



# STM32F101xC/D/E and STM32F103xC/D/E Errata sheet

## STM32F101xC/D/E and STM32F103xC/D/E high-density device limitations

### Silicon identification

This errata sheet applies to the revisions Z, Y and 1 of the STMicroelectronics STM32F101xC/D/E access line and STM32F103xC/D/E performance line high-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<sup>(1)</sup>**

Order code	Revision code <sup>(2)</sup> marked on device
STM32F103xC, STM32F103xD, STM32F103xE	“Z”, “Y” or “1”
STM32F101xC, STM32F101xD, STM32F101xE	“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.

**Table 2. Device summary**

Reference	Part number
STM32F101xCDE	STM32F101RC STM32F101VC STM32F101ZC
	STM32F101RD STM32F101VD STM32F101ZD
	STM32F101RE STM32F101VE STM32F101ZE
STM32F103xCDE	STM32F103RC STM32F103VC STM32F103ZC
	STM32F103RD STM32F103VD STM32F103ZD
	STM32F103RE STM32F103VE STM32F103ZE

# Contents

- 1      ARM™ 32-bit Cortex®-M3 limitations ..... 7**
- 1.1   Cortex-M3 limitations description for STM32F10xxx high-density devices 7
  - 1.1.1   Cortex-M3 LDRD with base in list may result in incorrect base register when interrupted or faulted ..... 8
  - 1.1.2   Cortex-M3 event register is not set by interrupts and debug ..... 8
  - 1.1.3   Cortex-M3 BKPT in debug monitor mode can cause DFSR mismatch .. 8
  - 1.1.4   Cortex-M3 may freeze for SLEEPONEXIT single instruction ISR ..... 9
  - 1.1.5   Interrupted loads to SP can cause erroneous behavior ..... 9
  - 1.1.6   SVC and BusFault/MemManage may occur out of order ..... 9
  
- 2      STM32F10xxx silicon limitations ..... 11**
- 2.1   Voltage glitch on ADC input 0 ..... 14
- 2.2   Flash memory read after WFI/WFE instruction ..... 14
- 2.3   Debug registers cannot be read by user software ..... 14
- 2.4   Debugging Stop mode and system tick timer ..... 15
- 2.5   Debugging Stop mode with WFE entry ..... 15
- 2.6   Alternate function ..... 15
  - 2.6.1   USART1\_RTS and CAN\_TX ..... 15
  - 2.6.2   SPI1 in slave mode and USART2 in synchronous mode ..... 16
  - 2.6.3   SPI1 in master mode and USART2 in synchronous mode ..... 16
  - 2.6.4   SPI2 in slave mode and USART3 in synchronous mode ..... 16
  - 2.6.5   SPI2 in master mode and USART3 in synchronous mode ..... 17
  - 2.6.6   SDIO with TIM8 ..... 17
  - 2.6.7   SDIO and TIM3\_REMAP ..... 17
  - 2.6.8   SDIO with USART3 remapped and UART4 ..... 18
  - 2.6.9   FSMC with I2C1 and TIM4\_CH2 ..... 18
  - 2.6.10   FSMC with USART2 remapped ..... 18
  - 2.6.11   FSMC with USART3 and TIM1 remapped ..... 19
  - 2.6.12   I2S2 in master/slave mode and USART3 in synchronous mode ..... 19
  - 2.6.13   USARTx\_TX pin usage ..... 19
- 2.7   PVD and USB wakeup events ..... 20
- 2.8   SPI3 in I<sup>2</sup>S slave mode: timing sensitivity between I2S3\_WS and I2S3\_CK ..... 20
- 2.9   Boundary scan TAP: wrong pattern sent out after the “capture IR” state . 20



2.10	Flash memory BSY bit delay versus STRT bit setting	21
2.11	I <sup>2</sup> C peripheral	21
2.11.1	Some software events must be managed before the current byte is being transferred	21
2.11.2	Wrong data read into data register	23
2.11.3	SMBus standard not fully supported	24
2.11.4	Wrong behavior of I2C peripheral in master mode after a misplaced Stop	24
2.11.5	Mismatch on the “Setup time for a repeated Start condition” timing parameter	25
2.11.6	Data valid time ( $t_{VD;DAT}$ ) violated without the OVR flag being set	25
2.12	SPI peripheral	26
2.12.1	CRC still sensitive to communication clock when SPI is in slave mode even with NSS high	26
2.12.2	SPI2/I2S2 slave mode wrong behavior in transmission and reception	27
2.13	I2S peripheral	27
2.13.1	Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronisation mode	27
2.13.2	In I2S slave mode, WS level must be set by the external master when enabling the I2S	27
2.13.3	I2S slave mode desynchronisation with the master during communication	28
2.14	USART peripheral	28
2.14.1	Parity Error flag (PE) is set again after having been cleared by software	28
2.14.2	Idle frame is not detected if receiver clock speed is deviated	29
2.14.3	In full duplex mode, the Parity Error (PE) flag can be cleared by writing the data register	29
2.14.4	Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection	29
2.14.5	Break frame is transmitted regardless of nCTS input line status	29
2.14.6	nRTS signal abnormally driven low after a protocol violation	30
2.15	Timers	30
2.15.1	Missing capture flag	30
2.15.2	Overcapture detected too early	30
2.15.3	General-purpose timer: regulation for 100% PWM	31
2.16	LSI clock stabilization time	31
2.17	FSMC limitations	31
2.17.1	Multimaster access on the FSMC memory map	31

