



STM32F20x and STM32F21x Errata sheet

STM32F205/207xx and STM32F215/217xx device limitations

Silicon identification

This errata sheet applies to the revision Y and X of STMicroelectronics STM32F205xx/STM32F207xx and STM32F215xx/STM32F217xx microcontroller families. In this document they will be referred to as STM32F20x and STM32F21x, respectively, unless otherwise specified.

The STM32F20x and STM32F21x feature revision r2p0-v2 of the ARM™ 32-bit Cortex®-M3 core, for which an errata notice is also available (see [Section 1](#) for details).

The full list of part numbers is shown in [Table 2](#). The products are identifiable as shown in [Table 1](#):

- by the revision code marked below the order code on the device package
- by the last three digits of the Internal order code printed on the box label

Table 1. Device Identification⁽¹⁾

Order code	Revision code marked on device ⁽²⁾
STM32F205xx, STM32F207xx	“Y” and “X”
STM32F215xx, STM32F217xx	“Y” and “X”

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the STM32F20x and STM32F21x reference manual for details on how to find the revision code).
2. Refer to [Appendix A: Revision code on device marking](#) for details on how to identify the revision code and the date code on the different packages.

Table 2. Device summary

Reference	Part number
STM32F205xx	STM32F205RB, STM32F205RC, STM32F205RE, STM32F205RF, STM32F205RG, STM32F205VB, STM32F205VC, STM32F205VE, STM32F205VF, STM32F205VG, STM32F205ZC, STM32F205ZE, STM32F205ZF, STM32F205ZG
STM32F207xx	STM32F207IC, STM32F207IE, STM32F207IF, STM32F207IG, STM32F207ZC, STM32F207ZE, STM32F207ZF, STM32F207ZG, STM32F207VC, STM32F207VE, STM32F207VF, STM32F207VG
STM32F215xx	STM32F215RG, STM32F215VG, STM32F215ZG, STM32F215RE, STM32F215VE, STM32F215ZE
STM32F217xx	STM32F217VG, STM32F217IG, STM32F217ZG, STM32F217VE, STM32F217IE, STM32F217ZE

Contents

1	ARM™ 32-bit Cortex®-M3 limitations	7
1.1	Cortex-M3 limitations description for the STM32F20x and STM32F21x devices	7
1.1.1	Cortex-M3 LDRD with base in list may result in incorrect base register when interrupted or faulted	7
1.1.2	Cortex-M3 event register is not set by interrupts and debug	8
1.1.3	Cortex-M3 interrupted loads to stack pointer can cause erroneous behavior	8
1.1.4	Cortex-M3 SVC and BusFault/MemManage may occur out of order	9
2	STM32F20x and STM32F21x silicon limitations	10
2.1	System limitations	12
2.1.1	ART Accelerator prefetch queue instruction is not supported	12
2.1.2	Debugging Stop mode and system tick timer	12
2.1.3	Debugging Stop mode with WFE entry	12
2.1.4	DBGMCU_CR register cannot be read by user software	13
2.1.5	Configuration of PH10 and PI10 as external interrupts is erroneous	13
2.1.6	DMA2 data corruption when managing AHB and APB peripherals in a concurrent way	13
2.1.7	Slowing down APB clock during a DMA transfer	14
2.1.8	MPU attribute to RTC and IWDG registers could be managed incorrectly	14
2.1.9	Delay after an RCC peripheral clock enabling	14
2.1.10	Battery charge monitoring lower than 2.4 V	15
2.1.11	Internal noise impacting the ADC accuracy	15
2.2	IWDG peripheral limitation	15
2.2.1	RVU and PVU flags are not reset in STOP mode	15
2.3	I2C peripheral limitations	16
2.3.1	SMBus standard not fully supported	16
2.3.2	Start cannot be generated after a misplaced Stop	16
2.3.3	Mismatch on the “Setup time for a repeated Start condition” timing parameter	16
2.3.4	Data valid time ($t_{VD;DAT}$) violated without the OVR flag being set	17
2.4	I2S peripheral limitations	17
2.4.1	Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronisation mode	17

2.4.2	In I2S slave mode, WS level must be set by the external master when enabling the I2S	17
2.4.3	I2S slave mode desynchronization with the master during communication	18
2.5	USART peripheral limitations	18
2.5.1	Idle frame is not detected if receiver clock speed is deviated	18
2.5.2	In full duplex mode, the Parity Error (PE) flag can be cleared by writing to the data register	19
2.5.3	Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection	19
2.5.4	Break frame is transmitted regardless of nCTS input line status	19
2.5.5	nRTS signal abnormally driven low after a protocol violation	19
2.6	OTG_FS peripheral limitations	20
2.6.1	Data in RxFIFO is overwritten when all channels are disabled simultaneously	20
2.6.2	OTG host blocks the receive channel when receiving IN packets and no TxFIFO is configured	20
2.6.3	Host channel-halted interrupt not generated when the channel is disabled	21
2.6.4	Error in software-read OTG_FS_DCFG register values	21
2.6.5	Minimum AHB frequency to guarantee correct operation of USB OTG FS peripheral	21
2.7	Ethernet peripheral limitations	21
2.7.1	Incorrect layer 3 (L3) checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads	21
2.7.2	The Ethernet MAC processes invalid extension headers in the received IPv6 frames	22
2.7.3	MAC stuck in the Idle state on receiving the TxFIFO flush command exactly 1 clock cycle after a transmission completes	22
2.7.4	Transmit frame data corruption	22
2.7.5	MCO PLL clock pins not compatible with Ethernet IEEE802.3 long term jitter specifications	23
2.8	FSMC peripheral limitations	23
2.8.1	Dummy read cycles inserted when reading synchronous memories	23
2.8.2	FSMC synchronous mode and NWAIT signal disabled	24
2.9	SDIO peripheral limitations	24
2.9.1	SDIO HW flow control	24
2.9.2	Wrong CCRCFAIL status after a response without CRC is received	24
2.9.3	SDIO clock divider BYPASS mode may not work properly	25
2.9.4	Data corruption in SDIO clock dephasing (NEGEDGE) mode	25

2.9.5 CE-ATA multiple write command and card busy signal management . . 25

2.10 DAC limitations 26

2.10.1 DMA underrun flag management 26

2.10.2 DMA request not automatically cleared by DMAEN=0 26

Appendix A Revision code on device marking 27

Revision history 33



List of tables

Table 1. Device Identification 1

Table 2. Device summary 1

Table 3. Cortex-M3 core limitations and impact on microcontroller behavior 7

Table 4. Summary of silicon limitations 10

Table 5. Document revision history 33

List of figures

Figure 1. WLCSP64+2 top package view 27

Figure 2. UFBGA176 top package view..... 28

Figure 3. LQFP176 top package view 29

Figure 4. LQFP144 top package view 30

Figure 5. LQFP100 top package view 31

Figure 6. LQFP64 top package view 32



1 ARM™ 32-bit Cortex®-M3 limitations

An errata notice of the STM32F20x and STM32F21x core is available from the following web address: <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.eat0420a/>.

The direct link to the errata notice pdf is:

<http://infocenter.arm.com/help/topic/com.arm.doc.eat0420c/Cortex-M3-Errata-r2p0-v2.pdf>.

All the described limitations are minor and related to the revision r2p0-v2 of the Cortex-M3 core. [Table 3](#) summarizes these limitations and their implications on the behavior of high-density STM32F20x and STM32F21x devices.

