STM32F10xx8 and STM32F10xxB Errata sheet

STM32F101x8/B, STM32F102x8/B and STM32F103x8/B medium-density device limitations

Silicon identification

This errata sheet applies to the revisions B, Z, Y and 1 of the STMicroelectronics medium-density STM32F101xx access line and STM32F103xx performance line products, and to revision Y of the STM32F102xx USB access line devices.
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 root 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
STM32F101xxx(3)	“B”, “Z” , “Y” or “1”
STM32F102xxx(3)	“Y” or “1”
STM32F103xxx(3)	“B”, “Z” , “Y” 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.
3. Are also concerned all devices with 32 KB of Flash memory that do not have the letter A in their order code.

Table 2. Device summary

Reference	Part number
STM32F101xx	STM32F101C8, STM32F101R8 STM32F101V8, STM32F101T8
	STM32F101RB, STM32F101VB, STM32F101CB
STM32F102xx	STM32F102C8, STM32F102R8
	STM32F102CB, STM32F102RB
STM32F103xx	STM32F103C8, STM32F103R8 STM32F103V8, STM32F103T8
	STM32F103RB STM32F103VB, STM32F103CB

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/help/index.jsp?topic=/com.arm.doc.eat0420a/*.
The direct link to the errata notice pdf is:
http://infocenter.arm.com/help/topic/com.arm.doc.eat0420a/Cortex-M3-Errata-r1p1-v0.2.pdf.
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 medium-density STM32F10xxx devices.

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

ARM ID	ARM category	ARM summary of errata	Impact on medium-density STM32F10xxx devices
752419	Cat 2	Interrupted loads to SP can cause erroneous behavior	Minor
740455	Cat 2	SVC and BusFault/MemManage may occur out of order	Minor
602117	Cat 2	LDRD with base in list may result in incorrect base register when interrupted or faulted	Minor
563915	Cat 2	Event register is not set by interrupts and debug	Minor
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	No

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

Only the limitations described below have an impact, even though minor, on the implementation of STM32F10xxx medium-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 medium-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 Interrupted loads to SP can cause erroneous behavior

Description

If an interrupt occurs during the data-phase of a single word load to the stack-pointer (SP/R13), erroneous behavior can occur. In all cases, returning from the interrupt will result in the load instruction being executed an additional time. For all instructions performing an update to the base register, the base register will be erroneously updated on each execution, resulting in the stack-pointer being loaded from an incorrect memory location.

The affected instructions are:

1. LDR SP,[Rn],#imm
2. LDR SP,[Rn,#imm]!
3. LDR SP,[Rn,#imm]
4. LDR SP,[Rn]
5. LDR SP,[Rn,Rm]

Workaround

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

Both issues may be worked around by replacing the direct load to the stack-pointer, with an intermediate load to a general-purpose register followed by a move to the stack-pointer.

Example: the following instruction "LDR SP, [R0]" can be replaced by

“LDR R2,[R0]

MOV SP,R2 "

1.1.6 SVC and BusFault/MemManage may occur out of order

Description

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

Workaround

A workaround is only required if the SVC handler will not return to the return address that has been stacked for the SVC exception and the instruction access after the SVC will fault. If this is the case then padding can be inserted between the SVC and the faulting area of code, for example, by inserting NOP instructions.

2 STM32F10xxx 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: Debug registers 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: Alternate function	*Section 2.6.1: USART1_RTS and CAN_TX*
	Section 2.6.2: SPI1 in slave mode and USART2 in synchronous mode
	Section 2.6.3: SPI1 in master mode and USART2 in synchronous mode
	Section 2.6.4: SPI2 in slave mode and USART3 in synchronous mode
	Section 2.6.5: SPI2 in master mode and USART3 in synchronous mode
	Section 2.6.6: I2C2 with SPI2 and USART3
	Section 2.6.7: I2C1 with SPI1 remapped and used in master mode
	Section 2.6.8: I2C1 and TIM3_CH2 remapped
	Section 2.6.9: USARTx_TX pin usage
Section 2.7: PVD and USB wakeup events	
Section 2.8: Compatibility issue with latest compiler releases	
Section 2.9: Boundary scan TAP: wrong pattern sent out after the “capture IR” state	
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: Wrong behavior of I2C peripheral in master mode 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 2.12: SPI peripheral	*Section 2.12.1: CRC still sensitive to communication clock when SPI is in slave mode even with NSS high*