---

2.17.2	Dummy read cycles inserted when reading synchronous memories . . .	32
2.17.3	1 dummy clock cycle inserted when writing to synchronous memories when CLKDIV=1	32
2.18	SDIO limitations . . . . .	32
2.18.1	Limited multibyte support with SDIO cards . . . . .	32
2.18.2	Data errors in SDIO hardware flow control mode . . . . .	33
2.19	USB limitations . . . . .	33
2.19.1	USB packet buffer memory: over/underrun or COUNTn_RX[9:0] field reporting incorrect number if APB1 frequency is below 13 MHz . . . . .	33
<b>Appendix A Revision code on device marking . . . . .</b>		<b>34</b>
<b>Revision history . . . . .</b>		<b>40</b>

## List of tables

Table 1.	Device Identification . . . . .	1
Table 2.	Device summary . . . . .	1
Table 3.	Cortex-M3 core limitations and impact on microcontroller behavior . . . . .	7
Table 4.	Summary of silicon limitations . . . . .	11
Table 5.	Document revision history . . . . .	40

## List of figures

Figure 1. LFBGA144 top package view ..... 34  
Figure 2. LFBGA100 top package view ..... 35  
Figure 3. WLCSP64 top package view ..... 36  
Figure 4. LQFP144 top package view ..... 37  
Figure 5. LQFP100 top package view ..... 38  
Figure 6. LQFP64 top package view ..... 39

# 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 high-density STM32F10xxx devices.

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

ARM ID	ARM category	ARM summary of errata	Impact on high-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 high-density devices

Only the limitations described below have an impact, even though minor, on the implementation of STM32F10xxx high-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 high-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 0xE00ED30 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. (● = limitation present in this rev.)

**Table 4. Summary of silicon limitations**

Links to silicon limitations		Rev Z	Rev Y and Rev 1
<a href="#">Section 2.1: Voltage glitch on ADC input 0</a>		●	●
<a href="#">Section 2.2: Flash memory read after WFI/WFE instruction</a>		●	●
<a href="#">Section 2.3: Debug registers cannot be read by user software</a>		●	●
<a href="#">Section 2.4: Debugging Stop mode and system tick timer</a>		●	●
<a href="#">Section 2.5: Debugging Stop mode with WFE entry</a>		●	●
<a href="#">Section 2.6: Alternate function</a>	<a href="#">Section 2.6.1: USART1_RTS and CAN_TX</a>	●	●
	<a href="#">Section 2.6.2: SPI1 in slave mode and USART2 in synchronous mode</a>	●	●
	<a href="#">Section 2.6.3: SPI1 in master mode and USART2 in synchronous mode</a>	●	●
	<a href="#">Section 2.6.4: SPI2 in slave mode and USART3 in synchronous mode</a>	●	●
	<a href="#">Section 2.6.5: SPI2 in master mode and USART3 in synchronous mode</a>	●	●
	<a href="#">Section 2.6.6: SDIO with TIM8</a>	●	●
	<a href="#">Section 2.6.7: SDIO and TIM3_REMAP</a>	●	●
	<a href="#">Section 2.6.8: SDIO with USART3 remapped and UART4</a>	●	●
	<a href="#">Section 2.6.9: FSMC with I2C1 and TIM4_CH2</a>	●	●
	<a href="#">Section 2.6.10: FSMC with USART2 remapped</a>	●	●
	<a href="#">Section 2.6.11: FSMC with USART3 and TIM1 remapped</a>	●	●
	<a href="#">Section 2.6.12: I2S2 in master/slave mode and USART3 in synchronous mode</a>	●	●
	<a href="#">Section 2.6.13: USARTx_TX pin usage</a>	●	●
<a href="#">Section 2.7: PVD and USB wakeup events</a>		●	●
<a href="#">Section 2.8: SPI3 in I2S slave mode: timing sensitivity between I2S3_WS and I2S3_CK</a>		●	●
<a href="#">Section 2.9: Boundary scan TAP: wrong pattern sent out after the "capture IR" state</a>		●	●
<a href="#">Section 2.10: Flash memory BSY bit delay versus STRT bit setting</a>		●	●

**Table 4. Summary of silicon limitations (continued)**

Links to silicon limitations		Rev Z	Rev Y and Rev 1
<i>Section 2.11: I2C peripheral</i>	<i>Section 2.11.1: Some software events must be managed before the current byte is being transferred</i>	•	•
	<i>Section 2.11.2: Wrong data read into data register</i>	•	•
	<i>Section 2.11.3: SMBus standard not fully supported</i>	•	•
	<i>Section 2.11.4: Wrong behavior of I2C peripheral in master mode after a misplaced Stop</i>	•	•
	<i>Section 2.11.5: Mismatch on the “Setup time for a repeated Start condition” timing parameter</i>	•	•
	<i>Section 2.11.6: Data valid time (tVD;DAT) violated without the OVR flag being set</i>	•	•
<i>Section 2.12: SPI peripheral</i>	<i>Section 2.12.1: CRC still sensitive to communication clock when SPI is in slave mode even with NSS high</i>	•	•
	<i>Section 2.12.2: SPI2/I2S2 slave mode wrong behavior in transmission and reception</i>	•	Fixed
<i>Section 2.13: I2S peripheral</i>	<i>Section 2.13.1: Wrong WS signal generation in 16-bit extended to 32-bit PCM long synchronisation mode</i>	•	•
	<i>Section 2.13.2: In I2S slave mode, WS level must be set by the external master when enabling the I2S</i>	•	•
	<i>Section 2.13.3: I2S slave mode desynchronisation with the master during communication</i>	•	•
<i>Section 2.14: USART peripheral</i>	<i>Section 2.14.1: Parity Error flag (PE) is set again after having been cleared by software</i>	•	•
	<i>Section 2.14.2: Idle frame is not detected if receiver clock speed is deviated</i>	•	•
	<i>Section 2.14.3: In full duplex mode, the Parity Error (PE) flag can be cleared by writing the data register</i>	•	•
	<i>Section 2.14.4: Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection</i>	•	•
	<i>Section 2.14.5: Break frame is transmitted regardless of nCTS input line status</i>	•	•
	<i>Section 2.14.6: nRTS signal abnormally driven low after a protocol violation</i>	•	•
<i>Section 2.15: Timers</i>	<i>Section 2.15.1: Missing capture flag</i>	•	•
	<i>Section 2.15.2: Overcapture detected too early</i>	•	•
	<i>Section 2.15.3: General-purpose timer: regulation for 100% PWM</i>	•	•
<i>Section 2.16: LSI clock stabilization time</i>		•	•

