STM32F42xx and STM32F43xx Errata sheet

STM32F427/437 and STM32F429/439 line limitations

Silicon identification

This errata sheet applies to the revision A, Y and 1 of STMicroelectronics STM32F427/437 and STM32F429/439 microcontroller lines.

The STM32F42xx and STM32F43xx devices feature an ARM® 32-bit Cortex™-M4 core with FPU, 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)
STM32F427xx, STM32F429xx	"A", "Y", "1"
STM32F437xx, STM32F439xx	

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the RM0090 STM32F4xx 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
STM32F427xx	STM32F427VG, STM32F427ZG, STM32F427IG, STM32F427AG, STM32F427VI, STM32F427ZI, STM32F427II, STM32F427AI
STM32F437xx	STM32F437VG, STM32F437ZG, STM32F437IG, STM32F437VI, STM32F437ZI, STM32F437II, STM32F437AI
STM32F429xx	STM32F429VG, STM32F429ZG, STM32F429IG, STM32F429VI, STM32F429ZI, STM32F429II, STM32F429AI, STM32F429AG, STM32F429BG, STM32F429BI, STM32F429NI, STM32F429NG, STM32F429VE, STM32F429ZE, STM32F429IE, STM32F429BE, STM32F429NE
STM32F439xx	STM32F439VI, STM32F439VG, STM32F439ZG, STM32F439ZI, STM32F439IG, STM32F439II, STM32F439AI, STM32F439BG, STM32F439BI, STM32F439NI, STM32F439NG

Contents

List of tables

List of figures

1 ARM 32-bit Cortex-M4 with FPU limitations

An errata notice of the STM32F42xx and STM32F43xx core is available from the following web address:
http://infocenter.arm.com/help/topic/com.arm.doc.ddi0439b_errata_01/index.html.

All the described limitations are minor and related to the revision r0p1-v1 of the CortexM4 core. *Table 3* summarizes these limitations and their implications on the behavior of STM32F42xx and STM32F43xx devices.

Table 3. Cortex-M4 core limitations and impact on microcontroller behavior

ARM ID	ARM category	ARM summary of errata	Impact on STM32F42xx and STM32F43xx
752419	Cat 2	Interrupted loads to SP can cause erroneous behavior	Minor

1.1 Cortex-M4 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 limitations 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

2 STM32F42xx and STM32F43xx 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		Revision A	Revision Y	Revision 1
Section 2.1: System limitations	*Section 2.1.1: Debugging Stop mode and system tick timer*	A	A	A
	Section 2.1.2: Debugging Stop mode with WFE entry	A	A	A
	Section 2.1.3: Wakeup sequence from Standby mode when using more than one wakeup source	A	A	A
	Section 2.1.4: Full JTAG configuration without NJTRST pin cannot be used	A	A	A
	Section 2.1.5: MPU attribute to RTC and IWDG registers could be managed incorrectly	A	A	A
	Section 2.1.6: Delay after an RCC peripheral clock enabling	A	A	A
	Section 2.1.7: Internal noise impacting the ADC accuracy	A	A	A
	Section 2.1.8: Over-drive and Under-drive modes unavailability	N	-	-
Section 2.2: IWDG peripheral limitation	*Section 2.2.1: RVU and PVU flags are not reset in STOP mode*	A	A	A
Section 2.3: I2C peripheral limitations	*Section 2.3.1: SMBus standard not fully supported*	A	A	A
	Section 2.3.2: Start cannot be generated after a misplaced Stop	A	A	A
	Section 2.3.3: Mismatch on the "Setup time for a repeated Start condition" timing parameter	A	A	A
	Section 2.3.4: Data valid time ($t_{VD;DAT}$) violated without the OVR flag being set	A	A	A
	Section 2.3.5: Both SDA and SCL maximum rise time (t_r) violated when VDD_I2C bus higher than ((VDD+0.3) / 0.7) V	A	A	A
Section 2.4: I2S peripheral limitation	*Section 2.4.1: In I2S slave mode, WS level must be set by the external master when enabling the I2S*	A	A	A

Table 4. Summary of silicon limitations (continued)

Links to silicon limitations		Revision A	Revision Y	Revision 1
Section 2.5: USART peripheral limitations	*Section 2.5.1: Idle frame is not detected if receiver clock speed is deviated*	N	N	N
	Section 2.5.2: In full duplex mode, the Parity Error (PE) flag can be cleared by writing to the data register	A	A	A
	Section 2.5.3: Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection	N	N	N
	Section 2.5.4: Break frame is transmitted regardless of nCTS input line status	N	N	N
	Section 2.5.5: nRTS signal abnormally driven low after a protocol violation	A	A	A
Section 2.6: OTG_FS peripheral limitations	*Section 2.6.1: Data in RxFIFO is overwritten when all channels are disabled simultaneously*	A	A	A
	Section 2.6.2: OTG host blocks the receive channel when receiving IN packets and no TxFIFO is configured	A	A	A
	Section 2.6.3: Host channel-halted interrupt not generated when the channel is disabled	A	A	A
	Section 2.6.4: Error in software-read OTG_FS_DCFG register values	A	A	A
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	A	A
	Section 2.7.2: The Ethernet MAC processes invalid extension headers in the received IPv6 frames	N	N	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	A	A
	Section 2.7.4: Transmit frame data corruption	A	A	A
	Section 2.7.5: Successive write operations to the same register might not be fully taken into account	A	A	A

