
ERRATA NOTES

CC2500

Table Of Contents

1	RC OSCILLATOR TOLERANCE	2
2	RX FIFO	3
3	CYCLIC REDUNDANCY CHECK (CRC).....	4
4	PLL LOCK DETECTOR OUTPUT	6
5	SPI READ SYNCHRONIZATION ISSUE.....	7
6	WOR TIMING LIMITATION ON SHORT TIMING INTERVALS.....	10
7	RXFIFO_OVERFLOW ISSUE.....	11
8	REDUCED LINK PERFORMANCE AT CERTAIN FREQUENCIES.....	13
9	EXTRA BYTE TRANSMITTED IN TX	14
10	WOR TIMER ISSUE.....	15
11	DOCUMENT HISTORY	16
12	CONTACT INFORMATION	16

1 RC Oscillator Tolerance

1.1 Description and Reason for the Problem

The RC oscillator contains an error in the calibration routine that statistically occurs in 17.3% of all calibrations performed. When the calibration error occurs the RC oscillator tolerance is typically +8% (+10% expected worst case) to -0.3%. This means that the Wake-On-Radio (WOR) function will exhibit degraded performance in certain aspects or not be able to operate as intended. The following issue must be expected:

- Increased average current consumption, since the necessary time needed in receive mode will increase due to the increased tolerance.

1.2 Suggested Workaround

There is currently no workaround to this problem.

1.3 Batches Affected

This errata note applies to all batches and revisions of the chip.

2 RX FIFO

2.1 Description and Reason for the Problem

If a received data byte is written to the RX FIFO at the exact same time as the last byte in the RX FIFO is read over the SPI interface, the RX FIFO pointer is not properly updated and the last read byte is duplicated.

2.2 Suggested Workaround

For packets below 64 bytes, it is recommended to wait until the complete packet has been received before reading it out. If this is not possible or the packet is longer than 64 bytes, it is recommended to use the following workaround:

The number of bytes in the RX FIFO can be read from the status registers RXBYTES.NUM_RXBYTES. To avoid receiving data while reading the last byte in the RX FIFO one should never empty the RX FIFO before the last byte of the packet is received. Due to issue 5 in this errata note, special care must be taken when reading the RXBYTES register during reception.

1. Read RXBYTES.NUM_RXBYTES repeatedly at a rate guaranteed to be at least twice that of which RF bytes are received until the same value is returned twice; store value in *n*.
2. If $n < \#$ of bytes remaining in packet, read $n-1$ bytes from the RX FIFO.
3. Repeat 1-2 until $n = \#$ of bytes remaining in packet.
4. Read the remaining bytes from the RX FIFO.

Pseudocode:

```

BYTE n, l, len, *pDataBuf;

// Get length byte in packet (safely)
n = SPI_READ(RXBYTES);
do { l = n; n = SPI_READ(RX_BYTES); } while (n<2 && n!=1);
*pDataBuf++ = len = SPI_READ(RX_FIFO);

// Copy rest of packet (safely)
while (len>1) {
    n = SPI_READ(RXBYTES);
    do { l = n; n = SPI_READ(RX_BYTES); } while (n<2 && n!=1);
    while (n>1) {
        *pDataBuf++ = SPI_READ(RX_FIFO);
        len--; n--;
    }
}
*pDataBuf++ = SPI_READ(RX_FIFO);

```

2.3 Batches Affected

This errata note applies to all batches and revisions of the chip.

3 Cyclic Redundancy Check (CRC)

3.1 Description and Reason for the Problem

For certain conditions the CRC flag in PKTSTATUS, LQI and on the GDOx pins will always indicate “CRC not OK”, regardless of actual CRC status.

If PKTCTRL0.CC2400_EN=0 and the second byte of the CRC value appended to a TX packet is zero, the CRC_OK flag available in the PKTSTATUS[7] register, in the LQI[7] status register, and on one of the general purpose control (GDO) pins (if outputs 7 (0x07) or 15 (0x0F) are selected), will always indicate “CRC not OK”, i.e. the CRC_OK flag=0, in the device receiving the packet.

