STM32F101xF/G STM32F103xF/G
Errata sheet

STM32F101xF/G and STM32F103xF/G XL-density device limitations

Silicon identification

This errata sheet applies to the revision A or 1 of the STMicroelectronics STM32F101xF/G access line and STM32F103xF/G performance line XL-density products. These families feature an 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(2) marked on device
STM32F103xF, STM32F103xG	“A” or “1”
STM32F101xF, STM32F101xG	“A” or “1”

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the STM32F10xxx 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 on the different packages.

Table 2. Device summary

Reference	Part number
STM32F101xFG	STM32F101RF STM32F101VF STM32F101ZF
	STM32F101RG STM32F101VG STM32F101ZG
STM32F103xFG	STM32F103RF STM32F103VF STM32F103ZF
	STM32F103RG STM32F103VG STM32F103ZG

Contents

List of tables

List of figures

1 ARM® 32-bit Cortex®-M3 limitations

An errata notice of the STM32F10xxx core is available from the following web address: *http://infocenter.arm.com.*

All the described limitations are minor and related to the revision r1p1-01rel0 of the Cortex-M3 core. *Table 3* summarizes these limitations and their implications on the behavior of XL-density STM32F10xxx devices.

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

ARM ID	ARM category	ARM summary of errata	Impact on XL-density STM32F10xxx devices
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
531064	impl	SWJ-DP missing POR reset sync	No
511864	Cat 3	Cortex-M3 may fetch instructions using incorrect privilege on return from an exception	No
532314	Cat 3	DWT CPI counter increments during sleep	No
538714	Cat 3	Cortex-M3 TPIU clock domain crossing	No
548721	Cat 3	Internal write buffer could be active whilst asleep	No
463763	Cat 3	BKPT in debug monitor mode can cause DFSR mismatch	Minor
463764	Cat 3	Core may freeze for SLEEPONEXIT single instruction ISR	Minor
463769	Cat 3	Unaligned MPU fault during a write may cause the wrong data to be written to a successful first access	Minor

1.1 Cortex-M3 limitations description for STM32F10xxx XL-density devices

Only the limitations described below have an impact, even though minor, on the implementation of STM32F10xxx XL-density 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 STM32F10xxx XL-density devices (Cortex-M3 r1p1-01rel0).

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 STM32F10xxx 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 STM32F10xxx 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 BKPT in debug monitor mode can cause DFSR mismatch

Description

A BKPT may be executed in debug monitor mode. This causes the debug monitor handler to be run. However, the bit 1 in the Debug fault status register (DFSR) at address 0xE000ED30 is not set to indicate that it was originated by a BKPT instruction. This only occurs if an interrupt other than the debug monitor is already being processed just before the BKPT is executed.

Workaround

If the DFSR register does not have any bit set when the debug monitor is entered, this means that we must be in this “corner case” and so, that a BKPT instruction was executed in debug monitor mode.

1.1.4 Cortex-M3 may freeze for SLEEPONEXIT single instruction ISR

Description

If the Cortex-M3 SLEEPONEXIT functionality is used and the concerned interrupt service routine (ISR) contains only a single instruction, the core becomes frozen. This freezing may occur if only one interrupt is active and it is preempted by an interrupt whose handler only contains a single instruction.

However, any new interrupt that causes a preemption would cause the core to become unfrozen and behave correctly again.

Workaround

This scenario does not happen in real application systems since all enabled ISRs should at least contain one instruction. Therefore, if an empty ISR is used, then insert an NOP or any other instruction before the exit instruction (BX or BLX).

1.1.5 Cortex-M3 unaligned MPU fault during a write may cause the wrong data to be written to a successful first access

Description

When an unaligned store is executed by Cortex-M3 the transaction is split up into either two or three aligned transactions forming constituent parts of the larger transaction. The MPU checks that these transactions are permitted and blocks them if necessary. If an unaligned transaction occurs where it overlaps two MPU regions then each region relating to the part of the transaction that hits that region will be checked.

If an unaligned store occurs that crosses an MPU region boundary and has an MPU permission fault for the second region check but not for the first region then it is possible for the second component’s data to be written for the first successful transaction in place of the first transaction’s data. This can occur for writes to either the D-Code or system bus.

However in this case, a MemManage fault occurs immediately, pointing to the instruction that caused the fault.

Workaround

Ensure that accesses do not cross the MPU region border or program the MPU correctly, in order to cover the respective data region as needed.

2 STM32F101xF/G and STM32F103xF/G silicon limitations

Table 4 gives quick references to all documented limitations.

Table 4. Summary of silicon limitations

Links to silicon limitations	
Section 2.1: Voltage glitch on ADC input 0	
Section 2.2: Flash memory read after WFI/WFE instruction	
Section 2.3: DBGMCU_CR register cannot be read by user software	
Section 2.4: Debugging Stop mode and system tick timer	
Section 2.5: Debugging Stop mode with WFE entry	
Section 2.6: Wakeup sequence from Standby mode when using more than one wakeup source	
Section 2.7: LSE start-up in harsh environments	
Section 2.8: Alternate function	*Section 2.8.1: SPI1 in slave mode and USART2 in synchronous mode*
	Section 2.8.2: SPI1 in master mode and USART2 in synchronous mode
	Section 2.8.3: SPI2 in slave mode and USART3 in synchronous mode
	Section 2.8.4: SPI2 in master mode and USART3 in synchronous mode
	Section 2.8.5: SDIO with TIM8
	Section 2.8.6: SDIO and TIM3_REMAP
	Section 2.8.7: SDIO with USART3 remapped and UART4
	Section 2.8.8: I2S2 in master/slave mode and USART3 in synchronous mode
	Section 2.8.9: USARTx_TX pin usage
Section 2.10: Flash memory BSY bit delay versus STRT bit setting	
Section 2.11: I2C peripheral	*Section 2.11.1: Some software events must be managed before the current byte is being transferred*
	Section 2.11.2: Wrong data read into data register
	Section 2.11.3: SMBus standard not fully supported
	Section 2.11.4: Start cannot be generated after a misplaced Stop
	Section 2.11.5: Mismatch on the “Setup time for a repeated Start condition” timing parameter
	Section 2.11.6: Data valid time (tVD;DAT) violated without the OVR flag being set
	Section 2.11.7: I2C analog filter may provide wrong value, locking BUSY flag and preventing master mode entry