**Table 4. Summary of silicon limitations (continued)**

Links to silicon limitations		Rev Z	Rev Y and Rev 1
<i>Section 2.17: FSMC limitations</i>	<i>Section 2.17.1: Multimaster access on the FSMC memory map</i>	•	Fixed
	<i>Section 2.17.2: Dummy read cycles inserted when reading synchronous memories</i>	•	•
	<i>Section 2.17.3: 1 dummy clock cycle inserted when writing to synchronous memories when CLKDIV=1</i>	•	•
<i>Section 2.18: SDIO limitations</i>	<i>Section 2.18.1: Limited multibyte support with SDIO cards</i>	•	•
	<i>Section 2.18.2: Data errors in SDIO hardware flow control mode</i>	•	•
<i>Section 2.19: USB limitations</i>	<i>Section 2.19.1: USB packet buffer memory: over/underrun or COUNTn_RX[9:0] field reporting incorrect number if APB1 frequency is below 13 MHz</i>	•	•

## 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 as a digital input, it will not be detected as a spurious transition, providing that PA0 is driven with an impedance lower than 5 k $\Omega$ . 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, 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.

### 2.6.1 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 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.

**2.6.7 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.

## 2.6.8 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.

## 2.6.9 FSMC with I2C1 and TIM4\_CH2

### Description

When the FSMC is being used, the NADV signal is set to 1 by default when the alternate function output is selected for this pin. TIM4\_CH2 and the I2C1 SDA signal are in conflict with the NADV signal.

### Workaround

Do not use TIM4\_CH2 as an output when the FSMC is being used.

Concerning I2C1, it is possible to use the remap functionality available on the PB8 and PB9 pins. Otherwise, disable the FSMC clock through the "RCC\_AHBENR" register prior using the I2C1.

## 2.6.10 FSMC with USART2 remapped

### Description

When the FSMC is being used and the alternate function output is selected on PD4, PD5 and PD7, the following signals are in conflict:

- USART2\_RTS remapped to PD4 and FSMC\_OEN
- USART2\_TX remapped to PD5 and FSMC\_WEN
- USART2\_CK remapped to PD7 and FSMC\_NE1/FSMC\_NCE2

### Workaround

Use the USART2 default configuration (no remap).

### 2.6.11 FSMC with USART3 and TIM1 remapped

#### Description

When the FSMC is being used in 8-bit mode and the alternate function output is selected on the unused FSMC data lines 8 to 15, the following signals are in conflict:

- USART3\_TX and USART3\_CK remapped to PD8 and PD10, respectively, are in conflict with FSMC\_D13 and FSMC\_D15, respectively.
- TIM1\_CH2, TIM1\_CH3N, TIM1\_CH3 and TIM1\_CH4 used as outputs and remapped to PE11 to PE14, respectively, are in conflict with FSMC\_D8 to FSMC\_D11, respectively.

#### Workaround

- Use the USART3's partial remap functionality or default configuration (no remap).
- Do not use TIM1\_CH2, TIM1\_CH3N, TIM1\_CH3 and TIM1\_CH4 as outputs when they are remapped to PE11 to PE14, respectively.

### 2.6.12 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).

### 2.6.13 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 SPI3 in I<sup>2</sup>S slave mode: timing sensitivity between I2S3\_WS and I2S3\_CK

### Description

When SPI3 is configured in I<sup>2</sup>S slave audio modemode 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.

## 2.9 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 lower 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<sup>2</sup>C 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 drainEV6\_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<sup>2</sup>C 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 I<sup>2</sup>C 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<sup>2</sup>C 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  $T_{su;sta}$  in the I<sup>2</sup>C specification) can be slightly violated when the I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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 deselected 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 SPI2/I2S2 slave mode wrong behavior in transmission and reception

### Description

When the SPI2/I2S2 slave mode is used:

- In transmit or in full-duplex mode, the TXE flag may go high two times before RXNE is set, that is, TXE may be asserted before the data already in the data register are transmitted. This may result in data loss during transmission.
- In receive or in full-duplex mode, with CRC calculation enabled, the CRC may be detected as wrong (the CRCERR flag is set) while the received data are correct.
- In transmit or in full-duplex mode, with CRC calculation enabled, the CRC may not be output on the data line after the CRCNEXT bit is set in CPU mode, or when the DMA in transmission reaches the end of transfer.

### Workarounds

- The transmit or full-duplex mode (CRC feature not used) may be used if the master stops the clock for every data transfer. The SPI2/I2S2 slave then has enough time to load the data register on the RXNE flag going high. DMA must not be used.
- In receive only mode, with CRC calculation enabled, at the end of the CRC reception, the software needs to check the CRCERR flag.
  - If it is found set, read back the SPI\_RXCRC:
    - If the value is 0, the complete data transfer is successful.
    - Otherwise, one or more errors have been detected during the data transfer by CPU or DMA.
  - If it is found reset, the complete data transfer is successful.
- In full duplex or transmit only configurations, the CRC feature cannot be used.