Table 4. Summary of silicon limitations (continued)

Links to silicon limitations		Revision A	Revision Y	Revision 1
Section 2.8: FMC peripheral limitation	*Section 2.8.1: Dummy read cycles inserted when reading synchronous memories*	N	N	N
	Section 2.8.2: FMC synchronous mode and NWAIT signal disabled	A	A	A
	Section 2.8.3: Read access to a non-initialized FMC_SDRAM bank	P	P	P
	Section 2.8.4: Corruption of data read from the FMC	A	-	-
	Section 2.8.5: Interruption of CPU read burst access to an end of SDRAM row	A	A	A
	Section 2.8.6: FMC NOR/PSRAM controller: asynchronous read access on bank 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set)	A	A	-
	Section 2.8.7: FMC dynamic and static bank switching	A	A	A
Section 2.9: SDIO peripheral limitations	*Section 2.9.1: SDIO HW flow control*	N	N	N
	Section 2.9.2: Wrong CCRCFAIL status after a response without CRC is received	A	A	A
	Section 2.9.3: Data corruption in SDIO clock dephasing (NEGEDGE) mode	N	N	N
	Section 2.9.4: CE-ATA multiple write command and card busy signal management	A	A	A
	Section 2.9.5: No underrun detection with wrong data transmission	A	A	A
Section 2.10: ADC peripheral limitations	*Section 2.10.1: ADC sequencer modification during conversion*	A	A	A
Section 2.11: DAC peripheral limitations	*Section 2.11.1: DMA underrun flag management*	A	A	A
	Section 2.11.2: DMA request not automatically cleared by DMAEN=0	A	A	A

2.1 System limitations

2.1.1 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.

