STM32L162xD
Errata sheet

STM32L162xD
ultralow power MCU limitations

Silicon identification

This errata sheet applies to the revision A and Z of the STMicroelectronics STM32L162xD ultralow power products. This family features an ARM™ 32-bit Cortex®-M3 core, for which an errata notice is also available (see *Section 1* for details).

A full list of root part numbers is shown in *Table 1*.

The products can be identified (see *Table 2*) by:

- The revision code marked below the sales type on the device package
- The last three digits of the internal sales type printed on the box label

Table 1. Device summary

Reference	Part number
STM32L162xD	STM32L162VD, STM32L162ZD, STM32L162QD

Table 2. Device identification(1)

Sales type	Revision code(2) marked on device
STM32L162xD	“A” or “Z”

1. The REV_ID bits in the DBGMCU_IDCODE register show the revision code of the device (see the *STM32L151xx, STM32L152xx and STM32L162xx high-density advanced ARM-based 32-bit MCUs* reference manual (RM0038) for details on how to find the revision code).
2. Refer to *Appendix A: Revision and date codes on device marking* for details on how to identify the revision code on the different packages.

Contents

List of tables

List of figures

1 ARM™ 32-bit Cortex®-M3 limitations

An ARM errata notice of the STM32L162xD core is available from the following web address: *http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.eat0420c/index.html.*

The direct link to the errata notice pdf is:
http://infocenter.arm.com/help/topic/com.arm.doc.eat0420c/Cortex-M3-Errata-r2p0-v2.pdf

All the described limitations are minor and relate to revision r2p0-00rel0 of the Cortex-M3 core. *Table 3* summarizes these limitations and their implications on the behavior of the STM32L162xD ultra low power devices.

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

ARM ID	ARM category	ARM summary of errata	Impact on STM32L162xD ultralow power devices
602117	Cat 2	LDRD with base in list may result in incorrect base register when interrupted or faulted	Minor
563915	Cat 2	Event register is not set by interrupts and debug	Minor

1.1 Cortex-M3 limitation description for the STM32L162xD ultra low power devices

Only the limitations described below have an impact, even though minor, on the implementation of STM32L162xD ultralow power devices.

All 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 the STM32L162xD ultralow power devices (Cortex-M3 r2p0-00rel0).

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 STM32L162xD 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, the compilers 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 wake from event (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 a WFE if no other event or interrupt occurs.

Workarounds:

1. For the following interrupt sources:
   - all external interrupts/events lines (EXTI)
   - PVD output on EXTI line 16 (if VREFINT is enabled only)
   - RTC Alarm on EXTI line 17
   - USB Wake-up on EXTI line 18
   - RTC tamper and timestamp on EXTI line 19
   - RTC Wake-up on EXTI line 20
   - Comparator 1 wake-up on EXTI line 21 (if VREFINT is enabled only)
   - Comparator 2 wake-up on EXTI line 22 (if VREFINT is enabled only)

   use STM32L15x external events instead of interrupts to wake up the core from a WFE by configuring an external or internal EXTI line in event mode.
2. For all other interrupt sources, a timer must be programmed to provide a timeout event and wake-up the core if the event is likely to arrive before the WFE instruction is executed.

2 STM32L162xD silicon limitations

Table 4 gives a summary of the fix status.

Legend for *Table 4*: A = workaround available; N = no workaround available; P = partial workaround available, '-' and grayed = fixed.

Table 4. Summary of silicon limitations

Links to silicon limitations		Rev A	Rev Z
Section 2.1: System limitations	*Section 2.1.1: AOP_RANGE bit is mapped on register COMP_CSR(28) instead of register AOP_CSR(28)*	A	-
	Section 2.1.2: Missing analog switch on GPIO PC10	N	N
	Section 2.1.3: Ports PG0 to PG15 and ports PF0 to PF5 sink current when VIN>VDD	A	-
	Section 2.1.4: If Debugger is connected in JTAG mode and JNTRST (PB4) pin configuration is changed, the connection is lost	A	A
	Section 2.1.5: Read protection: a mass erase occurs if the RDP register is written with Level0 when Level0 is already set	A	A
	Section 2.1.6: In STOP mode, RAMs are not in power down if DMA is enabled	A	-
	Section 2.1.7: Invalid read from Data EEPROM after write in Program Flash	A	-
	Section 2.1.8: Debugging Stop mode with WFE entry	A	A
Section 2.2: LCD peripheral limitations	*Section 2.2.1: High drive resistive network total value too low*	N	-
Section 2.3: I2C peripheral limitations	*Section 2.3.1: SMBus standard not fully supported*	A	A
	Section 2.3.2: Wrong behavior of I2C peripheral in Master mode after misplaced STOP	A	A
	Section 2.3.3: Violation of I2C "setup time for repeated START condition" parameter	A	A
	Section 2.3.4: In I2C slave "NOSTRETCH" mode, underrun errors may not be detected and may generate bus errors	A	A
Section 2.4: SPI/I2S peripheral limitations	*Section 2.4.1: In I2S slave mode, WS level must to be set by the external master when enabling the I2S*	A	A