Table 4. Summary of silicon limitations

Links to silicon limitations	
Section 2.13: USART peripheral	*Section 2.13.1: Parity Error flag (PE) is set again after having been cleared by software*
	Section 2.13.2: Idle frame is not detected if receiver clock speed is deviated
	Section 2.13.3: In full duplex mode, the Parity Error (PE) flag can be cleared by writing the data register
	Section 2.13.4: Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection
	Section 2.13.5: Break frame is transmitted regardless of nCTS input line status
	Section 2.13.6: nRTS signal abnormally driven low after a protocol violation
Section 2.14: Timers	*Section 2.14.1: Missing capture flag*
	Section 2.14.2: Overcapture detected too early
	Section 2.14.3: General-purpose timer: regulation for 100% PWM
Section 2.15: IWDG peripheral	*Section 2.15.1: RVU and PVU flags are not cleared in Stop mode*
Section 2.16: LSI clock stabilization time	
Section 2.17: USB packet buffer memory: over/underrun or COUNTn_RX[9:0] field reporting incorrect number if APB1 frequency is below 13 MHz	

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 Debug registers cannot be read by user software

Description

The DBGMCU_IDCODE and DBGMCU_CR debug registers are accessible only in debug mode (not accessible by the user software). When these registers are 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 Alternate function

In some specific cases, some potential weakness may exist between alternate functions mapped onto the same pin.

2.6.1 USART1_RTS and CAN_TX

Conditions

- USART1 is clocked
- CAN is not clocked
- I/O port pin PA12 is configured as an alternate function output.

Description

Even if CAN_TX is not used, this signal is set by default to 1 if I/O port pin PA12 is configured as an alternate function output.

In this case USART1_RTS cannot be used.

Workaround

When USART1_RTS is used, the CAN must be remapped to either another IO configuration when the CAN is used, or to the unused configuration (CAN_REMAP[1:0] set to “01”) when the CAN is not used.

2.6.2 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.

2.6.3 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.

2.6.4 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.

2.6.5 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,

2.6.6 I2C2 with SPI2 and USART3

Conditions

- I2C2 and SPI2 are clocked together or I2C2 and USART3 are clocked together
- I/O port pin PB12 is configured as an alternate function output

Description

- Conflict between the I2C2 SMBA signal (even if this function is not used) and SPI2_NSS in output mode.
- Conflict between the I2C2 SMBA signal (even if this function is not used) and USART3_CK.
- In these cases the I/O port pin PB12 is set to 1 by default if the I/O alternate function output is selected and I2C2 is clocked.

Workaround

I2C2 SMBA can be used as an output if SPI2 is configured in master mode with NSS in software mode.
I2C2 SMBA can be used in input mode if SPI2 is configured in master or slave mode with NSS managed by software.
SPI2 cannot be used in any other configuration when I2C2 is being used.

USART3 must *not* be used in synchronous mode when I2C2 is being used.

2.6.7 I2C1 with SPI1 remapped and used in master mode

Conditions

- I2C1 and SPI1 are clocked.
- SPI1 is remapped.
- I/O port pin PB5 is configured as an alternate function output.

Description

Conflict between the SPI1 MOSI signal and the I2C1 SMBA signal (even if SMBA is not used).

Workaround

Do not use SPI1 remapped in master mode and I2C1 together.

When using SPI1 remapped, the I2C1 clock must be disabled.

2.6.8 I2C1 and TIM3_CH2 remapped

Conditions

- I2C1 and TIM3 are clocked.
- I/O port pin PB5 is configured as an alternate function output.

Description

Conflict between the TIM3_CH2 signal and the I2C1 SMBA signal, (even if SMBA is not used).

In these cases the I/O port pin PB5 is set to 1 by default if the I/O alternate function output is selected and I2C1 is clocked. TIM3_CH2 cannot be used in output mode.

Workaround

To avoid this conflict, TIM3_CH2 can only be used in input mode.

2.6.9 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).

2.7 PVD and USB wakeup events

Description

PVD and USB Wakeup, which are internally linked to EXTI line16 and EXTI line18, respectively, cannot be used as event sources for the Cortex-M3 core. As a consequence, these signals cannot be used to exit the Sleep or the Stop mode (exit WFE).