Table 3. Cortex-M3 core limitations and impact on microcontroller behavior

ARM ID	ARM category	ARM summary of errata	Impact on STM32F20x and STM32F21x devices
752419	Cat 2	Interrupted loads to SP can cause erroneous behavior	Minor
740455	Cat 2	SVC and BusFault/MemManage may occur out of order	Minor
602117	Cat 2	LDRD with base in list may result in incorrect base register when interrupted or faulted	Minor
563915	Cat 2	Event register is not set by interrupts and debug	Minor

1.1 Cortex-M3 limitations description for the STM32F20x and STM32F21x devices

Only the limitations described below have an impact, even though minor, on the implementation of STM32F20x and STM32F21x devices.

All the other limitations described in the ARM errata notice (and summarized in [Table 3](#) above) have no impact and are not related to the implementation of STM32F20x and STM32F21x devices (Cortex-M3 r2p0-v2).

1.1.1 Cortex-M3 LDRD with base in list may result in incorrect base register when interrupted or faulted

Description

The Cortex-M3 Core has a limitation when executing an LDRD instruction from the system-bus area, with the base register in a list of the form LDRD Ra, Rb, [Ra, #imm]. The execution may not complete after loading the first destination register due to an interrupt before the second loading completes or due to the second loading getting a bus fault.

Workarounds

1. This limitation does not impact the STM32F20x and STM32F21x code execution when executing from the embedded Flash memory, which is the standard use of the microcontroller.
2. Use the latest compiler releases. As of today, they no longer generate this particular sequence. Moreover, a scanning tool is provided to detect this sequence on previous releases (refer to your preferred compiler provider).

1.1.2 Cortex-M3 event register is not set by interrupts and debug

Description

When interrupts related to a WFE occur before the WFE is executed, the event register used for WFE wakeup events is not set and the event is missed. Therefore, when the WFE is executed, the core does not wake up from WFE if no other event or interrupt occur.

Workaround

Use STM32F20x and STM32F21x external events instead of interrupts to wake up the core from WFE by configuring an external or internal EXTI line in event mode.

1.1.3 Cortex-M3 interrupted loads to stack pointer can cause erroneous behavior

Description

An interrupt occurring during the data-phase of a single word load to the stack pointer (SP/R13) can cause an erroneous behavior of the device. In addition, returning from the interrupt results in the load instruction being executed an additional time.

For all the instructions performing an update of the base register, the base register is erroneously updated on each execution, resulting in the stack pointer being loaded from an incorrect memory location.

The instructions affected by this limitation are the following:

- LDR SP, [Rn],#imm
- LDR SP, [Rn,#imm]!
- LDR SP, [Rn,#imm]
- LDR SP, [Rn]
- LDR SP, [Rn,Rm]

Workaround

As of today, no compiler generates these particular instructions. This limitation can only occur with hand-written assembly code.

Both issues can be solved by replacing the direct load to the stack pointer by an intermediate load to a general-purpose register followed by a move to the stack pointer.

Example:

Replace LDR SP, [R0] by

LDR R2,[R0]

MOV SP,R2

1.1.4 Cortex-M3 SVC and BusFault/MemManage may occur out of order

Description

If an SVC exception is generated by executing the SVC instruction while the next instruction fetch is faulted, then the MemManage or BusFault handler may be entered even though the faulted instruction following the SVC instruction should not have been executed.

Workaround

A workaround is required only if the SVC handler does not return to the return address that has been stacked for the SVC exception, and the instruction access after the SVC will fault. In this case, insert padding (e.g. NOP instructions) between the SVC instruction and the faulting code.

2 STM32F20x and STM32F21x silicon limitations

[Table 4](#) gives quick references to all documented limitations.

Legend for [Table 4](#): A = workaround available; N = no workaround available; P = partial workaround available, '-' and grayed = fixed.

Table 4. Summary of silicon limitations

Links to silicon limitations		Rev Y and X
Section 2.1: System limitations	Section 2.1.1: ART Accelerator prefetch queue instruction is not supported	N
	Section 2.1.2: Debugging Stop mode and system tick timer	A
	Section 2.1.3: Debugging Stop mode with WFE entry	A
	Section 2.1.4: DBGMCU_CR register cannot be read by user software	N
	Section 2.1.5: Configuration of PH10 and PI10 as external interrupts is erroneous	N
	Section 2.1.6: DMA2 data corruption when managing AHB and APB peripherals in a concurrent way	A
	Section 2.1.7: Slowing down APB clock during a DMA transfer	A
	Section 2.1.8: MPU attribute to RTC and IWDG registers could be managed incorrectly	A
	Section 2.1.9: Delay after an RCC peripheral clock enabling	A
	Section 2.1.10: Battery charge monitoring lower than 2.4 V	P
	Section 2.1.11: Internal noise impacting the ADC accuracy	A
Section 2.2: IWDG peripheral limitation	Section 2.2.1: RVU and PVU flags are not reset in STOP mode	A
Section 2.3: I2C peripheral limitations	Section 2.3.1: SMBus standard not fully supported	A
	Section 2.3.2: Start cannot be generated after a misplaced Stop	A
	Section 2.3.3: Mismatch on the "Setup time for a repeated Start condition" timing parameter	A
	Section 2.3.4: Data valid time ($t_{VD, DAT}$) violated without the OVR flag being set	A
Section 2.4: I2S peripheral limitations	Section 2.4.1: Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronisation mode	A
	Section 2.4.2: In I2S slave mode, WS level must be set by the external master when enabling the I2S	A
	Section 2.4.3: I2S slave mode desynchronization with the master during communication	A

Table 4. Summary of silicon limitations (continued)

Links to silicon limitations		Rev Y and X
Section 2.5: USART peripheral limitations	Section 2.5.1: Idle frame is not detected if receiver clock speed is deviated	N
	Section 2.5.2: In full duplex mode, the Parity Error (PE) flag can be cleared by writing to the data register	A
	Section 2.5.3: Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection	N
	Section 2.5.4: Break frame is transmitted regardless of nCTS input line status	N
	Section 2.5.5: nRTS signal abnormally driven low after a protocol violation	A
Section 2.6: OTG_FS peripheral limitations	Section 2.6.1: Data in RxFIFO is overwritten when all channels are disabled simultaneously	A
	Section 2.6.2: OTG host blocks the receive channel when receiving IN packets and no TxFIFO is configured	A
	Section 2.6.3: Host channel-halted interrupt not generated when the channel is disabled	A
	Section 2.6.4: Error in software-read OTG_FS_DCFG register values	A
	Section 2.6.5: Minimum AHB frequency to guarantee correct operation of USB OTG FS peripheral	N
Section 2.7: Ethernet peripheral limitations	Section 2.7.1: Incorrect layer 3 (L3) checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads	A
	Section 2.7.2: The Ethernet MAC processes invalid extension headers in the received IPv6 frames	N
	Section 2.7.3: MAC stuck in the Idle state on receiving the TxFIFO flush command exactly 1 clock cycle after a transmission completes	A
	Section 2.7.4: Transmit frame data corruption	A
	Section 2.7.5: MCO PLL clock pins not compatible with Ethernet IEEE802.3 long term jitter specifications	A
Section 2.8: FSMC peripheral limitations	Section 2.8.1: Dummy read cycles inserted when reading synchronous memories	N
	Section 2.8.2: FSMC synchronous mode and NWAIT signal disabled	A
Section 2.9: SDIO peripheral limitations	Section 2.9.1: SDIO HW flow control	N
	Section 2.9.2: Wrong CCRCFAIL status after a response without CRC is received	N
	Section 2.9.3: SDIO clock divider BYPASS mode may not work properly	A
	Section 2.9.4: Data corruption in SDIO clock dephasing (NEGEDGE) mode	N
	Section 2.9.5: CE-ATA multiple write command and card busy signal management	A
Section 2.10: DAC limitations	Section 2.10.1: DMA underrun flag management	A
	Section 2.10.2: DMA request not automatically cleared by DMAEN=0	A

Note: Revision X differs from revision Y as it improves the LSE (Low Speed External oscillator) power consumption, without fixing limitations. Refer to STM32F20xx and STM32F21xx datasheets.