Table 4. Summary of silicon limitations (continued)

Links to silicon limitations		Rev A	Rev Z
Section 2.5: USART peripheral limitations	*Section 2.5.1: Idle frame is not detected if receiver clock speed is deviated*	N	N
	Section 2.5.2: In full duplex mode, the Parity Error (PE) flag can be cleared by writing the data register	A	A
	Section 2.5.3: Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection	N	N
	Section 2.5.4: Break frame is transmitted regardless of nCTS input line status	N	N
	Section 2.5.5: nRTS signal abnormally driven low after a protocol violation	A	A
Section 2.6: SDIO peripheral limitations	*Section 2.6.1: SDIO hardware flow control*	A	A
	Section 2.6.2: Wrong CCRCFAIL status after a response without CRC	A	A

2.1 System limitations

2.1.1 AOP_RANGE bit is mapped on register COMP_CSR(28) instead of register AOP_CSR(28)

Description

The operational amplifier voltage range is controlled by COMP_CSR(28) instead of AOP_CSR(28). The COMP_CSR(28) bit keeps its original function which selects GPIO PC3 as ADC input channel CH13 together with the control of operational amplifier voltage range.

Workaround

When the operational amplifiers are enabled, ADC channel CH13 must not be used.

This limitation is planned to be fixed in the next revision.

2.1.2 Missing analog switch on GPIO PC10

Description

An analog switch is not present on port PC10. As a consequence bit RI_ASMR3(10) has no effect, so PC10 cannot be used for the touch sensing feature.

Workaround

None.

2.1.3 Ports PG0 to PG15 and ports PF0 to PF5 sink current when $V_{IN}>V_{DD}$

Description

Ports PG0 to PG15 and ports PF0 to PF5 exhibit leakage current when a voltage above V_{DD} is applied on them.

Workaround

When current consumption is important, do not apply voltage above V_{DD} on port PG0 to PG15 and port PF0 to PF5.

This limitation is planned to be fixed in the next revision.

2.1.4 If Debugger is connected in JTAG mode and JNTRST (PB4) pin configuration is changed, the connection is lost

Description

PB4 is configured by default in Alternate function mode after Reset.

When the configuration bit changes from Alternate function to Input, Analog or GPIO, the Reset signal connected to the CPU is tied to '0', and forces a Reset on the CPU TAP that stops the Debugger connection, even if the pin itself is pulled up.

Only JTAG mode with 4 wires is impacted. Serial Wire Debug (SWD) mode is not impacted.

Workaround

During the debug phase, when the debugger is connected in JTAG mode is used, I/O port PB4 should not be used by the application. If the application needs to use PB4 even during debug phase, the Debugger should use Serial Wire Debug (SWD) mode to connect.

2.1.5 Read protection: a mass erase occurs if the RDP register is written with Level0 when Level0 is already set

Description

The read protection ranges from the lowest level 0 (no read protection) to level 2 (disable debug/chip read protection). It is always possible to increase the read protection level without any side effects. It is not possible to decrease the protection level from level 2 to a lower level. It is possible to decrease the protection level from level 1 to level 0, but to avoid allowing access to data that were previously protected, a Mass Erase of the data EEPROM, program Flash and backup registers is performed.

The limitation appears when level 0 is requested (writing 0xAA in the Option byte RDP) while the protection Level is already at 0, in this case an unwanted mass erase is performed.

Workaround

Before setting Level0 in the RDP register, check that the current level is not equal to Level0.

- If the current level is not equal to Level0, Level0 can be activated.
- If the current level is Level0 then the RDP register must not be written again with Level0.

2.1.6 In STOP mode, RAMs are not in power down if DMA is enabled

Description

In STOP mode, the RAM memories (Main and USB) are not in power-down when DMA1 and/or DMA2 are enabled in the RCC registers. If DMA1 and/or DMA2 are disabled, the RAMs are switched off. This induces an overconsumption of about 200 nA in STOP mode.

