Power Modes in CC111xFx, CC243x, and CC251xFx

By Torgeir Sundet

Keywords

- *Power Modes*
- *Data Sheet*
- *Errata Note*
- *Work around*
- *Interrupt Service Routine*
- *External Port Interrupt*
- *Sleep Timer Interrupt*
- *CC1110*
- *CC1111*
- *CC2430*
- *CC2431*
- *CC2510*
- *CC2511*

1 Introduction

This design note combines relevant information from the CC111xFx/CC251xFx/CC2430 data sheets ([1], [2] and [3]), and CC111xFx/CC251xFx errata notes ([4] and [5]), in order to compile all key information regarding CC111xFx/CC251xFx/CC243x power mode.

The main objective is to explain the CC111xFx/CC251xFx/CC243x software instructions required to safely enter a power mode and resume active mode.

Table of Contents

2 Abbreviations

DMA	Direct Memory Access
HS XOSC	High Speed Xtal Oscillator, refer to relevant data sheets [1], [2] and [3] for actual frequency.
HS RCOSC	High Speed RC Oscillator, refer to relevant data sheets [1], [2] and [3] for actual frequency.
ISR	Interrupt Service Routine.
NOP	No Operation
PM	Power Mode, e.g. PM0, PM1, PM2, and PM3.
SFR	Special Function Register
SoC	System on Chip. A collective term used to refer to Texas Instruments ICs with on-chip MCU and RF transceiver. Used in this document to reference the CC1110, CC1111, CC2430, CC2431, CC2510 and CC2511.
RX	Receive. Used in this document to reference radio receive.
TX	Transmit. Used in this document to reference radio transmit.

3 Background

In the SoC data sheets ([1], [2], and [3]) it is stated that the SoC has one active mode and four power modes, called PM0, PM1, PM2, and PM3, where PM3 has the lowest power consumption. The power modes are shown in Table 1 together with voltage regulator and oscillator options.

Operating Mode	High-speed Oscillator	Low-speed Oscillator	Digital Voltage Regulator	CPU
Configuration	A None B High speed XOSC C HS RCOSC	A None B Low power RCOSC C 32.768 kHz XOSC		
Active	B and / or C	B or C	On	Running
PM0	B and / or C	B or C	On	Idle
PM1	A	B or C	On	Idle
PM2	A	B or C	Off	Idle
PM3	A	A	Off	Idle

Table 1: Operating Modes

Active mode: The full functional mode. The voltage regulator to the digital core is on and either the high speed RC oscillator or the high speed crystal oscillator or both are running. Either the Low power RC oscillator or the 32.768 kHz crystal oscillator is running.

PM0: Same as active mode, but the CPU is idle, meaning that no code is being executed.

PM1: The voltage regulator to the digital part is on. Neither the high speed crystal oscillator nor the high speed RC oscillator is running. Either the low power RC oscillator or the 32.768 kHz crystal oscillator is running. The system will go to active mode on reset or an external interrupt or when the Sleep Timer expires.

PM2: The voltage regulator to the digital core is turned off. Neither the high speed crystal oscillator nor the high speed RC oscillator is running. Either the low power RC oscillator or the 32.768 kHz crystal oscillator is running. The system will go to active mode on reset or an external interrupt or when the Sleep Timer expires. The CC2511Fx will lose all USB state information when PM2 is entered. *Thus, PM2 should not be used with USB.*

PM3: The voltage regulator to the digital core is turned off. None of the oscillators are running. The system will go to active mode on reset or an external interrupt. The CC2511Fx will lose all USB state information when PM3 is entered. *Thus, PM3 should not be used with USB*.

4 Switching Power Modes

This section describes the key elements of entering and exiting each SoC power mode. The desired power mode is selected by the SLEEP.MODE bits. Setting the IDLE bit in the PCON SFR after setting the SLEEP.MODE bits, makes the SoC enter the selected power mode. Please note that when SLEEP.MODE ≠ 0, interrupts and oscillator switching are gated (blocked).