2.1.2 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 }
```

2.1.3 Wakeup sequence from Standby mode when using more than one wakeup source

Description

The various wakeup sources are logically OR-ed in front of the rising-edge detector which generates the wakeup flag (WUF). The WUF needs to be cleared prior to Standby mode entry, otherwise the MCU wakes up immediately.

If one of the configured wakeup sources is kept high during the clearing of the WUF (by setting the CWUF bit), it may mask further wakeup events on the input of the edge detector. As a consequence, the MCU might not be able to wake up from Standby mode.

Workaround

To avoid this problem, the following sequence should be applied before entering Standby mode:

- Disable all used wakeup sources,
- Clear all related wakeup flags,
- Re-enable all used wakeup sources,
- Enter Standby mode

Note: Be aware that, when applying this workaround, if one of the wakeup sources is still kept high, the MCU enters Standby mode but then it wakes up immediately generating a power reset.

2.1.4 Full JTAG configuration without NJTRST pin cannot be used

Description

When using the JTAG debug port in debug mode, the connection with the debugger is lost if the NJTRST pin (PB4) is used as a GPIO. Only the 4-wire JTAG port configuration is impacted.

Workaround

Use the SWD debug port instead of the full 4-wire JTAG port.

2.1.5 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.

2.1.6 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).

2.1.7 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

To adapt the accuracy level to the application requirements, set one of the following options:

- Option1
  Set the ADCDC1 bit in the PWR_CR register.
- Option2
  Set the corresponding ADCxDC2 bit in the SYSCFG_PMC register.

Only one option can be set at a time.

For more details on option 1 and option2 mechanisms, refer to AN4073.

2.1.8 Over-drive and Under-drive modes unavailability

Description

The Over-drive and Under-drive modes are not available on revision A devices.

Workaround

None.

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 I^2C 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 limitation can occur only in the following configuration:

- in Master mode
- in Standard mode at a frequency between 88 kHz and 100 kHz (no limitation 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 limitation 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 limitation 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.

