ERRATA NOTES

CC2500

Table Of Contents

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

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 errate note, special care must be taken when reading the RXBYTES register during reception. It is recommended to wait until the complete packet has been received before reading it out or, if this is not possible, to use the following software fix:

1. Read RXBYTES.NUM_RXBYTES repeatedly at a rate guaranteed to b e 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!=l);
*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!=l);
    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.

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).

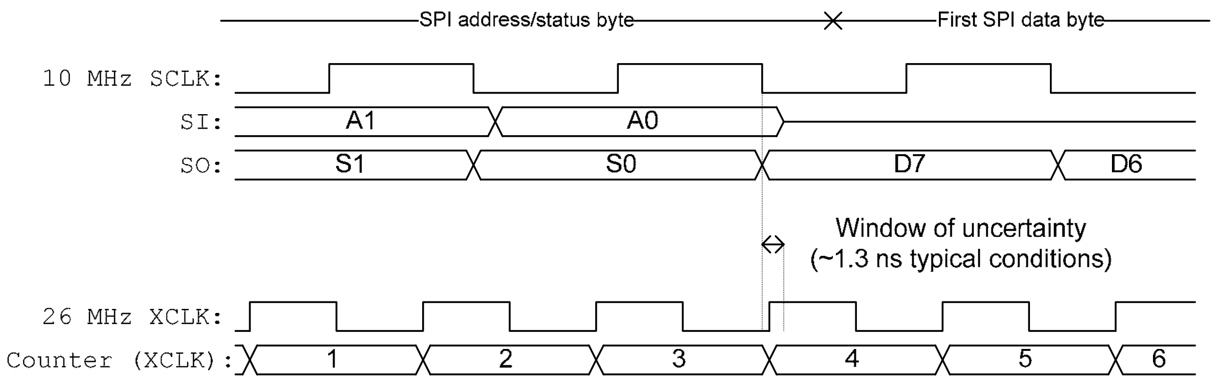

Figure 1 Window of uncertainty (drawing not to scale)

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 (FREQEST, 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 FREQEST 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 WORTIME 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:

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

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 $P_{corrupt} = T_{WU} f_c = 1.3\,\text{ns} \cdot (500\,\text{kbps} / 8\text{b}) \approx 80\,\text{ppm}$ 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 an f_{xosc}/750 kHz (34.7 kHz with f_{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 bug the timer is pre-incremented even if the device 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 RX 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 have been received (the RX FIFO and the pre-fetch buffer is full) without reading the RX FIFO, the radio will enter RX_OVERFLOW state. There are however some cases where the radio will indicate RX state instead of 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 RX FIFO overflows. In the cases where the radio is permanent in RX state, the GDOx pin will not be de-asserted.

	# of bytes to be put in RX FIFO	MARCSTATE	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 ≤ 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). In receive mode the CC2500 exhibits increased packet error rate and thereby reduced link performance 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

There is no workaround for this problem and these frequencies should be avoided.

8.3 Batches Affected

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

9 Document History

Revision	Date	Description/Changes
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

10 Contact Information

TEXAS INSTRUMENTS

Web sites:	http://www.ti.com/lpw
Worldwide Product Information Centers:	Contact TI PIC Centers for Technical Support