Table 4. Summary of silicon limitations (continued)

Links to silicon limitations	
Section 2.12: SPI peripheral	*Section 2.12.1: CRC sensitive to communication clock when SPI is in slave mode even with NSS high*
	Section 2.12.2: SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI is under DMA transaction close to the end of transfer or end of transfer -1
Section 2.13: I2S peripheral	*Section 2.13.1: Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronization mode*
	Section 2.13.2: In I2S slave mode, WS level must to be set by the external master when enabling the I2S
	Section 2.13.3: I2S slave mode desynchronisation with the master during communication
Section 2.14: USART peripheral	*Section 2.14.1: Parity Error flag (PE) is set again after having been cleared by software*
	Section 2.14.2: Idle frame is not detected if receiver clock speed is deviated
	Section 2.14.3: In full duplex mode, the Parity Error (PE) flag can be cleared by writing the data register
	Section 2.14.4: Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection
	Section 2.14.5: Break frame is transmitted regardless of nCTS input line status
	Section 2.14.6: nRTS signal abnormally driven low after a protocol violation
Section 2.15: SDIO peripheral	*Section 2.15.1: SDIO hardware flow control*
	Section 2.15.2: Wrong CCRCFAIL status after a response without CRC is received
	Section 2.15.3: Data corruption in SDIO clock dephasing (NEGEDGE) mode
	Section 2.15.4: CE-ATA multiple write command and card busy signal management
	Section 2.15.5: No underrun detection with wrong data transmission
Section 2.16: Timers	*Section 2.16.1: Missing capture flag*
	Section 2.16.2: Overcapture detected too early
	Section 2.16.3: General-purpose timer: regulation for 100% PWM
Section 2.17: LSI clock stabilization time	
Section 2.18: FSMC limitations	*Section 2.18.1: Dummy read cycles inserted when reading synchronous memories*
	Section 2.18.2: 1 dummy clock cycle inserted when writing to synchronous memories when CLKDIV=1

2.1 Voltage glitch on ADC input 0

Description

A low-amplitude voltage glitch may be generated (on ADC input 0) on the PA0 pin, when the ADC is converting with injection trigger. It is generated by internal coupling and synchronized to the beginning and the end of the injection sequence, whatever the channel(s) to be converted.

The glitch amplitude is less than 150 mV with a typical duration of 10 ns (measured with the I/O configured as high-impedance input and left unconnected). If PA0 is used as a digital output, this has no influence on the signal. If PA0 is used has a digital input, it will not be detected as a spurious transition, providing that PA0 is driven with an impedance lower than 5 kΩ. This glitch does not have any influence on the remaining port A pin or on the ADC conversion injection results, in single ADC configuration.

When using the ADC in dual mode with injection trigger, and in order to avoid any side effect, it is advised to distribute the analog channels so that Channel 0 is configured as an injected channel.

Workaround

None.

2.2 Flash memory read after WFI/WFE instruction

Conditions

- Flash prefetch on
- Flash memory timing set to 2 wait states
- FLITF clock stopped in Sleep mode

Description

If a WFI/WFE instruction is executed during a Flash memory access and the Sleep duration is very short (less than 2 clock cycles), the instruction fetch from the Flash memory may be corrupted on the next wakeup event.

Workaround

When using the Flash memory with two wait states and prefetch on, the FLITF clock must *not* be stopped during the Sleep mode – the FLITFEN bit in the RCC_AHBENR register must be set (keep the reset value).

2.3 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.

2.4 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.

2.5 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 }

2.6 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 flag needs to be cleared prior to the Standby mode entry, otherwise the MCU wakes up immediately.

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

Workaround

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

1. Disable all used wakeup sources.
2. Clear all related wakeup flags.
3. Re-enable all used wakeup sources.
4. Enter Standby mode.

Be aware that, when applying this workaround, if one of the wakeup sources is still kept high, the MCU will enter the Standby mode, but then it will wake up immediately generating the power reset.

2.7 LSE start-up in harsh environments

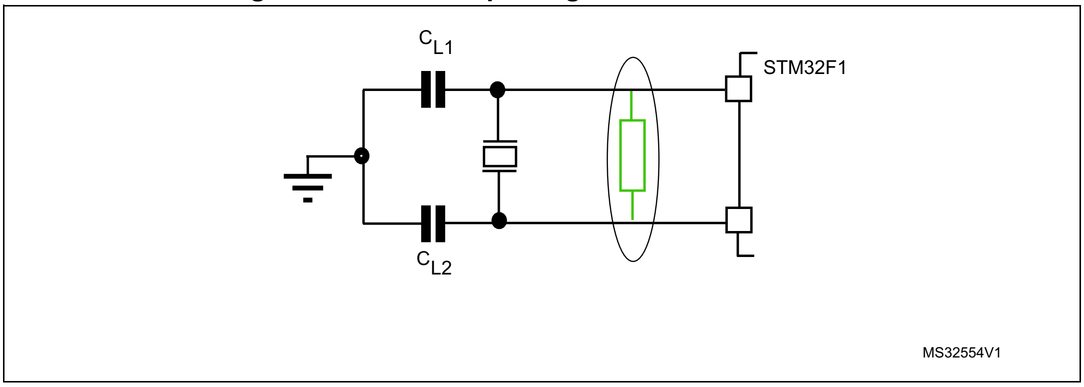
Description

The LSE (Low Speed External) oscillator system has been designed to minimize the overall power consumption of the STM32F1 microcontroller. It is extremely important to take specific care in the design of the PCB to ensure this low power oscillator starts in harsh conditions. In some PCB designs without coating, an induced low leakage may prevent the LSE to start-up, regardless of the 32.768 KHz crystal used. This phenomenon is amplified in humid environments that create frost on the OSC32_IN/OSC32_OUT tracks. This unwanted behavior may happen only at the first back-up domain power-on of the device.

Workaround

It is recommended to mount an additional parallel feedback resistor (from 16 MΩ to 22 MΩ) on board to help the oscillation start-up in all cases (see *Figure 1*). For more details on compatible crystals and hardware techniques on PCB, refer to AN2867 application note.