Note that a “CRC OK” indication can always be trusted. However, the “CRC not OK” indication from any of the three sources mentioned above can not be relied on when the second byte of the CRC value appended to a TX packet is zero. Then all packets will always result in a “CRC not OK” indication in the receiving device. Retransmission of identical packet(s) does not help, as the same situation will occur every time the same packet is received. For a packet that is affected, toggling any single bit in the packet (except for some bits in the last byte of the packet) before retransmission will cause CRC indication in RX to work properly.

The probability of the CRC bug affecting a packet with random data is 1/256 (not dependent on packet length). As mentioned above, the CRC bug is completely deterministic, only dependent on packet data. If there is a lot of variation in packet data from one packet to the next, system performance may not be badly affected, but it is difficult to guarantee that long sequences of packets all affected by the bug can not occur.

The bug does NOT affect the “CRC OK” indication, i.e. the CRC_OK flag, that can be appended to the packet in the RX FIFO (when PKTCTRL1.APPEND_STATUS=1).

3.2 Suggested Workaround

Checking the CRC flag by reading PKTSTATUS[7] register, LQI[7] status register, GDOx_CFG=0x07 and GDOx_CFG=0x15 should not be done if PKTCTRL0.CC2400_EN=0. If PKTCTRL0.CC2400_EN=0 a software workaround is needed in applications currently using one of these sources and relying on the respective CRC_OK flag. It is possible to check the CRC status in 2 different ways:

- 1) Set PKTCTRL1.APPEND_STATUS=1 and read the CRC_OK flag in the MSB of the second byte appended to the RX FIFO after the packet data. This requires double buffering of the packet, i.e. the entire packet content of the RX FIFO must be completely read out before it is possible to check whether the CRC indication is OK or not.

- 2) To avoid reading the whole RX FIFO, another solution is to use the PKTCTRL1.CRC_AUTOFLUSH feature. If this feature is enabled, the entire RX FIFO will be flushed if the CRC check fails. If GDOx_CFG=0x06 the GDOx pin will be asserted when a sync word is found. The GDOx pin will be de-asserted at the end of the packet. When the latter occurs the MCU should read the number of bytes in the RX FIFO. This can be read from the RXBYTES.NUM_RXBYTES status register or from the status byte returned on the MISO line when sending a byte on the SPI bus (FIFO_BYTES_AVAILABLE). If the number of bytes in RX FIFO is 0 the CRC check failed and the FIFO was flushed. If the number of bytes in RX FIFO > 0 the CRC check was OK and data can be read out of the FIFO.

The CRC_OK flag available in the PKTSTATUS[7] register, in the LQI[7] status register, and on one of the general purpose control (GDO) pins (if outputs 7 (0x07) or 15 (0x0F) are selected), can always be trusted in the device receiving the packet if PKTCTRL0.CC2400_EN=1. Note that neither whitening nor autoflush can be used with PKTCTRL0.CC2400_EN=1. More details can be found in the CC2500 data sheet.

3.3 Batches Affected

This errata note applies to all batches and revisions of the chip.

4 PLL Lock Detector Output

4.1 Description and Reason for the Problem

The PLL lock detector output is not 100% reliable and might toggle even if the PLL is in lock. The PLL is in lock if the lock detector output has a positive transition or is constantly logic high. The PLL is not in lock if the lock detector output is constantly logic low. It is not recommended to check for PLL lock by reading PKTSTATUS[0] with GDOx_CFG=0x0A or PKTSTATUS[2] register with GDOx_CFG=0x0A (x = 0 or 2).

4.2 Suggested Workaround

PLL lock can be checked reliably as follows:

1) Program register IOCFGx.GDOx_CFG=0x0A and use the lock detector output available on the GDOx pin as an interrupt for the MCU. A positive transition on the GDOx pin means that the PLL is in lock. It is important to disable for interrupt when waking the chip from SLEEP state as the wake-up might cause the GDOx pin to toggle when it is programmed to output the lock detector.

or

2) Read register FSCAL1. The PLL is in lock if the register content is different from 0x3F.