Workaround

Use interrupt sources and the WFI instruction if the application must be woken up from the Sleep or the Stop mode by PVD or USB Wakeup.

2.8 Compatibility issue with latest compiler releases

Description

Compilers with improved optimizations for the STM32F10xxx have been recently released on the market. Revisions Z and B of the medium-density STM32F10xxx devices (STM32F10xx8/B) do not support some of the sequences associated with the high-level optimizations done in these compilers. Revision Y and 1 are not affected by this limitation.

Workaround

This behavior is fully deterministic, and should be detected during firmware development or the validation phase. Consequently, systems already developed, validated and delivered to the field with previous silicon revisions are not affected.

For code update of revision Z and B devices already in the field, do not use these new compilers. To date, compilers known to generate these sequences are:

- IAR EWARM rev 5.20 and later
- GNU rev 4.2.3 and later

For new developments associated with these compilers, revision Y or 1 of the STM32F10xx8/B must be used.

2.9 Boundary scan TAP: wrong pattern sent out after the “capture IR” state

Description

After the “capture IR” state of the boundary scan TAP, the two least significant bits in the instruction register should be loaded with “01” for them to be shifted out whenever a next instruction is shifted in.
However, the boundary scan TAP shifts out the latest value loaded into the instruction register, which could be “00”, “01”, “10” or “11”.

Workaround

The data shifted out, after the capture IR state, in the boundary scan flow should therefore be ignored and the software should check not only the two least significant bits (XXX01) but all register bits (XXXXX).

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 I^2C 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 Wrong behavior of I2C peripheral in master mode after a misplaced Stop

Description

If a misplaced Stop is generated on the bus, the peripheral cannot enter master mode properly:

- If a void message is received (START condition immediately followed by a STOP): the BERR (bus error) flag is not set, and the I2C peripheral is not able to send a start condition on the bus after the write to the START bit in the I2C_CR2 register.
- In the other cases of a misplaced STOP, the BERR flag is set. If the START bit is already set in I2C_CR2, the START condition is not correctly generated on the bus and can create bus errors.

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.

In case of a noisy environment in which unwanted bus errors can occur, it is recommended to implement a timeout to ensure that after the START control bit is set, the SB (start bit) flag is set. In case the timeout has elapsed, the peripheral must be reset by setting the SWRST bit in the I2C_CR2 control register. It should also be reset in the same way if a BERR is detected while the START bit is set in I2C_CR2.

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. Moreover, if the data register is written too late and close to the SCL rising edge, an error can be generated on the bus (SDA toggles while SCL is high). These violations 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 in writing the DR data register, but not late enough to set the OVR flag (the data register is written before the SCL rising edge).

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 writes to the data register fast enough after TXE or ADDR events. For instance, use an interrupt on the TXE or ADDR flag and boost its priority to the higher level, or use DMA. Use this "NOSTRETCH" mode with a slow I2C bus speed.

Note: The first data byte to transmit must be written in the data register after the ADDR flag is cleared, and before the next SCL rising edge, so that the time window for writing the first data byte in the data register is less than t_{LOW}.

If this is not possible, a workaround can be used:

Clear the ADDR flag

Wait for the OVR flag to be set

Clear OVR and write the first data byte.

Then the time window for writing the next data byte will be the time to transfer one byte. In this case, the master must discard the first received data byte.

2.12 SPI peripheral

2.12.1 CRC still 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)

2.13 USART peripheral

2.13.1 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.

2.13.2 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.

2.13.3 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.

2.13.4 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.

2.13.5 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.

2.13.6 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.

Workarounds

Note: These workarounds are needed only if the other UART device has violated the protocol. In most systems (no limitation on the other device), the USART works fine and no workaround is needed.

Workaround 1: After data reception and before reading the data in the data register, the software takes control of the nRTS pin using the GPIO registers and keeps it high as long as needed. If the application knows the USART is not ready and that further data received reception from the other device may be discarded, it keeps the nRTS pin at high level. It then releases the nRTS pin when the USART is ready to continue reception.

Workaround 2: Ensure that the received data is always read in a time window less than the duration of the 2nd data reception. One solution would be to handle all data reception by DMA.

2.14 Timers

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

2.14.1 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. Possibly, a missed capture will be detected by the EXTI peripheral.

2.14.2 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.

2.14.3 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.