Figure 1. LSE start-up using an additional resistor



2.8 Alternate function

In some specific cases, a potential weakness may exist between alternate function outputs mapped onto the same pin. On those IOs the EVENTOUT Cortex output feature cannot be used at the same time as another alternate function.

2.8.1 SPI1 in slave mode and USART2 in synchronous mode

Conditions

- SPI1 and USART2 are clocked
- I/O port pin PA4 is configured as an alternate function output.

Description

USART2 cannot be used in synchronous mode (USART2_CK signal), if SPI1 is used in slave mode.

Workaround

None.

2.8.2 SPI1 in master mode and USART2 in synchronous mode

Conditions

- SPI1 and USART2 are clocked
- I/O port pin PA4 is configured as an alternate function output.

Description

USART2 cannot be used in synchronous mode (USART2_CK signal) if SPI1 is used in master mode and SP1_NSS is configured in software mode. In this case USART2_CK is not output on the pin.

Workaround

In order to output USART2_CK, the SSOE bit in the SPI1_CR2 register must be set to configure the pin in output mode.

2.8.3 SPI2 in slave mode and USART3 in synchronous mode

Conditions

- SPI2 and USART3 are clocked
- I/O port pin PB12 is configured as an alternate function output.

Description

USART3 cannot be used in synchronous mode (USART3_CK signal) if SPI2 is used in slave mode.

Workaround

None.

2.8.4 SPI2 in master mode and USART3 in synchronous mode

Conditions

- SPI2 and USART3 are clocked
- I/O port pin PB12 is configured as an alternate function output.

Description

USART3 cannot be used in synchronous mode (USART3_CK signal) if SPI2 is used in master mode and SP2_NSS is configured in software mode. In this case USART3_CK is not output on the pin.

Workaround

In order to output USART3_CK, the SSOE bit in the SPI2_CR2 register must be set to configure the pin in output mode,

2.8.5 SDIO with TIM8

Description

Conflicts occur when:

- the SDIO is configured in 1- or 4-bit mode and TIM8_CH4 is configured as an output

The signals that conflict are the following:

- TIM8_CH4 and SDIO_D1

Workaround

Do not use TIM8_CH4 as an output when the SDIO is being used.

2.8.6 SDIO and TIM3_REMAP

Description

When SDIO is configured in 1- or 4-bit mode, and TIM3 channels are remapped to PC6 to PC9, and configured as outputs, a conflict occurs between:

- TIM3_CH4 and SDIO_D1

Workaround

Do not use TIM3_CH4 as an output when the SDIO is being used.

2.8.7 SDIO with USART3 remapped and UART4

Description

When SDIO is configured in 1-bit mode, there are conflicts with the USART3_TX pin remapped and with the UART4_TX pin. Conflicts are between the following signals:

- USART3_TX and SDIO_D2
- UART4_TX and SDIO_D2

Workaround

Use USART3_TX either in the default configuration (on the PB10 I/O) or remap USART3_TX to PD8 when the SDIO is being used.

Do not use UART4_TX when the SDIO is being used.

2.8.8 I2S2 in master/slave mode and USART3 in synchronous mode

Conditions

- USART3 in synchronous mode is clocked
- I2S2 is not clocked
- I/O port pin PB12 is configured as an alternate function output

Description

If I2S2 was used prior to operating USART3 in synchronous mode, a conflict occurs between the I2S2_WS and USART3_CK signal even though the I2S2 clock was disabled.

Workaround

To use USART3 in synchronous mode, first disable the I2S2 clock, then perform a software reset of SPI2(I2S2).

2.8.9 USARTx_TX pin usage

Description

In USART receive-mode-only communication (TE = 0 in the USARTx_CR1 register), even when the USARTx_TX pin is not being used, the corresponding I/O port pin cannot be used to output another alternate function (in this mode the USARTx_TX output is set to 1 and thus no other alternate function output can be used).

This limitation applies to all USARTx_TX pins that share another alternate function output.

Workaround

Do not use the corresponding I/O port of the USARTx_TX pin in alternate function output mode. Only the input mode can be used (TE bit in the USARTx_CR1 has to be cleared).

2.9 SPI3 in I²S slave mode: timing sensitivity between I2S3_WS and I2S3_CK

Description

When SPI3 is configured in I²S slave audio mode in I2S Philips or PCM modes, If the I2S3_WS signal arrives too early with respect to the active edge of I2S3_CK, a wrong communication starting too soon may result: then, depending on the clock polarity and the Audio mode selected, it is either shifted by one bit from start to end or, the first left and right data items are lost and the others, shifted.

Workaround

None. Use I2S3 in slave mode in the MSB/LSB justified mode only.

2.10 Flash memory BSY bit delay versus STRT bit setting

Description

When the STRT bit in the Flash memory control register is set (to launch an erase operation), the BSY bit in the Flash memory status register goes high one cycle later.

Therefore, if the FLASH_SR register is read immediately after the FLASH_CR register is written (STRT bit set), the BSY bit is read as 0.

Workaround

Read the BSY bit at least one cycle after setting the STRT bit.

2.11 I²C peripheral

2.11.1 Some software events must be managed before the current byte is being transferred

Description

When the EV7, EV7_1, EV6_1, EV6_3, EV2, EV8, and EV3 events are not managed before

the current byte is being transferred, problems may be encountered such as receiving an extra byte, reading the same data twice or missing data.