4.3 Batches Affected

This errata note applies to all batches and revisions of the chip.

5 SPI Read Synchronization Issue

A bug affecting the synchronization mechanism between the SPI clock domain (using a user supplied SCLK) and the internal 26 MHz clock domain (XCLK in this document) will sometimes result in incorrect read values for register fields that are continuously updated. The frequency with which this occurs is very low and guidelines for application design to avoid this issue are given in this chapter. The issue does **not** affect the data read from the RX FIFO as it uses a different and more robust synchronization mechanism. Neither does the issue affect writes to registers or the TX FIFO at any time.

5.1 Symptoms

When reading multi-bit register fields that are updated by the radio hardware such as the MARCSTATE or TXBYTES registers over the SPI interface, occasionally nonsensical or erroneous values will be read.

For example, in an application that sends packets longer than the 64 byte TX FIFO, the TX FIFO must be topped up with additional data during packet transmission. Assuming this is done by initially transferring 64 bytes to the TX FIFO, starting transmission, and then continuously polling TXBYTES to see when space for additional bytes is available, and then transferring the required number of bytes until the end of the packet. In this case the expected sequence of values read from TXBYTES would be:

64, 64, ..., 63, (write byte), 64, 64, ..., 63, (write byte), 64, ...

Due to the SPI synchronization issue the following might (infrequently) be seen instead:

64, 64, ..., 63, (write byte), 64, 64, ..., 64, **89**, 63, ...

The erroneous value read is highlighted in red. The register read is changing from the value 64 (01000000b) to the value 63 (00111111b) on the XCLK clock at the same time that its value is latched into the SPI output shift register on the SCLK clock. If the two clock edges occur sufficiently close in time, the improper synchronization mechanism will latch some bit values from the previous register value and some bits from the next register value, resulting in the erroneous value 89 (01011001b).

5.2 Description

During an SPI read transaction, the SPI output register latches the read value on the last falling edge of SCLK during an SPI address byte. For a burst read operation, subsequent register values are latched on the falling edge of SCLK in the last bit of each previous data byte.

Due to this synchronization issue, if the register being read changes value (synchronously with XCLK) during a certain period of time after this falling edge of SCLK then some of the bits in the read value will come from the **previous** value and some from the **next** value. This so-called window of uncertainty is about 1.3 ns for typical conditions and increases to about 2.0 ns for worst-case conditions (1.8 V VDD, 85 °C).

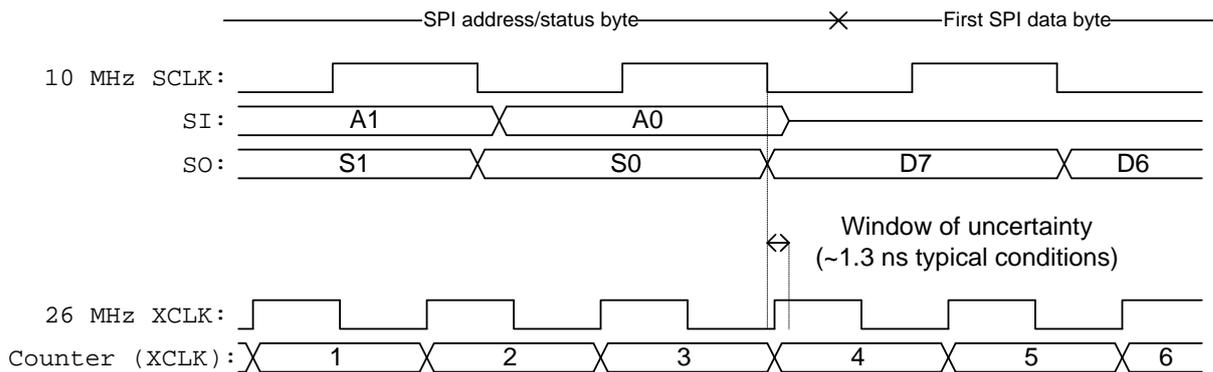


Figure 1 Window of uncertainty (drawing not to scale)

Figure 1 shows a timing diagram of an SPI read that fails when reading a fictitious counter being updated internally each **XCLK**. Since the counter update from value 3 (011b) to 4 (100b) within the window of uncertainty, the read value could be any one of 0-7 (000b, 001b, 010b, 011b, 100b, 101b, 110b, 111b) depending on exactly when the positive edge of **XCLK** falls within the window of uncertainty.