2.3.5 Both SDA and SCL maximum rise time (t_r) violated when VDD_I2C bus higher than ((VDD+0.3) / 0.7) V

Description

When an external legacy I^{2}C bus voltage (VDD_I2C) is set to 5 V while the MCU is powered from V_{DD}, the internal 5-Volt tolerant circuitry is activated as soon the input voltage (V_{IN}) reaches the V_{DD} + diode threshold level. An additional internal large capacitance then prevents the external pull-up resistor (R_P) from rising the SDA and SCL signals within the maximum timing (t_r) which is 300 ns in fast mode and 1000 ns in Standard mode.

The rise time (t_r) is measured from V_{IL} and V_{IH} with levels set at 0.3VDD_I2C and 0.7VDD_I2C.

Workaround

The external VDD_I2C bus voltage should be limited to a maximum value of ((VDD+0.3) / 0.7) V. As a result, when the MCU is powered from V_{DD}=3.3 V, VDD_I2C should not exceed 5.14 V to be compliant with I^{2}C specifications.

2.4 I2S peripheral limitation

2.4.1 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.

2.5 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 × MPSIZ + 2 × 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 supports 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.

2.7 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 Successive write operations to the same register might not be fully taken into account

Description

A write to a register might not be fully taken into account if a previous write to the same register is performed within a time period of four TX_CLK/RX_CLK clock cycles. When this error occurs, reading the register returns the most recently written value, but the Ethernet MAC continues to operate as if the latest write operation never occurred.

See *Table 5: Impacted registers and bits* for the registers and bits impacted by this limitation.

Table 5. Impacted registers and bits

Register name	Bit number	Bit name
DMA registers		
ETH_DMABMR	7	EDFE
ETH_DMAOMR	26	DTCEFD
	25	RSF
	20	FTF
	7	FEF
	6	FUGF
	4:3	RTC
GMAC registers		
ETH_MACCR	25	CSTF
	23	WD
	22	JD
	19:17	IFG
	16	CSD
	14	FES
	13	ROD
	12	LM
	11	DM
	10	IPCO
	9	RD
	7	APCS
	6:5	BL
	4	DC
	3	TE
	2	RE
ETH_MACFFR		MAC frame filter register
ETH_MACHTHR	31:0	Hash Table High Register
ETH_MACHTLR	31:0	Hash Table Low Register
ETH_MACFCR	31:16	PT
	7	ZQPD
	5:4	PLT
	3	UPFD
	2	RFCE
	1	TFCE
	0	FCB/BPA

Table 5. Impacted registers and bits (continued)