2.15 IWDG peripheral

2.15.1 RVU and PVU flags are not cleared in Stop mode

Description

The RVU and PVU flags in the IWDG_SR register are set by hardware after a write access to the IWDG_RLR or the IWDG_PR registers, respectively. If MCU enters Stop mode immediately after the write access, the RVU and PVU flags are not cleared by hardware. Consequently the next time the application attempts to write to the IWDG_RLR or the IWDG_PR registers, it waits in an infinite loop for the RVU and PVU flags to be cleared and the IWDG generates a reset after the programmed time-out period.

Workaround

The application has to wait until the RVU and PVU flags in the IWDG_SR register are cleared before entering Stop mode.

2.16 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.

2.17 USB packet buffer memory: over/underrun or COUNTn_RX[9:0] field reporting incorrect number if APB1 frequency is below 13 MHz

Description

The USB peripheral's packet buffer memory is expected to operate at a minimum APB1 frequency of 8 MHz.

It may however happen that, when OUT transactions are sent by the Host with a data payload size exactly equal to the maximum packet size already programmed in the COUNTn_RX packet buffer memory (via the BLSIZE and NUM_BLOCK[4:0] fields), the packet and all bytes from the Host are correctly received and stored into the packet buffer memory, but, the COUNTn_RX[9:0] field indicates an incorrect number (one byte less).

Workaround

This limitation concerns applications that check the exact number of bytes received in the packet buffer memory. In order to avoid that these applications interpret a Host error and so, stall the OUT endpoint even if no data reception error actually occurred, it is recommended to:

1. increase the APB1 frequency to a minimum of 13 MHz, or
2. increase the APB1 frequency to a minimum of 10 MHz. Then program USB_COUNTn_RX (via the BLSIZE and NUM_BLOCK[4:0] fields) to have more than the number of bytes in the maximum packet size allocated for reception in the packet buffer memory

Appendix A Revision code on device marking

Figure 1, *Figure 2*, *Figure 3*, *Figure 4* and *Figure 5* show the marking compositions for the LFBGA100, LQFP100, LQFP64, LQFP48, VFQFPN36 and VFQFPN48 packages, respectively. Only the Additional field containing the Revision code is shown.