5.2.1 What Kinds of Register Fields Are Affected?

This issue does **not** affect:

- Reading of received data from the RXFIFO at any time.
- Reading of the static configuration registers (registers 0x00-0x2F)
- Reading static status registers (PARTNUM, VERSION) or status registers whose values should only be read after packet reception/transmission or FS calibration (FREEST, LQI, VCO_VC_DAC)
- Single-bit fields (all fields in PKTSTATUS, TXBYTES.TXFIFO_UNDERFLOW, RXBYTES.RXFIFO_OVERFLOW)
- Reading of any register whose value is known not to change at the time of the read operation (e.g. reading RXBYTES or RSSI after having received a packet)

This issue **does** affect:

- The SPI status byte (shifted out while the host MCU supplies the address byte) fields STATE and FIFO_BYTES_AVAILABLE.
- Reading FREEST or RSSI while the receiver is active.
- Reading MARCSTATE at any other time than when the device is inactive (IDLE).
- Reading RXBYTES when receiving a packet or TXBYTES while transmitting a packet.
- Reading WORTIME1 and WORTIME0 at any time.

5.2.2 How Often Does the Issue Corrupt Read Values?

The probability of reading a corrupt value is given by the frequency with which the read value changes, f_c , and the length of window of uncertainty, T_{WU} (typically 1.3 ns). The probability that the two events overlap, and thus that the read value is potentially corrupted, is given by:

$$P_{\text{corrupt}} = \frac{T_{\text{WU}}}{T_c} = T_{\text{WU}} f_c$$

In the example given in section 5.1, the probability of any single read from TXBYTES being corrupt, assuming the maximum datarate is used, is approximately $P_{\text{corrupt}} = T_{\text{WU}} f_c = 1.3 \text{ ns} \cdot (500 \text{ kbps} / 8\text{b}) \approx 80 \text{ ppm}$ or less than once every 10000 reads. In many situations the underlying received packet failure rate in the communication system is so much higher that any packet transmission/reception failure attributable to the issue described here will be negligible.

5.3 Suggested Workaround

In a typical radio system a packet error rate of at least 1 % should be tolerated in order to ensure robustness. In light of this, the negligible contribution to the number of packets lost due to, for example, occasionally reading incorrect FIFO byte count values or the wrong radio state from MARCSTATE, can probably be ignored in most applications. However, care should be taken to ensure that reading an incorrect value does not jeopardize an application. Examples of commonsense things to do include:

- For packets longer than the TX FIFO, configure the device to signal on a GDO pin when there is enough room to fill up with a new block of data (using the TXFIFO_THR threshold). If polling TXBYTES is necessary due to pin constraints, read TXBYTES repeatedly until the same value is returned twice in succession – such a value can always be trusted.
- Always perform a length check on the number of bytes reported in the RX FIFO to avoid a buffer overrun when copying the data to your MCU. A buffer overrun could make your firmware behave erratically or become deadlocked.
- Do not rely on the internal radio state machine through transient states (e.g. CALIBRATE – SETTling – TX – IDLE). It is, however, perfectly safe to poll for the end of transmission by waiting for MARCSTATE=IDLE.
- Always average RSSI and LQI values over several packets before using them in decision algorithms (e.g. for FH channel selection).
- Avoid using the SPI status byte STATE and FIFO_BYTES_AVAILABLE fields during packet transmission.