Register name	Bit number	Bit name
ETH_MACVLANTR	16	VLANTC
	15:0	VLANTI
ETH_MACRWUFFR		all remote wakeup registers
ETH_MACPMTCSR	31	WFFRPR
	9	GU
	2	WFE
	1	MPE
	0	PD
ETH_MACA0HR		MAC address 0 high register
ETH_MACA0LR		MAC address 0 low register
ETH_MACA1HR		MAC address 1 high register
ETH_MACA1LR		MAC address 1 low register
ETH_MACA2HR		MAC address 2 high register
ETH_MACA2LR		MAC address 2 low register
ETH_MACA3HR		MAC address 3 high register
ETH_MACA3LR		MAC address 3 low register
IEEE 1588 time stamp registers		
ETH_PTPTSCR	18	TSPFFMAE
	17:16	TSCNT
	15	TSSMRME
	14	TSSEME
	13	TSSIPV4FE
	12	TSSIPV6FE
	11	TSSPTPOEFE
	10	TSPTPPSV2E
	9	TSSSR
	8	TSSARFE
	5	TSARU
	3	TSSTU
	2	TSSTI
	1	TSFCU
	0	TSE

Workaround

Two workarounds could be applicable:

- Ensure a delay of four TX_CLK/RX_CLK clock cycles between the successive write operations to the same register.
- Make several successive write operations without delay, then read the register when all the operations are complete, and finally reprogram it after a delay of four TX_CLK/RX_CLK clock cycles.

2.8 FMC peripheral limitation

2.8.1 Dummy read cycles inserted when reading synchronous memories

Description

When performing a burst read access to a synchronous memory, two 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 FMC and there is no functional failure.

Workaround

None.

2.8.2 FMC synchronous mode and NWAIT signal disabled

Description

When the FMC synchronous mode operates with the NWAIT signal disabled, if the polarity (WAITPOL in the FMC_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 FMC_BCRx register).

2.8.3 Read access to a non-initialized FMC_SDRAM bank

Description

When a read access is performed to an SDRAM bank while the SDRAM controller is not yet initialized, the system hangs and no fault is generated.

Workaround

Read access to an SDRAM bank must be performed only when the SDRAM controller initialization is complete.

2.8.4 Corruption of data read from the FMC

Description

When the FMC is used as stack, heap or variable data, an interrupt occurring during a CPU read access to the FMC may results in read data corruption or hard fault exception. This problem does not occur when read accesses are performed by another master or when FMC accesses are done when the interrupts are disabled.

Workaround

Two workarounds can be applied:

- Do not use the FMC as stack or heap, and make sure CPU read accesses to the FMC are performed while interrupts are disabled
