

# **Sensor Sequencing Using the CC13x2 and CC26x2 Sensor Controller**

---

---

---

Mandus Börjesson

## **ABSTRACT**

The purpose of this application note is to demonstrate how current consumption can be minimized, by making the Sensor Controller automatically enable a sensor at a designated time period before the sensor is to be sampled. This implementation allows the Sensor Controller to stay in the lowest-possible power mode while the sensor boots and then start up when the sensor is ready. In this application note, the idea of enabling a sensor at specific times is referred to as sensor sequencing.

---

## **Contents**

1	Introduction .....	2
2	Limitations .....	3
3	Practical Example.....	4
Appendix A	Code Implementation.....	6

## **List of Figures**

1	Sensor Sequencing With One and Multiple Sensors .....	2
2	Timer and Event Configuration Visualization .....	3
3	Full Sensor Sequencing Using CC2642R1 Sensor Controller Without Sensors Connected .....	4
4	Full Sensor Sequencing Using the CC2642R1 Sensor Controller .....	5
5	Lazy Sensor Sequencing – All Sensors Powered at Same Time .....	5

## **List of Tables**

1	Energy Consumed Depending on Configuration .....	4
---	--	---

## **Trademarks**

SimpleLink, Code Composer Studio are trademarks of Texas Instruments.  
All other trademarks are the property of their respective owners.

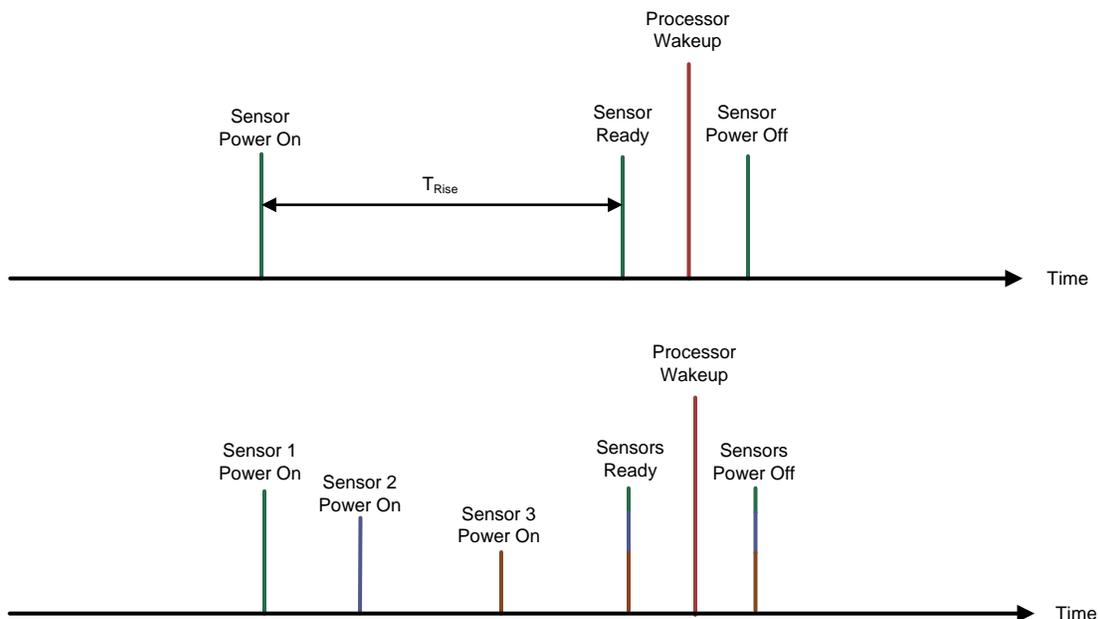
## 1 Introduction

The CC13x2 and CC26x2 microcontroller units (MCUs) have quite a few improvements compared to their predecessors, the CC13x0 and CC26x0 MCUs. One improvement is the upgraded Sensor Controller that resides in a separate power domain of the MCU. The Sensor Controller is a part of the CC13xx and CC26xx SimpleLink™ wireless MCUs. The Sensor Controller is a sensor interface with a programmable CPU that is optimized for low-power operation.

To implement sensor sequencing, the sensor controller must be able to do the following:

- Wake up periodically
- Enable sensors at the right time (preferably without waking up from sleep)
- Read sensor data at processor wakeup
- Power off sensors when data collection is complete

Figure 1 shows the wake-up sequences for both one sensor and multiple sensors.