## 2.13 I2S peripheral

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

#### Description

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

#### Workaround

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

### 2.13.2 In I2S slave mode, WS level must be set by the external master when enabling the I2S

#### Description

In slave mode the WS signal level is used only to start the communication. If the I2S (in slave mode) is enabled while the master is already sending the clock and the WS signal

level is low (for I2S protocol) or is high (for the LSB or MSB-justified mode), the slave starts communicating data immediately. In this case the master and slave will be desynchronized throughout the whole communication.

#### Workaround

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

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

### 2.13.3 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 Timers

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

### 2.15.1 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.

### 2.15.2 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.

### 2.15.3 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.

### 2.16 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  $\mu$ s should be added.

### 2.17 FSMC limitations

#### 2.17.1 Multimaster access on the FSMC memory map

##### Description

When multimaster accesses are performed on the FSMC memory map (address space from 0x6000 0000 to 0xA000 0FFF), an error could be generated, leading to either a bus fault or a DMA transfer error.

A multimaster can be:

- DMA1 and DMA2
- DMA1 and CPU
- DMA2 and CPU

##### Workaround

If multimaster accesses are required on the FSMC address space, the software must ensure that accesses are performed one at a time, and not at the same time.

## 2.17.2 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.

## 2.17.3 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-asserted 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.

## 2.18 SDIO limitations

### 2.18.1 Limited multibyte support with SDIO cards

#### Description

The SDIO standard allows multibyte transfers (to transfer any number of data bytes from 1, 2, 3 up to 512), which extends the block transfer of the SD standard.

In a multibyte transfer operation, CMD53 will transfer only a number of bytes that is equal to a power of 2 (1, 2, 4, 8...512) between 1 and 512.

#### Workaround

In case the application needs to read a number of bytes N that is not a power of 2, and if the SDIO device supports the multiblock mode, it is then possible to read the N bytes using the CMD53 multiblock command:

- Configure the SDIO block size to 1 byte
- Read N bytes using the CMD53 multiblock command



## 2.18.2 Data errors in SDIO hardware flow control mode

### Description

When HW Flow Control is enabled by setting bit 14 of SDIO\_CLKCR register, some glitches can occur on the SDIO\_CK output clock resulting in wrong data being written in the SD/MMC Card or the SDIO device. As consequence a data CRC error is reported to the SD/SDIO MMC host interface (DCRCFAIL bit in the SDIO\_STA register is set).

### Workaround

None. HW Flow Control must not be used. Software has to manage Overrun Errors (Rx mode) and FIFO underrun (Tx mode).

## 2.19 USB limitations

### 2.19.1 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 to Figure 6 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 1. LFBGA144 top package view