If it is important to **ensure** that read values are not corrupted, reading of one of the affected registers should be done repeatedly until the same value is read twice in succession. If the rate at which the register is read is guaranteed to be at least twice as fast as the expected register update rate, then an upper bound on the number of required reads is four and the average number of reads slightly more than two.

The same method can be used to ensure that the SPI status byte fields that provide simplified radio FSM state and saturated FIFO byte count are correct. This only makes sense when polling the status byte with SNOP as the address.

5.4 Batches Affected

This errata note applies to all chip batches and revisions of the chip.

6 WOR Timing Limitation on Short Timing Intervals

6.1 Description and Reason for the Problem

The Wake on Radio (WOR) timer is a very low power timer. It uses a $f_{\text{xosc}}/750$ kHz (34.7 kHz with $f_{\text{xosc}} = 26$ MHz) clock source for the timer and compare logic. In power down mode this clock source is divided by 128 to achieve a 270.8 Hz clock frequency for the timer, given that f_{xosc} is 26 MHz.

The WOR timer runs on 270.8 Hz in power down to save power. Shortly before reaching the programmed timeout value the timer automatically resumes 34.7 kHz operation. This is achieved by pre-incrementing the actual timer value before entering power down mode, to allow match logic to resume 34.7 kHz timer operation.

For timeouts less than approximately 11 ms (detailed timing is shown in application note AN047), the timeout period is too short to switch to the 270.8 Hz clocking scheme. Due to a design error the timer is pre-incremented even if the CC2500 does not switch to the 270.8 Hz clocking scheme, effectively shortening the timeout by one 270.8 Hz clock period (~3.7 ms).

6.2 Suggested Workaround

WOR usage is described in application note AN047.

6.3 Batches Affected

This errata note applies to all batches and revisions of the chip.

7 RXFIFO_OVERFLOW Issue

7.1 Description and Reason for the Problem

In addition to having a 64 bytes long RX FIFO, the CC2500 has a one byte long pre-fetch buffer between the FIFO and the SPI module. It also has buffers for status registers, CRC bytes, and buffers used when FEC is enabled. If more than 65 bytes has been received (the FIFO and the pre-fetch buffer is full) without reading the RX FIFO, the radio will enter RXFIFO_OVERFLOW state. There are however some cases where the radio will be stuck in RX state instead of entering RXFIFO_OVERFLOW state, as it should. Below is a table showing the register settings that will cause this problem. APPEND_STATUS is found in the PKTCTRL1 register, CRC_EN is found in the PKTCTRL0 register, and FEC_EN is in the MDMCFG1 register. IOCFGx=0x06, which means that the pin should be de-asserted when the RXFIFO overflows. In the cases where the radio is stuck in RX state, the GDOx pin will not be de-asserted.

When the radio is stuck in RX state like this, it will draw current as in RX state, but it will not be able to receive any more data. The only way to get out of this state is to issue an SIDLE strobe and then flush the FIFO (SFRX).

	# of bytes to be put in RX FIFO	MARCSSTATE	RXBYTES		GDOx
			RXFIFO_OVERFLOW	NUM_RXBYTES	
APPEND_STATUS = 1	64	IDLE	0	64	OK
CRC_EN = 0	65	IDLE	0	65	OK
FEC_EN = 1	66	RX	0	65	-
	67	RX	0	65	-
	68	RX	0	65	-
	69	RX	0	65	-
	70	RX	0	65	-
	71	RXFIFO_OVERFLOW	1	65	OK
APPEND_STATUS = 1	64	IDLE	0	64	OK
CRC_EN = 1	65	IDLE	0	65	OK
FEC_EN = 1	66	RX	0	65	-
	67	RX	0	65	-
	68	RX	0	65	-
	69	RX	0	65	-
	70	RXFIFO_OVERFLOW	1	65	OK
APPEND_STATUS = 0	64	IDLE	0	64	OK
CRC_EN = 0	65	IDLE	0	65	OK
FEC_EN = 1	66	RX	0	65	-
	67	RX	0	65	-
	68	RX	0	65	-
	69	RXFIFO_OVERFLOW	1	65	OK
APPEND_STATUS = 0	64	IDLE	0	64	OK