**Figure 1. Sensor Sequencing With One and Multiple Sensors**

Timer 2 is used because of its four capture/compare channels, which allow event setting and clearing without waking the sensor controller. The four channels of timer 2 can therefore be used to wake up the device at a user-defined interval, as well as control I/Os.

The first criterion was achieved by using capture/compare channel 3 to trigger at the value, which represents the desired period, and set an event when the value is reached. By setting up an event trigger, the Sensor Controller wakes up when the event is set. The timer was set up in *Up repeatedly* mode, and the target value was the same as the capture/compare value.

The second criterion was achieved similarly to the first, but instead of setting up an event trigger, the sensor power pins were each tied to one event. This setup caused the pins to go high when the appropriate value was reached. For the sensor power pins, capture/compare registers zero, one, and two were used.

The third criterion depends on implementation. For the use case shown in this application note, two analog pins were read and an accelerometer was polled through the SPI.

Because the sensor power pins are tied to certain events, the fourth criterion was achieved by manually clearing the events.

**NOTE:** Any sensor reading must be done before clearing the events, because the sensors lose power when the event is cleared.

Figure 2 shows how the timer was set up to set the events (and their related pins) in relation to the desired power-on sequence. The period of the wake-up is determined by CC3. With timer 2 set up to run at 32 kHz, the value of CC3 must be set to that in Equation 1, to wake up every  $T_{desired, ms}$ .

$$32 \times T_{desired, ms} \tag{1}$$

CC0, CC1, and CC2 determine the amount of time the sensor is powered before the processor wakes up, and is defined in Equation 2.

$$CCN = (T_{desired, ms} - T_{riseN, ms}) \times 32$$

where

- $T_{riseN, ms}$  is the setup time for sensor N. (2)

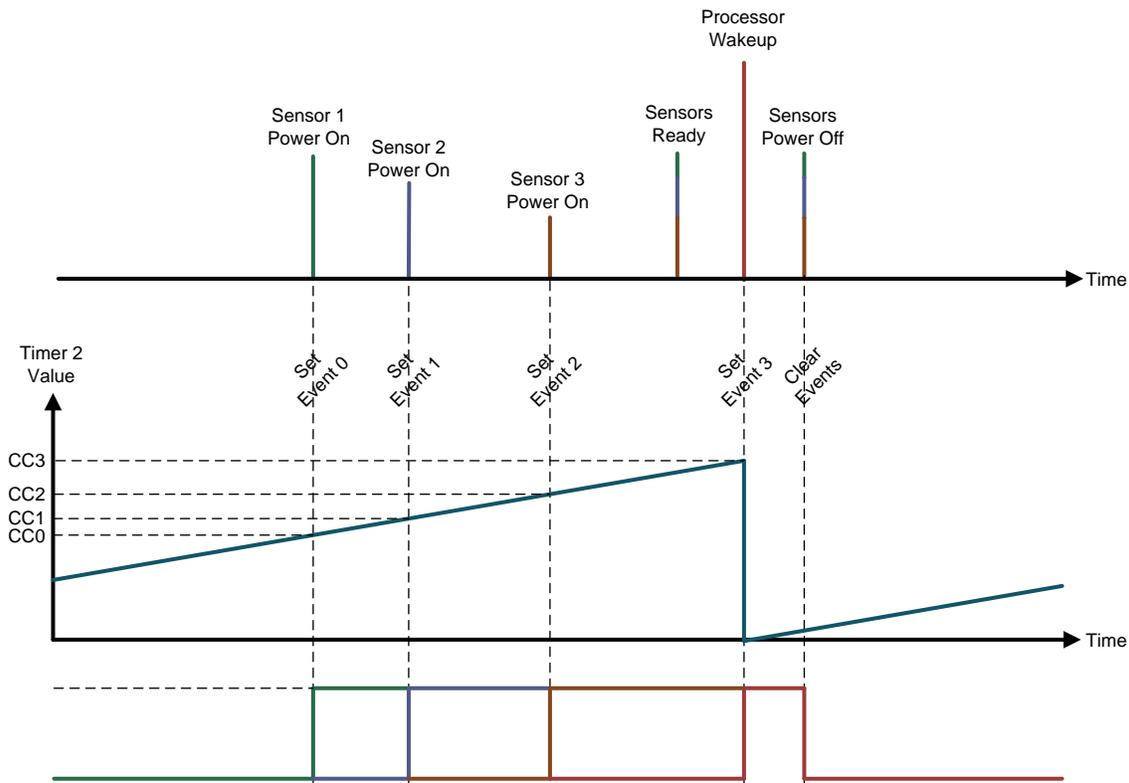


Figure 2. Timer and Event Configuration Visualization

## 2 Limitations

Given that timer 2 is set to run at the 32-kHz oscillator, and that CC0, CC1, CC2 and CC3 are used to determine the wake-up times, the maximum resolution is 1/32 milliseconds. However, for ease-of-use and readability, the setup times are given in whole milliseconds in the example.

