CC1101 Silicon Errata

Table 1. Advisory List

RX FIFO

RX FIFO pointer is not properly updated and the last read byte is duplicated

Revision(s) Affected

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

Details

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.

Workaround(s)

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 the SPI read synchronization issue described later 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 the 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++ = len = SPI_READ(RX_FIFO);
                len--; n--;
        }
}
*pDataBuf++ = SPI_READ(RX_FIFO);
```

PLL Lock Detector Output *PLL lock detector output is not 100% reliable*

Revision(s) Affected

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

Details

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).

Workaround(s)

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.
2. Read register FSCAL1. The PLL is in lock if the register content is different from 0x3F. With both of the above workarounds, the CC1101 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.

SPI Read Synchronization Issue *SPI read synchronization results in incorrect read values for register fields that are continuously updated*

Revision(s) Affected

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

Details

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.

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).

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).

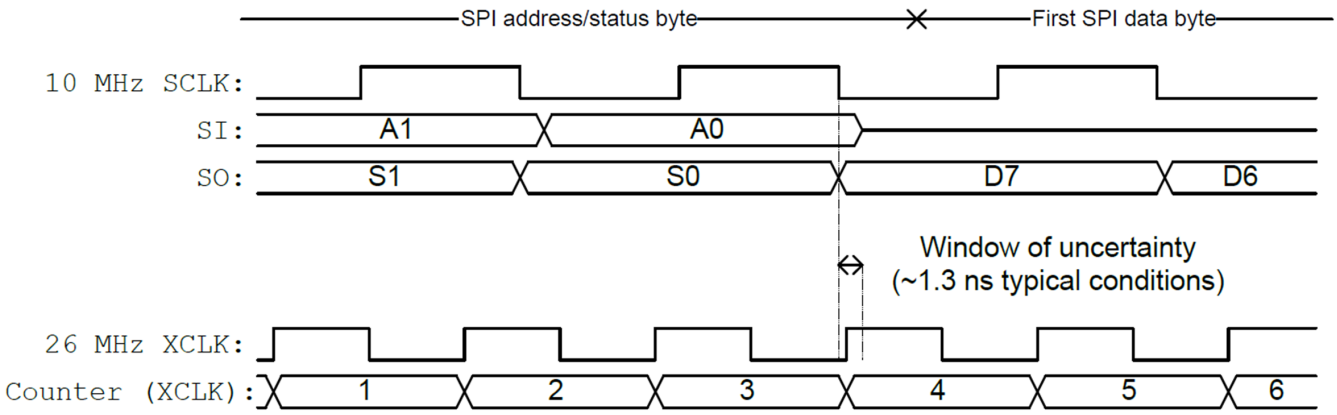


Figure 1. Window of Uncertainty (drawing not to scale)

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.

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/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 (that is, 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 WORTIME1 and WORTIME0 at any time.

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, fc, and the length of window of uncertainty, TWU (typically 1.3 ns). The probability that the two events overlap, and thus that the read value is potentially corrupted, is given by:

$$P_{corrupt} = \frac{T_{WU}}{T_c} = T_{WU} f_c \quad (1)$$

In the example given in the symptoms above, the probability of any single read from TXBYTES being corrupt, assuming the maximum data rate is used, is approximately $P_{corrupt} = T_{WU} f_c$ = 1.3 ns · 500 kbps / 8b ≈ 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 transmission/reception failure attributable to the issue described here will be negligible.

Workaround(s)

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 TX FIFO 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 (that is, 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 (that is, 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.

WOR Timing Error *WOR timing error on short timing intervals*

Revision(s) Affected

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

Details

The Wake on Radio timer is a very low power timer. It uses a 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 timer runs on 270.8 Hz in power down to save power. Some time 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 (SWRA126)), 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 device does not switch to the 270.8 Hz clocking scheme, effectively shortening the timeout by one 270.8 Hz clock period.

Workaround(s)

WOR usage is described in application note AN047 (SWRA126).

RXFIFO_OVERFLOW Issue *Radio stays in RX state instead of entering RXFIFO_OVERFLOW state*

Revision(s) Affected

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

Details

In addition to having a 64 bytes long RX FIFO, the CC1101 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).

	Number of bytes to be put in RX FIFO	MARCSTATE	RXBYTES		GDOx
			RXFIFO_OVERFLOW	NUM_RXBYTES	
