

## PSSI
- Bus error if FIFO overrun occurs during master access to the FIFO

## TIM
- One-pulse mode trigger not detected in master-slave reset + trigger configuration

## LPTIM
- MCU may remain stuck in LPTIM interrupt when entering Stop mode
- MCU may remain stuck in LPTIM interrupt when clearing event flag

## RTC and TAMP
- Calendar initialization may fail in case of consecutive INIT mode entry
- Alarm flag may be repeatedly set when the core is stopped in debug

## I2C
- Wrong data sampling when data setup time (\(t_{SU;DAT}\)) is shorter than one I2C kernel clock period
- Spurious bus error detection in master mode
- Spurious master transfer upon own slave address match
- START bit is cleared upon setting ADDRCF, not upon address match

## USART
- Receiver timeout counter wrong start in two-stop-bit configuration

## SPI2S
- Master data transfer stall at system clock much faster than SCK
- Corrupted CRC return at non-zero UDRDET setting
- TXP interrupt occurring while SPI disabled

### Revision history

<table>
<thead>
<tr>
<th>Revision</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ES0478 - Rev 7</td>
<td>page 18/19</td>
</tr>
</tbody>
</table>