Timer 2 is a 16-bit timer that gives it a maximum value of 65535. Using the lowest-frequency clock (32 kHz) and no prescaler, the maximum interval is shown in Equation 3. Using the prescaler, it is possible to increase the interval by a factor of up to 256. However, using the prescaler causes the maximum resolution to increase.

$$65535 / 32000 = 2.048 \text{ s} = 2048 \text{ ms} \tag{3}$$

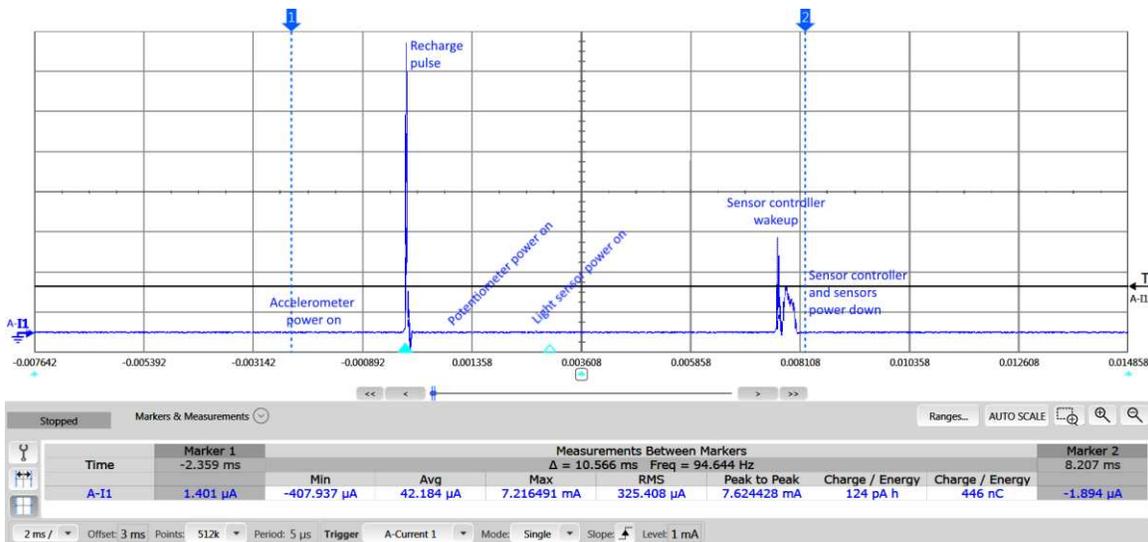
### 3 Practical Example

To showcase the use of sensor sequencing, a simple example that read two analog sensors (a potentiometer and an analog light sensor) and polled an accelerometer was created using the previously introduced ideas. The sensor wake-up times were 10, 7, and 5 ms. A simple solution where all sensors are powered on at the same time was also done to highlight the differences in current consumption and energy use (see Table 1). Remember that power consumed depends on the sensors used, their power consumption and wake-up times, and the power mode of the sensor controller. The recharge pulse present in all cases is not part of the sensor sequencing, but rather an automatically reoccurring event required to prevent the Sensor Controller from completely losing power.

**Table 1. Energy Consumed Depending on Configuration**

Configuration	Energy Per Complete Wakeup
No sensor connected (only recharge and sensor controller energy)	0.45 $\mu$ C
Complete sensor sequencing	7.47 $\mu$ C
Lazy sensor sequencing (all sensors on at the same time)	10.77 $\mu$ C

Figure 3 displays the current consumption when no sensors are connected; notice that the sensor controller does not need to wake up to set the sensor power pins high. Figure 4 shows the current consumption with all sensors connected. Figure 5 shows the current consumption when all sensors are on at the same time.