Workaround

Before entering STOP mode, if the DMA1EN and DMA2EN bits in the RCC_AHBENR register are enabled then they have to be reset. When the MCU wakes up from STOP mode, the DMA1EN and DMA2EN bits in the RCC_AHBENR register have to be enabled again (this can be done by the interrupt subroutine).

2.1.7 Invalid read from Data EEPROM after write in Program Flash

Description

When a read from data EEPROM in one bank is performed after a write in the program code, the value of the read might be wrong.

This problem can occur even if the write and the read access are separated by a delay of several CPU clock periods.

Workaround

In this case, perform two read operations from the same address location in data EEPROM, the first one should be discarded, the second one is correct and the value can be used.

2.1.8 Debugging Stop mode with WFE entry

Description

When the Stop debug mode is enabled (DBG_STOP bit set in the DBGMCU_CR register) this allows software debugging during Stop mode. However, if the application software uses the WFE instruction to enter Stop mode, after wakeup some instructions could be missed if the WFE is followed by sequential instructions. This affects only Stop debug mode with WFE entry.

Workaround

To debug Stop mode with WFE entry, the WFE instruction must be inside a dedicated function with 1 instruction (NOP) between the execution of the WFE and the Bx LR.

Example: __asm void _WFE(void) {

WFE

NOP

BX lr }

2.2 LCD peripheral limitations

2.2.1 High drive resistive network total value too low

Description

The value of the high drive resistive network is 60 kΩ instead of 240 kΩ, which leads to higher current consumption when used.

Workaround

None.

2.3 I^{2}C peripheral limitations

2.3.1 SMBus standard not fully supported

Description

The I^{2}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

The following higher-level mechanisms should be used to verify that a write operation is being performed correctly at the target device:

1. The SMBA pin if supported by the host
2. The alert response address (ARA) protocol
3. The host notify protocol

2.3.2 Wrong behavior of I^{2}C peripheral in Master mode after misplaced STOP

The I2C peripheral does not enter Master mode properly if a misplaced STOP is generated on the bus and the START bit is already set in I2C_CR2. In this case the START condition is not correctly generated on the bus and can create bus errors.

Workaround

In the I2C standard, it is not allowed to send a STOP before the full byte is transmitted (8 bits + acknowledge). Other derived protocols like CBUS allow it, but they are not supported by the I^{2}C peripheral.

In case of noisy environment in which unwanted bus errors can occur, it is recommended to reset the I2C peripheral by setting the SWRST bit in the I2C_CR2 control register if a BERR is detected while the START bit is set in I2C_CR2.

No fix is planned for this limitation.

2.3.3 Violation of I^{2}C “setup time for repeated START condition” parameter

Description

In case of a repeated Start, the “setup time for repeated START condition” parameter (named $t_{SU(STA)}$ in the datasheet and Tsu:sta in the I^{2}C specifications) may be slightly violated when the I^{2}C operates in Master Standard mode at a frequency ranging from 88 to 100 kHz. $t_{SU(STA)}$ minimum value may be 4 µs instead of 4.7 µs.

The issue occurs under the following conditions:

1. The I^{2}C peripheral operates in Master Standard mode at a frequency ranging from 88 to 100 kHz (no issue in Fast mode)
2. and the SCL rise time meets one of the following conditions:
   – The slave does not stretch the clock and the SCL rise time is more than 300 ns (the issue cannot occur when the SCL rise time is less than 300 ns).
   – or the slave stretches the clock.

Workaround

Reduce the frequency down to 88 kHz or use the I^{2}C Fast mode if it is supported by the slave.

2.3.4 In I^{2}C slave “NOSTRETCH” mode, underrun errors may not be detected and may generate bus errors

Description

The data valid time ($t_{VD;DAT}$, $t_{VD;ACK}$) described by the I^{2}C specifications may be violated as well as the maximum current data hold time ($t_{HD;DAT}$) under the conditions described below. In addition, if the data register is written too late and close to the SCL rising edge, an error may 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 occurs under the following conditions:

1. The I^2C peripheral operates In Slave transmit mode with clock stretching disabled (NOSTRETCH=1)
2. and the application is late to write 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 supports it, use the clock stretching mechanism by programming the bit NOSTRETCH=0 in the I2C_CR1 register.

If the master device does not support it, ensure that the write operation to the data register is performed just after TXE or ADDR events. You can use an interrupt on the TXE or ADDR flag and boost its priority to the higher level or use DMA.