Figure 1. LFBGA100 top package view

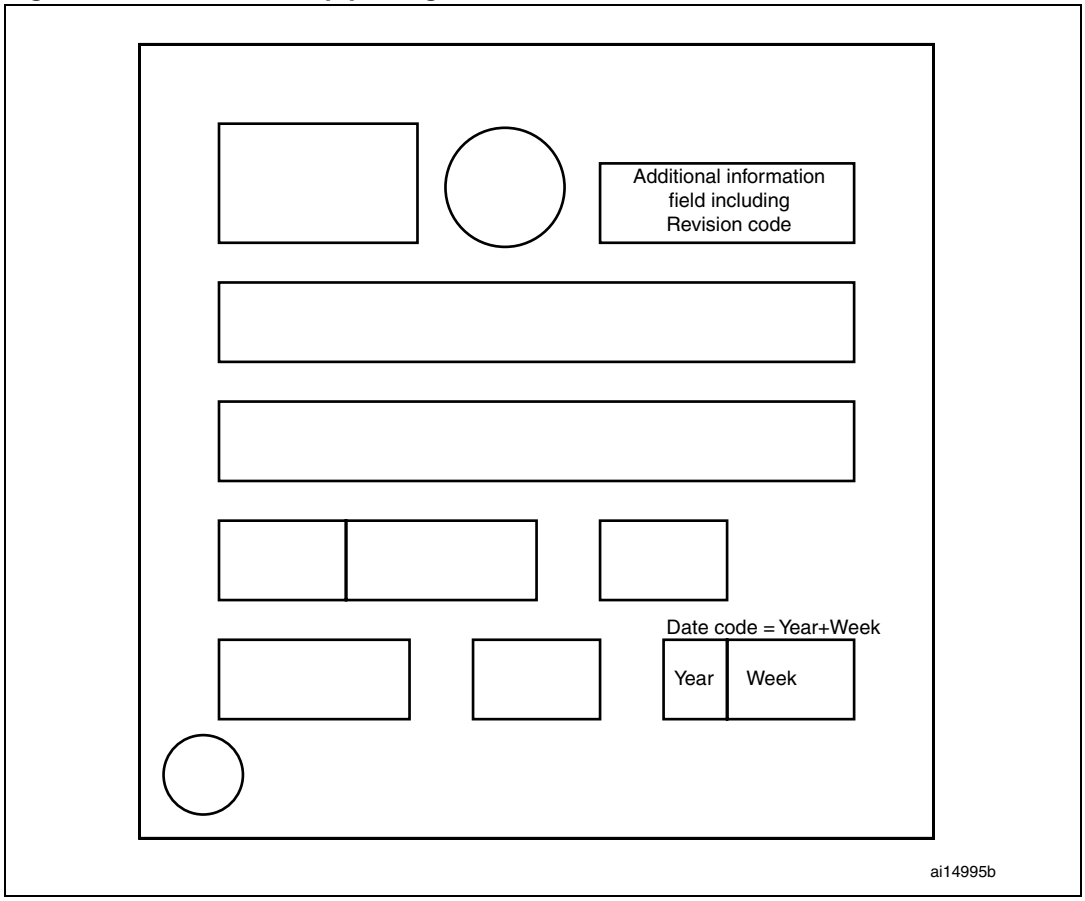


Figure 2. LQFP100 top package view