Also note the minimum requirement on HS XOSC power down guard time in all modes of operation for CC111xFx/CC251xFx, see Table 11 in the corresponding CC111xFx/CC251xFx data sheets ([1] and [2]). Spending less time in PM than the required HS XOSC power down guard time might cause RF packet error/loss, if RX or TX mode is entered when resuming active mode.

4.1 Entering Power Mode (PM)

As already introduced in section 4, SLEEP.MODE and PCON.IDLE are used to make the SoC enter PM. Nevertheless, note that in order for the SoC to wake up from PM{0 - 3} it is required to enable the global SoC interrupt and the relevant peripheral interrupt. For PM0 the SoC will wake up upon any enabled interrupt, while PM{1 – 3} requires that the Sleep Timer or Port 0/1/2 interrupt is enabled.

4.1.1 Entering PM0 - CC111xFx/CC251xFx/CC243x

In order to safely enter PM0 it is only necessary to set SLEEP.MODE = 00, and PCON.IDLE = 1. Since PM0 does not power down the oscillator(s) the application does not have to implement any particular oscillator control associated with PM0 entry/exit. Thus the resulting code is very simple, as shown in Figure 1.

```
// C language code:

void main(void)
{
    // Setup + enable the interrupt source(s) which is/are intended to wake-up
    // the SoC from PM0. Any SoC peripheral interrupt will wake up the SoC from
    // PM0

    // Set SLEEP.MODE to PM0.
    SLEEP = (SLEEP & 0xFC) | 0x00;

    // Set PCON.IDLE to enter PM0
    PCON |= 0x01;

    // The SoC is now in PM0 and will only wake up upon any enabled SoC interrupt
}
```

Figure 1: Code for Entering PM0 - CC111xFx/CC251xFx/CC243x

4.1.2 Entering PM{1 - 3} - CC111xFx/CC251xFx/CC243x

With reference to the SoC data sheet ([1], [2], and [3]), the code sequence shown in Figure 2 must be applied to ensure safe PM{1 - 3} entry. Note that the CC243x must use the HS RCOSC as system clock source prior to entering PM{1 - 3}. Also, be aware of the particular code requirements shown in Figure 3; CC111xFx/CC251xFx PM{1 - 2} entry using the Sleep Timer, and the DMA setup needed for CC111xFx/CC251xFx PM{2 - 3} entry, shown in Figure 4.

```
// C language code:

void main(void)
{
    // Here the application should implement the relevant code shown in
    // Figure 5/Figure 6.

    // Setup + enable the interrupt source(s) (Sleep Timer, Port) which is/are
    // intended to wake-up the SoC from PM.

    ////////////////////////////////////////////////////////////////////////
    // NOTE:
    // For entering PM{1 – 2} in CC111xFx/CC251xFx, using the Sleep timer,
    // the following code section must be replaced by the code shown in
    // Figure 3 !
    // For entering PM{2 – 3} in CC111xFx/CC251xFx the following code
    // section must be replaced by the code shown in Figure 4 !
    ////////////////////////////////////////////////////////////////////////

    ////////////////////////////////////////////////////////////////////////
    ////////////////////// Code section begin //////////////////////////////
    ////////////////////////////////////////////////////////////////////////

    // Set SLEEP.MODE according to desired PM, e.g. PM1.
    SLEEP = (SLEEP & 0xFC) | 0x01;

    // Apply three NOPs to allow the corresponding interrupt blocking to take
    // effect, before verifying the SLEEP.MODE bits below. Note that all
    // interrupts are blocked when SLEEP.MODE ≠ 0, thus the time between
    // setting SLEEP.MODE ≠ 0, and asserting PCON.IDLE should be as short as
    // possible. If an interrupt occurs before the NOPs have completed, then
    // the enabled ISR shall clear the SLEEP.MODE bits, according to the code
    // in Figure 7.
    asm("NOP");
    asm("NOP");
    asm("NOP");

    // If no interrupt was executed in between the above NOPs, then all
    // interrupts are effectively blocked when reaching this code position.

    // If the SLEEP.MODE bits have been cleared at this point, which means
    // that an ISR has indeed executed in between the above NOPs, then the
    // application should not enter PM{1 – 3} !
    if (SLEEP & 0x03)
    {
        // Set PCON.IDLE to enter the selected PM, e.g. PM1.
        PCON |= 0x01;

        // The SoC is now in PM and will only wake up upon Sleep Timer interrupt
        // or external Port interrupt.

        // First instruction upon exiting PM.
        asm("NOP");
    }

    ////////////////////////////////////////////////////////////////////////
    ////////////////////// Code section end ////////////////////////////////
    ////////////////////////////////////////////////////////////////////////

    // Here the application should implement the relevant code shown in
    // Figure 5/Figure 6.
    ...
}
```