Using the “NOSTRETCH” mode with a slow I^2C bus speed can prevent the application from being late to write the DR register (second condition).

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

If this is not possible, a possible workaround can be the following:

1. Clear the ADDR flag
2. Wait for the OVR flag to be set
3. Clear OVR and write the first data.

The time window for writing the next data is then the time to transfer one byte. In that case, the master must discard the first received data.

2.4 SPI/I2S peripheral limitations

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

Description

In slave mode the WS signal level is used only to start the communication. If the I2S (in slave mode) is enabled while the master is already sending the clock and the WS signal level is low (for I2S protocol) or is high (for the LSB or MSB-justified mode), the slave starts communicating data immediately. In this case the master and slave will be desynchronized throughout the whole communication.

Workaround

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

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

2.5 USART peripheral limitations

2.5.1 Idle frame is not detected if receiver clock speed is deviated

Description

If the USART receives an idle frame followed by a character, and the clock of the transmitter device is faster than the USART receiver clock, the USART receive signal falls too early when receiving the character start bit, with the result that the idle frame is not detected (IDLE flag is not set).

Workaround

None.

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

2.5.3 Parity Error (PE) flag is not set when receiving in Mute mode using address mark detection

Description

The USART receiver is in Mute mode and is configured to exit the Mute mode using the address mark detection. When the USART receiver recognizes a valid address with a parity error, it exits the Mute mode without setting the Parity Error flag.

Workaround

None.

2.5.4 Break frame is transmitted regardless of nCTS input line status

Description

When CTS hardware flow control is enabled (CTSE = 1) and the Send Break bit (SBK) is set, the transmitter sends a break frame at the end of current transmission regardless of nCTS input line status.

Consequently, if an external receiver device is not ready to accept a frame, the transmitted break frame is lost.

Workaround

None.

2.5.5 nRTS signal abnormally driven low after a protocol violation

Description

When RTS hardware flow control is enabled, the nRTS signal goes high when 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.

2.6 SDIO peripheral limitations

2.6.1 SDIO hardware flow control

Description

When enabling hardware flow control by setting bit 14 of the SDIO_CLKCR register to '1',

glitches can occur on the SDIOCLK output clock resulting in wrong data being written to the SD/MMC card or to the SDIO device. As a consequence, a CRC error is returned to the SD/SDIO MMC host interface (DCRCFAIL bit set to '1' in SDIO_STA register).

Workaround

There is no workaround. Do not use hardware flow control. Overrun errors (Rx mode) and FIFO underrun (Tx mode) should be managed by the application software.

2.6.2 Wrong CCRCFAIL status after a response without CRC

Description

When a command is followed by a response which does not contain a CRC field, the CRC is calculated anyway. Consequently, after the SDIO command IO_SEND_OP_COND (CMD5), the CCRCFAIL bit of the SDIO_STA register is set.

Workaround

In this case the CCRCFAIL bit in the SDIO_STA register must be ignored by software. CCRCFAIL must be cleared by setting the CCRCFAILC bit in the SDIO_ICR register after reception of the response to the CMD5 command.

Appendix A Revision and date codes on device marking

Figure 1, *Figure 2* and *Figure 3* show the marking compositions for the UFBGA132, LFP144 and LQFP100 packages, respectively. The only fields shown are the “additional” field, containing the revision code, and the “year” and “week” fields making up the date code.

