

## **MSP430G2153 Device Erratasheet**

---



---



---

### **1 Revision History**

✓ The check mark indicates that the issue is present in the specified revision.

Errata Number	Rev C	Rev B	Rev A
<a href="#">BCL12</a>	✓	✓	✓
<a href="#">CPU4</a>	✓	✓	✓
<a href="#">EEM20</a>	✓	✓	✓
<a href="#">SYS15</a>	✓	✓	✓
<a href="#">TA12</a>	✓	✓	✓
<a href="#">TA16</a>	✓	✓	✓
<a href="#">TA21</a>	✓	✓	✓
<a href="#">TAB22</a>	✓	✓	✓
<a href="#">USCI20</a>	✓	✓	✓
<a href="#">USCI22</a>	✓	✓	✓
<a href="#">USCI23</a>	✓	✓	✓
<a href="#">USCI24</a>	✓	✓	✓
<a href="#">USCI25</a>	✓	✓	✓
<a href="#">USCI26</a>	✓	✓	✓
<a href="#">USCI29</a>	✓	✓	✓
<a href="#">USCI30</a>	✓	✓	✓
<a href="#">USCI35</a>	✓	✓	✓
<a href="#">USCI40</a>	✓	✓	✓
<a href="#">XOSC5</a>	✓	✓	✓

## 2 Package Markings

### N20

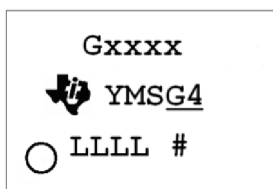
#### PDIP (N), 20 Pin



YM = Year and Month Date Code  
 LLLL = Assembly Lot Code  
 S = Assembly Site Code  
 # = DIE Revision

### PW20

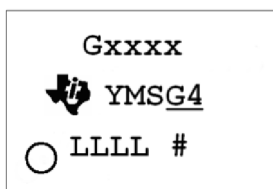
#### TSSOP (PW), 20 Pin



YM = Year and Month Date Code  
 LLLL = Assembly Lot Code  
 S = Assembly Site Code  
 # = DIE Revision  
 o = PIN 1

### PW28

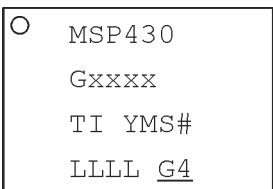
#### TSSOP (PW), 28 Pin



YM = Year and Month Date Code  
 LLLL = Assembly Lot Code  
 S = Assembly Site Code  
 # = DIE Revision  
 o = PIN 1

### RHB32

#### QFN (RHB), 32 Pin



TI = TI Letters  
 YM = Year and Month Date Code  
 LLLL = Assembly Lot Code  
 S = Assembly Site Code  
 # = Die Revision  
 O = Pin 1

### 3 Detailed Bug Description

#### BCL12 *BCS Module*

<b>Function</b>	Switching RSELx or modifying DCOCTL can cause DCO dead time or a complete DCO stop
<b>Description</b>	<p>After switching RSELx bits (located in register BCSTL1) from a value of &gt;13 to a value of &lt;12 OR from a value of &lt;12 to a value of &gt;13, the resulting clock delivered by the DCO can stop before the new clock frequency is applied. This dead time is approximately 20 us. In some instances, the DCO may completely stop, requiring a power cycle.</p> <p>Furthermore, if all of the RSELx bits in the BCSTL1 register are set, modifying the DCOCTL register to change the DCOx or the MODx bits could also result in DCO dead time or DCO hang up.</p>
<b>Workaround</b>	- When switching RSEL from >13 to <12, use an intermediate frequency step. The intermediate RSEL value should be 13.

Current RSEL	Target RSEL	Recommended Transition Sequence
15	14	Switch directly to target RSEL
14 or 15	13	Switch directly to target RSEL
14 or 15	0 to 12	Switch to 13 first, and then to target RSEL (two step sequence)
0 to 13	0 to 12	Switch directly to target RSEL

AND

- When switching RSEL from <12 to >13 it's recommended to set RSEL to its default value first (RSEL = 7) before switching to the desired target frequency.

AND

- In case RSEL is at 15 (highest setting) it's recommended to set RSEL to its default value first (RSEL = 7) before accessing DCOCTL to modify the DCOx and MODx bits. After the DCOCTL register modification the RSEL bits can be manipulated in an additional step.

