ERRATA NOTES
CC113L

Table Of Contents

1 RX FIFO

1.1 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.

1.2 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 register `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 3 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 specified 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!=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);
```

1.3 Batches Affected

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

2 PLL Lock Detector Output

2.1 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.GDO0` with `IOCFGx.GDO0_CFG=0x0A` or `PKTSTATUS.GDO2` register with `GDO2_CFG=0x0A`.

2.2 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 (x = 0 or 2).

or

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

With both of the above workarounds the CC113L PLL calibration should be carried out with the correct settings for `TEST0.VCO_SEL_CAL_EN` and `FSCAL2.VCO_CORE_H_EN`. These settings are depending on the operating frequency, and is calculated automatically by SmartRF™ Studio.

It must be noted that the `TEST0` register content is not retained in SLEEP state, and thus it is necessary to write to this register as described above when returning from the SLEEP state.

2.3 Batches Affected

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

3 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.

3.1 Symptoms

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

For example, in an application that receives packets longer than the 64 byte RX FIFO, the RX FIFO must be read during packet reception. Assume this is done by reading the `RXBYTES` register at a given interval checking if there are data in the RX FIFO to be read. If there are for example 31 (00011111b) bytes in the RX FIFO, the reception of another byte will cause the `RXBYTES` register to change to 32 (00100000b). If the `RXBYTES` register is changing from 31 to 32 on the XCLK clock at the same time that its value is latched into the SPI output shift register on the SCLK clock, the improper synchronization mechanism might latch some bit values from the previous register value and some bits from the next register value if the two clock edges occur sufficiently close in time. This might result in the erroneous value 60 (00111100b) or any other combination of the two value.

3.2 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).

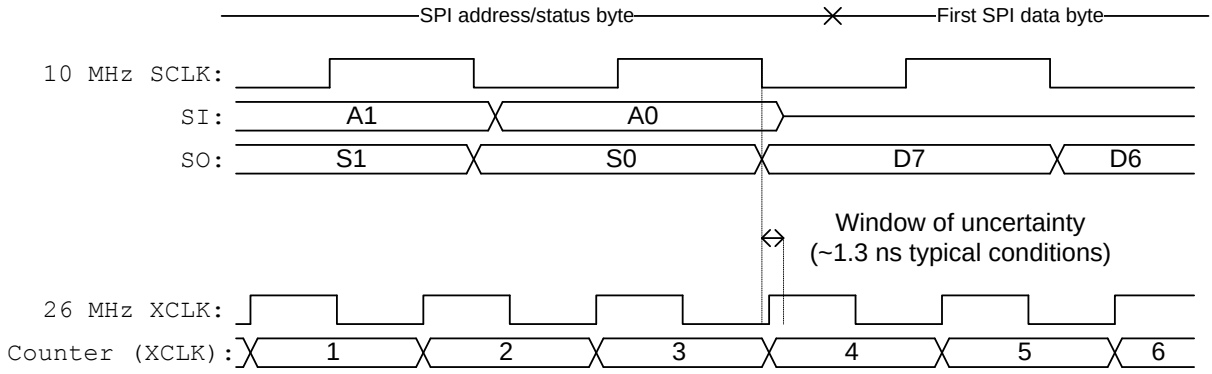

Figure 1: Window of Uncertainty (drawing not to scale)

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.

3.2.1 What Kinds of Register Fields Are Affected?

This issue does **not** affect:

- Reading of received data from the RX FIFO at any time.
- Reading of the static configuration registers (registers 0x00 - 0x2E)
- Reading static status registers (PARTNUM, VERSION) or status registers whose values should only be read after packet reception or FS calibration
- Single-bit fields (all fields in PKTSTATUS, 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 while receiving a packet

3.2.2 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 3.1, the probability of any single read from RXBYTES being corrupt, assuming the maximum data rate is used, is approximately $P_{corrupt} = T_{WU} \cdot f_c = 1.3\ ns \cdot (500\ kbps/8b) \approx 80$ 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 reception failure attributable to the issue described here will be negligible.

3.3 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 RX FIFO, configure the device to signal on a GDO pin when there is a certain number of bytes in the FIFO to be read (using the RX FIFO threshold). If polling RXBYTES is necessary due to pin constraints, read RXBYTES repeatedly until the same value is returned twice in succession - such a value can always be trusted.
- Do not rely on the internal radio state machine through transient states (e.g. CALIBRATE - SETTLING - RX - IDLE). It is, however, perfectly safe to poll for the end of transmission by waiting for MARCSTATE = IDLE.
- Always average RSSI 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 specified 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.

3.4 Batches Affected

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

4 RXFIFO_OVERFLOW Issue

4.1 Description and Reason for the Problem

In addition to having a 64 bytes long RX FIFO, the CC113L has a one byte long pre-fetch buffer between the FIFO and the SPI module. It also has buffers for status registers and CRC bytes. 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 and `CRC_EN` is found in the `PKTCTRL0` 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	MARCSTATE	RXBYTES		GDOx
			RXFIFO_OVERFLOW	NUM_RXBYTES	
APPEND_STATUS = 1 CRC_EN = 1	64	IDLE	0	64	OK
	65	IDLE	0	65	OK
	66	RX	0	65	-
	67	RX	0	65	-
	68	RXFIFO_OVERFLOW	1	65	OK
APPEND_STATUS = 0 CRC_EN = 1	64	IDLE	0	64	OK
	65	IDLE	0	65	OK
	66	RXFIFO_OVERFLOW	1	65	OK
APPEND_STATUS = 1 CRC_EN = 0	64	IDLE	0	64	OK
	65	IDLE	0	65	OK
	66	RXFIFO_OVERFLOW	1	65	OK
APPEND_STATUS = 0 CRC_EN = 0	64	IDLE	0	64	OK
	65	IDLE	0	65	OK
	66	RXFIFO_OVERFLOW	1	65	OK

4.2 Suggested Workaround

In applications where the packets are short enough to fit in the RX FIFO the following can be done:

a) Variable packet length mode (`PKTCTRL0.LENGTH_CONFIG=1`): the `PKTLEN` register should be set to 61 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)
b) Fixed packet length mode (`PKTCTRL0.LENGTH_CONFIG=0`): `PKTLEN` ≤ 62

In application where the packets do not fit in the RX FIFO, one must start reading the RX FIFO before it reaches its limit (64 bytes).

4.3 Batches Affected

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

5 General Information

5.1 Document History

Revision	Date	Description/Changes
SWRZ038	05.24.2011	Initial Release

Table 1: Document History

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Audio	www.ti.com/audio	Communications and Telecom	www.ti.com/communications
Amplifiers	amplifier.ti.com	Computers and Peripherals	www.ti.com/computers
Data Converters	dataconverter.ti.com	Consumer Electronics	www.ti.com/consumer-apps
DLP® Products	www.dlp.com	Energy and Lighting	www.ti.com/energy
DSP	dsp.ti.com	Industrial	www.ti.com/industrial
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Security	www.ti.com/security
Logic	logic.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Power Mgmt	power.ti.com	Transportation and Automotive	www.ti.com/automotive
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com	Wireless	www.ti.com/wireless-apps
RF/IF and ZigBee® Solutions	www.ti.com/lprf		
	TI E2E Community Home Page		e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265