- Use only DMAs to perform read accesses to the FMC.

2.8.5 Interruption of CPU read burst access to an end of SDRAM row

Description

If an interrupt occurs during an CPU AHB burst read access to an end of SDRAM row, it may result in wrong data read from the next row if all the conditions below are met:

- The SDRAM data bus is 16-bit or 8-bit wide. 32-bit SDRAM mode is not affected.
- RBURST bit is reset in the FMC_SDCR1 register (read FIFO disabled).
- An interrupt occurs while CPU is performing an AHB incrementing bursts read access of unspecified length (using LDM = Load Multiple instruction).
- The address of the burst operation includes the end of an SDRAM row.

Workaround

Enable the read FIFO by setting the RBURST bit in the FMC_SDCR1 register.

2.8.6 FMC NOR/PSRAM controller: asynchronous read access on bank 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set)

Description

If an interrupt occurs during a CPU AHB read access to one NOR/PSRAM bank (bank2 to 4) which is enabled in asynchronous mode, while bank 1 of the NOR/PSRAM controller is configured in synchronous read mode (BURSTEN bit set to ‘1’), then the FMC NOR/PSRAM controller returns wrong data.

This limitation does not occur when using the DMA or when only bank1 is used in synchronous mode.

Workaround

If multiple banks are enabled in mixed asynchronous and synchronous modes, use any NOR/PSRAM bank for synchronous read accesses, except for bank 1. As a consequence the continuous clock feature is not available in asynchronous mode.

2.8.7 FMC dynamic and static bank switching

Description

The dynamic and static banks cannot be accessed concurrently.

Workaround

Do not use dynamic and static banks at the same time. The SDRAM device must be in self-refresh before switching to the static memory mapped on the NOR/PSRAM or NAND/PC-Card controller. Before switching from static memory to SDRAM, issue a Normal command to wake-up the device from self-refresh mode.

This limitation will be fixed in next silicon revision.

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.

Note: 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 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.

2.9.4 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.

2.9.5 No underrun detection with wrong data transmission

Description