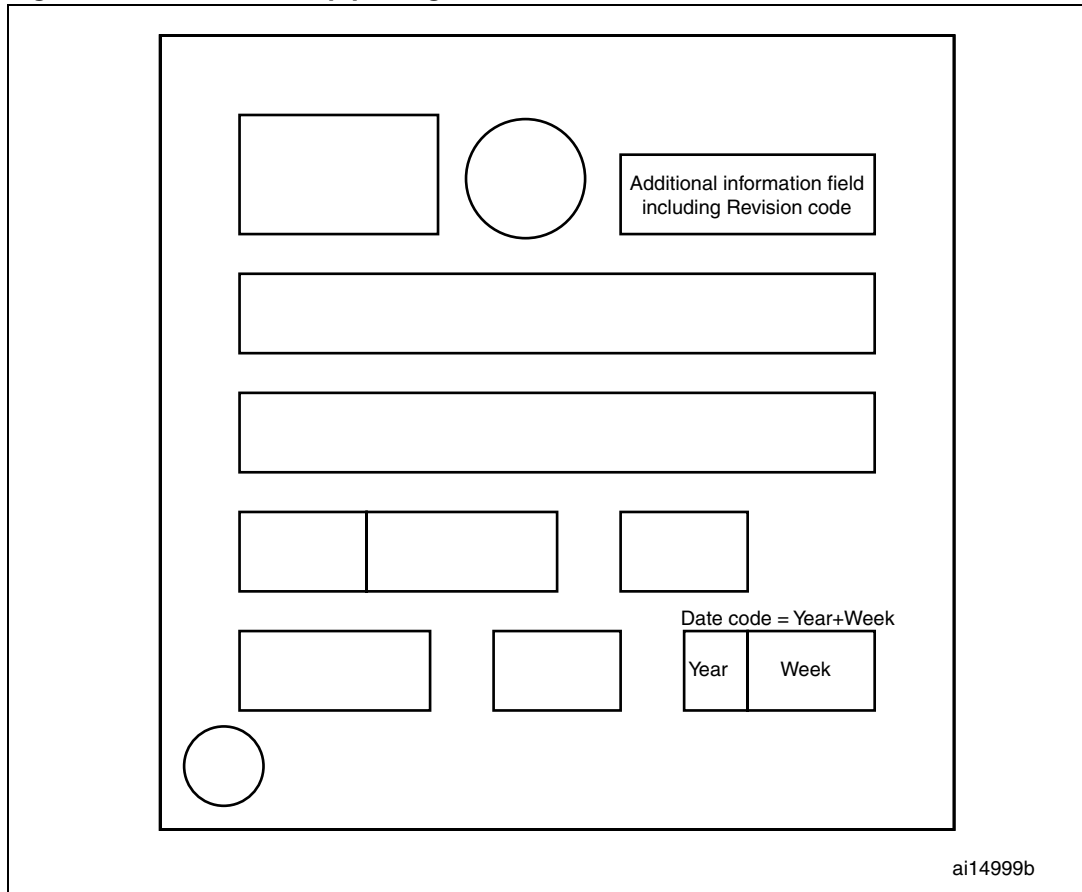


Figure 2. LFBGA100 top package view

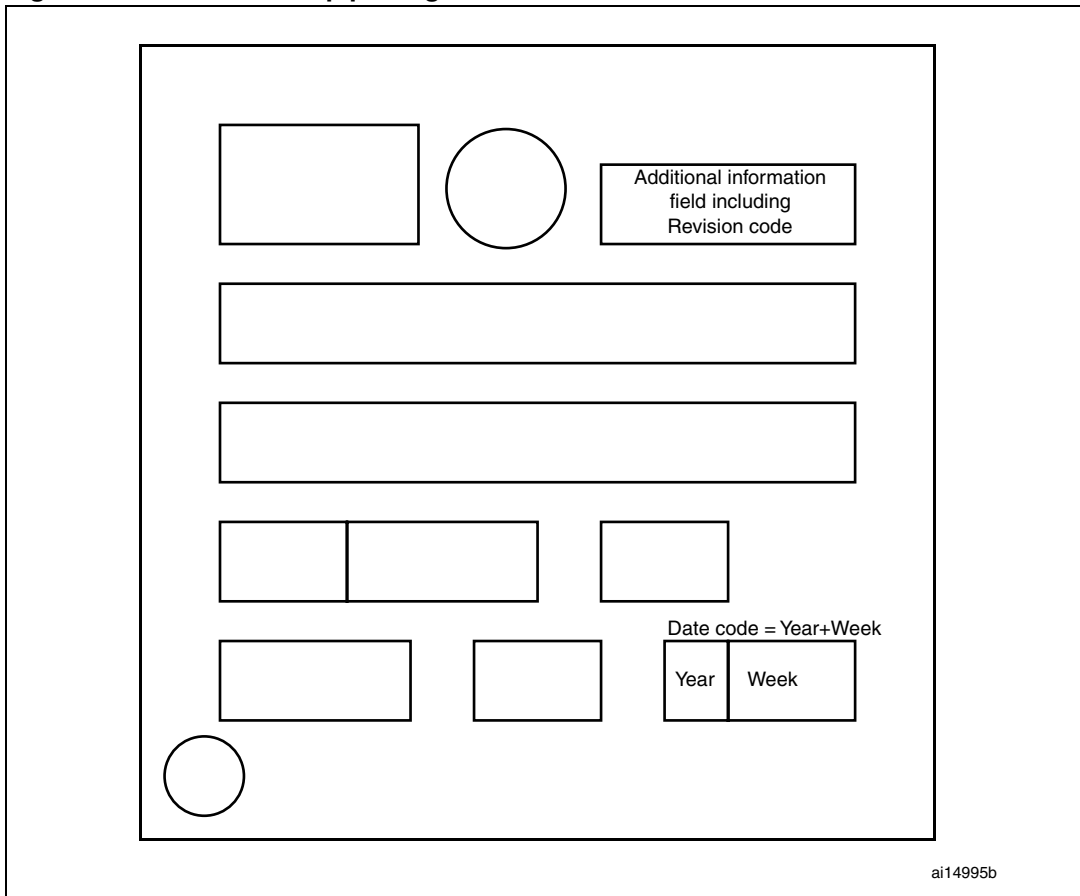