Figure 2: Code for Entering PM{1 - 3} - CC111xFx/CC251xFx/CC243x

4.1.3 Entering PM{1 - 2} using the Sleep Timer - CC111xFx/CC251xFx

The “Sleep Timer and Power Modes” chapter in the CC111xFx/CC251xFx data sheets ([1] and [2]) states that Entering PM{1 - 2} has to be aligned to a positive edge on the 32 kHz clock source. Any update to the compare value, EVENT0, has to happen prior to this positive edge. There has to be at least two positive edges on the 32 kHz clock source between WORCTRL.WOR_RESET being asserted and updating EVENT0 or entering PM{1 - 2}. If EVENT0 is changed to a value lower than the current counter value, WORCTRL.WOR_RESET has to be asserted first. Code alternative 1 or 2 in Figure 3 should be used in order to update EVENT0 and entering PM{1 - 2} correctly:

```
// C language code:

void main(void)
{
    char temp = WORTIME0;

    // Here the application should implement the relevant code shown in
    // Figure 5/Figure 6.

    // Set + verify SLEEP.MODE (see Figure 2 for detailed explanation).
    SLEEP = (SLEEP & 0xFC) | 0x01;  // Choose PM, e.g. PM1.
    asm("NOP");
    asm("NOP");
    asm "NOP");
    if (SLEEP & 0x03)
    {
        ////////////////////////////////////////////////////////////////////
        ////////////////////////////////////////////////////////////////////
        // Alternative 1:
        // Alignment of entering PM{1 - 2} to a positive edge on the 32 kHz clock
        // source and updating Event0 to a value higher than current Sleep Timer value.
        while(temp == WORTIME0);    // Wait until a positive 32 kHz edge.
        WOREVT1 = desired event0;   // Set Event0, high byte.
        WOREVT0 = desired event0;   // Set Event0, low byte.
        ////////////////////////////////////////////////////////////////////
        ////////////////////////////////////////////////////////////////////

        ////////////////////////////    OR    ////////////////////////////////

        ////////////////////////////////////////////////////////////////////
        ////////////////////////////////////////////////////////////////////
        // Alternative 2:
        // Reset the Sleep Timer and align the entering of PM{1 - 2} to a positive edge
        // on the 32 kHz clock source. Update Event0 to a value lower than current
        // Sleep Timer value.
        WORCTRL |= 0x04;            // Reset Sleep Timer.
        temp = WORTIME0;
        while(temp == WORTIME0);    // Wait until a positive 32 kHz edge.
        temp = WORTIME0;
        while(temp == WORTIME0);    // Wait until a positive 32 kHz edge.
        WOREVT1 = desired event0;   // Set Event0, high byte.
        WOREVT0 = desired event0;   // Set Event0, low byte.
        ////////////////////////////////////////////////////////////////////
        ////////////////////////////////////////////////////////////////////

        // Set PCON.IDLE to enter the selected PM.
        PCON |= 0x01;               // Enter PM, e.g. PM1.

        // The SoC is now in PM and will only wake up upon Sleep Timer interrupt.

        // First instruction upon exiting PM. See Figure 2 for reference.
        asm("NOP");
    }

    // Here the application should implement the relevant code shown in
    // Figure 5/Figure 6.
    ...
}
```

Figure 3: Code for PM{1 - 2} Entry using Sleep Timer - CC111xFx/CC251xFx

4.1.4 Entering PM{2 - 3} - CC111xFx/CC251xFx

The Errata Notes for CC111xFx/CC251xFx ([4] and [5]) states that, when waking up from PM{2 - 3} there is a small chance that the SLEEP.MODE bits are faulty set to a value other than zero before the PCON.IDLE bit is cleared by the CPU. This causes the CC111xFx/CC251xFx to re-enter PM{2 - 3} immediately. Since an enabled interrupt is pending at this point, the CC111xFx/CC251xFx will wake up and re-enter PM{2 - 3} continuously and appear to hang. Once the CC111xFx/CC251xFx hangs, only a system reset will get the CC111xFx/CC251xFx back to normal operation.