In case there is an ongoing data transfer from the SDIO host to the SD card and the hardware flow control is disabled (bit 14 of the SDIO_CLKCR is not set), if an underrun condition occurs, the controller may transmit a corrupted data block (with wrong data word) without detecting the underrun condition when the clock frequencies have the following relationship:

[3 x period(PCLK2) + 3 x period(SDIOCLK)] >= (32 / (BusWidth)) x period(SDIO_CK)

Workaround

Avoid the above-mentioned clock frequency relationship, by:

- Incrementing the APB frequency
- or decreasing the transfer bandwidth
- or reducing SDIO_CK frequency

2.10 ADC peripheral limitations

2.10.1 ADC sequencer modification during conversion

Description

If an ADC conversion is started by software (writing the SWSTART bit), and if the ADC_SQRx or ADC_JSQRx registers are modified during the conversion, the current conversion is reset and the ADC does not restart a new conversion sequence automatically.

If an ADC conversion is started by hardware trigger, this limitation does not apply. The ADC restarts a new conversion sequence automatically.

Workaround

When an ADC conversion sequence is started by software, a new conversion sequence can be restarted only by setting the SWSTART bit in the ADC_CR2 register.

2.11 DAC peripheral limitations

2.11.1 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.

2.11.2 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* and *Figure 5*, *Figure 6*, *Figure 7* show the marking compositions for the TFBGA216, WLCSP143, LQFP208, UFBGA176, LQFP176, LQFP144 and LQFP100 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. TFBGA216 top package view

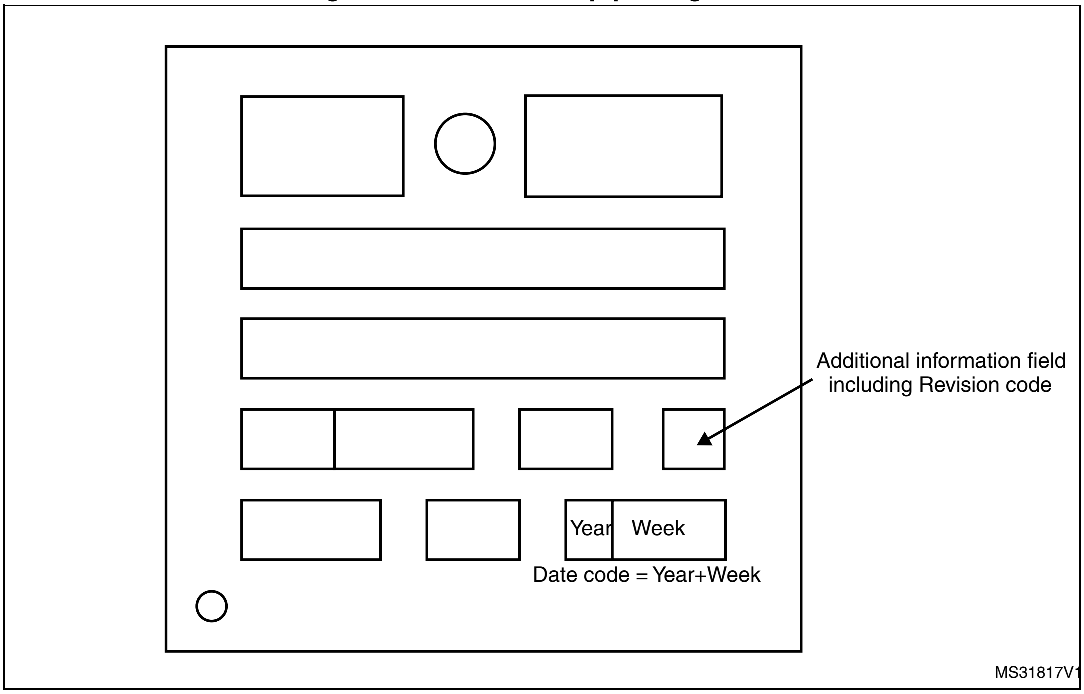


Figure 2. WLCSP143 top package view

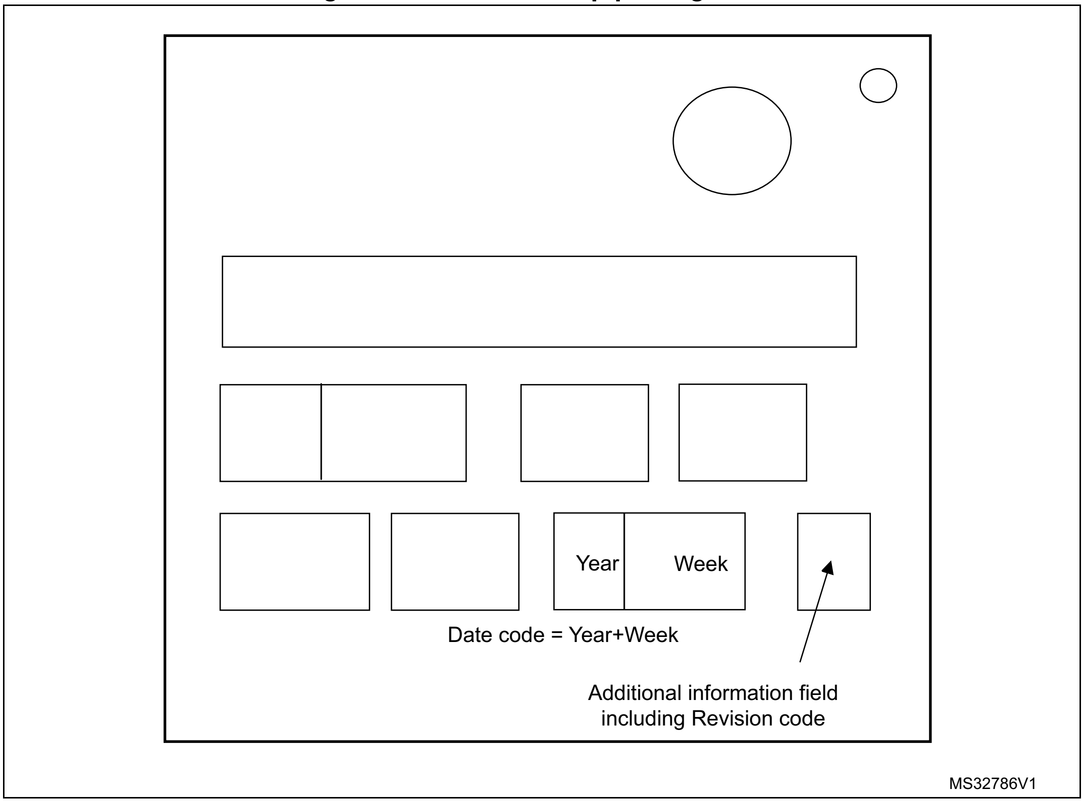


Figure 3. LQFP208 top package view

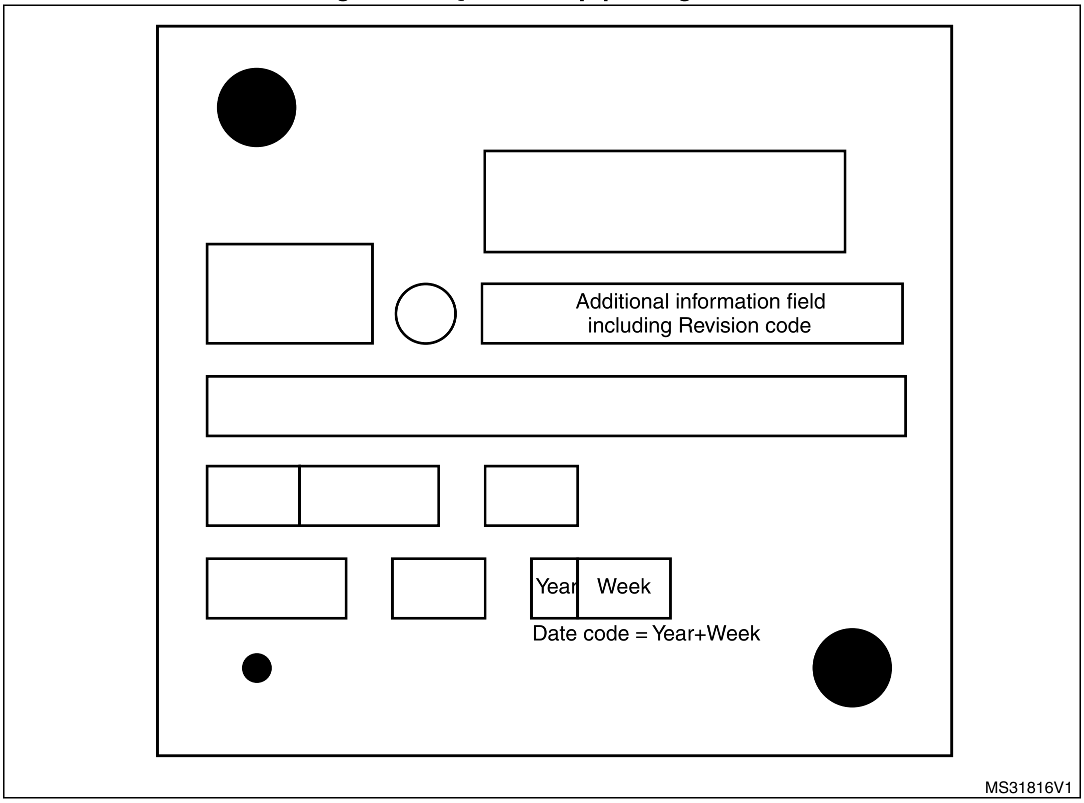


Figure 4. UFBGA176 top package view

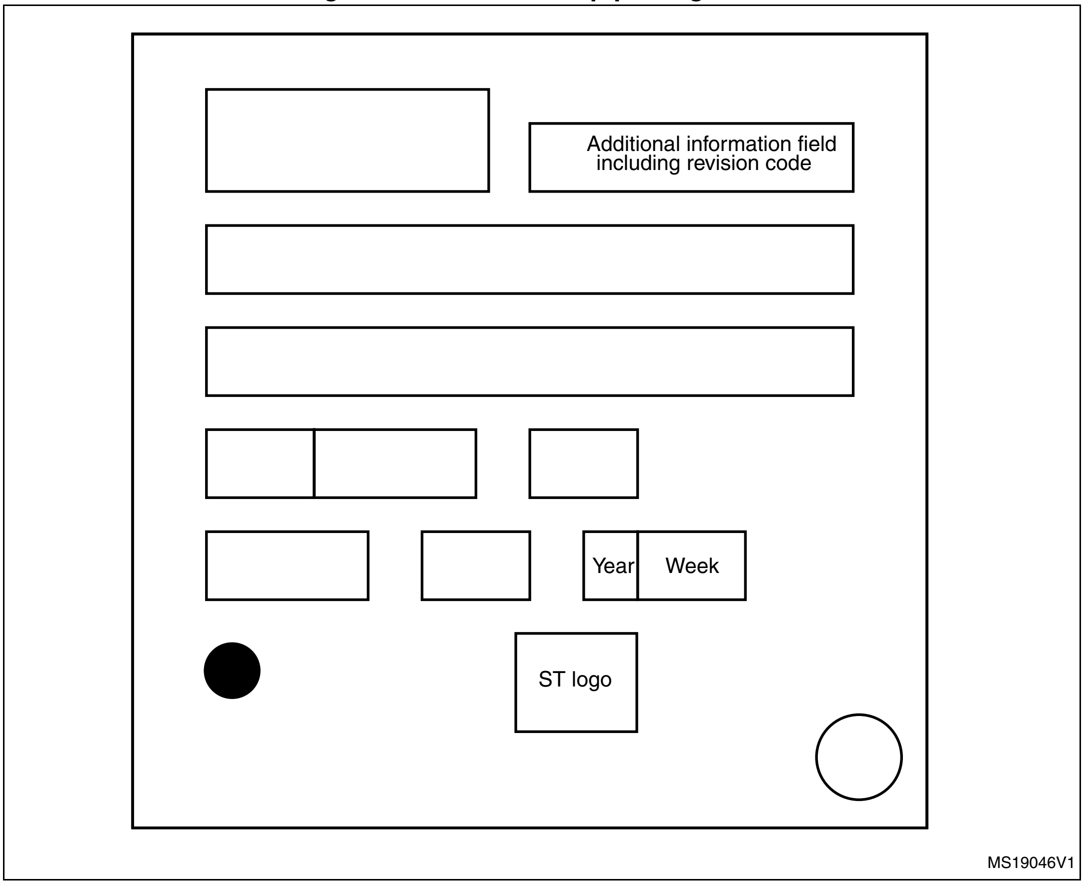


Figure 5. LQFP176 top package view

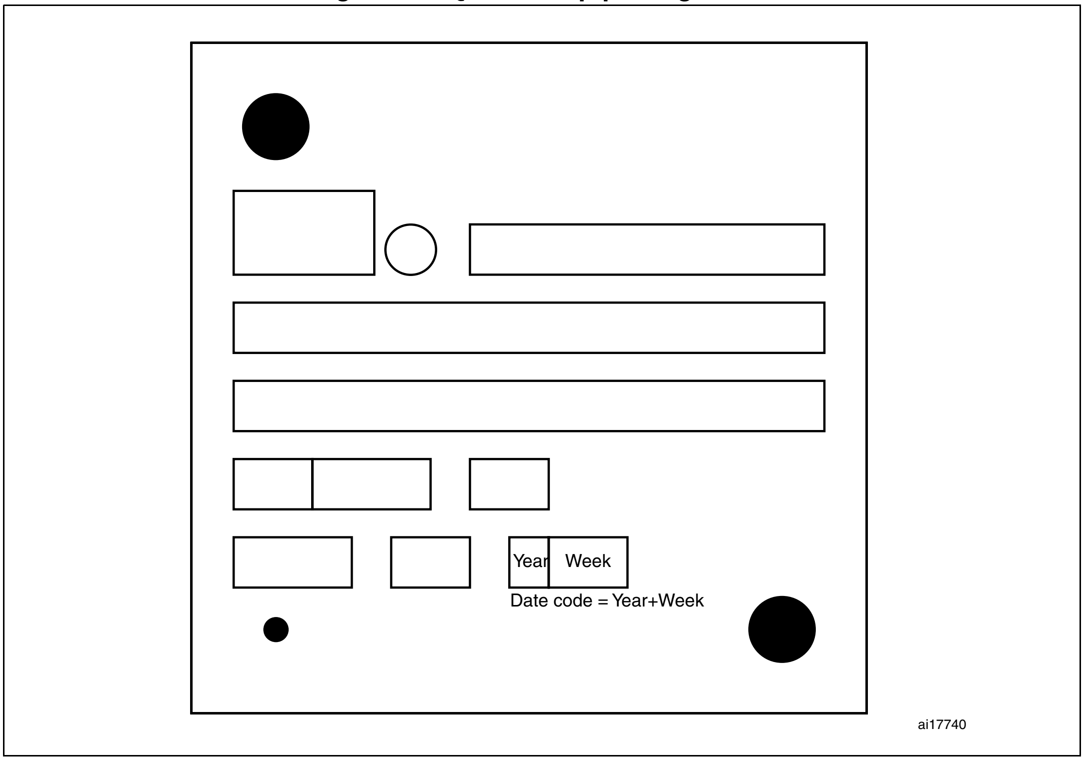


Figure 6. LQFP144 top package view

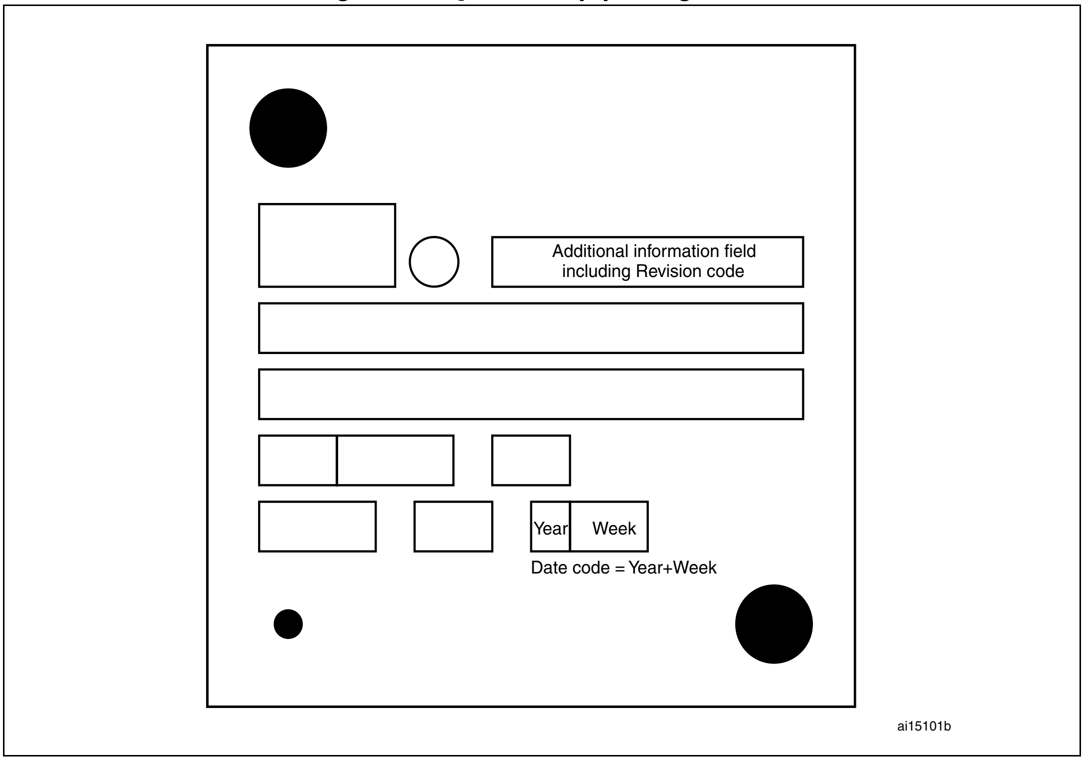
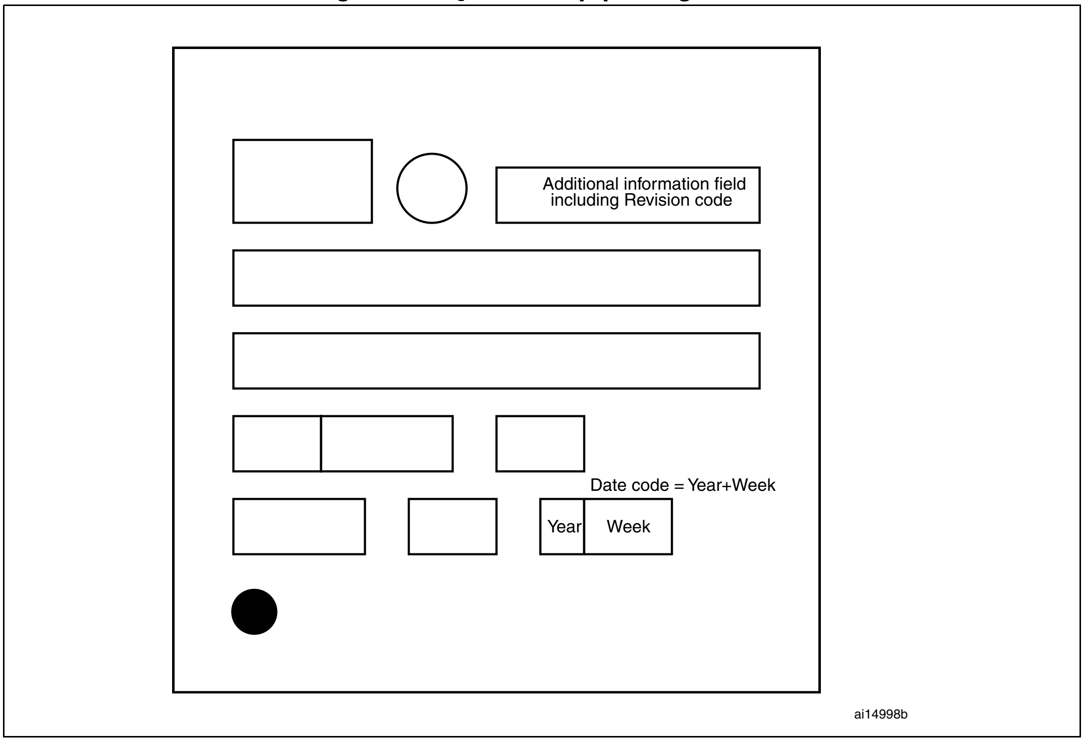


Figure 7. LQFP100 top package view



Revision history

Table 6. Document revision history

Date	Revision	Changes
11-Feb-2013	1	Initial release.
25-Feb-2013	2	Document converted to new template. Added *Section 2.8.4: Corruption of data read from the FMC*