Workarounds

When it is not possible to manage the EV7, EV7_1, EV6_1, EV6_3, EV2, EV8, and EV3 events before the current byte transfer and before the acknowledge pulse when changing the ACK control bit, it is recommended to:

- **Workaround 1**
  Use the I2C with DMA in general, except when the Master is receiving a single byte.
- Workaround 2
  Use I2C interrupts and boost their priorities to the highest one in the application to make them uninterruptible
- **Workaround 3** (only for EV6_1 and EV6_3 events used in method 2)
  EV6_1 event (used in master receiver 2 bytes):
  Stretch SCL line between ADDR bit is cleared and ACK is cleared:
  a) ADDR=1
  b) Configure SCL I/O as GPIO open-drain output low
  c) Clear ADDR by reading SR1 register followed by reading SR3
  d) Program ACK=0
  e) Configure SCL I/O as Alternate Function open drain
  EV6_3 event (used in master receiver 1 byte):
  Stretch SCL line between ADDR bit is cleared and STOP bit programming:
  a) ADDR=1
  b) Program ACK=0
  c) Configure SCL I/O as GPIO open-drain output low
  d) Clear ADDR by reading SR1 register followed by reading SR3
  e) Program STOP=1
  f) Configure SCL I/O as Alternate Function open drain

2.11.2 Wrong data read into data register

In Master Receiver mode, when closing the communication using method 2, the content of the last read data can be corrupted. The following two sequences are concerned by the limitation:

- **Sequence 1:** Transfer sequence for master receiver when N = 2:
  a) BTF = 1(Data N-1 in DR and Data N in shift register)
  b) Program STOP = 1,
  c) Read DR twice (Read Data N-1 and Data N) just after programming the STOP.
- **Sequence 2:** Transfer sequence for master receiver when N > 2:
  a) BTF = 1 (Data N-2 in DR and Data N-1 in shift register)
  b) Program ACK = 0,
  c) Read DataN-2 in DR.
  d) Program STOP = 1,
  e) Read DataN-1.

If the user software is not able to read the data N-1 before the STOP condition is generated on the bus, the content of the shift register (data N) will be corrupted (data N is shifted 1-bit to the left).

Workarounds

- Workaround 1

  Stretch the SCL line by configuring SCL I/O as a general purpose I/O, open-drain output low level, before the SET STOP in sequence 1 and before the READ Data N-2 in séquence 2. Then configure back the SCL I/O as alternate function open-drain after the READ Data N-1. The sequences become:

  Sequence 1:

  a) BTF = 1(Data N-1 in DR and Data N in shift register)
  b) Configure SCL I/O as GPIO open-drain output low
  c) Program STOP = 1
  d) Read Data N-1
  e) Configure SCL I/O as Alternate Function open drain
  f) Read Data N

Sequence 2:

a) BTF = 1 (Data N-2 in DR and Data N-1 in shift register)

b) Program ACK = 0

c) Configure SCL I/O as GPIO open-drain output low

d) Read Data N-2 in DR.

e) Program STOP = 1,

f) Read Data N-1.

g) Configure SCL I/O as Alternate Function open drain.

- Workaround 2

  Mask all active interrupts between the SET STOP and the READ data N-1 for sequence 1; and between the READ data N-2, the SET STOP and the READ data N-1 for Sequence 2.
- Workaround 3

  Manage I2C RxNE events with DMA or interrupts with the highest priority level, so that the condition BTF = 1 never occurs.

2.11.3 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

2.11.4 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 I^2C 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 I^2C peripheral.

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

2.11.5 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.

2.11.6 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.

2.11.7 I2C analog filter may provide wrong value, locking BUSY flag and preventing master mode entry

Description

The I2C analog filters embedded in the I2C I/Os may be tied to low level, whereas SCL and SDA lines are kept at high level. This can occur after an MCU power-on reset, or during ESD stress. Consequently, the I2C BUSY flag is set, and the I2C cannot enter master mode (START condition cannot be sent). The I2C BUSY flag cannot be cleared by the SWRST control bit, nor by a peripheral or a system reset. BUSY bit is cleared under reset, but it is set high again as soon as the reset is released, because the analog filter output is still at low level. This issue occurs randomly.

Note: Under the same conditions, the I2C analog filters may also provide a high level, whereas SCL and SDA lines are kept to low level. This should not create issues as the filters output will be correct after next SCL and SDA transition.

Workaround

The SCL and SDA analog filter output is updated after a transition occurs on the SCL and SDA line respectively. The SCL and SDA transition can be forced by software configuring the I2C I/Os in output mode. Then, once the analog filters are unlocked and output the SCL and SDA lines level, the BUSY flag can be reset with a software reset, and the I2C can enter master mode. Therefore, the following sequence must be applied:

1. Disable the I2C peripheral by clearing the PE bit in I2Cx_CR1 register.
2. Configure the SCL and SDA I/Os as General Purpose Output Open-Drain, High level (Write 1 to GPIOx_ODR).
3. Check SCL and SDA High level in GPIOx_IDR.
4. Configure the SDA I/O as General Purpose Output Open-Drain, Low level (Write 0 to GPIOx_ODR).
5. Check SDA Low level in GPIOx_IDR.
6. Configure the SCL I/O as General Purpose Output Open-Drain, Low level (Write 0 to GPIOx_ODR).
7. Check SCL Low level in GPIOx_IDR.
8. Configure the SCL I/O as General Purpose Output Open-Drain, High level (Write 1 to GPIOx_ODR).
9. Check SCL High level in GPIOx_IDR.
10. Configure the SDA I/O as General Purpose Output Open-Drain , High level (Write 1 to GPIOx_ODR).
11. Check SDA High level in GPIOx_IDR.
12. Configure the SCL and SDA I/Os as Alternate function Open-Drain.
13. Set SWRST bit in I2Cx_CR1 register.
14. Clear SWRST bit in I2Cx_CR1 register.
15. Enable the I2C peripheral by setting the PE bit in I2Cx_CR1 register.

2.12 SPI peripheral

2.12.1 CRC sensitive to communication clock when SPI is in slave mode even with NSS high

Description

When the SPI is configured in slave mode with the CRC feature enabled, the CRC is calculated even if the NSS pin deselects the SPI (high level applied on the NSS pin).

Workaround

The CRC has to be cleared on both Master and Slave sides between the slave deselection (high level on NSS) and the slave selection (low level on NSS), in order to resynchronize the Master and Slave for their respective CRC calculation.

To procedure to clear the CRC is the following:

1. disable the SPI (SPE = 0)
2. clear the CRCEN bit
3. set the CRCEN bit
4. enable the SPI (SPE = 1)

2.12.2 SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI is under DMA transaction close to the end of transfer or end of transfer -1

Description

In the following conditions, the CRC may be frozen before the CRCNEXT bit is written, resulting in a CRC error:

- SPI is slave or master.
- Full duplex or simplex mode is used.
- CRC feature is enabled.
- SPI is configured to manage data transfers by software (interrupt or polling).
- A peripheral, mapped on the same DMA channel as the SPI, is executing DMA transfers.