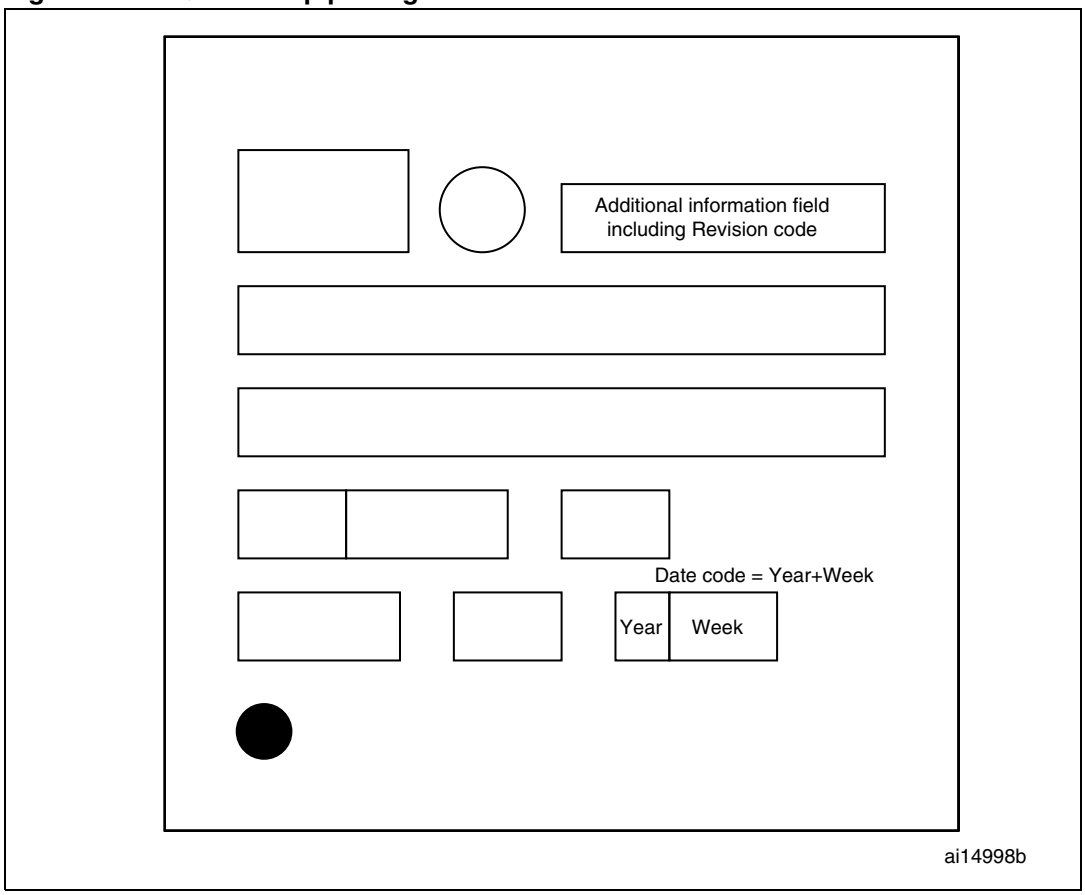


Figure 3. LQFP64 top package view

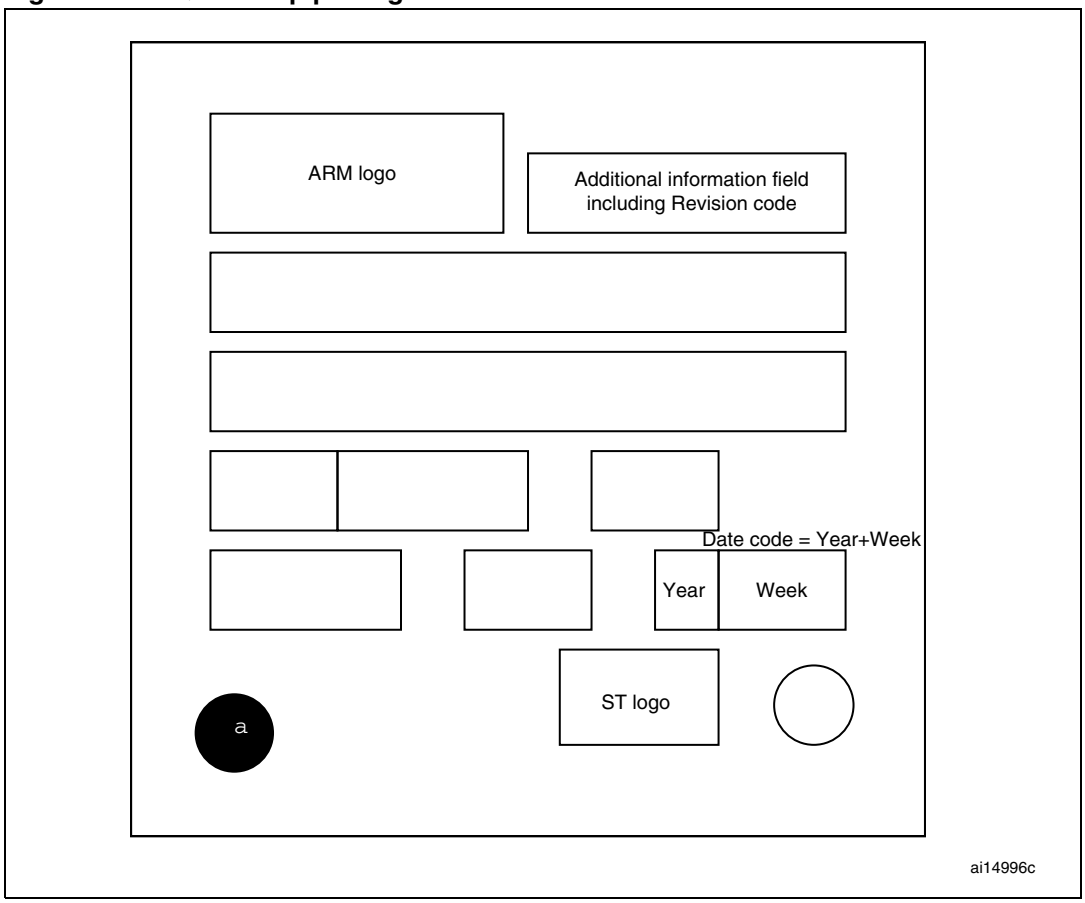