26-Apr-2013	3	Added Silicon revision Y. Removed the reference to 'Cortex-M4F' in the whole document. Updated *Section 2.8.1: Dummy read cycles inserted when reading synchronous memories*. Added *Section 2.1.3: Wakeup sequence from Standby mode when using more than one wakeup source*, *Section 2.7.5: Successive write operations to the same register might not be fully taken into account* and *Section 2.8.3: FSMC NOR Flash/PSRAM controller asynchronous access on bank 2 to 4 when bank 1 is in synchronous mode (CBURSTRW bit is set)*. Removed limitation 2.10.3 SDIO clock divider BYPASS mode may not work properly. Updated *Section 2.9.5: No underrun detection with wrong data transmission*.
19-Sep-2013	4	Added STM32F429xx and STM32F439xx devices. Removed FSMC limitations. Added *Section 2.3.5: Both SDA and SCL maximum rise time (t_r) violated when VDD_I2C bus higher than ((VDD+0.3) / 0.7) V*. Updated *Section 2.8.5: Interruption of CPU read burst access to an end of SDRAM row*. Added *Section 2.8.1: Dummy read cycles inserted when reading synchronous memories*, *Section 2.8.2: FMC synchronous mode and NWAIT signal disabled*, *Section 2.8.3: Read access to a non-initialized FMC_SDRAM bank*, *Section 2.8.4: Corruption of data read from the FMC*, *Section 2.8.5: Interruption of CPU read burst access to an end of SDRAM row*, *Section 2.8.6: FMC NOR/PSRAM controller: asynchronous read access on bank 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set)* and *Section 2.8.7: FMC dynamic and static bank switching*. Added *Figure 1: TFBGA216 top package view*, *Figure 2: WLCSP143 top package view*, and *Figure 3: LQFP208 top package view*.

Table 6. Document revision history (continued)

Date	Revision	Changes
23-Sep-2013	5	Updated workaround in *Section 2.8.6: FMC NOR/PSRAM controller: asynchronous read access on bank 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set)*.
09-Jan-2014	6	Added silicon revision 1. Added STM32F429xE, STM32F427Ax, STM32F437Ax, STM32F429Ax, and STM32F439Ax part numbers. Removed mention of limitation fix in *Section 2.1.8: Over-drive and Under-drive modes unavailability*, *Section 2.8.4: Corruption of data read from the FMC* and *Section 2.8.6: FMC NOR/PSRAM controller: asynchronous read access on bank 2 to 4 returns wrong data when bank 1 is in synchronous mode (BURSTEN bit is set)*. Updated *Section 2.8.7: FMC dynamic and static bank switching* to indicate the limitation will be fixed in next silicon revision.

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