CRC_EN = 1	65	IDLE	0	65	OK
FEC_EN = 1	66	RX	0	65	-
	67	RX	0	65	-
	68	RXFIFO_OVERFLOW	1	65	OK
APPEND_STATUS = 1	64	IDLE	0	64	OK
CRC_EN = 1	65	IDLE	0	65	OK
FEC_EN = 0	66	RX	0	65	-
	67	RX	0	65	-
	68	RXFIFO_OVERFLOW	1	65	OK
APPEND_STATUS = 0	64	IDLE	0	64	OK
CRC_EN = 1	65	IDLE	0	65	OK
FEC_EN = 0	66	RXFIFO_OVERFLOW	1	65	OK
APPEND_STATUS = 1	64	IDLE	0	64	OK
CRC_EN = 0	65	IDLE	0	65	OK
FEC_EN = 0	66	RXFIFO_OVERFLOW	1	65	OK
APPEND_STATUS = 0	64	IDLE	0	64	OK
CRC_EN = 0	65	IDLE	0	65	OK
FEC_EN = 0	66	RXFIFO_OVERFLOW	1	65	OK

Note: (-) GDOx pin is not de-asserted when the RX FIFO overflows

7.2 Suggested Workaround

In applications where the packets are short enough to fit inside the RX FIFO and it is desirable to wait for the entire packet to be received before starting to read the RX FIFO, the PKTLEN register should be set to 61 for variable packet length mode (PKTCTRL0.LENGTH_CONFIG = 1) to make sure the whole packet including status bytes are 64 bytes or less (length byte (61) + 61 payload bytes + 2 status bytes = 64 bytes) or $PKTLEN \leq 62$ if fixed packet length mode is used (PKTCTRL0.LENGTH_CONFIG = 0). In applications where the packets do not fit inside the RX FIFO, reading from the RX FIFO must start before it reaches its limit (64 bytes).

7.3 Batches Affected

This errata note applies to all batches and revisions of the chip.

8 Reduced Link Performance at certain Frequencies

8.1 Description and Reason for the Problem

There will be spurious signals at $n/2$ -crystal oscillator frequency (n is an integer number). As a result the CC2500 exhibits increased packet error rate in receive mode and thereby reduced link performance for data rates below 250 kBaud at RF frequencies which are equal to $n/2$ -crystal oscillator frequency (e.g. 2405, 2418, 2431, 2444, 2457, 2470 and 2483 MHz when using a 26 MHz crystal).

8.2 Suggested Workaround

A workaround to this problem is to avoid using 200 kHz on each side of the frequencies listed above. Using e.g. every half MHz (2405.5, 2418.5, ...) instead of integer MHz (2405, 2418, ...) frequencies also gives better co-existence in an environment with e.g. Bluetooth products.

If the use of the exact frequencies listed above is required, another solution is to use another crystal frequency. With e.g. 27 MHz, the problem frequencies will appear at 13.5 MHz intervals instead of 13 MHz.

8.3 Batches Affected

This errata note applies to all batches and revisions of the chip.

9 Extra Byte Transmitted in TX

9.1 Description and Reason for the Problem

If one abort a transmission (exits TX mode) during the transmission of the first half of any byte, there will be a repetition of the first byte of the next transmission. This issues is caused by a state machine controlling the `mod_rd_data` signal in the modulator. This signal asserts at the start of transmission of each full byte, then deasserts after half the byte has been transmitted. If transmission is aborted after a byte has started but before half the byte is transmitted this signal remains asserted and the first byte in the next transmission will be repeated.

9.2 Suggested Workaround

As long as the packet handling features of the CC2500 are used, this will not be a problem since the chip will always exit TX mode after the transmission of the last bit in the last byte of the packet. If, however, one disables the packet handling features (`MDMCFG2.SYNC_MODE=0`) and wants to exit TX mode manually by strobing IDLE, one should make sure that the IDLE strobe is being issued after clocking out 12 dummy bits (8 dummy bits are necessary due to the TX latency, but since this would mean that transmission is aborted within the first half of a byte, 4 extra bits are added).