2.1 System limitations

2.1.1 ART Accelerator prefetch queue instruction is not supported

Description

The ART Accelerator prefetch queue instruction is not supported when V_{DD} is lower than 2.1 V.

This limitation does not prevent the ART Accelerator from using the cache enable/disable capability and the selection of the number of wait states according to the system frequency.

Workaround

None. Refer to application note AN3430 for details on how to adjust performance and power consumption.

2.1.2 Debugging Stop mode and system tick timer

Description

If the system tick timer interrupt is enabled during the Stop mode debug (DBG_STOP bit set in the DBGMCU_CR register), it will wake up the system from Stop mode.

Workaround

To debug the Stop mode, disable the system tick timer interrupt.

2.1.3 Debugging Stop mode with WFE entry

Description

When the Stop debug mode is enabled (DBG_STOP bit set in the DBGMCU_CR register), this allows software debugging during Stop mode.

However, if the application software uses the WFE instruction to enter Stop mode, after wakeup some instructions could be missed if the WFE is followed by sequential instructions. This affects only Stop debug mode with WFE entry.

Workaround

To debug Stop mode with WFE entry, the WFE instruction must be inside a dedicated function with 1 instruction (NOP) between the execution of the WFE and the Bx LR.

Example:

```
__asm void _WFE(void) {  
    WFE  
    NOP
```

BX lr }

2.1.4 DBGMCU_CR register cannot be read by user software

Description

The DBGMCU_CR debug register is accessible only in debug mode (not accessible by the user software). When this register is read in user mode, the returned value is 0x00.

Workaround

None.

2.1.5 Configuration of PH10 and PI10 as external interrupts is erroneous

Description

PH10 or PI10 are selected as the source for EXTI10 external interrupt by setting bits EXTI10[3:0] of SYSCFG_EXTICR3 register to 0x0111 or 0x1000, respectively. However, this erroneous operation enables PH2 and PI2 as external interrupt inputs.

As a result, it is not possible to use PH10/PI10 as interrupt sources if PH2/PI2 are not selected as the interrupt source, as well. This means that bits EXTI10[3:0] of SYSCFG_EXTICR3 register and bits EXTI2[3:0] of SYSCFG_EXTICR1 should be programmed to the same value:

- 0x0111 to select PH10/PH2
- 0x1000 to select PI10/PI2

Workaround

None.

2.1.6 DMA2 data corruption when managing AHB and APB peripherals in a concurrent way

Description

When the DMA2 is managing AHB Peripherals (only peripherals embedding FIFOs) and also APB transfers in a concurrent way, this generates a data corruption (multiple DMA access).

When this condition occurs:

- The data transferred by the DMA to the AHB peripherals could be corrupted in case of a FIFO target.
- For memories, it will result in multiple access (not visible by the Software) and the data is not corrupted.
- For the DCMI, a multiple unacknowledged request could be generated, which implies an unknown behavior of the DMA.

AHB peripherals embedding FIFO are DCMI, CRYPTO, and HASH. On sales types without CRYPTO, only the DCMI is impacted. External FIFO controlled by the FSMC is also impacted.

Workaround

Avoid concurrent AHB (DCMI, CRYPTO, HASH, FSMC with external FIFO) and APB transfer management using the DMA2.

2.1.7 Slowing down APB clock during a DMA transfer**Description**

When the CPU modifies the APB clock (slows down the clock: changes AHB/APB prescaler from 1 to 2, 1 to 4, 1 to 8 or 1 to 16) while the DMA is performing a write access to the same APB peripherals, the current DMA transfer will be blocked. Only system reset will recover.

Workaround

Before slowing down the APB clock, wait until the end of the DMA transfer on this APB.

2.1.8 MPU attribute to RTC and IWDG registers could be managed incorrectly**Description**

If the MPU is used and the non bufferable attribute is set to the RTC or IWDG memory map region, the CPU access to the RTC or IWDG registers could be treated as bufferable, provided that there is no APB prescaler configured (AHB/APB prescaler is equal to 1).

Workaround

If the non bufferable attribute is required for these registers, the software could perform a read after the write to guaranty the completion of the write access.

2.1.9 Delay after an RCC peripheral clock enabling**Description**

A delay between an RCC peripheral clock enable and the effective peripheral enabling should be taken into account in order to manage the peripheral read/write to registers.

This delay depends on the peripheral's mapping:

- If the peripheral is mapped on AHB: the delay should be equal to 2 AHB cycles.
- If the peripheral is mapped on APB: the delay should be equal to 1 + (AHB/APB prescaler) cycles.

Workarounds

1. Use the DSB instruction to stall the Cortex-M CPU pipeline until the instruction is completed.
2. Insert "n" NOPs between the RCC enable bit write and the peripheral register writes (n = 2 for AHB peripherals, n = 1 + AHB/APB prescaler in case of APB peripherals).

2.1.10 Battery charge monitoring lower than 2.4 V

Description

If ($V_{DD} = V_{DDA}$) is lower than or equal to 2.4 V, the V_{BAT} conversion correctness is not guaranteed in full temperature and voltage ranges. When V_{BAT} is set, the voltage divider bridge is enabled and $V_{BAT}/2$ is connected to the ADC input. In order to monitor the battery charge correctly, the input of the ADC must not be higher than ($V_{DDA} - 0.6$ V).

Thus, $V_{BAT}/2 < V_{DD} - 0.6$ V implies that $V_{DD} > 2.4$ V.

Workaround

None. ($V_{DD} = V_{DDA}$) should be greater than 2.4 V.

2.1.11 Internal noise impacting the ADC accuracy

Description

An internal noise generated on V_{DD} supplies and propagated internally may impact the ADC accuracy.

This noise is always active whatever the power mode of the MCU (RUN or Sleep).

Workarounds

Two steps could be followed to adapt the accuracy level to the application requirements:

1. Configure the Flash ART as Prefetch OFF and (Data + Instruction) cache ON.
2. Use averaging and filtering algorithms on ADC output codes.

For more workaround details of this limitation, please refer to AN4073.

2.2 IWDG peripheral limitation

2.2.1 RVU and PVU flags are not reset in STOP mode

Description

The RVU and PVU flags of the IWDG_SR register are set by hardware after a write access to the IWDG_RLR and the IWDG_PR registers, respectively. If the Stop mode is entered immediately after the write access, the RVU and PVU flags are not reset by hardware.

Before performing a second write operation to the IWDG_RLR or the IWDG_PR register, the application software must wait for the RVU or PVU flag to be reset. However, since the RVU/PVU bit is not reset after exiting the Stop mode, the software goes into an infinite loop and the independent watchdog (IWDG) generates a reset after the programmed timeout period.

Workaround

Wait until the RVU or PVU flag of the IWDG_SR register is reset before entering the Stop mode.

2.3 I²C peripheral limitations

2.3.1 SMBus standard not fully supported

Description

The I²C peripheral is not fully compliant with the SMBus v2.0 standard since It does not support the capability to NACK an invalid byte/command.

Workarounds

A higher-level mechanism should be used to verify that a write operation is being performed correctly at the target device, such as:

1. Using the SMBAL pin if supported by the host
2. the alert response address (ARA) protocol
3. the Host notify protocol

2.3.2 Start cannot be generated after a misplaced Stop

Description

If a master generates a misplaced Stop on the bus (bus error), the peripheral cannot generate a Start anymore.

Workaround

In the I²C standard, it is allowed to send a Stop only at the end of the full byte (8 bits + acknowledge), so this scenario is not allowed. Other derived protocols like CBUS allow it, but they are not supported by the I²C peripheral.

A software workaround consists in asserting the software reset using the SWRST bit in the I2C_CR1 control register.

