STM32F40x and STM32F41x Errata sheet

STM32F405/407xx and STM32F415/417xx revision A device limitations

Silicon identification

This errata sheet applies to the revision A of STMicroelectronics STM42F405xx/STM42F407xx and STM42F415xx/STM42F417xx microcontroller families. In this document they will be referred to as STM32F40x and STM32F41x, respectively, unless otherwise specified.

The STM32F40x and STM32F41x families feature an ARM™ 32-bit Cortex®M4F 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)
STM42F405xx, STM42F407xx	"A"
STM42F415xx, STM42F417xx	

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the STM32F40x and STM32F41x reference manual for details on how to find the revision code).
2. Refer to *Appendix A: Revision and date codes 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
STM42F405xx	STM42F405RG, STM42F405VG, STM42F405ZG
STM42F407xx	STM42F407IG, STM42F407VG, STM42F407ZG, STM42F407ZE, STM42F407IE, STM42F407VE
STM42F415xx	STM42F415RG, STM42F415VG, STM42F415ZG
STM42F417xx	STM42F417VG, STM42F417IG, STM42F417ZG, STM42F417VE, STM42F417ZE, STM42F417IE

Contents

List of tables

List of figures

1 ARM™ 32-bit Cortex®M4F limitations

An errata notice of the STM32F40x and STM32F41x core is available from the following web address: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0439c/index.html.

All the described limitations are minor and related to the revision r0p1-v1 of the CortexM4F core. *Table 3* summarizes these limitations and their implications on the behavior of STM32F40x and STM32F41x devices.

Table 3. CortexM4F core limitations and impact on microcontroller behavior

ARM ID	ARM category	ARM summary of errata	Impact on STM32F40x and STM32F41x
752419	Cat 2	Interrupted loads to SP can cause erroneous behavior	Minor

1.1 CortexM4F 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 caused 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

2 STM32F40x and STM32F41x 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 A
Section 2.1: System limitations	*Section 2.1.1: ART Accelerator prefetch queue instruction is not supported*	N
	Section 2.1.2: MCU device ID is incorrect	A
	Section 2.1.3: Debugging Stop mode and system tick timer	A
	Section 2.1.4: Debugging Stop mode with WFE entry	A
Section 2.2: I2C peripheral limitations	*Section 2.2.1: SMBus standard not fully supported*	A
	Section 2.2.2: Start cannot be generated after a misplaced Stop	A
	Section 2.2.3: Mismatch on the "Setup time for a repeated Start condition" timing parameter	A
	Section 2.2.4: Data valid time (tVD;DAT) violated without the OVR flag being set	A
Section 2.3: I2S peripheral limitations	*Section 2.3.1: In I2S slave mode, WS level must to be set by the external master when enabling the I2S*	A
Section 2.4: USART peripheral limitations	*Section 2.4.1: Idle frame is not detected if receiver clock speed is deviated*	N
	Section 2.4.2: In full duplex mode, the Parity Error (PE) flag can be cleared by writing the data register	A
	Section 2.4.3: Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection	N
	Section 2.4.4: Break frame is transmitted regardless of nCTS input line status	N
	Section 2.4.5: nRTS signal abnormally driven low after a protocol violation	A
Section 2.5: OTG_FS limitations	*Section 2.5.1: Data in RxFIFO are overwritten when all channels are disabled simultaneously*	A
	Section 2.5.2: OTG host blocks the receive channel when receiving IN packets and no TxFIFO is configured	A
	Section 2.5.3: Host channel-halted interrupt not generated when the channel is disabled	A
	Section 2.5.4: Error in software-read OTG_FS_DCFG register values	A

Table 4. Summary of silicon limitations (continued)

Links to silicon limitations		Rev A
Section 2.6: Ethernet limitations	*Section 2.6.1: Incorrect layer 3 (L3) checksum is inserted in transmitted IPv6 packets without TCP, UDP or ICMP payloads*	A
	Section 2.6.2: The Ethernet MAC processes invalid extension headers in the received IPv6 frames	N
	Section 2.6.3: MAC stuck in the Idle state on receiving the TxFIFO flush command exactly 1 clock cycle after a transmission completes	A
	Section 2.6.4: Transmit frame data corruption	A
Section 2.7: FSMC limitations	*Section 2.7.1: Dummy read cycles inserted when reading synchronous memories*	N
Section 2.8: SDIO limitation	*Section 2.8.1: SDIO HW flow control*	N

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.

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.

This limitation will be fixed in next silicon revision.

2.1.2 MCU device ID is incorrect

Description

The STM32F40x and STM32F41x have the same MCU device ID as the STM32F20x and STM32F21x devices. The device ID can be read from address 0xE004 2000.