9.3 Batches Affected

This errata note applies to all batches and revisions of the chip.

10 WOR Timer Issue

10.1 Description and Reason for the Problem

The WOR timer uses two clock sources, one 32 kHz and one 256 Hz. If an SWOR strobe is executed at the exact same time as an edge of the 256 Hz clock, there is a small probability that the timer enters a lock state where the WOR timer runs 128 times too slow, i.e. the sleep period, t_{Event0} , is 128 times longer than expected. To recover from this state the WOR timer needs to be reset by an SRES strobe command.

10.2 Suggested Workaround

To get around this issue it is necessary to avoid entering SLEEP mode (after issuing a SWOR strobe) on a 256 Hz edge. This can be done by outputting CLK_256 on a GDO pin by setting IOCFGx = 0x26 and then wait for an edge on this signal before issuing the SWOR strobe. One must make sure that the duration of the SPI access when transmitting the SWOR strobe is less than half a 256 Hz period to avoid the next edge.

Please note that when running on a 26 MHz crystal the RCOSC is running on $26\text{ MHz}/750 = 34.666\text{ kHz}$ and not 32 kHz and the 256 Hz clock is actually 270.83 Hz. This means that the strobe command must be executed in less than 1.84 ms.

10.3 Batches Affected

This errata note applies to all batches and revisions of the chip.

11 Document History

Revision	Date	Description/Changes
SWRZ002E	2013-10-07	Added Section 10 "WOR Timer Issue"
SWRZ002D	2007-11-29	Section 9.2: "bytes" changed to "bits"
SWRZ002C	2007-05-09	Added Chapter 9 with info related to the transmission of an extra byte in TX. Chapters 4 and 8 are updated with more detailed workaround information. WORTIME in Chapter 5 changed to WORTIME1 and WORTIME0. Added info about bug related to RXFIFO_OVERFLOW in Chapter 7. Document revision history in SWRZ002B regarding Chapter 1 and Chapter 3.2 was wrong: Chapter 1: The correct should be "The RC oscillator tolerance in Chapter 1 has been changed from "-8% (-10% expected worst case) to +0.3%" to "+8% (+10% expected worst case) to -0.3%". Chapter 3.2: The second suggested workaround was not removed so there should not have been a reference to this chapter. The RX FIFO autoflush function works when PKTCTRL0.CC2400_EN = 0.
SWRZ002B	2007-02-12	Added Chapter 6-8. The RC oscillator tolerance in Chapter 1 has been changed from "-8% (-10% expected worst case) to +3%" to "+8% (+10% expected worst case) to -3%". The second suggested workaround in Chapter 3.2 has been removed since RX FIFO autoflush function does not work when PKTCTRL0.CC2400_EN = 0.
1.1	2006-05-23	Chapter 2 with RX FIFO is updated and Chapter 5 is new
1.0	2006-01-26	First edition

12 Contact Information

TEXAS INSTRUMENTS

Web sites: <http://www.ti.com/lpw>

Worldwide Product Information Centers: Contact TI PIC Centers for Technical Support

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products

Audio	www.ti.com/audio
Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DLP® Products	www.dlp.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
OMAP Applications Processors	www.ti.com/omap
Wireless Connectivity	www.ti.com/wirelessconnectivity

Applications

Automotive and Transportation	www.ti.com/automotive
Communications and Telecom	www.ti.com/communications
Computers and Peripherals	www.ti.com/computers
Consumer Electronics	www.ti.com/consumer-apps
Energy and Lighting	www.ti.com/energy
Industrial	www.ti.com/industrial
Medical	www.ti.com/medical
Security	www.ti.com/security
Space, Avionics and Defense	www.ti.com/space-avionics-defense
Video and Imaging	www.ti.com/video

TI E2E Community

e2e.ti.com