In the majority of cases switching directly to intermediate RSEL steps as described above will prevent the occurrence of BCL12. However, a more reliable method can be implemented by changing the RSEL bits step by step in order to guarantee safe function without any dead time of the DCO.

Note that the 3-step clock startup sequence consisting of clearing DCOCTL, loading the BCSTL1 target value, and finally loading the DCOCTL target value as suggested in the in the "TLV Structure" chapter of the [MSP430x2xx Family User's Guide](#) is not affected by BCL12 if (and only if) it is executed after a device reset (PUC) prior to any other modifications being made to BCSTL1 since in this case RSEL still is at its default value of 7. However any further changes to the DCOx and MODx bits will require the consideration of the workaround outlined above.

#### CPU4 *CPU Module*

<b>Function</b>	PUSH #4, PUSH #8
<b>Description</b>	The single operand instruction PUSH cannot use the internal constants (CG) 4 and 8. The other internal constants (0, 1, 2, -1) can be used. The number of clock cycles is different:

PUSH #CG uses address mode 00, requiring 3 cycles, 1 word instruction  
 PUSH #4/#8 uses address mode 11, requiring 5 cycles, 2 word instruction

**Workaround** Workaround implemented in assembler.

## **EEM20** ***EEM Module***

**Function** Debugger might clear interrupt flags

**Description** During debugging read-sensitive interrupt flags might be cleared as soon as the debugger stops. This is valid in both single-stepping and free run modes.

**Workaround** None.

## **SYS15** ***SYS Module***

**Function** LPM3 and LPM4 currents exceed specified limits

**Description** LPM3 and LPM4 currents may exceed specified limits if the SMCLK source is switched from DCO to VLO or LFXT1 just before the instruction to enter LPM3 or LPM4 mode.

**Workaround** After clock switching, a delay of at least four new clock cycles (VLO or LFXT1) must be implemented to complete the clock synchronization before going into LPM3 or LPM4.

## **TA12** ***TIMER\_A Module***

**Function** Interrupt is lost (slow ACLK)

**Description** Timer\_A counter is running with slow clock (external TACLK or ACLK) compared to MCLK. The compare mode is selected for the capture/compare channel and the CCRx register is incremented by one with the occurring compare interrupt (if TAR = CCRx). Due to the fast MCLK the CCRx register increment (CCRx = CCRx+1) happens before the Timer\_A counter has incremented again. Therefore the next compare interrupt should happen at once with the next Timer\_A counter increment (if TAR = CCRx + 1). This interrupt gets lost.

**Workaround** Switch capture/compare mode to capture mode before the CCRx register increment. Switch back to compare mode afterwards.

## **TA16** ***TIMER\_A Module***

**Function** First increment of TAR erroneous when IDx > 00

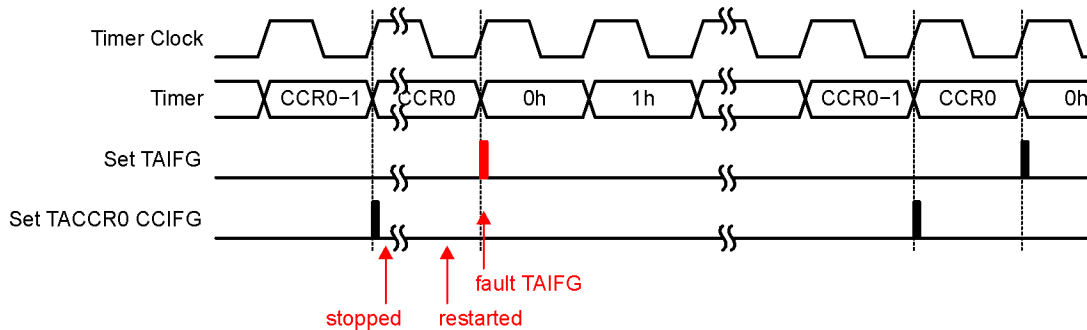
**Description** The first increment of TAR after any timer clear event (POR/TACLK) happens immediately following the first positive edge of the selected clock source (INCLK, SMCLK, ACLK or TACLK). This is independent of the clock input divider settings (ID0, ID1). All following TAR increments are performed correctly with the selected IDx settings.