APPEND_STATUS = 1 CRC_EN = 0 FEC_EN = 1	64	IDLE	0	64	OK
	65	IDLE	0	65	OK
	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 CRC_EN = 1 FEC_EN = 1	64	IDLE	0	64	OK
	65	IDLE	0	65	OK
	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 CRC_EN = 0 FEC_EN = 1	64	IDLE	0	64	OK
	65	IDLE	0	65	OK
	66	RX	0	65	-
	67	RX	0	65	-
	68	RX	0	65	-
	69	RXFIFO_OVERFIEW	1	65	OK
APPEND_STATUS = 0 CRC_EN = 1 FEC_EN = 1	64	IDLE	0	64	OK
	65	IDLE	0	65	OK
	66	RX	0	65	-
	67	RX	0	65	-
	68	RXFIFO_OVERFIEW	1	65	OK
APPEND_STATUS = 1 CRC_EN = 1 FEC_EN = 0	64	IDLE	0	64	OK
	65	IDLE	0	65	OK
	66	RX	0	65	-
	67	RX	0	65	-
	68	RXFIFO_OVERFIEW	1	65	OK

	Number of bytes to be put in RX FIFO	MARCSTATE	RXBYTES		GDOx
			RXFIFO_OVERFLOW	NUM_RXBYTES	
APPEND_STATUS = 0 CRC_EN = 1 FEC_EN = 0	64	IDLE	0	64	OK
	65	IDLE	0	65	OK
	66	RXFIFO_OVERFIEW	1	65	OK
APPEND_STATUS = 1 CRC_EN = 0 FEC_EN = 0	64	IDLE	0	64	OK
	65	IDLE	0	65	OK
	66	RXFIFO_OVERFIEW	1	65	OK
APPEND_STATUS = 0 CRC_EN = 0 FEC_EN = 0	64	IDLE	0	64	OK
	65	IDLE	0	65	OK
	66	RXFIFO_OVERFIEW	1	65	OK

Workaround(s)

In applications where the packets are short enough to fit in the RX FIFO and one wants to wait for the whole packet to be received before starting to read the RX FIFO, for 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) or PKTLEN ≤ 62 if fixed packet length mode is used (PKTCTRL0.LENGTH_CONFIG=0). 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).

Extra Byte Transmitted in TX *Repetition of the first byte in the next transmission*

Revision(s) Affected This errata note applies to all batches and revisions of the chip.

Details If one aborts a transmission (exits TX mode) during the transmission of the first half of any byte, there will be a repetition of the first byte in 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 de-asserts 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.

Workaround(s) As long as the packet handling features of the CC1101 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).

WOR Timer Issue when RCOSC Calibration is Disabled *WOR timer runs too slow*

Revision(s) Affected

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

Details

The WOR timer uses two clock sources, one 32 kHz and one 256 Hz. If WORCTRL.RC_CAL = 0 and 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, for example, 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.

Workaround(s)

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. It must be made 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 than when running on a 26 MHz crystal the RCOSC is running on 26 MHz/750 = 34.666 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.

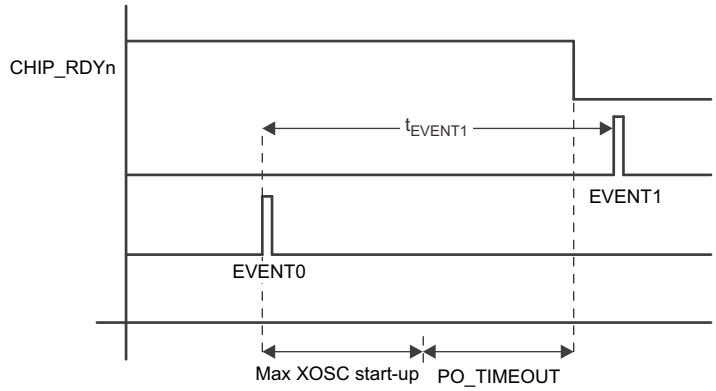
WOR Issue when t_{Event1} is too Short *Radio not entering RX in WOR mode*

Revision(s) Affected This errata note applies to all batches and revisions of the chip.

Details If EVENT1 occurs after the XOSC has stabilized but before CHP_RDYn goes low (during the PO_TIMEOUT period), there is a small probability that the radio enters an illegal state, causing the radio to miss EVENT1. If this happens, the internal SRX strobe never takes place and the chip does not enter RX mode. On the next EVENT1, the crystal is running and CHIP_RDYn is already asserted, so the radio will enter RX as it should, and then go back to SLEEP (if no packets are received).

Workaround(s) To avoid this happening, t_{Event1} must be programmed to be larger than the maximum startup time of the crystal + PO_TIMEOUT.

$$t_{Event1} > \text{Max XOSC Startup} + \text{PO\_TIMEOUT} \quad (2)$$



Revision History

Changes from D Revision (September 2013) to E Revision	Page
• Added condition for WOR Issue when t_{Event1} is too Short.	11

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

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

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