2.3.3 Mismatch on the “Setup time for a repeated Start condition” timing parameter

Description

In case of a repeated Start, the “Setup time for a repeated Start condition” (named Tsu;sta in the I²C specification) can be slightly violated when the I²C operates in Master Standard mode at a frequency between 88 kHz and 100 kHz.

The issue can occur only in the following configuration:

- in Master mode
- in Standard mode at a frequency between 88 kHz and 100 kHz (no issue in Fast-mode)
- SCL rise time:
 - If the slave does not stretch the clock and the SCL rise time is more than 300 ns (if the SCL rise time is less than 300 ns, the issue cannot occur)
 - If the slave stretches the clock

The setup time can be violated independently of the APB peripheral frequency.

Workaround

Reduce the frequency down to 88 kHz or use the I²C Fast-mode, if supported by the slave.

2.3.4 Data valid time ($t_{VD;DAT}$) violated without the OVR flag being set**Description**

The data valid time ($t_{VD;DAT}$, $t_{VD;ACK}$) described by the I²C standard can be violated (as well as the maximum data hold time of the current data ($t_{HD;DAT}$)) under the conditions described below. This violation cannot be detected because the OVR flag is not set (no transmit buffer underrun is detected).

This issue can occur only under the following conditions:

- in Slave transmit mode
- with clock stretching disabled (NOSTRETCH=1)
- if the software is late to write the DR data register, but not late enough to set the OVR flag (the data register is written before)

Workaround

If the master device allows it, use the clock stretching mechanism by programming the bit NOSTRETCH=0 in the I2C_CR1 register.

If the master device does not allow it, ensure that the software is fast enough when polling the TXE or ADDR flag to immediately write to the DR data register. For instance, use an interrupt on the TXE or ADDR flag and boost its priority to the higher level.

2.4 I2S peripheral limitations**2.4.1 Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronisation mode****Description**

When I2S is master with PCM long synchronization is selected as 16-bit data frame extended to 32-bit, the WS signal is generated every 16 bits rather than every 32 bits.

Workaround

Only the 16-bit mode with no data extension can be used when the I2S is master and when the selected mode has to be PCM long synchronization mode.

2.4.2 In I2S slave mode, WS level must be set by the external master when enabling the I2S**Description**

In slave mode, the WS signal level is used only to start the communication. If the I2S (in slave mode) is enabled while the master is already sending the clock and the WS signal level is low (for I2S protocol) or is high (for the LSB or MSB-justified mode), the slave starts

communicating data immediately. In this case, the master and slave will be desynchronized throughout the whole communication.

Workaround

The I2S peripheral must be enabled when the external master sets the WS line at:

- High level when the I2S protocol is selected.
- Low level when the LSB or MSB-justified mode is selected.

2.4.3 I2S slave mode desynchronization with the master during communication

Description

In I2S slave mode, if glitches on SCK or WS signals are generated at an unexpected time, a desynchronization of the master and the slave occurs. No error is reported to allow audio system to re-synchronize.

Workaround

The following workarounds can be applied in order to detect and react after a desynchronization by disabling and enabling I2S peripheral in order to resynchronize with the master.

1. Monitoring the I2S WS signal through an external Interrupt to check the I2S WS signal status.
2. Monitoring the I2S clock signal through an input capture interrupt to check the I2S clock signal status.
3. Monitoring the I2S clock signal through an input capture interrupt and the I2S WS signal via an external interrupt to check the I2S clock and I2S WS signals status.

2.5 USART peripheral limitations

2.5.1 Idle frame is not detected if receiver clock speed is deviated

Description

If the USART receives an idle frame followed by a character, and the clock of the transmitter device is faster than the USART receiver clock, the USART receive signal falls too early when receiving the character start bit, with the result that the idle frame is not detected (IDLE flag is not set).

Workaround

None.

2.5.2 In full duplex mode, the Parity Error (PE) flag can be cleared by writing to the data register

Description

In full duplex mode, when the Parity Error flag is set by the receiver at the end of a reception, it may be cleared while transmitting by reading the USART_SR register to check the TXE or TC flags and writing data to the data register.

Consequently, the software receiver can read the PE flag as '0' even if a parity error occurred.

Workaround

The Parity Error flag should be checked after the end of reception and before transmission.

2.5.3 Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection

Description

The USART receiver is in Mute mode and is configured to exit the Mute mode using the address mark detection. When the USART receiver recognizes a valid address with a parity error, it exits the Mute mode without setting the Parity Error flag.

Workaround

None.

2.5.4 Break frame is transmitted regardless of nCTS input line status

Description

When CTS hardware flow control is enabled (CTSE = 1) and the Send Break bit (SBK) is set, the transmitter sends a break frame at the end of the current transmission regardless of nCTS input line status.

Consequently, if an external receiver device is not ready to accept a frame, the transmitted break frame is lost.

Workaround

None.

2.5.5 nRTS signal abnormally driven low after a protocol violation

Description

When RTS hardware flow control is enabled, the nRTS signal goes high when data is received. If this data was not read and new data is sent to the USART (protocol violation), the nRTS signal goes back to low level at the end of this new data.

Consequently, the sender gets the wrong information that the USART is ready to receive further data.

On USART side, an overrun is detected which indicates that data has been lost.

Workaround

Workarounds are required only if the other USART device violates the communication protocol, which is not the case in most applications.

Two workarounds can be used:

- After data reception and before reading the data in the data register, the software takes over the control of the nRTS signal as a GPIO and holds it high as long as needed. If the USART device is not ready, the software holds the nRTS pin high, and releases it when the device is ready to receive new data.
- The time required by the software to read the received data must always be lower than the duration of the second data reception. For example, this can be ensured by treating all the receptions by DMA mode.

2.6 OTG_FS peripheral limitations

2.6.1 Data in RxFIFO is overwritten when all channels are disabled simultaneously

Description

If the available RxFIFO is just large enough to host 1 packet + its data status, and is currently occupied by the last received data + its status and, at the same time, the application requests that more IN channels be disabled, the OTG_FS peripheral does not first check for available space before inserting the disabled status of the IN channels. It just inserts them by overwriting the existing data payload.

Workaround

Use one of the following recommendations:

1. Configure the RxFIFO to host a *minimum* of $2 \times \text{MPSIZ} + 2 \times \text{data status entries}$.
2. The application has to check the RXFLVL bit (RxFIFO non-empty) in the OTG_FS_GINTSTS register before disabling each IN channel. If this bit is not set, then the application can disable an IN channel at a time. Each time the application disables an IN channel, however, it first has to check that the RXFLVL bit = 0 condition is true.

2.6.2 OTG host blocks the receive channel when receiving IN packets and no TxFIFO is configured

Description

When receiving data, the OTG_FS core erroneously checks for available TxFIFO space when it should only check for RxFIFO space. If the OTG_FS core cannot see any space allocated for data transmission, it blocks the reception channel and no data is received.

Workaround

Set at least one TxFIFO equal to the maximum packet size. In this way, the host application, which intends to support only IN traffic, also has to allocate some space for the TxFIFO.

Since a USB host is expected to support any kind of connected endpoint, it is good practice to always configure enough TxFIFO space for OUT endpoints.

2.6.3 Host channel-halted interrupt not generated when the channel is disabled

Description

When the application enables, then immediately disables the host channel before the OTG_FS host has had time to begin the transfer sequence, the OTG_FS core, as a host, does not generate a channel-halted interrupt. The OTG_FS core continues to operate normally.

Workaround

Do not disable the host channel immediately after enabling it.

2.6.4 Error in software-read OTG_FS_DCFG register values

Description

When the application writes to the DAD and PFIVL bitfields in the OTG_FS_DCFG register, and then reads the newly written bitfield values, the read values may not be correct.

The values written by the application, however, are correctly retained by the core, and the normal operation of the device is not affected.