Figure 1. UFBGA132 top package view

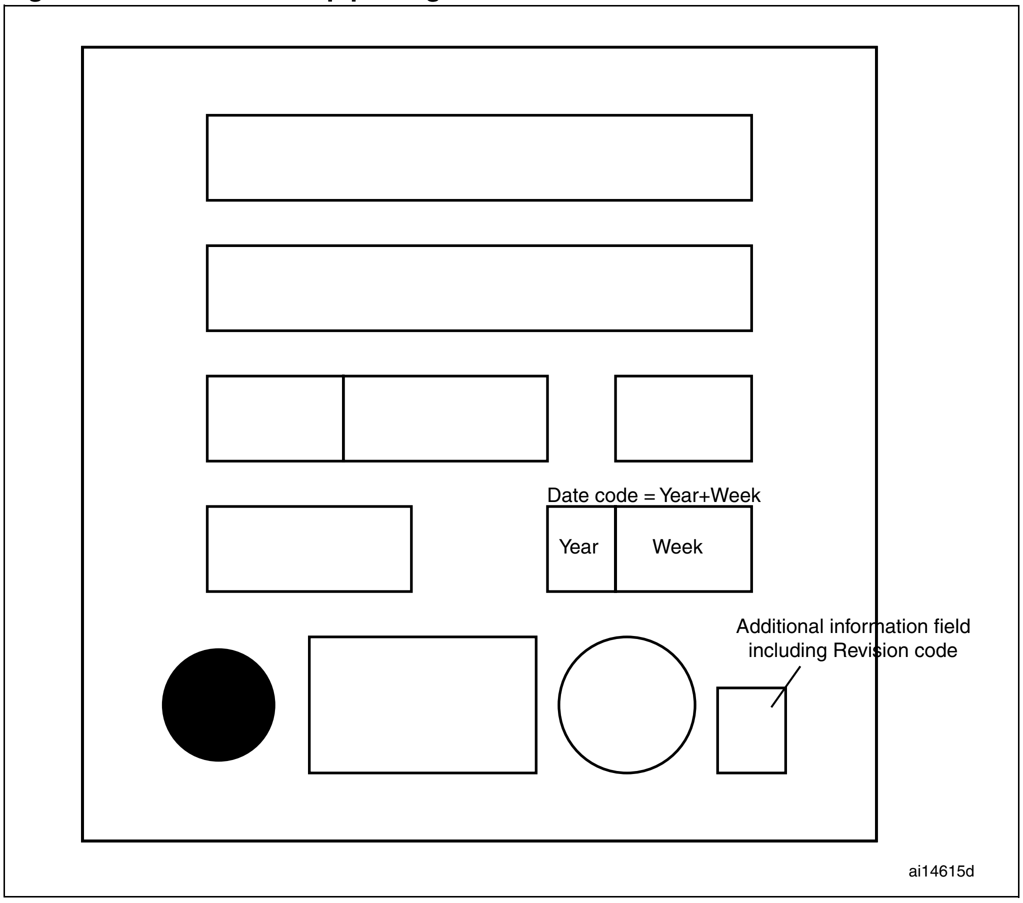


Figure 2. LQFP144 top package view

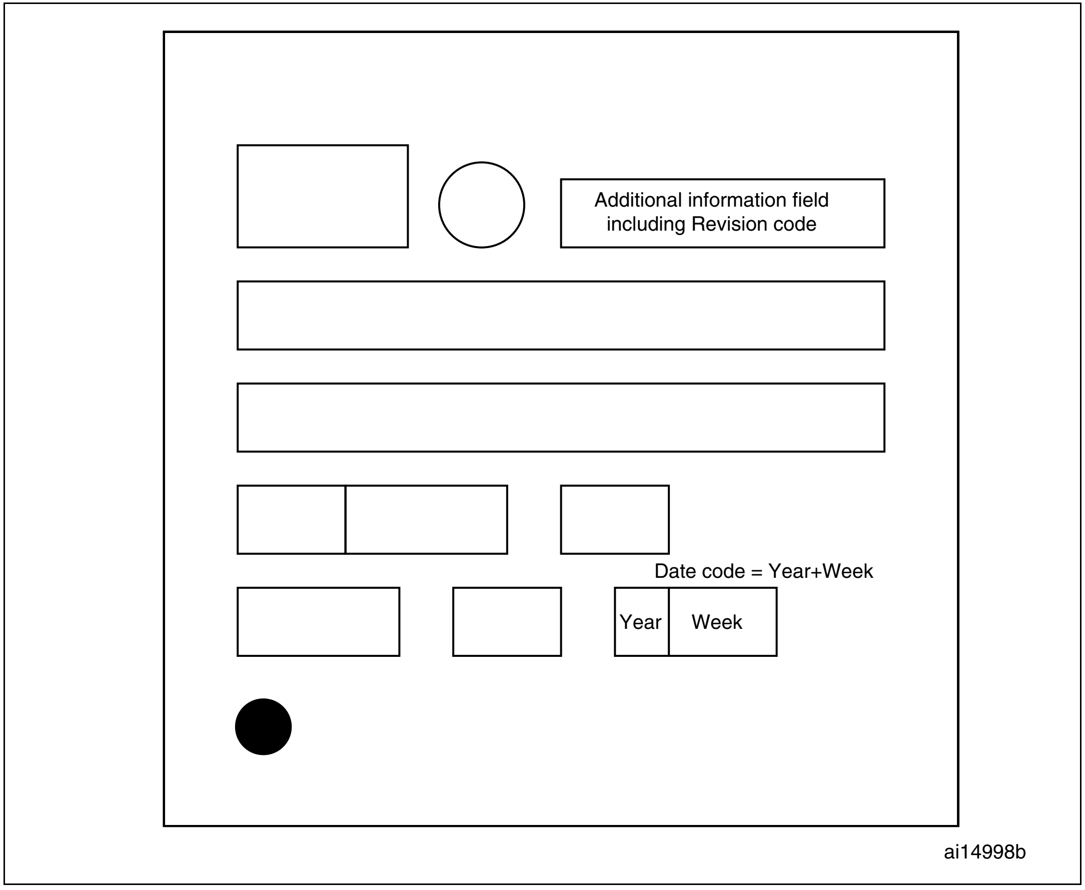
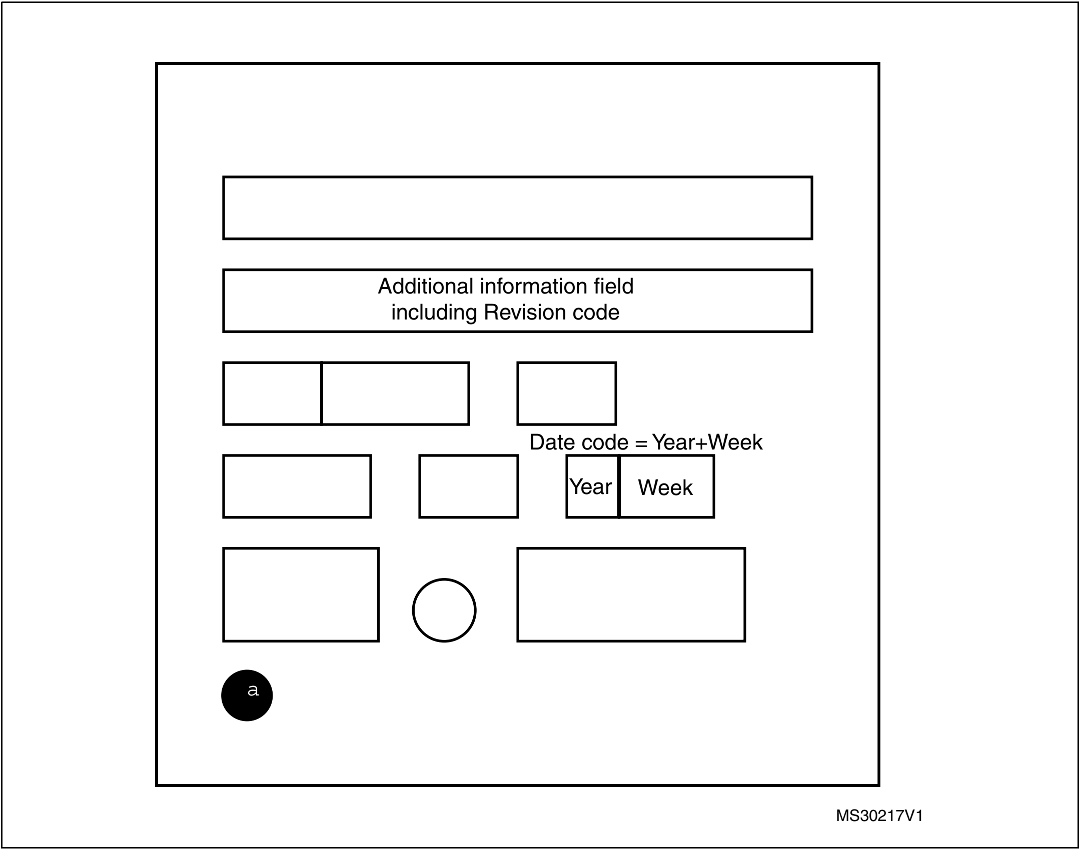


Figure 3. LQFP100 top package view



Revision history

Table 5. Document revision history

Date	Revision	Changes
17-Nov-2011	1	Initial release.
18-Jan-2012	2	Updated *Section Table 4.: Summary of silicon limitations* for silicon Rev Z. Added *Section 2.3.1: SMBus standard not fully supported* *Section 2.3.3: Violation of I2C “setup time for repeated START condition” parameter* *Section 2.3.4: In I2C slave “NOSTRETCH” mode, underrun errors may not be detected and may generate bus errors* *Section 2.6.2: Wrong CCRCFAIL status after a response without CRC*