**Workaround** None

## **TA21** ***TIMER\_A Module***

**Function** TAIFG Flag is erroneously set after Timer A restarts in Up Mode

**Description** In Up Mode, the TAIFG flag should only be set when the timer counts from TACCR0 to zero. However, if the Timer A is stopped at TAR = TACCR0, then cleared (TAR=0) by setting the TACLRL bit, and finally restarted in Up Mode, the next rising edge of the TACLK will erroneously set the TAIFG flag.



**Workaround** None.

## TAB22 *TIMER\_A/TIMER\_B Module*

**Function** Timer\_A/Timer\_B register modification after Watchdog Timer PUC

**Description** Unwanted modification of the Timer\_A/Timer\_B registers TACTL/TBCTL and TAIV/TBIV can occur when a PUC is generated by the Watchdog Timer(WDT) in Watchdog mode and any Timer\_A/Timer\_B counter register TACCRx/TBCCRx is incremented/decremented (Timer\_A/Timer\_B does not need to be running).

**Workaround** Initialize TACTL/TBCTL register after the reset occurs using a MOV instruction (BIS/BIC may not fully initialize the register). TAIV/TBIV is automatically cleared following this initialization.

Example code:

```
MOV.W #VAL, &TACTL
```

or

```
MOV.W #VAL, &TBCTL
```

Where, VAL=0, if Timer is not used in application otherwise, user defined per desired function.

## USCI20 *USCI Module*

**Function** I2C Mode Multi-master transmitter issue

**Description** When configured for I2C master-transmitter mode, and used in a multi-master environment, the USCI module can cause unpredictable bus behavior if all of the following four conditions are true:

1 - Two masters are generating SCL

And

2 - The slave is stretching the SCL low phase of an ACK period while outputting NACK on SDA

And

3 - The slave drives ACK on SDA after the USCI has already released SCL, and then the SCL bus line gets released

And

4 - The transmit buffer has not been loaded before the other master continues communication by driving SCL low

The USCI will remain in the SCL high phase until the transmit buffer is written. After the transmit buffer has been written, the USCI will interfere with the current bus activity and may cause unpredictable bus behavior.

**Workaround**

1 - Ensure that slave doesn't stretch the SCL low phase of an ACK period

Or

2 - Ensure that the transmit buffer is loaded in time

Or

3 - Do not use the multi-master transmitter mode

**USCI22**
**USCI Module**
**Function**

I2C Master Receiver with 10-bit slave addressing

**Description**

Unexpected behavior of the USCI\_B can occur when configured in I2C master receive mode with 10-bit slave addressing under the following conditions:

1) The USCI sends first byte of slave address, the slave sends an ACK and when second address byte is sent, the slave sends a NACK.

2) Master sends a repeat start condition (If UCTXSTT=1).

3) The first address byte following the repeated start is acknowledged.

However, the second address byte is not sent, instead the Master incorrectly starts to receive data and sets UCBxRXIFG=1.

**Workaround**

Do not use repeated start condition instead set the stop condition UCTXSTP=1 in the NACK ISR prior to the following start condition (USTXSTT=1).

**USCI23**
**USCI Module**
**Function**

UART transmit mode with automatic baud rate detection

**Description**

Erroneous behavior of the USCI\_A can occur when configured in UART transmit mode with automatic baud rate detection. During transmission if a "Transmit break" is initiated (UCTXBRK=1), the USCI\_A will not deliver a stop bit of logic high, instead, it will send a logic low during the subsequent synch period.

**Workaround**

1) Follow User's Guide instructions for transmitting a break/synch field following UCSWRST=1.

Or,

2) Set UCTXBRK=1 before an active transmission, i.e. check for bit UCBUSY=0 and then set UCTXBRK=1.