Workaround

Do not read from the OTG_FS_DCFG register's DAD and PFIVL bitfields just after programming them.

2.6.5 Minimum AHB frequency to guarantee correct operation of USB OTG FS peripheral

Description

In order to guarantee correct operation of the USB OTG FS peripheral, the AHB frequency should be configured to be not less than 14.2 MHz.

Workaround

None.

2.7 Ethernet peripheral limitations

2.7.1 Incorrect layer 3 (L3) checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads

Description

The application provides the per-frame control to instruct the MAC to insert the L3 checksums for TCP, UDP and ICMP packets. When automatic checksum insertion is enabled and the input packet is an IPv6 packet without the TCP, UDP or ICMP payload, then the MAC may incorrectly insert a checksum into the packet. For IPv6 packets without a TCP, UDP or ICMP payload, the MAC core considers the next header (NH) field as the extension header and continues to parse the extension header. Sometimes, the payload data in such

packets matches the NH field for TCP, UDP or ICMP and, as a result, the MAC core inserts a checksum.

Workaround

When the IPv6 packets have a TCP, UDP or ICMP payload, enable checksum insertion for transmit frames, or bypass checksum insertion by using the CIC (checksum insertion control) bits in TDES0 (bits 23:22).

2.7.2 The Ethernet MAC processes invalid extension headers in the received IPv6 frames

Description

In IPv6 frames, there can be zero or some extension headers preceding the actual IP payload. The Ethernet MAC processes the following extension headers defined in the IPv6 protocol: Hop-by-Hop Options header, Routing header and Destination Options header. All extension headers, except the Hop-by-Hop extension header, can be present multiple times and in any order before the actual IP payload. The Hop-by-Hop extension header, if present, has to come immediately after the IPv6's main header.

The Ethernet MAC processes all (valid or invalid) extension headers including the Hop-by-Hop extension headers that are present after the first extension header. For this reason, the GMAC core will accept IPv6 frames with invalid Hop-by-Hop extension headers. As a consequence, it will accept any IP payload as valid IPv6 frames with TCP, UDP or ICMP payload, and then incorrectly update the Receive status of the corresponding frame.

Workaround

None.

2.7.3 MAC stuck in the Idle state on receiving the TxFIFO flush command exactly 1 clock cycle after a transmission completes

Description

When the software issues a TxFIFO flush command, the transfer of frame data stops (even in the middle of a frame transfer). The TxFIFO read controller goes into the Idle state (TFRS=00 in ETH_MACDBGR) and then resumes its normal operation.

However, if the TxFIFO read controller receives the TxFIFO flush command exactly one clock cycle after receiving the status from the MAC, the controller remains stuck in the Idle state and stops transmitting frames from the TxFIFO. The system can recover from this state only with a reset (e.g. a soft reset).

Workaround

Do not use the TxFIFO flush feature.

If TxFIFO flush is really needed, wait until the TxFIFO is empty prior to using the TxFIFO flush command.

2.7.4 Transmit frame data corruption

Frame data corrupted when the TxFIFO is repeatedly transitioning from non-empty to empty and then back to non-empty.

Description

Frame data may get corrupted when the TxFIFO is repeatedly transitioning from non-empty to empty for a very short period, and then from empty to non-empty, without causing an underflow.

This transitioning from non-empty to empty and back to non-empty happens when the rate at which the data is being written to the TxFIFO is almost equal to or a little less than the rate at which the data is being read.

This corruption cannot be detected by the receiver when the CRC is inserted by the MAC, as the corrupted data is used for the CRC computation.

Workaround

Use the Store-and-Forward mode: TSF=1 (bit 21 in ETH_DMAOMR). In this mode, the data is transmitted only when the whole packet is available in the TxFIFO.

2.7.5 MCO PLL clock pins not compatible with Ethernet IEEE802.3 long term jitter specifications

Description

When the clock source output by the microcontroller on the MCO pin is issued from the PLL, the MCO pin cannot be used to deliver a 50 MHz RMII clock input or a 25 MHz MII clock input to the ethernet PHY compliant with the long term jitter maximum value for 1.4 ns specified in the IEEE802.3 standard.

This limitation applies both to MCO1 and MCO2 pins and PLLs.

Workaround

- In MII mode
Use a 25 MHz external crystal to generate the HSE clock and output the clock signal on the MCO pin to clock the PHY
- In RMII mode
Either use an external 50 MHz oscillator to clock the PHY or select a PHY with an internal PLL that is able to generate the 50 MHz RMII clock.

2.8 FSMC peripheral limitations

2.8.1 Dummy read cycles inserted when reading synchronous memories

Description

When performing a burst read access to a synchronous memory, some dummy read accesses are performed at the end of the burst cycle, whatever the type of AHB burst access. However, the extra data values which are read are not used by the FSMC and there is no functional failure. The number of dummy reads corresponds to the AHB data size.

Example: if AHB data size = 32bit and MEMSIZE= 16bit, two extra 16-bit reads will be performed.

Workaround

None.

2.8.2 FSMC synchronous mode and NWAIT signal disabled**Description**

When the FSMC synchronous mode operates with the NWAIT signal disabled, if the polarity (WAITPOL in the FSMC_BCRx register) of the NWAIT signal is identical to that of the NWAIT input signal level, the system hangs and no fault is generated.

Workaround

PD6 (NWAIT signal) must not be connected to AF12 and the NWAIT polarity must be configured to active high (set WAITPOL bit to 1 in FSMC_BCRx register).

2.9 SDIO peripheral limitations**2.9.1 SDIO HW flow control****Description**

When enabling the HW flow control by setting bit 14 of the SDIO_CLKCR register to '1', glitches can occur on the SDIOCLK output clock resulting in wrong data to be written into the SD/MMC card or into the SDIO device. As a consequence, a CRC error will be reported to the SD/SDIO MMC host interface (DCRCFAIL bit set to '1' in SDIO_STA register).

Workaround

None.

Do not use the HW flow control. Overrun errors (Rx mode) and FIFO underrun (Tx mode) should be managed by the application software.

2.9.2 Wrong CCRCFAIL status after a response without CRC is received**Description**

The CRC is calculated even if the response to a command does not contain any CRC field.

As a consequence, after the SDIO command IO_SEND_OP_COND (CMD5) is sent, the CCRCFAIL bit of the SDIO_STA register is set.

Workaround

The CCRCFAIL bit in the SDIO_STA register shall be ignored by the software. CCRCFAIL must be cleared by setting CCRCFAILC bit of the SDIO_ICR register after reception of the response to the CMD5 command.

2.9.3 SDIO clock divider BYPASS mode may not work properly

Description

In high speed communication mode, when SDIO_CLK is equal to 48 MHz (PLL48_output = 48 MHz), the BYPASS bit is equal to '1' and the NEGEDGE bit is equal to '0' (respectively bit 10 and bit 13 in the SDIO_CLKCR register), the hold timing at the I/O pin is not inline with the SD/MMC 2.0 specifications.

Workaround

When not using USB nor RNG, PLL48_output (SDIOCLK) frequency can be raised up to 75 MHz, allowing to reach 37.5 MHz on SDIO_CLK in high speed mode. The BYPASS bit, the CLKDIV bit and the NEGEDGE bit are equal to '0'.

2.9.4 Data corruption in SDIO clock dephasing (NEGEDGE) mode

Description

When NEGEDGE bit is set to '1', it may lead to invalid data and command response read.

Workaround

None. A configuration with the NEGEDGE bit equal to '1' should not be used.

2.9.5 CE-ATA multiple write command and card busy signal management

Description

The CE-ATA card may inform the host that it is busy by driving the SDIO_D0 line low, two cycles after the transfer of a write command (RW_MULTIPLE_REGISTER or RW_MULTIPLE_BLOCK). When the card is in a busy state, the host must not send any data until the BUSY signal is de-asserted (SDIO_D0 released by the card).