Workaround

To differentiate the STM32F4xxx from the STM32F2xxx series, read the MCU device ID and the Core Device.

- For STM32F2xxx
  MCU device ID = STM32F2xxx device ID
  Core Device = CortexM3
- For STM32F4xxx
  MCU device ID = STM32F4xxx device ID
  Core Device = CortexM4

This limitation will be fixed in next silicon revision.

2.1.3 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 wakeup the system from Stop mode.

Workaround

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

2.1.4 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 }
```

2.2 I2C peripheral limitations

2.2.1 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

2.2.2 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.

2.2.3 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.

2.2.4 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.

2.3 I2S peripheral limitations

2.3.1 In I2S slave mode, WS level must to 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 USART peripheral limitations

2.4.1 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.

2.4.2 In full duplex mode, the Parity Error (PE) flag can be cleared by writing 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 in 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.

2.4.3 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.

2.4.4 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 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.

2.4.5 nRTS signal abnormally driven low after a protocol violation

Description

When RTS hardware flow control is enabled, the nRTS signal goes high when a data is received. If this data was not read and a 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 some data has been lost.

Workaround

The lost data should be resent to the USART.

2.5 OTG_FS limitations

2.5.1 Data in RxFIFO are 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.

2.5.2 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 are 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.

2.5.3 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.

2.5.4 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 Ethernet limitations

2.6.1 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).

2.6.2 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.

2.6.3 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.

2.6.4 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 are being written to the TxFIFO is almost equal to or a little less than the rate at which the data are being read.

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

Workaround

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

2.7 FSMC limitations

2.7.1 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 SDIO limitation