Figure 3. WLCSP64 top package view

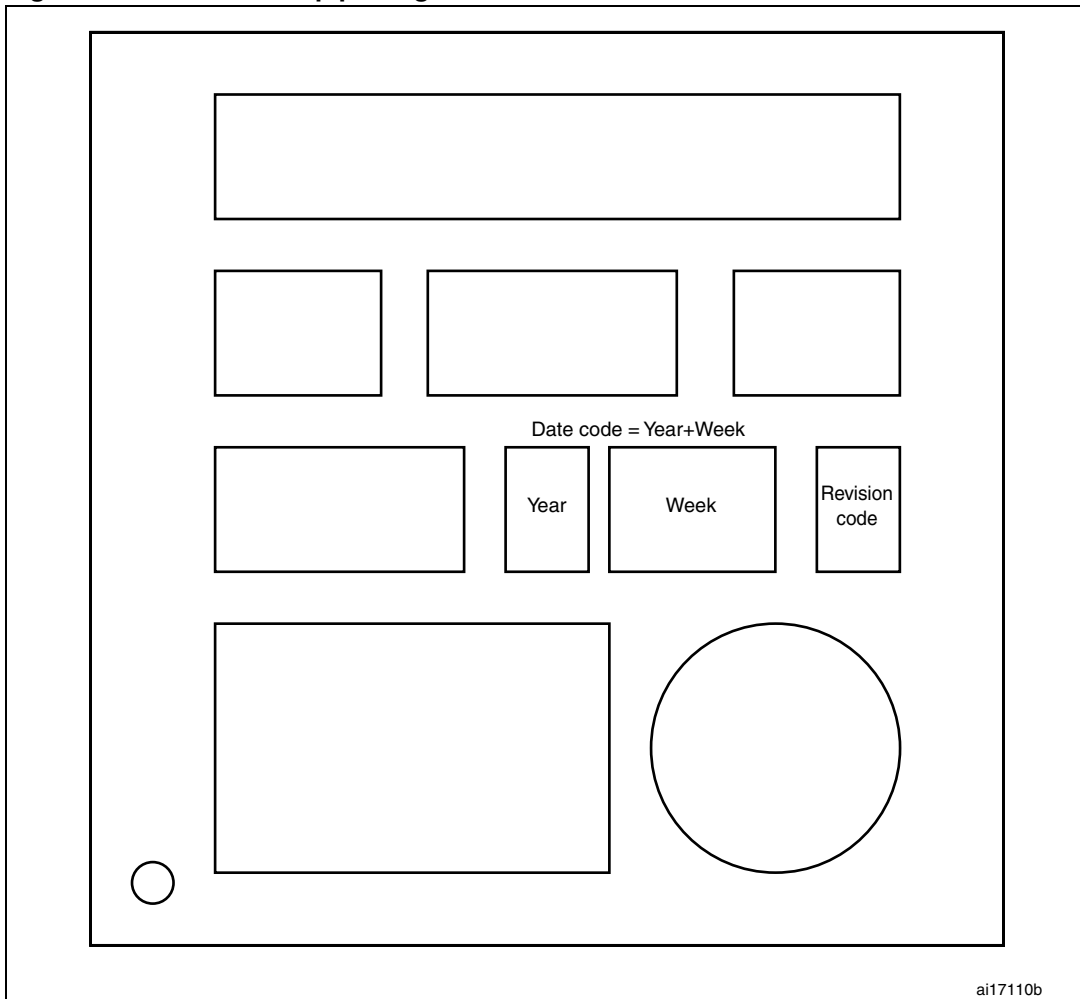


Figure 4. LQFP144 top package view