This condition is not respected if the data state machine leaves the IDLE state (Write operation programmed and started, DTEN = 1, DTDIR = 0 in SDIO_DCTRL register and TXFIFOE = 0 in SDIO_STA register).

As a consequence, the write transfer fails and the data lines are corrupted.

Workaround

After sending the write command (RW_MULTIPLE_REGISTER or RW_MULTIPLE_BLOCK), the application must check that the card is not busy by polling the BSY bit of the ATA status register using the FAST_IO (CMD39) command before enabling the data state machine.

2.10 DAC limitations

2.10.1 DMA underrun flag management

Description

If the DMA is not fast enough to input the next digital data to the DAC, as a consequence, the same digital data is converted twice. In these conditions, the DMAUDR flag is set, which usually leads to disable the DMA data transfers. This is not the case: the DMA is not disabled by DMAUDR=1, and it keeps servicing the DAC.

Workaround

To disable the DAC DMA stream, reset the EN bit (corresponding to the DAC DMA stream) in the DMA_SxCR register.

2.10.2 DMA request not automatically cleared by DMAEN=0

Description

if the application wants to stop the current DMA-to-DAC transfer, the DMA request is not automatically cleared by DMAEN=0, or by DACEN=0.

If the application stops the DAC operation while the DMA request is high, the DMA request will be pending while the DAC is reinitialized and restarted; with the risk that a spurious unwanted DMA request is serviced as soon as the DAC is re-enabled.

Workaround

To stop the current DMA-to-DAC transfer and restart, the following sequence should be applied:

1. Check if DMAUDR is set.
2. Clear the DAC/DMAEN bit.
3. Clear the EN bit of the DAC DMA/Stream
4. Reconfigure by software the DAC, DMA, triggers etc.
5. Restart the application.

Appendix A Revision code on device marking

Figure 1, Figure 2, Figure 3, Figure 4, Figure 5 and Figure 6 show the marking compositions for the WLCSP64+2, UFBGA176, LQFP176, LQFP144, LQFP100 and LQFP64 packages, respectively. The only fields shown are the Additional field containing the revision code and the Year and Week fields making up the date code.