2.8.1 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.

Appendix A Revision code on device marking

Figure 1, *Figure 2*, *Figure 3*, *Figure 4* and *Figure 5* show the marking compositions for the 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. UFBGA176 top package view

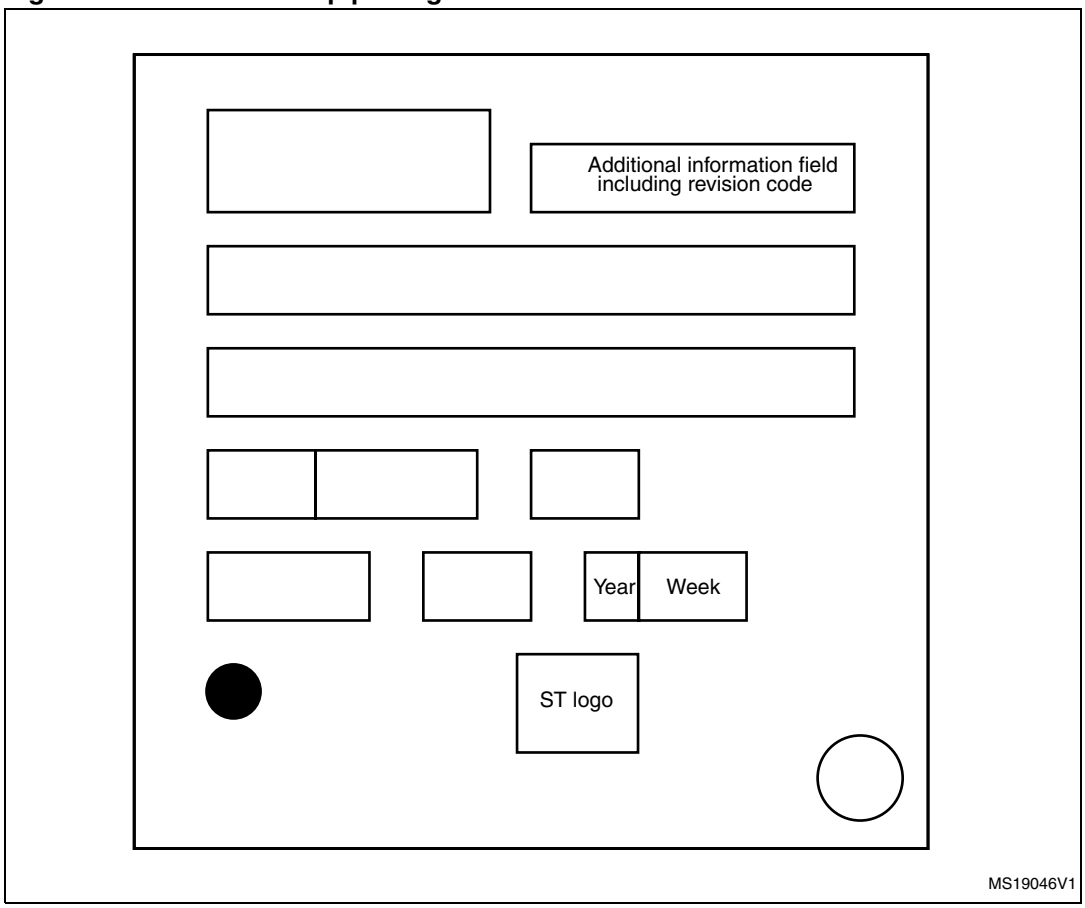


Figure 2. LQFP176 top package view

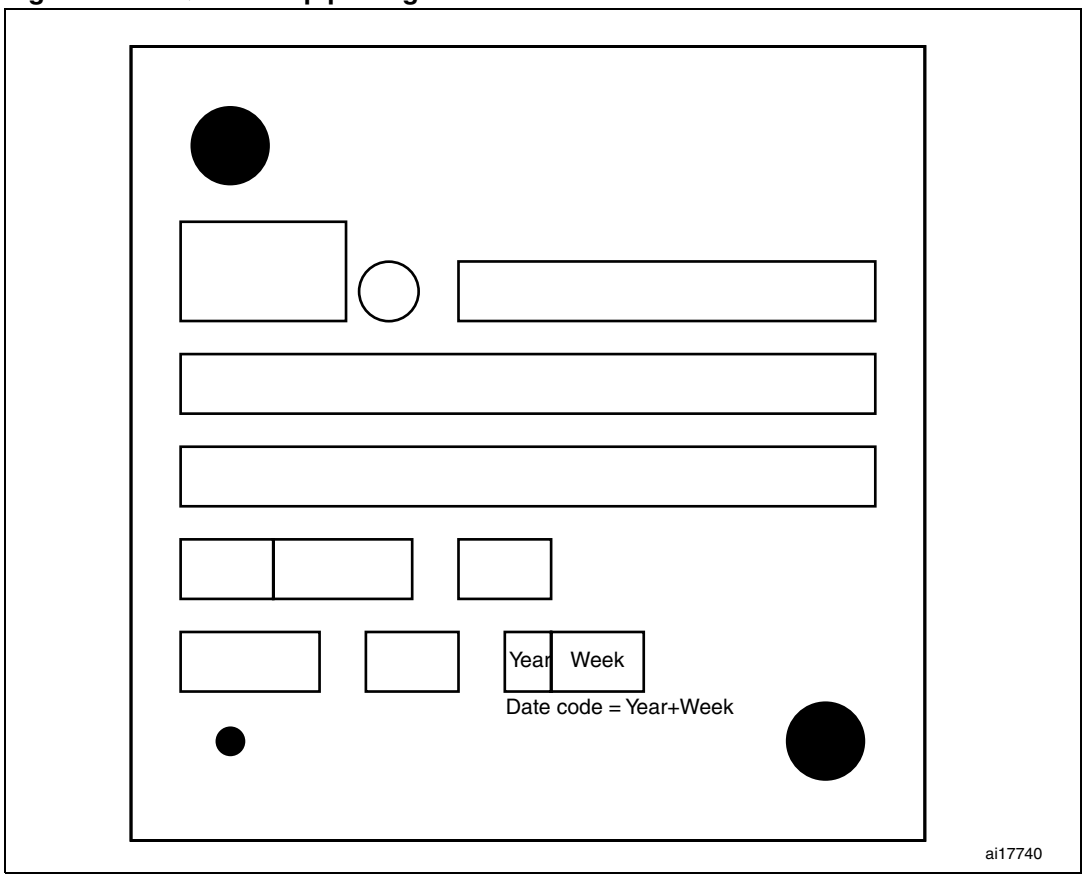


Figure 3. LQFP144 top package view

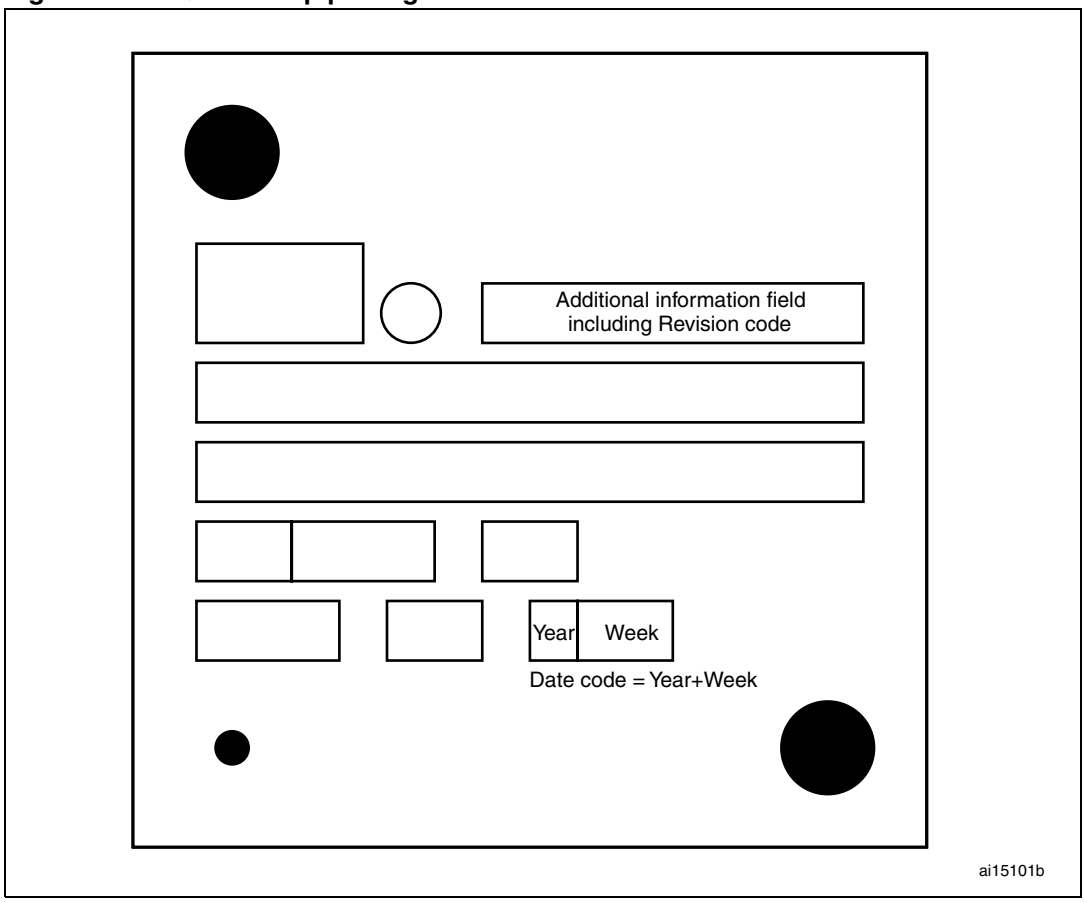


Figure 4. LQFP100 top package view

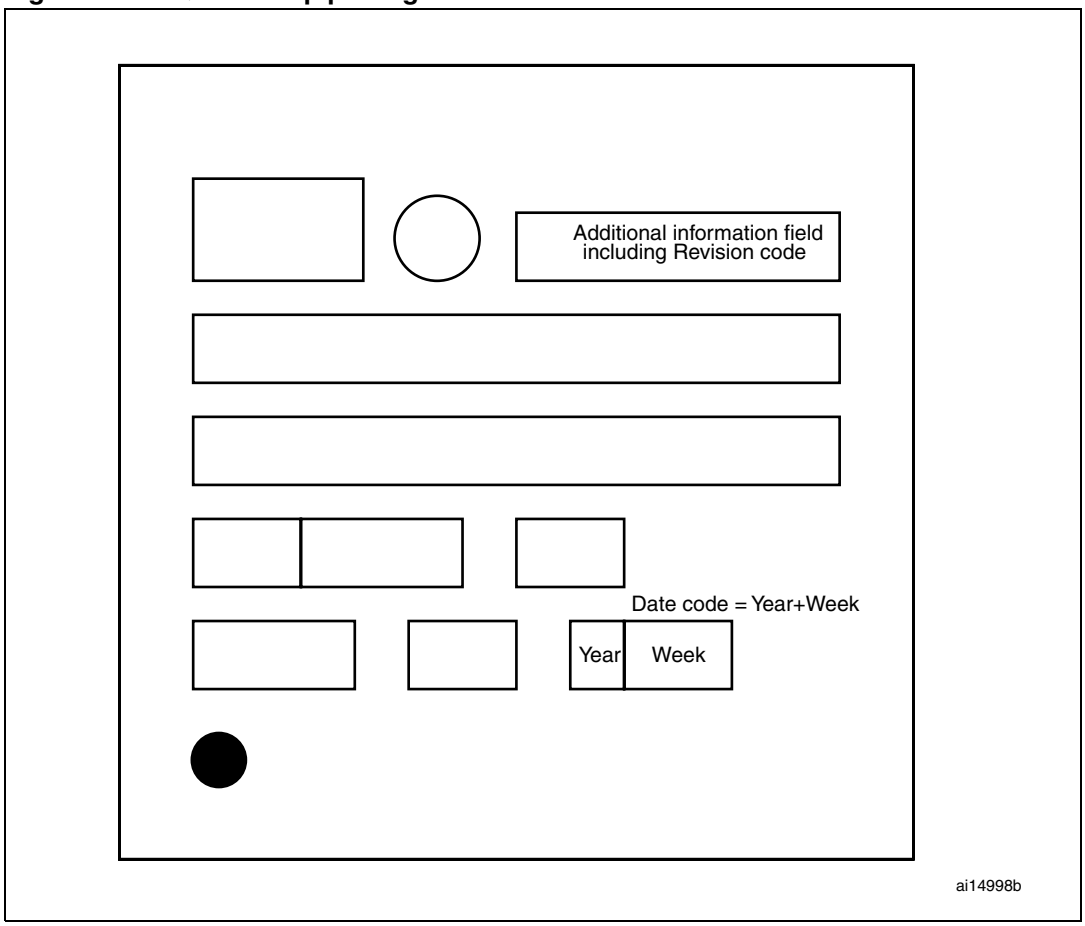
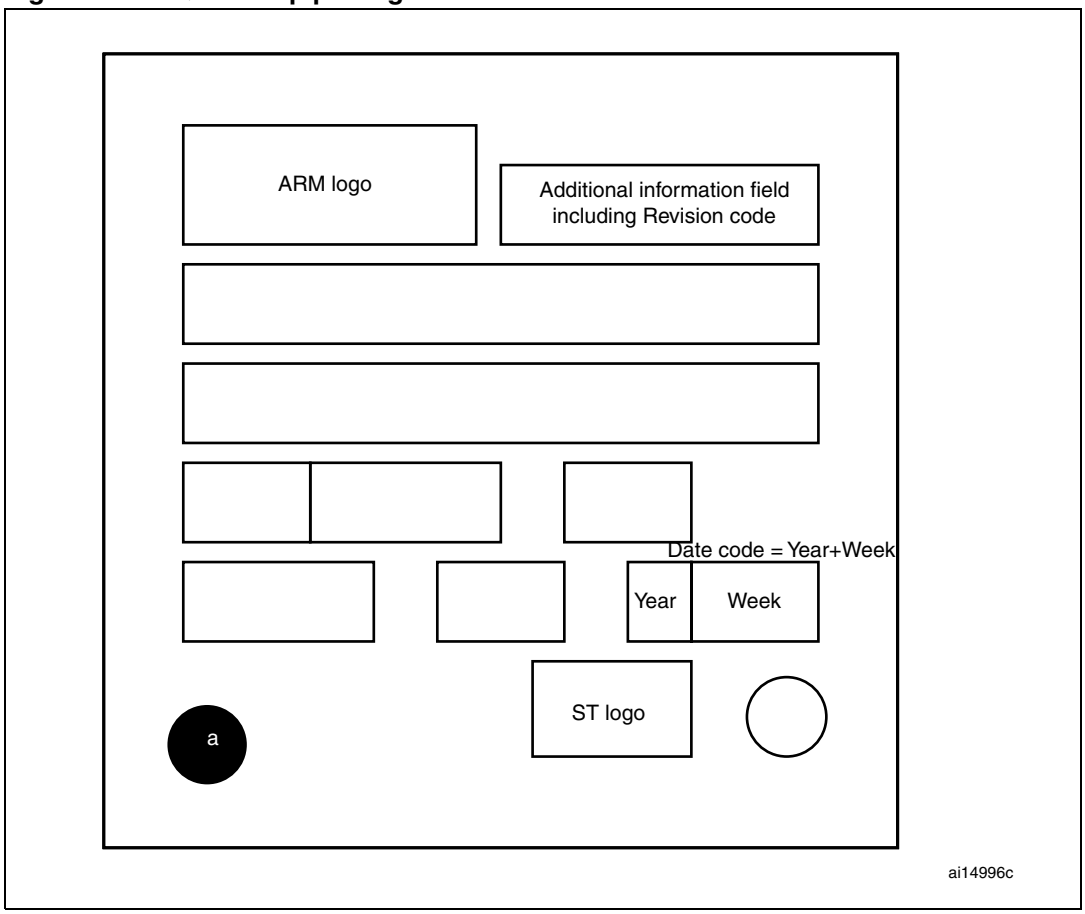


Figure 5. LQFP64 top package view



Revision history

Table 5. Document revision history

Date	Revision	Changes
19-Sep-2011	1	Initial release.

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.

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