<b>USCI24</b>	<b><i>USCI Module</i></b>
<b>Function</b>	Incorrect baud rate information during UART automatic baud rate detection mode
<b>Description</b>	Erroneous behavior of the USCI_A can occur when configured in UART mode with automatic baud rate detection. After automatic baud rate measurement is complete, the UART updates UCAxBR0 and UCAxBR1. Under Oversampling mode (UCOS16=1), for baud rates that should result in UCAxBRx=0x0002, the UART incorrectly reports it as UCAxBRx=0x5555.
<b>Workaround</b>	When break/synch is detected following the automatic baud rate detection, the flag UCBRK flag is set to 1. Check if UCAxBRx=0x5555 and correct it to 0x0002.
<b>USCI25</b>	<b><i>USCI Module</i></b>
<b>Function</b>	TXIFG is not reset when NACK is received in I2C mode
<b>Description</b>	When the USCI_B module is configured as an I2C master transmitter the TXIFG is not reset after a NACK is received if the master is configured to send a restart (UCTXSTT=1 & UCTXSTP=0).
<b>Workaround</b>	Reset TXIFG in software within the NACKIFG interrupt service routine
<b>USCI26</b>	<b><i>USCI Module</i></b>
<b>Function</b>	Tbuf parameter violation in I2C multi-master mode
<b>Description</b>	<p>In multi-master I2C systems the timing parameter Tbuf (bus free time between a stop condition and the following start) is not guaranteed to match the I2C specification of 4.7us in standard mode and 1.3us in fast mode. If the UCTXSTT bit is set during a running I2C transaction, the USCI module waits and issues the start condition on bus release causing the violation to occur.</p> <p>Note: It is recommended to check if UCBBUSY bit is cleared before setting UCTXSTT=1.</p>
<b>Workaround</b>	None
<b>USCI29</b>	<b><i>USCI Module</i></b>
<b>Function</b>	Timing of USCI I2C interrupts may result in call to a reserved ISR location
<b>Description</b>	<p>When certain USCI I2C interrupt flags (IFG) are set and an automatic flag-clearing event on the I2C bus occurs, the device makes a call to the TRAPINT interrupt vector. This will only happen if the IFG is cleared within a critical time window (~6 CPU clock cycles) after a USCI interrupt request occurs and before the interrupt servicing is initiated. The affected interrupts are UCBxTXIFG, UCSTPIFG, UCSTTIFG and UCNACKIFG.</p> <p>The automatic flag-clearing scenarios are described in the following situations:</p> <p>(1) A pending UCBxTXIFG interrupt request is cleared on the falling SCL clock edge following a NACK.</p> <p>(2) A pending UCSTPIFG, UCSTTIFG, or UCNACKIFG interrupt request is cleared by a following Start condition.</p>
<b>Workaround</b>	(1) Poll the affected flags instead of enabling the interrupts.

(2) Define an ISR for the interrupt vector TRAPINT. If the failure condition occurs; a call to the TRAPINT ISR is made. After the interrupt is serviced, the device returns to the application code and continues execution.

Include the following ISR definition in the application code.

```
#pragma vector= TRAPINT_VECTOR
__interrupt void TRAPINT_ISR(void)
{
    __no_operation();
}
```

For IDE versions earlier than IAR V4.22 and CCS V4.2 in addition to the above code; include the ISR definition in the device header file.

In IAR Embedded Workbench include the following line in the device header file msp430xxx.h.

```
/******
```

```
* Interrupt Vectors (offset from 0xFFE0)
```

```
*****/
```

```
#define TRAPINT_VECTOR (0 * 2u) /*INCLUDE THIS LINE IN .h FILE*/
```

```
#define PORT1_VECTOR (2 * 2u) /* 0xFFE4 Port 1 */
```

In Code Composer Essentials/Studio include the following line in the device header file msp430xxx.h.

```
/******
```

```
* Interrupt Vectors (offset from 0xFFE0)
```

```
*****/
```

```
#define TRAPINT_VECTOR (0 * 1u) /*INCLUDE THIS LINE IN .h FILE*/
```

```
#define PORT1_VECTOR (2 * 1u) /* 0xFFE4 Port 1 */
```

## USCI30

### USCI Module

#### Function

I2C mode master receiver / slave receiver

#### Description

When the USCI I2C module is configured as a receiver (master or slave), it performs a double-buffered receive operation. In a transaction of two bytes, once the first byte is moved from the receive shift register to the receive buffer the byte is acknowledged and the state machine allows the reception of the next byte.

If the receive buffer has not been cleared of its contents by reading the UCBxRXBUF register while the 7th bit of the following data byte is being received, an error condition may occur on the I2C bus. Depending on the USCI configuration the following may occur:

1) If the USCI is configured as an I2C master receiver, an unintentional repeated start condition can be triggered or the master switches into an idle state (I2C communication aborted). The reception of the current data byte is not successful in this case.