Workaround

If the application allows it, you can use the DMA for SPI transfers.

2.13 I2S peripheral

2.13.1 Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronization mode

Description

When I2S is master with PCM long synchronization is selected as16-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.

2.13.2 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.

2.13.3 I2S slave mode desynchronisation 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.

2.14 USART peripheral

2.14.1 Parity Error flag (PE) is set again after having been cleared by software

Description

The parity error flag (PE) is set at the end of the last data bit. It should be cleared by software by making a read access to the status register followed by reading the data in the data register.

Once the PE flag is set by hardware, if it is cleared by software before the middle of the stop bit, it will be set again. Consequently, the software may jump several times to the same interrupt routine for the same parity error.

Workaround

Before clearing the Parity Error flag, the software must wait for the RXNE flag to be set.

2.14.2 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.

2.14.3 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.

2.14.4 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.

2.14.5 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.

2.14.6 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.

2.15 SDIO peripheral

2.15.1 SDIO hardware flow control

Description

When enabling the hardware 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 hardware flow control. Overrun errors (Rx mode) and FIFO underrun (Tx mode) should be managed by the application software.

2.15.2 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.

2.15.3 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.

2.15.4 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.

2.15.5 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.

2.16 Timers

These limitations apply only to TIM1, TIM2, TIM3, TIM4, TIM5 and TIM8.

2.16.1 Missing capture flag

Description

In capture mode, when a capture occurs while the CCRx register is being read, the capture flag (CCxIF) may be cleared without the overcapture flag (CCxOF) being set. The new data are actually captured in the capture register.

Workaround

An external interrupt can be enabled on the capture I/O just before reading the capture register (in the capture interrupt), and disabled just after reading the captured data. A missed capture will be detected by the EXTI peripheral.

2.16.2 Overcapture detected too early

Description

In capture mode, the overcapture flag (CCxOF) can be set even though no data have been lost.

Conditions

If a capture occurs while the capture register is being read, an overcapture is detected even though the previously captured data are correctly read and the new data are correctly stored into the capture register.

The system is at the limit of an overcapture but no data are lost.

Workaround

None.

2.16.3 General-purpose timer: regulation for 100% PWM

Description

When the OCREF_CLR functionality is activated, the OCxREF signal becomes de-asserted (and consequently OCx is deasserted / OCxN is asserted) when a high level is applied on the OCREF_CLR signal. The PWM then restarts (output re-enabled) at the next counter overflow.

But if the PWM is configured at 100% (CCxR > ARR), then it does not restart and OCxREF remains de-asserted.

Workaround

None.

2.17 LSI clock stabilization time

Description

When the LSIRDY flag is set, the clock may still be out of the specified frequency range (f_{LSI} parameter, see LSI oscillator characteristics in the product datasheet).

Workaround

To have a fully stabilized clock in the specified range, a software temporization of 100 µs should be added.

2.18 FSMC limitations

2.18.1 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.

2.18.2 1 dummy clock cycle inserted when writing to synchronous memories when CLKDIV=1

Description

When performing a write access to a synchronous memory and CLKDIV=1 (in FSMC_BTRx register), one dummy clock cycle is generated after nWE is de-assserted whatever the type of write burst access. However, there is no dummy write to the memory since the extra clock is generated while nWE is de-asserted.

Workaround

None.

Appendix A Revision code on device marking

Figure 2, *Figure 3*, *Figure 4*, *Figure 5*, *Figure 6* and *Figure 7* show the marking compositions for the LFBGA144, LFBGA100, WLCSP64, 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 2. LFBGA144 top package view