Figure 4. LQFP48 top package view

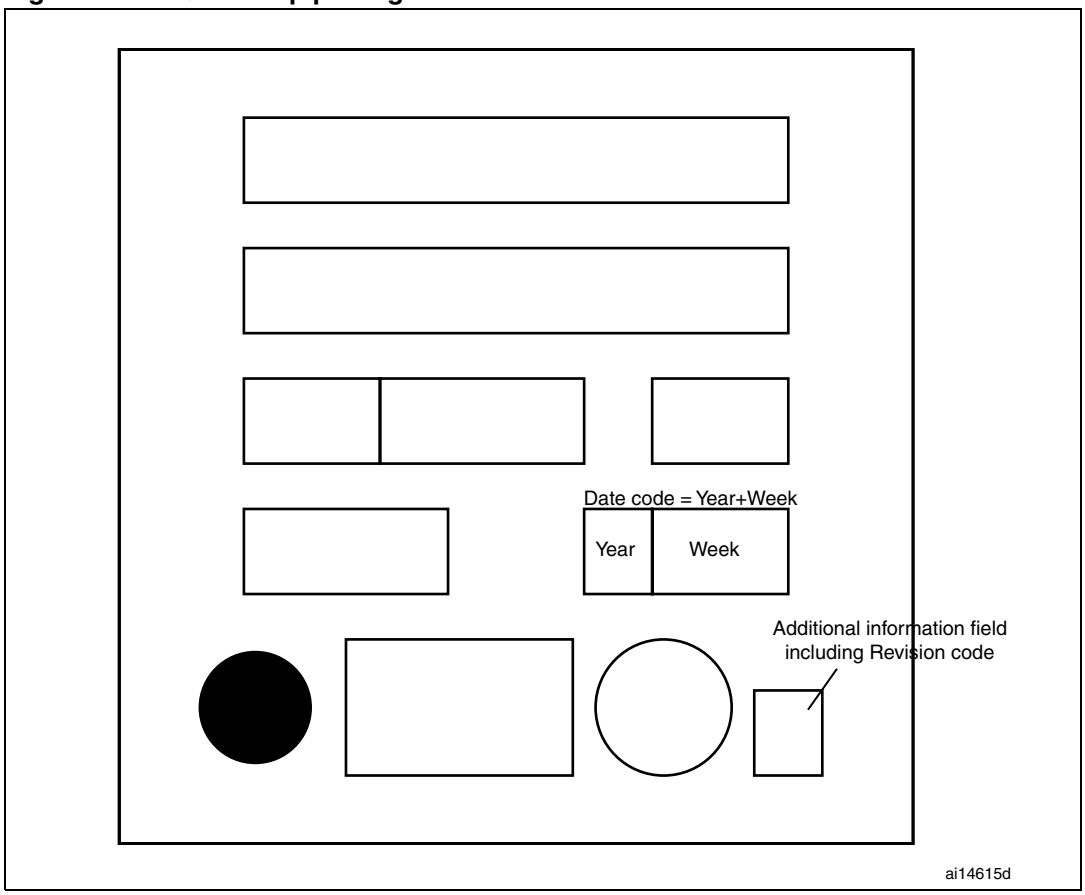
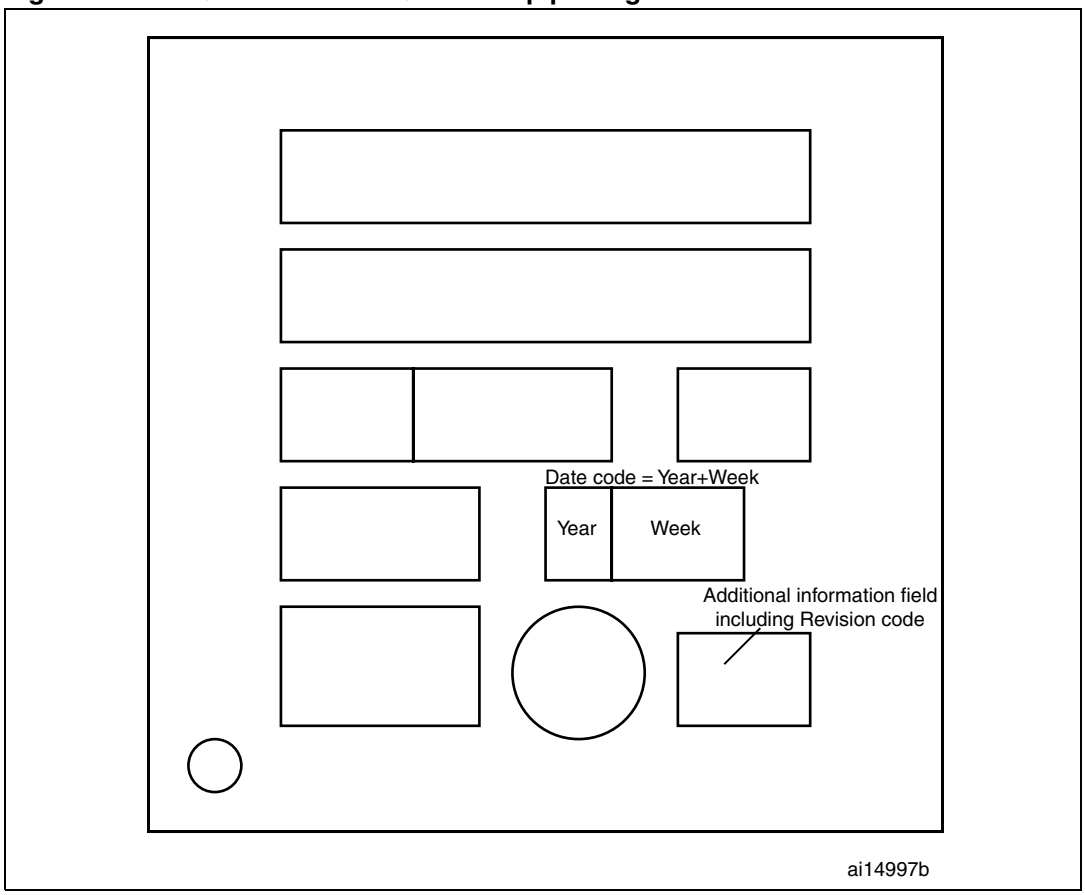