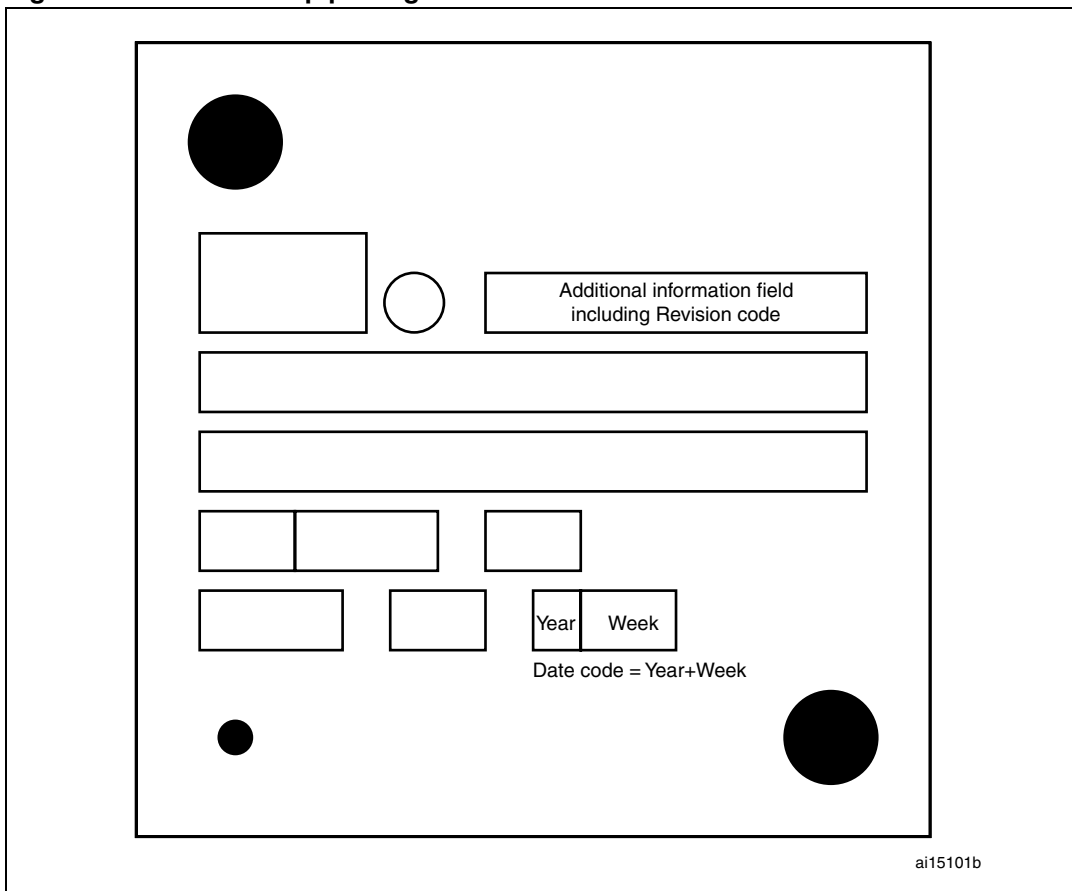
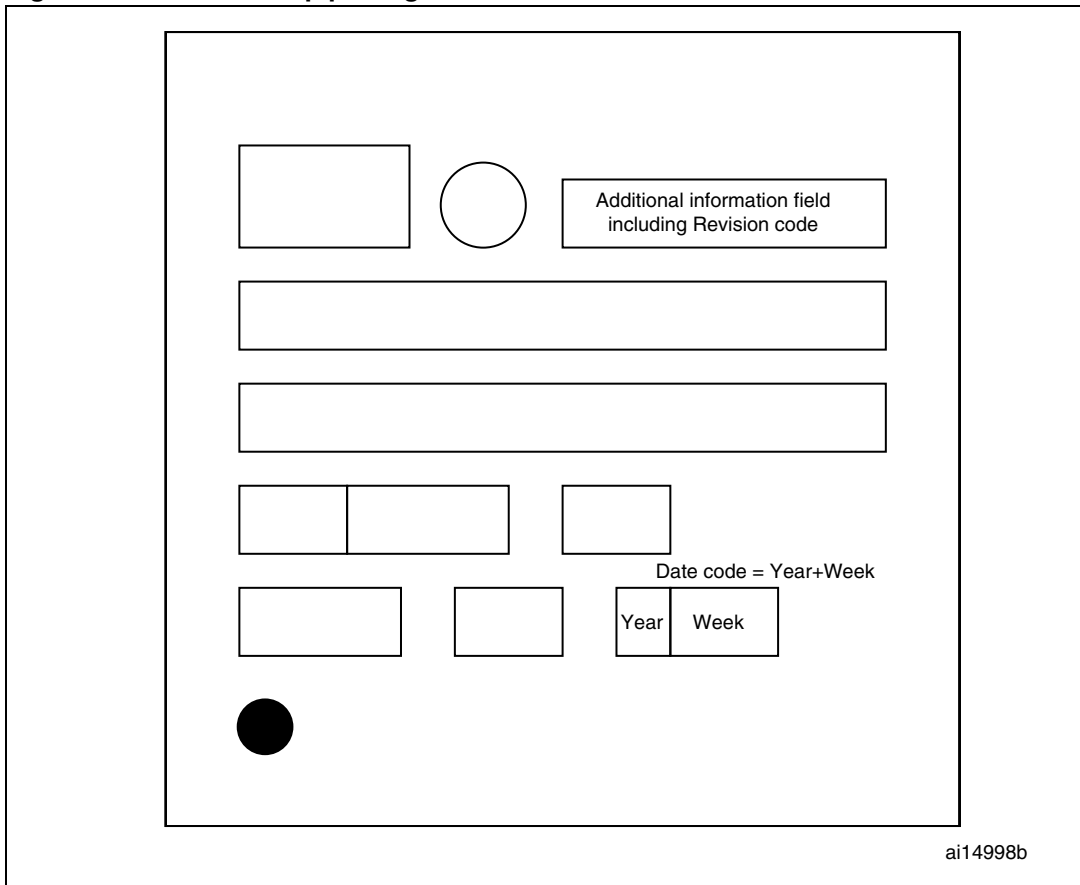
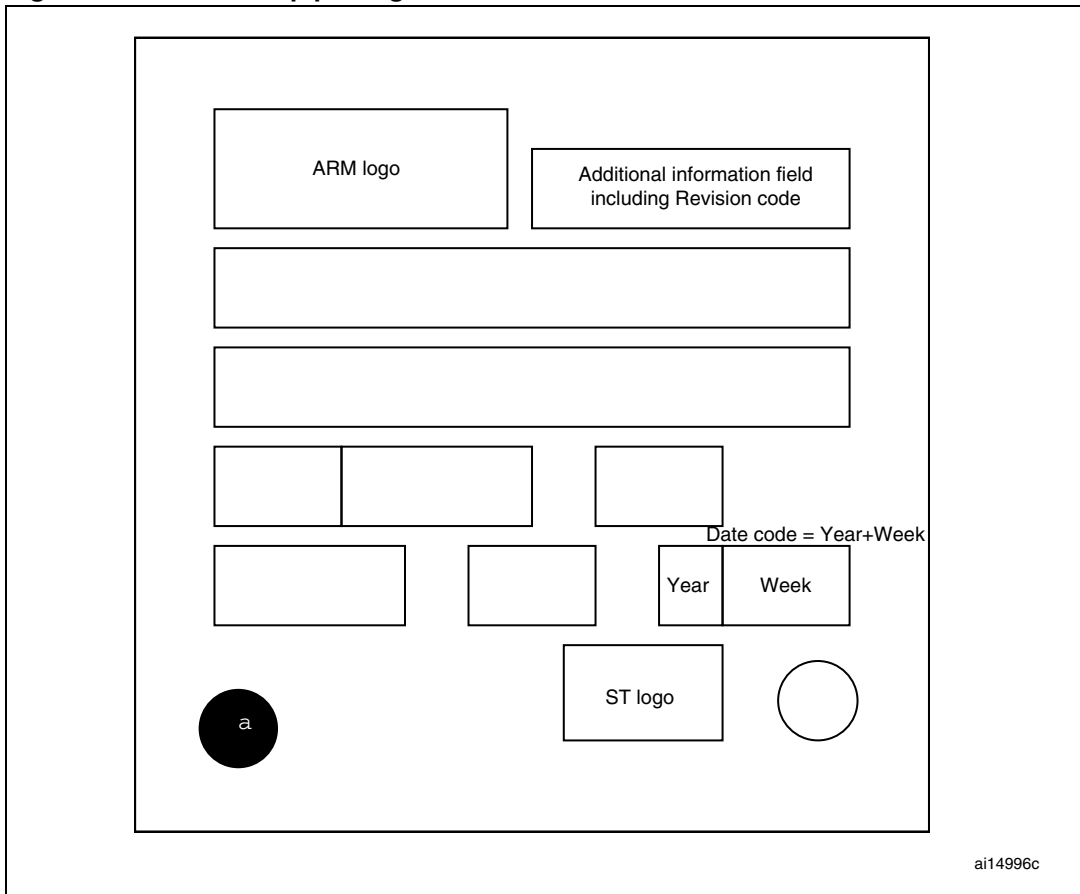