Figure 1. WLCSP64+2 top package view

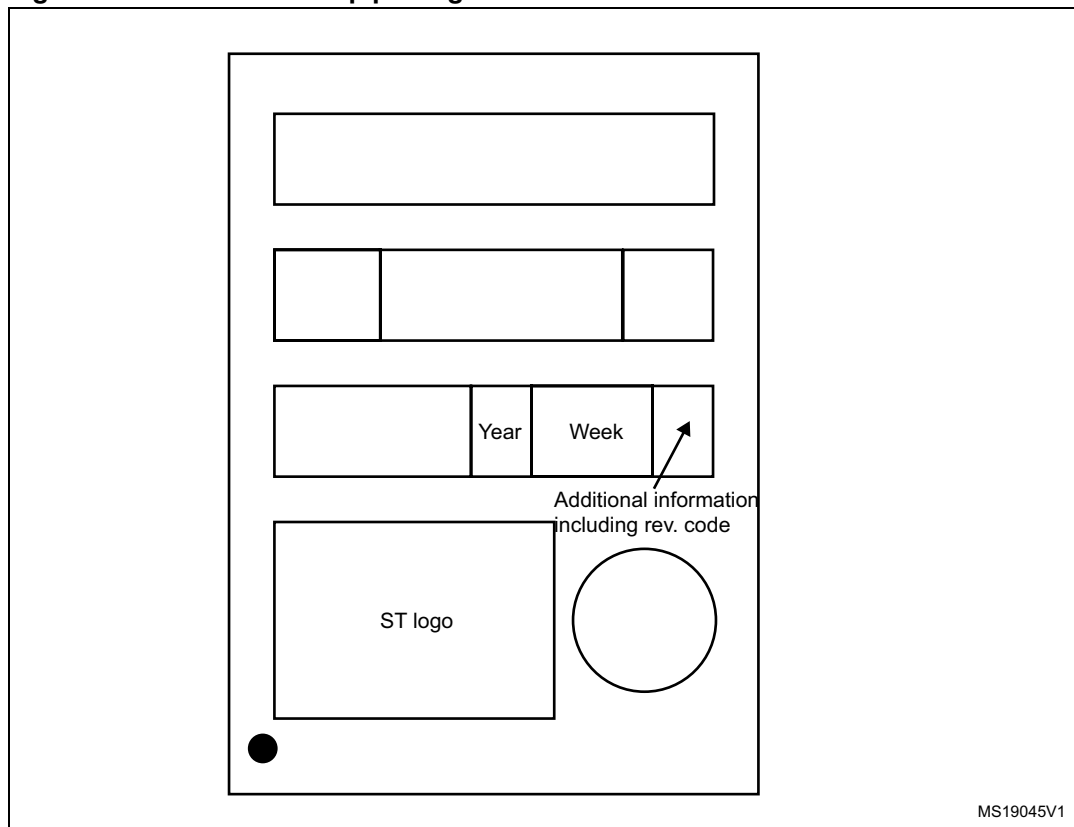


Figure 2. UFBGA176 top package view

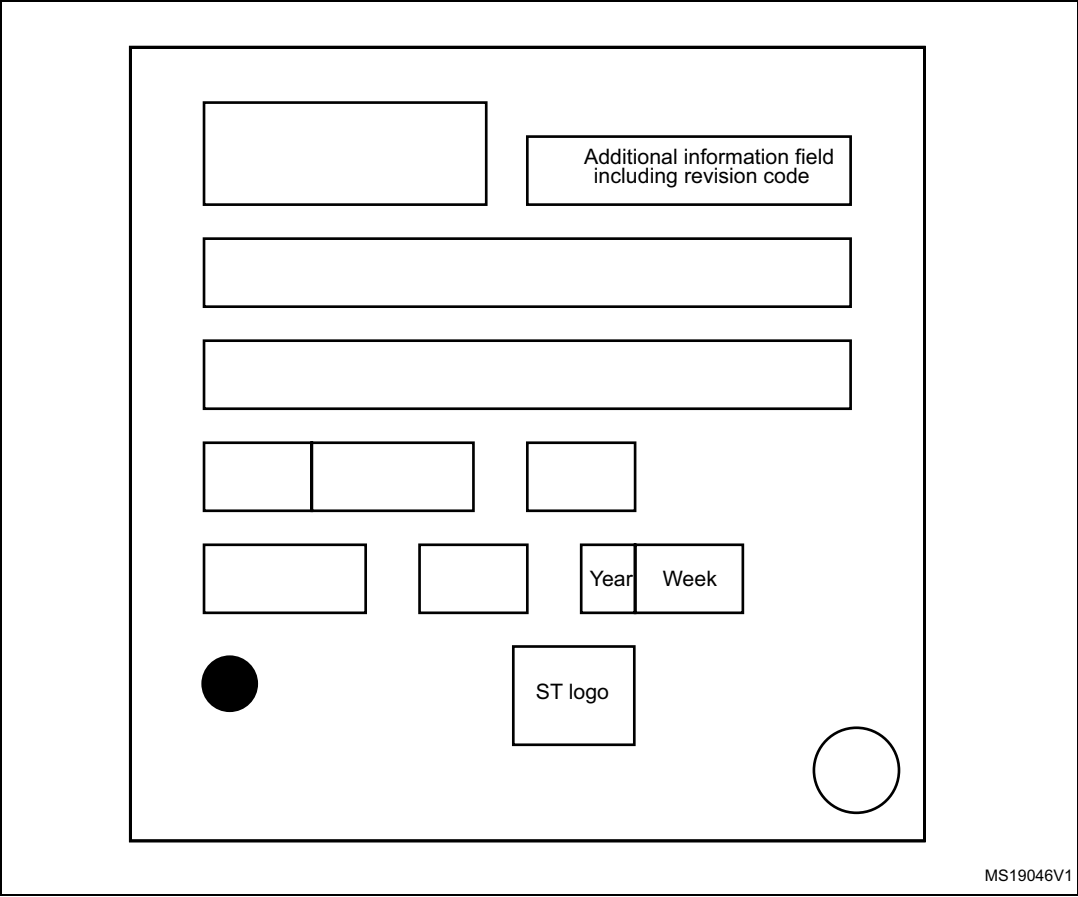


Figure 3. LQFP176 top package view

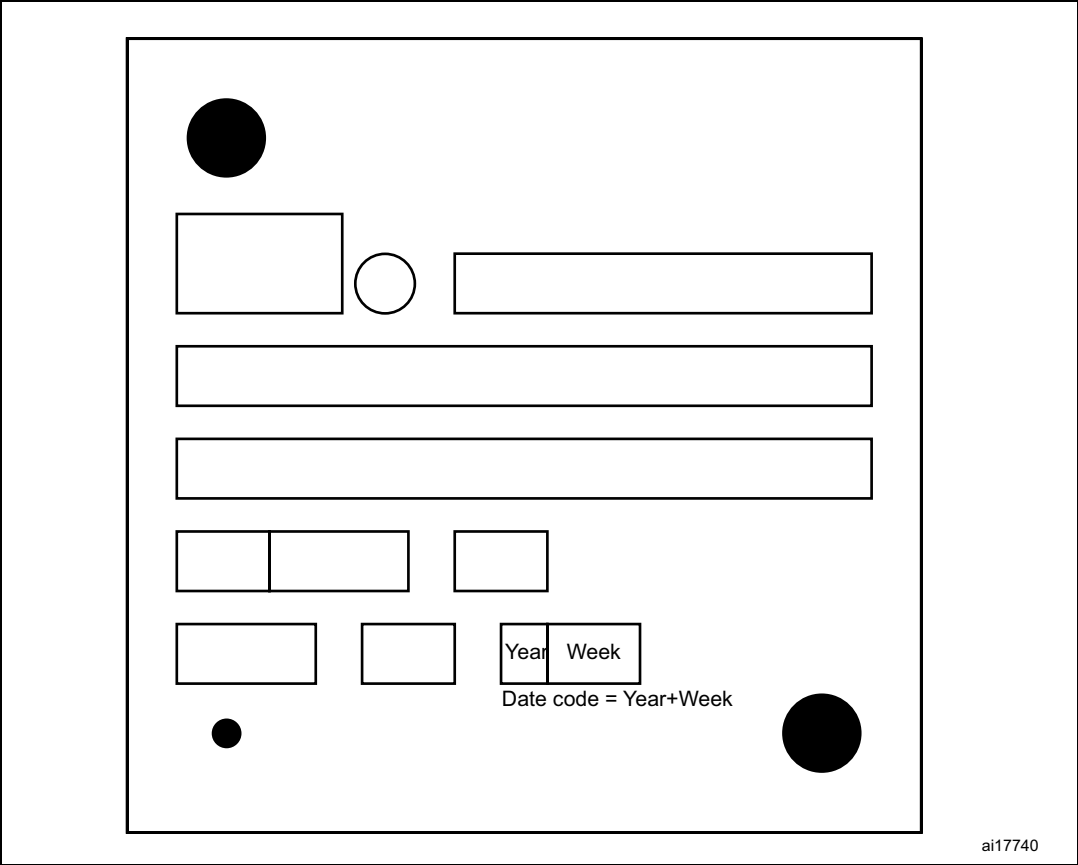


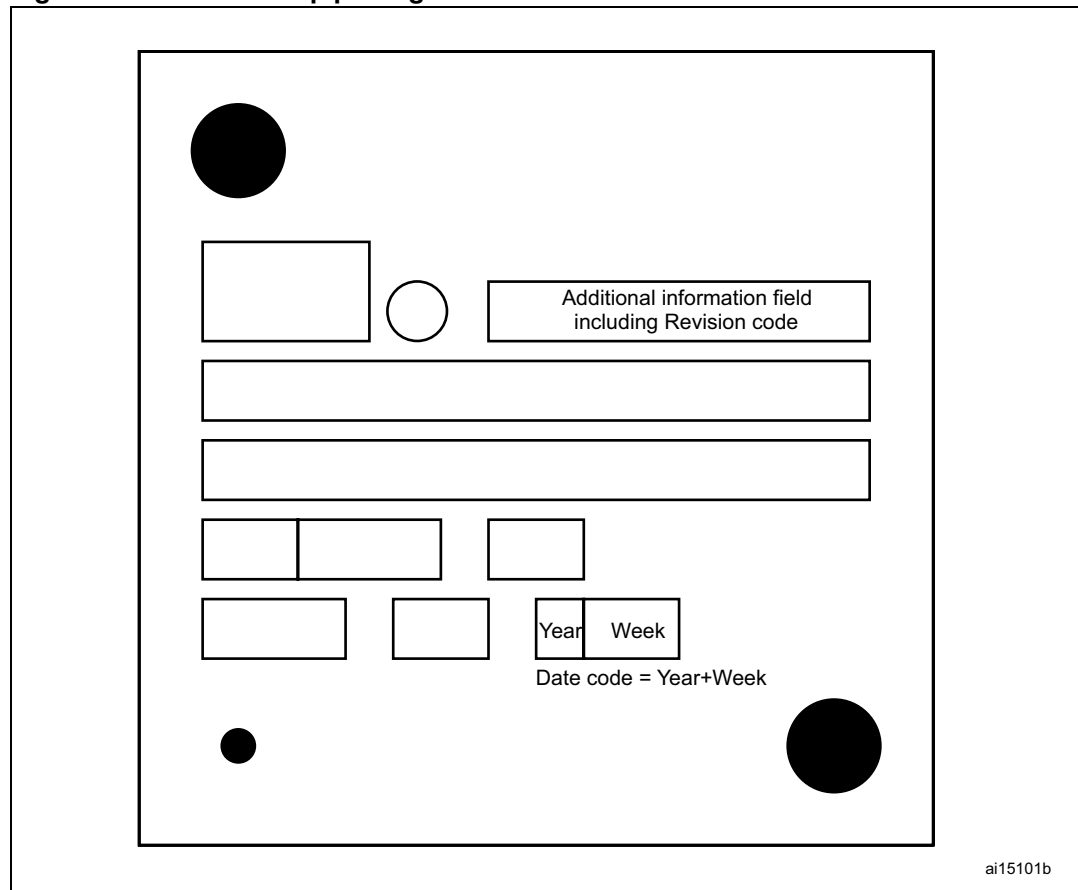
Figure 4. LQFP144 top package view

Figure 5. LQFP100 top package view

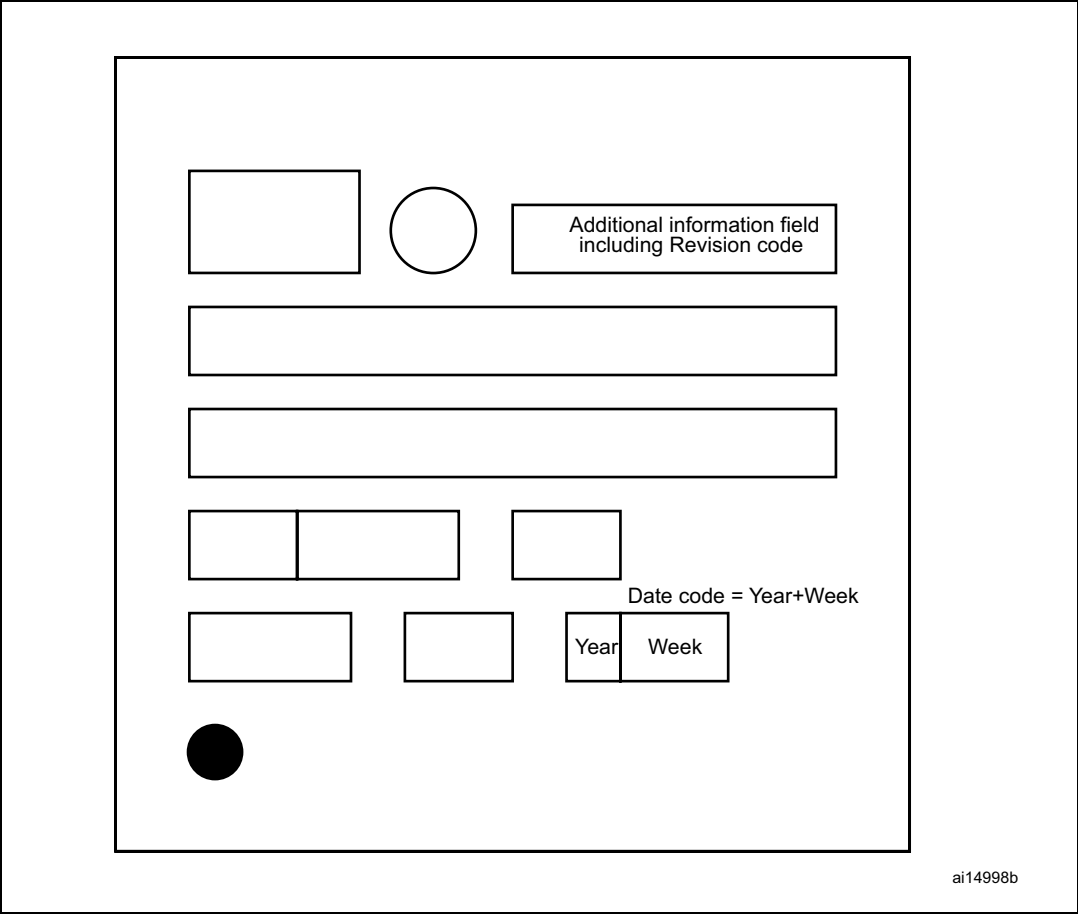
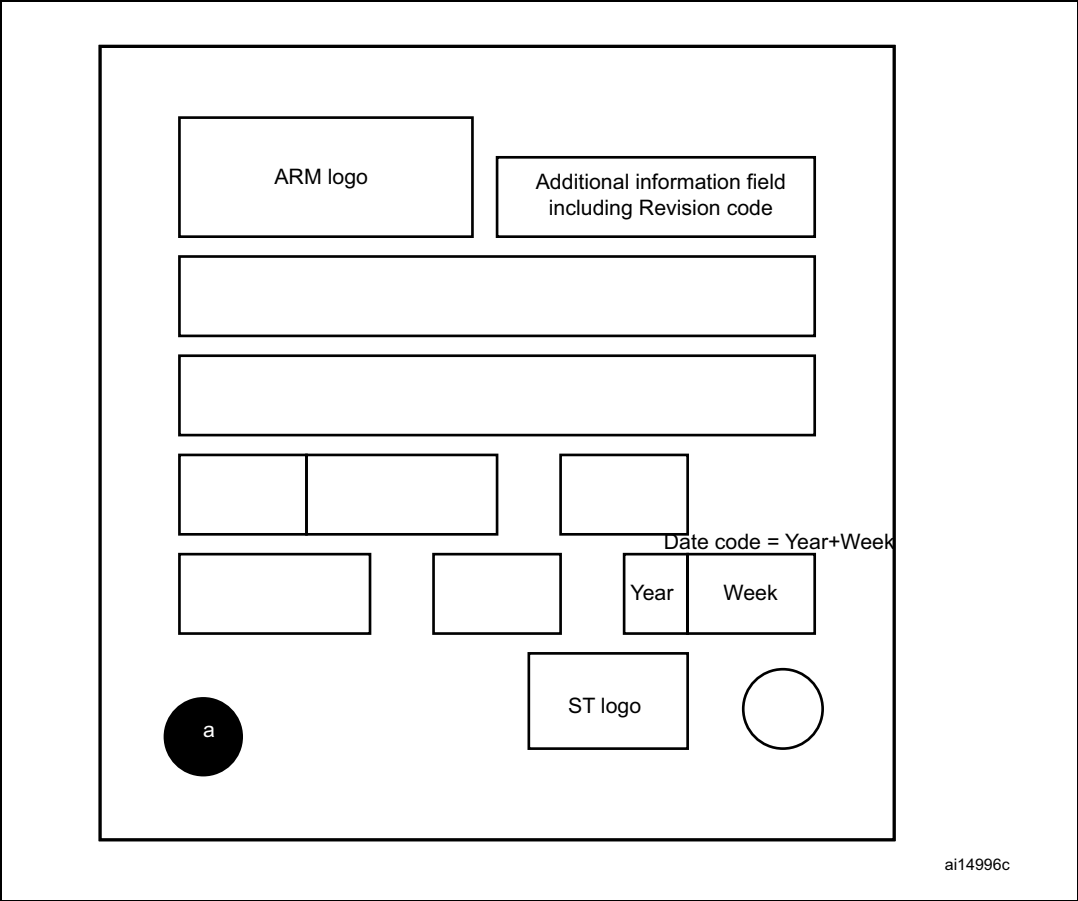


Figure 6. LQFP64 top package view



Revision history

Table 5. Document revision history

Date	Revision	Changes
03-Jun-2011	1	Initial release.
20-Dec-2011	2	<p>Removed cut number in the whole document.</p> <p>Added <i>Section 2.1.1: ART Accelerator prefetch queue instruction is not supported</i>, <i>Section 2.2.1: RVU and PVU flags are not reset in STOP mode</i>, <i>Section 2.1.5: Configuration of PH10 and PI10 as external interrupts is erroneous</i>, and <i>Section 2.9.2: Wrong CCRCFAIL status after a response without CRC is received</i>.</p> <p>Updated <i>Section 2.5.5: nRTS signal abnormally driven low after a protocol violation</i>, <i>Section 2.7.5: MCO PLL clock pins not compatible with Ethernet IEEE802.3 long term jitter specifications</i>,</p>