By ensuring that the SLEEP.MODE bits are written to zero at the instant the CC111xFx/CC251xFx wakes up from PM{2 - 3}, the CC111xFx/CC251xFx will never re-enter PM{2 - 3} unintentionally, see “Suggested Workaround” section in the CC111xFx/CC251xFx Errata Notes ([4] and [5]). This can be done by setting up a DMA transfer of a certain number of xdata bytes to the SLEEP register that is manually triggered (by writing to the relevant DMAREQ.DMAREQx bit) right before writing the PCON.IDLE bit: However, the work around requires that the following conditions are met:

- The CC111xFx/CC251xFx is running at the HS RC oscillator at the highest possible clock speed setting
- The HS XOSC is powered down
- Flash Cache is disabled

Please note that the requirements stated in the “Power Management Control” and “Sleep Timer and Power Modes” chapters of the CC111xFx/CC251xFx data sheets ([1] and [2]) still apply.

NOTE! The code in Figure 4 assumes the CC111xFx/CC251xFx is already running on the HS RCOSC with the highest clock speed setting possible.

```
// C language code:

// Initialization of source buffers and DMA descriptor for the DMA transfer
unsigned char __xdata PM2_BUF[7] = {0x06,0x06,0x06,0x06,0x06,0x06,0x04};
unsigned char __xdata PM3_BUF[7] = {0x07,0x07,0x07,0x07,0x07,0x07,0x04};
unsigned char __xdata dmaDesc[8] = {0x00,0x00,0xDF,0xBE,0x00,0x07,0x20,0x42};

void main(void)
{
    // Store current DMA channel 0 descriptor and abort any ongoing transfers,
    // if the channel is in use.
    unsigned char storedDescHigh = DMA0CFGH;
    unsigned char storedDescLow = DMA0CFGL;
    DMAARM |= 0x81;

    // Update descriptor with correct source.
    // NB! Replace &PM2_BUF with &PM3_BUF if powermode 3 is chosen instead.
    dmaDesc[0] = (unsigned int)& PM2_BUF >> 8;
    dmaDesc[1] = (unsigned int)& PM2_BUF;

    // Associate the descriptor with DMA channel 0 and arm the DMA channel
    DMA0CFGH = (unsigned int)&dmaDesc >> 8;
    DMA0CFGL = (unsigned int)&dmaDesc;
    DMAARM = 0x01;

    ////////////////////////////////////////////////////////////////////////////
    // NOTE! At this point, make sure all interrupts that will not be used to
    // wake from PM are disabled as described in the "Power Management Control"
    // chapter of the data sheet.
    // The following code is timing critical and should be done in the
    // order as shown here with no intervening code.

    // Align with positive 32 kHz clock edge as described in the
    // "Sleep Timer and Power Modes" chapter of the data sheet.

    char temp = WORTIME0;
    while(temp == WORTIME0);

    // Make sure XOSC is powered down when entering PM{2 - 3} and that the
    // flash cache is disabled.
    // NB! Replace 0x06 with 0x07 if power mode 3 is chosen instead.
    MEMCTR |= 0x02;
    SLEEP = 0x06;

    // Enter power mode as described in chapter "Power Management Control"
    // in the data sheet. Make sure DMA channel 0 is triggered just before
    // setting PCON.IDLE.
    asm("NOP");
    asm("NOP");
    asm("NOP");

    if(SLEEP & 0x03)
    {
        asm("MOV 0xD7,#0x01");  // DMAREQ = 0x01;
        asm("NOP");             // Needed to perfectly align the DMA transfer.
        asm("ORL 0x87,#0x01");  // PCON |= 0x01;
        asm("NOP");
    }
    // End of timing critical code
    ////////////////////////////////////////////////////////////////////////////

    // Enable Flash Cache.
    MEMCTR &= ~0x02;

    // Update DMA channel 0 with original descriptor and arm channel if it was
    // in use before PM was entered.
    DMA0CFGH = storedDescHigh;
    DMA0CFGL = storedDescLow;
    DMAARM = 0x01;
    ...
}
```