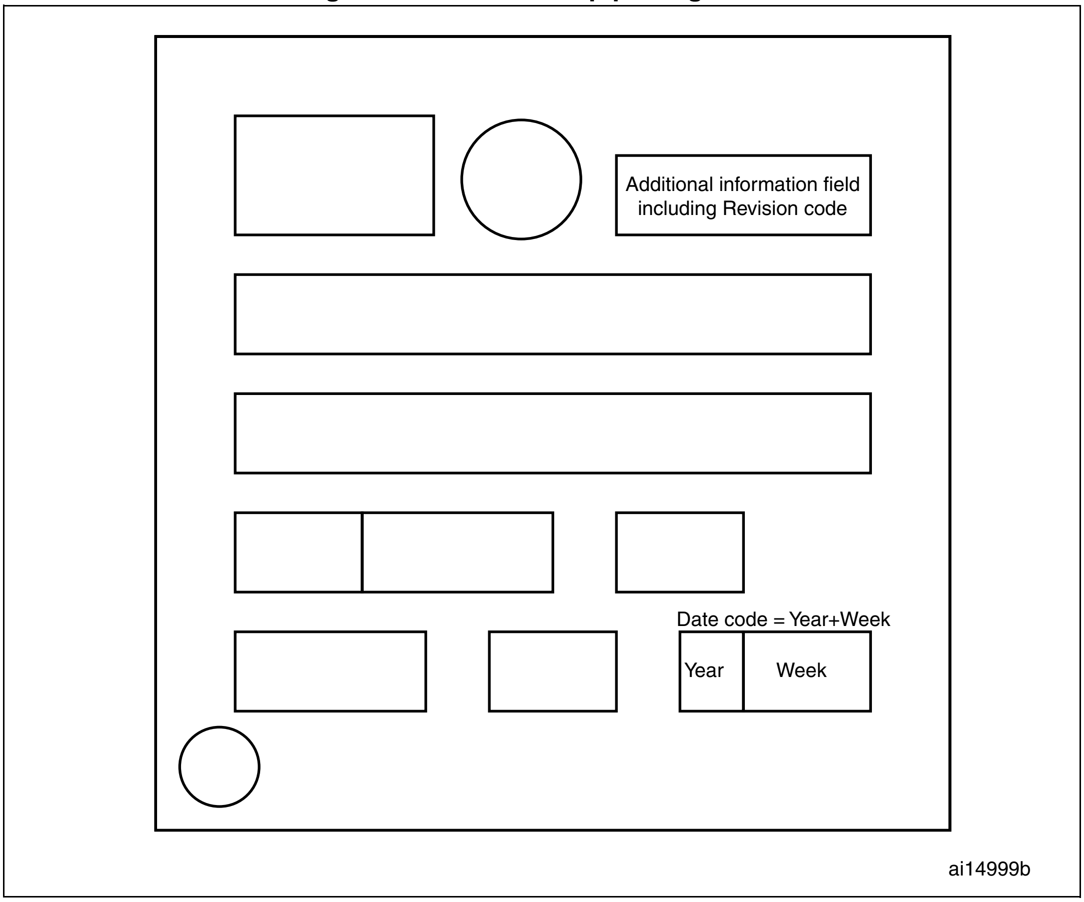


Figure 3. LFBGA100 top package view

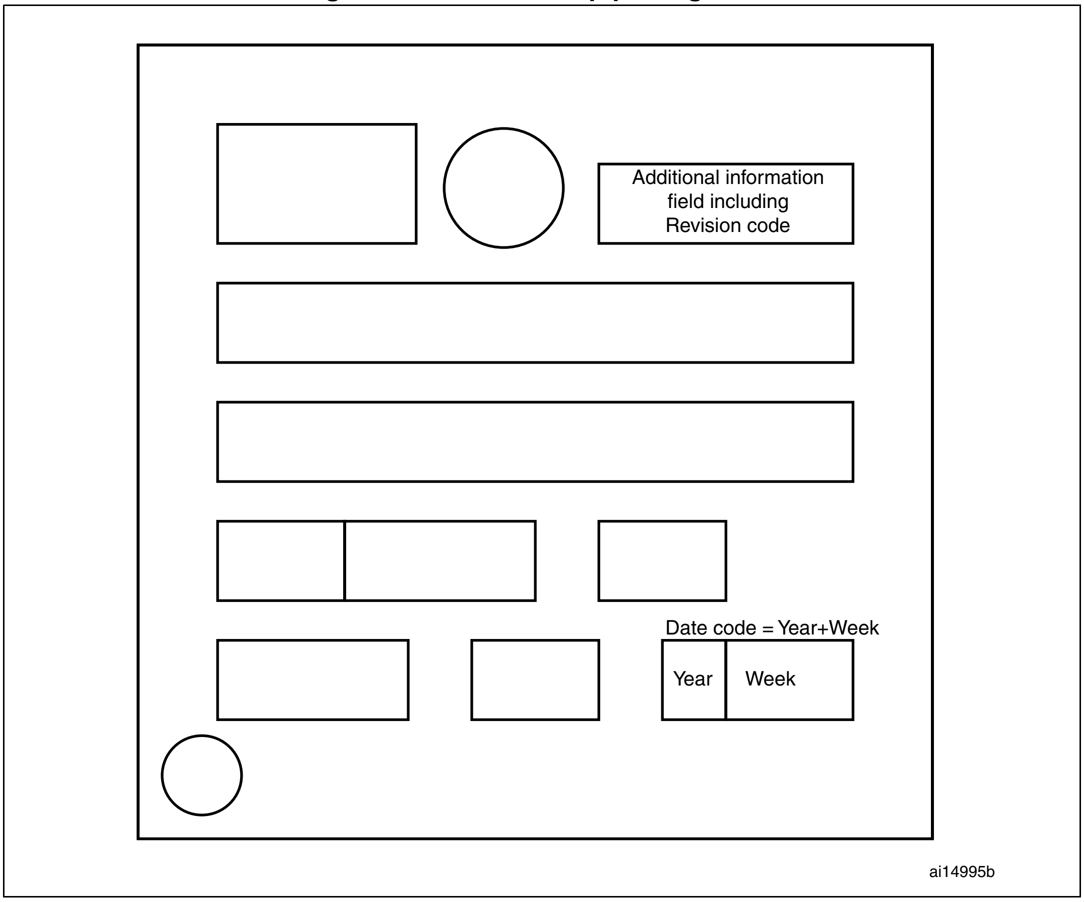


Figure 4. WLCSP64 top package view

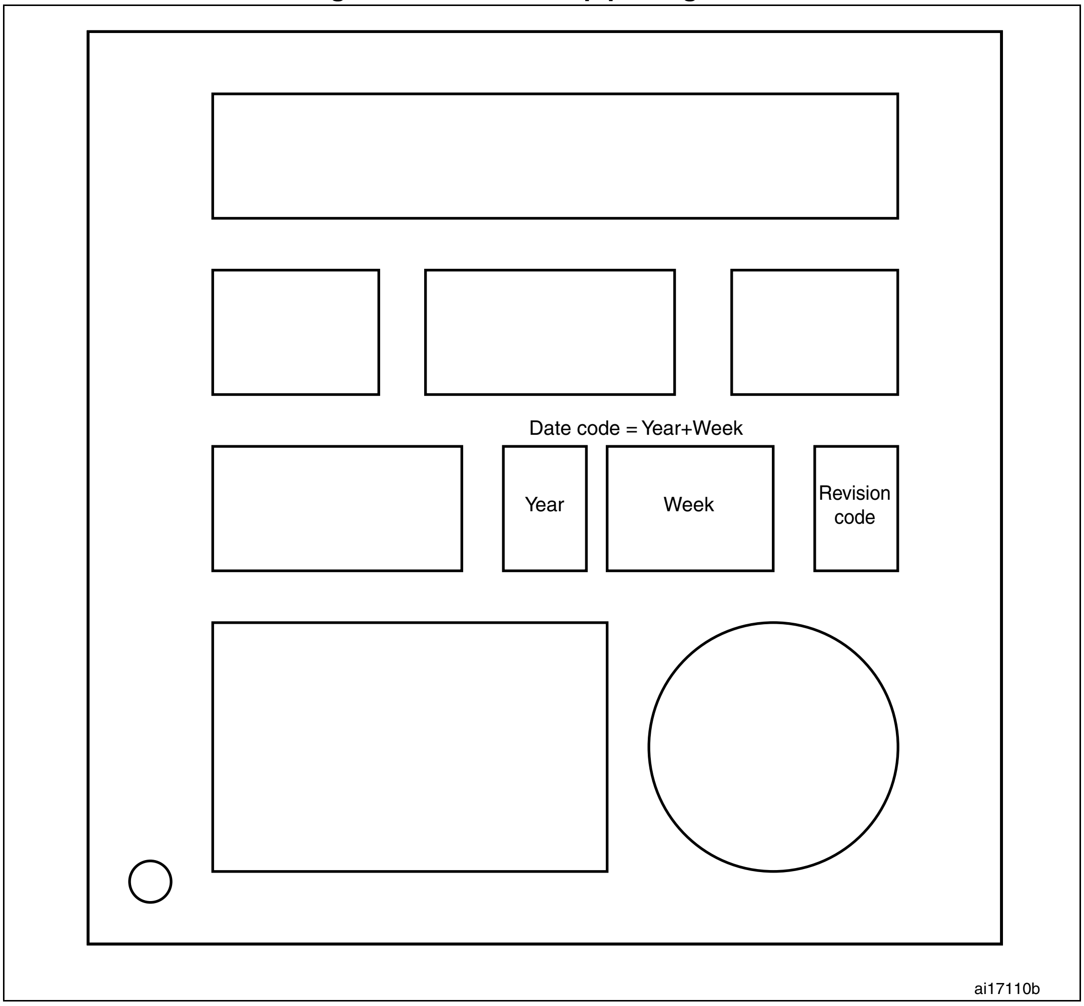


Figure 5. LQFP144 top package view

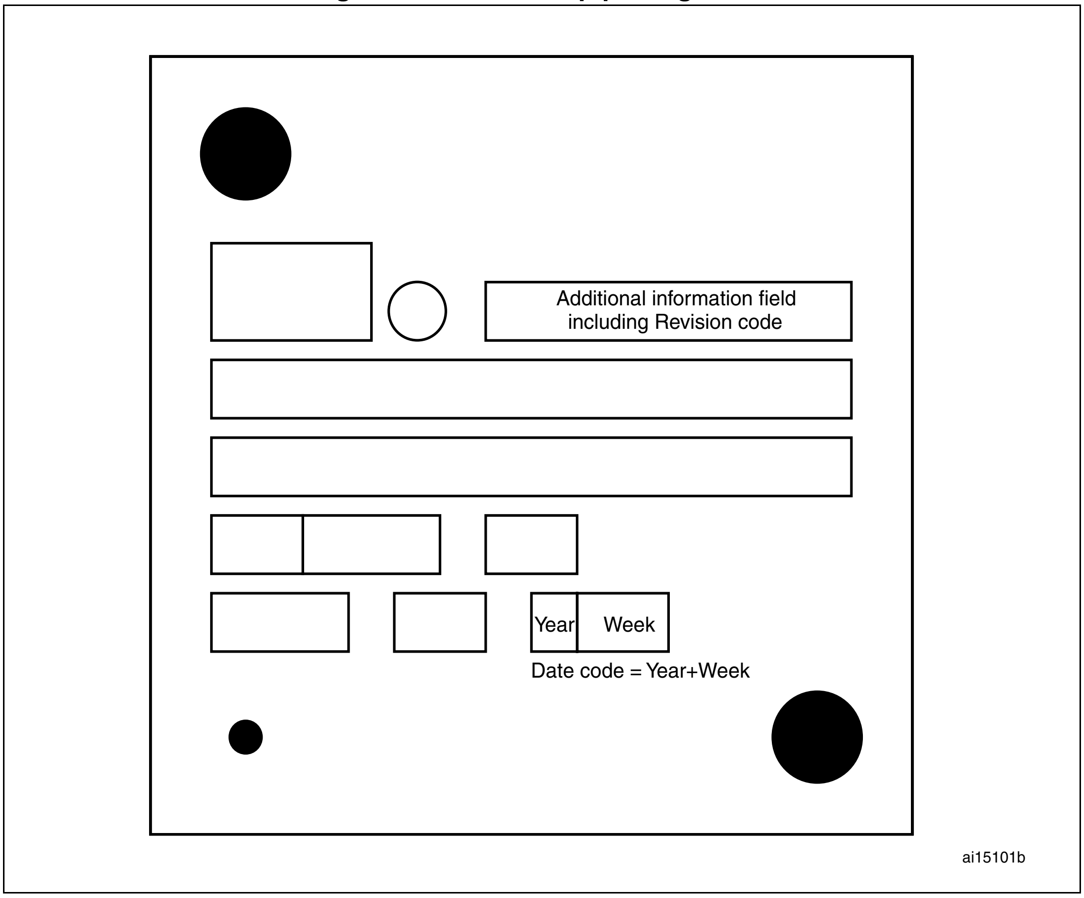


Figure 6. LQFP100 top package view

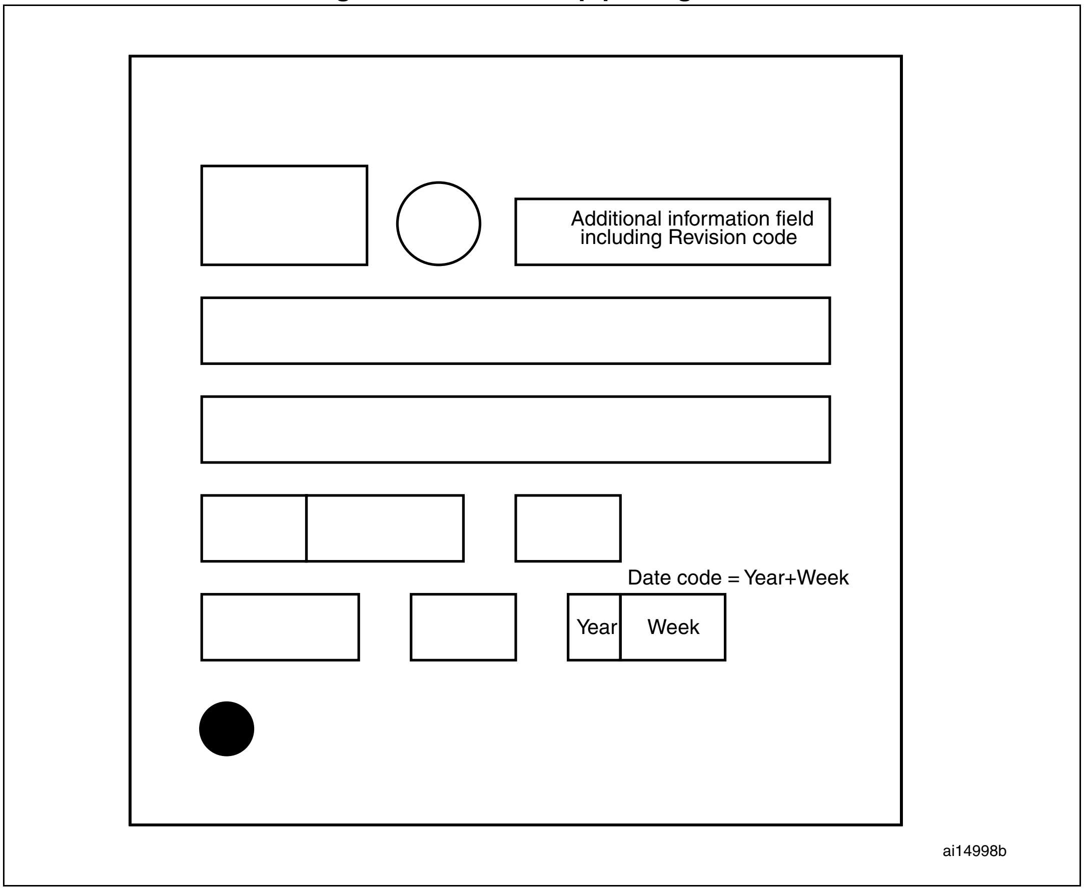
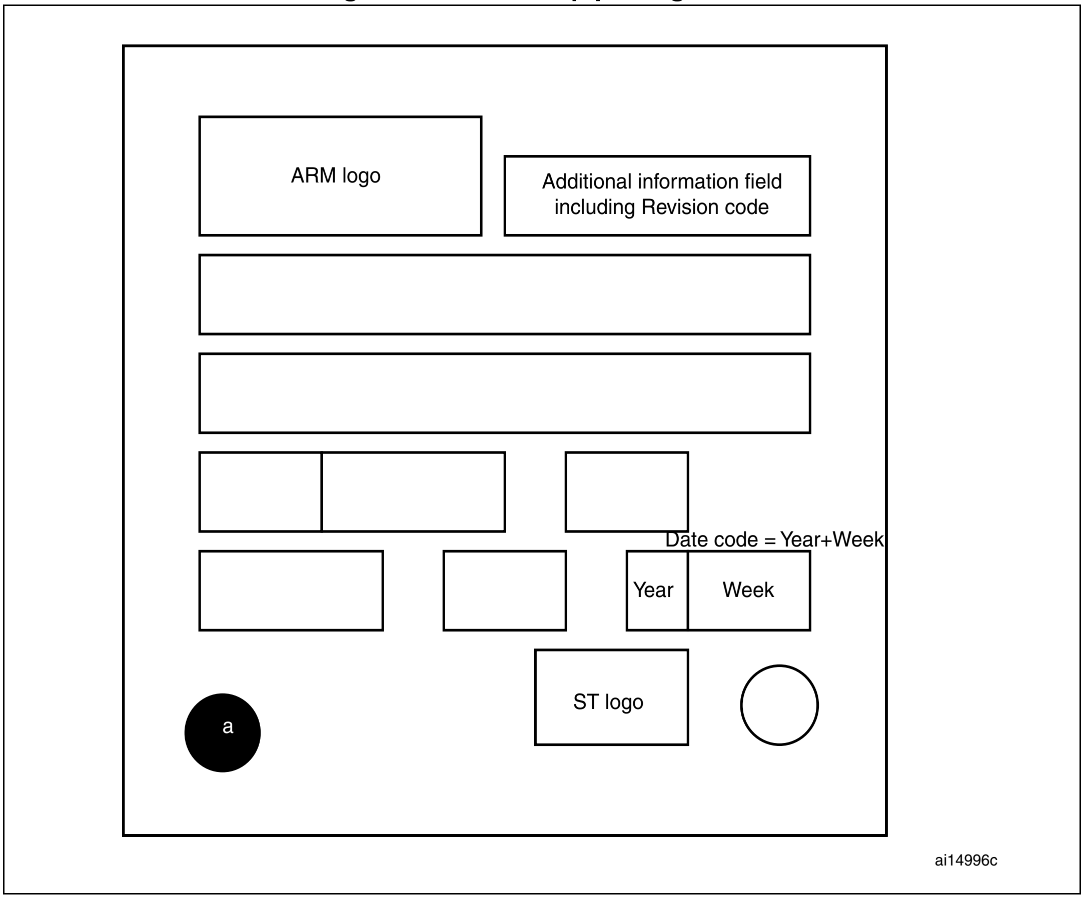


Figure 7. LQFP64 top package view



Revision history

Table 5. Document revision history

Date	Revision	Changes
14-May-2010	1	Initial release.
18-Jun-2010	2	Added *Section 2.4: Debugging Stop mode and system tick timer* Added *Section 2.5: Debugging Stop mode with WFE entry* Added *Section 2.14: USART peripheral*
07-Feb-2011	3	Updated workarounds in *Section 2.11.1: Some software events must be managed before the current byte is being transferred* and *Section 2.11.2: Wrong data read into data register* Added *Section 2.13: I2S peripheral* Added *Section 2.14.6: nRTS signal abnormally driven low after a protocol violation*
26-Sep-2013	4	Added: – *Section 2.6: Wakeup sequence from Standby mode when using more than one wakeup source* – *Section 2.7: LSE start-up in harsh environments* – *Section 2.11.7: I2C analog filter may provide wrong value, locking BUSY flag and preventing master mode entry* – *Section 2.12.2: SPI CRC may be corrupted when a peripheral connected to the same DMA channel of the SPI is under DMA transaction close to the end of transfer or end of transfer -1* – *Section 2.15: SDIO peripheral*. Updated *Table 4: Summary of silicon limitations*.
18-Nov-2015	5	Updated: – *Silicon identification* – *Table 1: Device identification*