03-Aug-2012	3	<p>Added <i>Section 2.1.6: DMA2 data corruption when managing AHB and APB peripherals in a concurrent way</i>, <i>Section 2.1.7: Slowing down APB clock during a DMA transfer</i>, <i>Section 2.1.8: MPU attribute to RTC and IWDG registers could be managed incorrectly</i>, <i>Section 2.1.9: Delay after an RCC peripheral clock enabling</i>, <i>Section 2.1.10: Battery charge monitoring lower than 2.4 V</i> and <i>Section 2.1.11: Internal noise impacting the ADC accuracy</i>.</p> <p>Added <i>Section 2.8.2: FSMC synchronous mode and NWAIT signal disabled</i>.</p> <p>Added <i>Section 2.9.3: SDIO clock divider BYPASS mode may not work properly</i>, <i>Section 2.9.4: Data corruption in SDIO clock dephasing (NEGEDGE) mode</i> and <i>Section 2.9.5: CE-ATA multiple write command and card busy signal management</i>.</p> <p>Added <i>Section 2.10: DAC limitations</i> with <i>Section 2.10.1: DMA underrun flag management</i> and <i>Section 2.10.2: DMA request not automatically cleared by DMAEN=0</i>.</p>

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.

UNLESS EXPRESSLY APPROVED IN WRITING BY TWO AUTHORIZED ST REPRESENTATIVES, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER'S OWN RISK.

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.

© 2012 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