2) If the USCI is configured as I2C slave receiver, the slave can switch to an idle state stalling I2C communication. The reception of the current data byte is not successful in this case. The USCI I2C state machine will notify the master of the aborted reception with a NACK.

Note that the error condition described above occurs only within a limited window of the 7th bit of the current byte being received. If the receive buffer is read outside of this



window (before or after), then the error condition will not occur.

**Workaround**

a) The error condition can be avoided altogether by servicing the UCBxRXIFG in a timely manner. This can be done by (a) servicing the interrupt and ensuring UCBxRXBUF is read promptly or (b) Using the DMA to automatically read bytes from receive buffer upon UCBxRXIFG being set.

OR

b) In case the receive buffer cannot be read out in time, test the I2C clock line before the UCBxRXBUF is read out to ensure that the critical window has elapsed. This is done by checking if the clock line low status indicator bit UCSCLOW is set for atleast three USCI bit clock cycles i.e.  $3 \times t(\text{BitClock})$ .

Note that the last byte of the transaction must be read directly from UCBxRXBUF. For all other bytes follow the workaround:

Code flow for workaround

- (1) Enter RX ISR for reading receiving bytes
- (2) Check if UCSCLOW.UCBxSTAT == 1
- (3) If no, repeat step 2 until set
- (4) If yes, repeat step 2 for a time period  $> 3 \times t(\text{BitClock})$  where  $t(\text{BitClock}) = 1/f(\text{BitClock})$
- (5) If window of  $3 \times t(\text{BitClock})$  cycles has elapsed, it is safe to read UCBxRXBUF

**USCI35**
**USCI Module**
**Function**

Violation of setup and hold times for (repeated) start in I2C master mode

**Description**

In I2C master mode, the setup and hold times for a (repeated) START,  $t_{\text{SU,STA}}$  and  $t_{\text{HD,STA}}$  respectively, can be violated if SCL clock frequency is greater than 50kHz in standard mode (100kbps). As a result, a slave can receive incorrect data or the I2C bus can be stalled due to clock stretching by the slave.

**Workaround**

If using repeated start, ensure SCL clock frequencies is  $< 50\text{kHz}$  in I2C standard mode (100 kbps).

**USCI40**
**USCI Module**
**Function**

SPI Slave Transmit with clock phase select = 1

**Description**

In SPI slave mode with clock phase select set to 1 (UCAxCTLW0.UCCKPH=1), after the first TX byte, all following bytes are shifted by one bit with shift direction dependent on UCMSB. This is due to the internal shift register getting pre-loaded asynchronously when writing to the USCIA TXBUF register. TX data in the internal buffer is shifted by one bit after the RX data is received.

**Workaround**

Reinitialize TXBUF before using SPI and after each transmission.

If transmit data needs to be repeated with the next transmission, then write back previously read value:

```
UCAxTXBUF = UCAxTXBUF;
```

**XOSC5**
**XOSC Module**

---

<b>Function</b>	LF crystal failures may not be properly detected by the oscillator fault circuitry
<b>Description</b>	The oscillator fault error detection of the LFXT1 oscillator in low frequency mode (XTS = 0) may not work reliably causing a failing crystal to go undetected by the CPU, i.e. OFIFG will not be set.
<b>Workaround</b>	None

## 4 Document Revision History

Changes from family erratasheet to device specific erratasheet.

1. Errata TA22 was renamed to TAB22
2. Description for TAB22 was updated

Changes from device specific erratasheet to document Revision A.

1. BCL12 Workaround was updated.

Changes from document Revision A to Revision B.

1. Errata TA21 was added to the errata documentation.

Changes from document Revision B to Revision C.

1. Silicon Revision B was added to the errata documentation.
2. USCI29 Workaround was updated.
3. USCI29 Function was updated.
4. Silicon Revision C was added to the errata documentation.
5. USCI29 Description was updated.

Changes from document Revision C to Revision D.

1. Errata USCI35 was added to the errata documentation.

Changes from document Revision D to Revision E.

1. Package Markings section was updated.

Changes from document Revision E to Revision F.

1. Errata USCI40 was added to the errata documentation.

Changes from document Revision F to Revision G.

1. TA21 Description was updated.

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

### Applications

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

### TI E2E Community

[e2e.ti.com](http://e2e.ti.com)