Figure 5. VFQFPN36 and VFQFPN48 top package view



Revision history

Table 5. Document revision history

Date	Revision	Changes
28-Mar-2008	1	Initial release.
07-Apr-2008	2	*Section 2.2: Flash memory read after WFI/WFE instruction on page 12* added. Workaround specified in *Section 2.6.1: USART1_RTS and CAN_TX*.
23-May-2008	3	The errata sheet also applies to Revision Y devices. *Section 2.1: PD0 and PD1 use in output mode*, *Section 2.2: ADC auto-injection channel* and *Section 2.3: ADC combined injected simultaneous+interleaved* removed from errata sheet. *Section 2.3: Debug registers cannot be read by user software on page 12* added. Small text changes.
18-Jul-2008	4	*Section 2.7: PVD and USB wakeup events* added.
01-Oct-2008	5	This errata sheet also applies to STM32F102xx medium-density devices. Though medium-density devices with 32 Kbyte of Flash were removed, the errata sheet still applies to devices whose commercial code does not contain an “A”. *Section 2.8: Compatibility issue with latest compiler releases* added. *Figure 1: LFBGA100 top package view* added. *Figure 3: LQFP64 top package view* and *Figure 4: LQFP48 top package view* corrected.
11-Feb-2009	6	*Section 1: ARM™ 32-bit Cortex®-M3 limitations* specified (*Table 3: Cortex-M3 core limitations and impact on microcontroller behavior* added limitations described). Added limitations: – *Boundary scan TAP: wrong pattern sent out after the “capture IR” state* – *Flash memory BSY bit delay versus STRT bit setting* – *I2C peripheral* – *Timers* – *LSI clock stabilization time* *Table 4: Summary of silicon limitations on page 10* added.
11-Jan-2010	7	Added limitations: – *Section 2.6.9: USARTx_TX pin usage* – *Section 2.11.4: Wrong behavior of I2C peripheral in master mode 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 2.12.1: CRC still sensitive to communication clock when SPI is in slave mode even with NSS high* – *Section 2.17: USB packet buffer memory: over/underrun or COUNTn_RX[9:0] field reporting incorrect number if APB1 frequency is below 13 MHz* Date code added to *Figure 1* to *Figure 5*.

Table 5. Document revision history (continued)

Date	Revision	Changes
21-Jun-2010	8	Added *Section 2.4: Debugging Stop mode and system tick timer* Added *Section 2.5: Debugging Stop mode with WFE entry* Added *Section 2.11.2: Wrong data read into data register* Updated *Section 2.11.4: Wrong behavior of I2C peripheral in master mode after a misplaced Stop* Added *Section 2.13: USART peripheral* Updated *Section 2.11.6: Data valid time (tVD;DAT) violated without the OVR flag being set*
22-Feb-2011	9	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 *Section 2.13.6: nRTS signal abnormally driven low after a protocol violation*
17-Jun-2011	10	Added reference to revision code 1 in *Table 1* Added *Section 1.1.5: Interrupted loads to SP can cause erroneous behavior* *Section 1.1.6: SVC and BusFault/MemManage may occur out of order* Added VFQFPN48 package to *Figure 5*
15-Nov-2011	11	Added reference to revision code 1 in *Section 2.8: Compatibility issue with latest compiler releases* Added *Section 2.15: IWDG peripheral* Updated *Section 2.13.6: nRTS signal abnormally driven low after a protocol violation*

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