Figure 4: Code for Entering PM{2 - 3} - CC111xFx/CC251xFx

4.1.5 Entering PM{2 - 3} - CC243x

The CC243x does not have the same issue (described in section 4.1.4) as CC111xFx/CC251xFx. Thus the procedure for making CC243x enter PM{2 - 3} is the same as described in section 4.1.2. Still, note that the CC243x must enter PM on the HS RCOSC.

4.2 Resuming Active Mode

When initializing/resuming active mode it is important to apply correct oscillator initialization/monitoring, such that the SoC operates on the required oscillator during, e.g. RF operation. This requirement does not apply to PM0, since the oscillator(s) are not powered down in this mode.

4.2.1 Oscillator Initialization/Monitoring - CC111xFx/CC251xFx

The following oscillator function/requirement only applies for PM{1 – 3}, see the CC111xFx/CC251xFx data sheets ([1] and [2]) for details:

1. Prior to entering PM1 the CC111xFx/CC251xFx can use either the HS RCOSC or HS XOSC as system clock source, while for PM{2 - 3} the HS RCOSC must be used as system clock source.
2. When resuming Active mode the CC111xFx/CC251xFx will initially always use the HS RCOSC as system clock source.
3. If the HS XOSC was running prior to entering PM, then the CC111xFx/CC251xFx will automatically power it up again when resuming active mode, no matter if the HS XOSC was actually used as system clock source. If the HS XOSC was indeed the system clock source prior to entering PM, then the CC111xFx/CC251xFx will also automatically switch from HS RCOSC to HS XOSC when the HS XOSC is stable. However, in order to ensure that the HS XOSC is stable before continuing code execution the application should still monitor `SLEEP.XOSC_STB`.

The corresponding code is show in Figure 5.

```
// C language code:

void main(void)
{
    ...

    // The following relevant code lines should be executed at the very
    // beginning, as well as after the PM{1 - 3} resume position, of the
    // application program:

    // Clear CLKCON.OSC to make the CC111xFx/CC251xFx operate on the HS XOSC.
    // This will only be necessary if the HS XOSC was not running prior
    // to entering PM.
    CLKCON &= 0xBF;

    // Monitor SLEEP.XOSC_STB to ensure the HS XOSC is stable before continuing
    // code execution (e.g. before operating the RF module).
    while(!(SLEEP & 0x40));

    // Set SLEEP.OSC_PD to power down the HS RCOSC.
    SLEEP |= 0x04;

    ...
}
```

Figure 5: Code for Oscillator Initialization/Monitoring - CC111xFx/CC251xFx

4.2.2 Oscillator Initialization/Monitoring - CC243x

The following oscillator function/requirement only applies for PM{1 – 3}, see the CC2430 data sheet ([3]) for details:

1. The CC243x must always use the HS RCOSC as system clock source prior to entering PM.
2. The CC243x always resumes active mode on HS RCOSC.
3. If the application wants to use the HS XOSC as system clock source in active mode, then it has to be manually powered up, by setting the `CLKCON.OSC` bit. Note that the `SLEEP.XOSC_STB` is not 100% reliable, so it is necessary to add 64 µs between monitoring `SLEEP.XOSC_STB` and clearing the `CLKCON.OSC` bit (selecting HS XOSC).