Figure 5. LQFP100 top package view



ai14998b

Figure 6. LQFP64 top package view



## Revision history

**Table 5. Document revision history**

Date	Revision	Changes
23-May-2008	1	Initial release.
21-Jul-2008	2	<a href="#">Section 2.7: PVD and USB wakeup events</a> added.
29-Sep-2008	3	<a href="#">Figure 6: LQFP64 top package view on page 39</a> corrected.
11-Feb-2009	4	<p><a href="#">Section 1: ARM™ 32-bit Cortex®-M3 limitations</a> specified (<a href="#">Table 3: Cortex-M3 core limitations and impact on microcontroller behavior</a> added limitations described).</p> <p>Limitation added: <a href="#">SPI3 in I2S slave mode: timing sensitivity between I2S3_WS and I2S3_CK on page 20</a>.</p> <p>Added limitations:</p> <ul style="list-style-type: none"> <li>– <a href="#">Boundary scan TAP: wrong pattern sent out after the “capture IR” state</a></li> <li>– <a href="#">Flash memory BSY bit delay versus STRT bit setting</a></li> <li>– <a href="#">I2C peripheral</a></li> <li>– <a href="#">Timers</a></li> <li>– <a href="#">LSI clock stabilization time</a></li> <li>– <a href="#">Section 2.17.1: Multimaster access on the FSMC memory map</a></li> <li>– <a href="#">Section 2.18.1: Limited multibyte support with SDIO cards</a></li> </ul> <p><a href="#">Table 4: Summary of silicon limitations on page 11</a> added.</p>
22-Jun-2009	5	<p>Introduction text updated in <a href="#">Section 2.6: Alternate function</a>.</p> <p><a href="#">Section 2.6.10: FSMC with USART2 remapped</a> updated.</p> <p><a href="#">Section 2.6.11: FSMC with USART3 and TIM1 remapped</a> added.</p> <p><a href="#">Section 2.18.1: Limited multibyte support with SDIO cards</a> clarified.</p> <p><a href="#">Section 2.19.1: USB packet buffer memory: over/underrun or COUNTn_RX[9:0] field reporting incorrect number if APB1 frequency is below 13 MHz</a> added.</p> <p><a href="#">Figure 3: WLCSP64 top package view</a> added.</p>
21-Jul-2009	6	<a href="#">Section 2.6.12: I2S2 in master/slave mode and USART3 in synchronous mode</a> added.



Table 5. Document revision history (continued)

Date	Revision	Changes
11-Jan-2010	7	<p>Workaround modified in <a href="#">Section 2.8: SPI3 in I2S slave mode: timing sensitivity between I2S3_WS and I2S3_CK</a>.</p> <p>Added limitations:</p> <ul style="list-style-type: none"> <li>– <a href="#">Section 2.6.13: USARTx_TX pin usage</a></li> <li>– <a href="#">Section 2.11.4: Wrong behavior of I2C peripheral in master mode after a misplaced Stop</a></li> <li>– <a href="#">Section 2.11.5: Mismatch on the “Setup time for a repeated Start condition” timing parameter</a></li> <li>– <a href="#">Section 2.11.6: Data valid time (tVD;DAT) violated without the OVR flag being set</a></li> <li>– <a href="#">Section 2.12.1: CRC still sensitive to communication clock when SPI is in slave mode even with NSS high</a></li> <li>– <a href="#">Section 2.19.1: USB packet buffer memory: over/underrun or COUNTn_RX[9:0] field reporting incorrect number if APB1 frequency is below 13 MHz</a></li> </ul> <p>Date code added to <a href="#">Figure 1</a> to <a href="#">Figure 6</a>.</p>
17-Jun-2010	8	<p>Updated for rev Y silicon</p> <p>Updated <a href="#">Table 4: Summary of silicon limitations</a> with fix status.</p> <p>Added <a href="#">Section 2.4: Debugging Stop mode and system tick timer</a></p> <p>Added <a href="#">Section 2.5: Debugging Stop mode with WFE entry</a></p> <p>Added <a href="#">Section 2.17: FSMC limitations</a></p> <p>Added <a href="#">Section 2.11.2: Wrong data read into data register</a></p> <p>Updated <a href="#">Section 2.11.4: Wrong behavior of I2C peripheral in master mode after a misplaced Stop</a></p> <p>Modified section <a href="#">Section 2.12.2: SPI2/I2S2 slave mode wrong behavior in transmission and reception</a></p> <p>Added <a href="#">Section 2.14: USART peripheral</a></p> <p>Updated <a href="#">Section 2.11.6: Data valid time (tVD;DAT) violated without the OVR flag being set</a></p>
21-Jan-2011	9	<p>Updated <a href="#">Section 2.6.9: FSMC with I2C1 and TIM4_CH2</a></p> <p>Added <a href="#">Section 2.13: I2S peripheral</a></p> <p>Added <a href="#">Section 2.14.6: nRTS signal abnormally driven low after a protocol violation</a></p>
07-Feb-2011	10	<p>Updated workarounds in <a href="#">Section 2.11.1: Some software events must be managed before the current byte is being transferred</a> and <a href="#">Section 2.11.2: Wrong data read into data register</a></p>
20-Jun-2011	11	<p>Added reference to revision code 1 in <a href="#">Table 1</a></p> <p>Added <a href="#">Section 1.1.5: Interrupted loads to SP can cause erroneous behavior</a></p> <p><a href="#">Section 1.1.6: SVC and BusFault/MemManage may occur out of order</a></p> <p><a href="#">Section 2.18.2: Data errors in SDIO hardware flow control mode</a></p>

**Please Read Carefully:**

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

**UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.**

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

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2011 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Philippines - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

[www.st.com](http://www.st.com)