The corresponding code is show in Figure 6.

```
// C language code:

void main(void)
{
    ...

    // The following relevant code lines should be executed at the very
    // beginning, and after the PM{1 - 3} resume position, of the application
    // program. However, note that they are only necessary if the application
    // needs to run the HS XOSC in Active mode:

    // Set SLEEP.OSC_PD to power up the HS XOSC (HS RCOSC is already running).
    SLEEP &= 0xFB;

    // Monitor SLEEP.XOSC_STB to ensure HS XOSC is stable before continuing
    // code execution.
    while(!(SLEEP & 0x40));

    // According to the CC243x data sheet ([3]), apply relevant number of NOPs
    // here, which represents 64 µs safety time before actually switching to
    // the HS XOSC.
    asm("NOP");

    // Clear CLKCON.OSC to make the CC243x operate on the HS XOSC.
    CLKCON &= 0xBF;

    // Monitor CLKCON.OSC to ensure CC243x has actually switched to the HS XOSC.
    while(CLKCON & 0x40);

    // Set SLEEP.OSC_PD to power down the HS RCOSC.
    SLEEP |= 0x04;

    ...
}
```

Figure 6: Code for Oscillator Initialization/Monitoring - CC243x

4.2.3 Resuming Active mode - CC111xFx/CC251xFx/CC243x

In order to ensure that the SoC resumes code execution consistently after exiting a power mode (waking up), an ISR, as shown in Figure 7, must be implemented for each enabled wake-up source. For PM{1 – 3} this means external port interrupt and/or Sleep Timer interrupt. Note that for PM3, the only wake-up source is external port interrupt. Except from PM0 it is critical that the ISR clears the SLEEP.MODE bits (00).

```
// C language code:

// Sleep Timer Interrupt Service Routine (ISR)

_PRAGMA(vector=VECT(5, 0x2B)) __near_func __interrupt void ST_ISR(void);
_PRAGMA(vector=VECT(5, 0x2B)) __near_func __interrupt void ST_ISR(void)
{
    // Clear IRCON.STIF (Sleep Timer CPU interrupt flag)
    IRCON &= 0x7F;

    // Clear WORIRQ.EVENT0_FLAG (Sleep Timer peripheral interrupt flag)
    // This is required for the CC111xFx/CC251xFx only!
    WORIRQ &= 0xFE;

    ...

    // Clear the SLEEP.MODE bits, because an interrupt can also occur before
    // the SoC has actually entered PM. If this interrupt occurs in between the
    // three NOPs (that is; before the corresponding interrupt blocking has
    // actually taken effect) in Figure 2, then this clearing of the SLEEP.MODE
    // bits will ensure that the application does not enter PM{1 – 3}.
    SLEEP &= 0xFC;  // Not required when resuming from PM0
}

// External Port Interrupt Service Routine (ISR) for e.g. Port1
_PRAGMA(vector=VECT(15, 0x7B)) __near_func __interrupt void P1_ISR(void);
_PRAGMA(vector=VECT(15, 0x7B)) __near_func __interrupt void P1_ISR(void)
{
    // Clear P1IFG.bit1 (Port1.Pin1 peripheral interrupt flag)
    P1IFG = 0xFD;

    // Clear IRCON2.P1IF (Port1 CPU interrupt flag)
    IRCON2 &= 0xF7;

    ...

    // Clear the SLEEP.MODE bits, because an interrupt can also occur before
    // the SoC has actually entered PM. If this interrupt occurs in between the
    // three NOPs (that is; before the corresponding interrupt blocking has
    // actually taken effect) in Figure 2, then this clearing of the SLEEP.MODE
    // bits will ensure that the application does not enter PM{1 – 3}.
    SLEEP &= 0xFC;  // Not required when resuming from PM0
}
```

Figure 7: Code for Resuming Active Mode - CC111xFx/CC251xFx/CC243x

5 References

[1] CC1110Fx/CC1111Fx Data Sheet (SWRS033)

[2] CC2510Fx/CC2511Fx Data Sheet (SWRS055)

[3] CC2430 Data Sheet (SWRS036)

[4] CC1110Fx/CC1111Fx Errata Note (SWRZ022)

[5] CC2510Fx/CC2511Fx Errata Note (SWRZ014)

6 General Information

6.1 Document History

Revision	Date	Description/Changes
SWRA162A	2008.03.19	Introduced separate section for Power Mode 0 entry, and updated the existing sections accordingly. Updated color on code comments.
SWRA162	2007.11.23	Initial release.

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**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Clocks and Timers	www.ti.com/clocks	Digital Control	www.ti.com/digitalcontrol
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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