**Figure 3. Full Sensor Sequencing Using CC2642R1 Sensor Controller Without Sensors Connected**

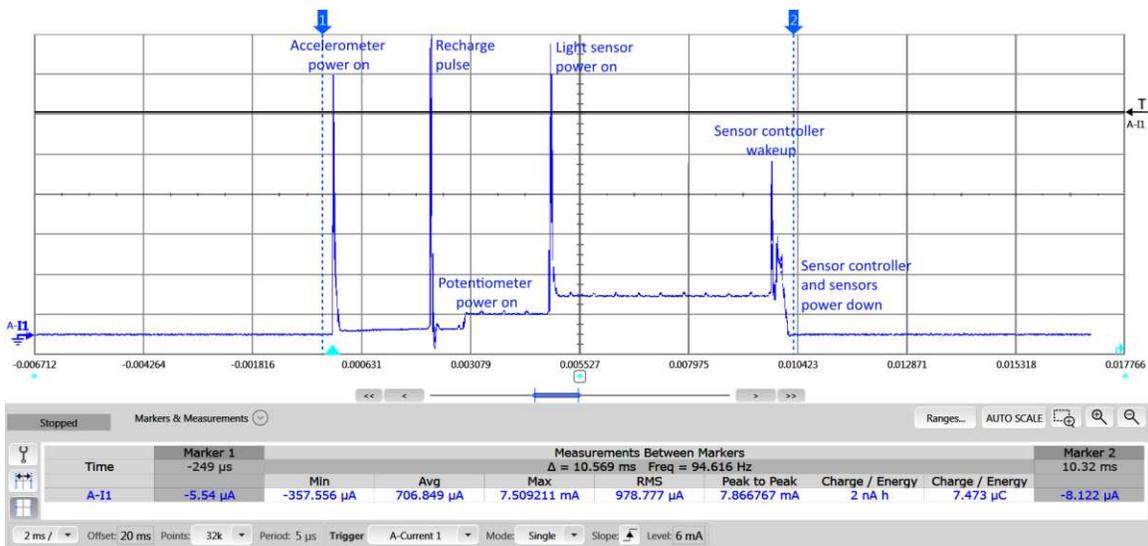


Figure 4. Full Sensor Sequencing Using the CC2642R1 Sensor Controller

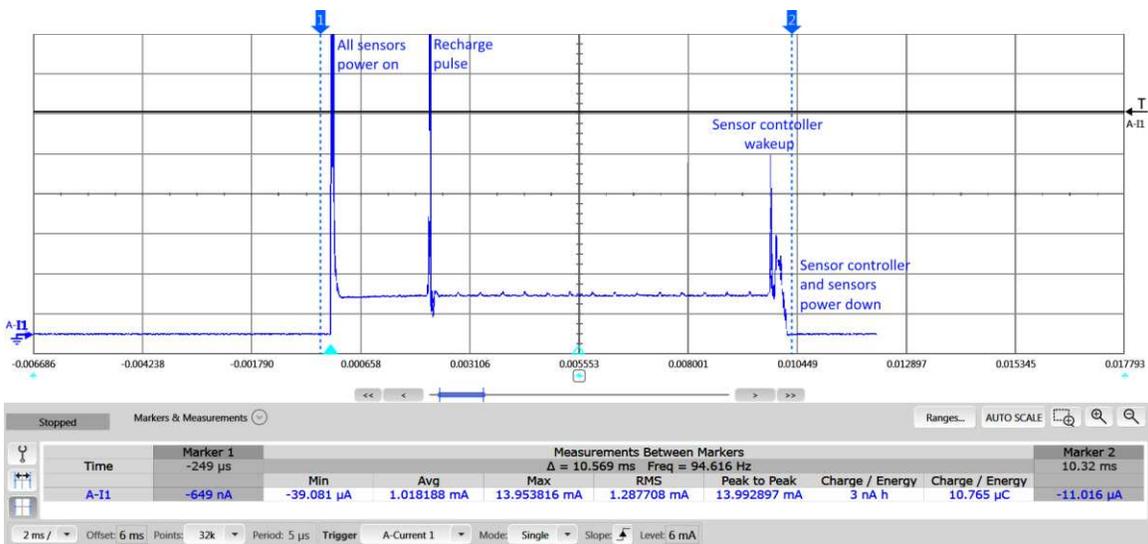


Figure 5. Lazy Sensor Sequencing – All Sensors Powered at Same Time

## Code Implementation

---

---

---

### A.1 Overview

The following data structures and constants were created, and task resources included:

- `cfg`
  - `SensorSetupTimes[3]` – Contains the setup times for each sensor in milliseconds
  - `period` – Period of wake-ups in ms, initially 2040
- Constants
  - `TIMER2_ALL_CLEAR` – Used to clear event 0..3, set to 0x0055 (85 dec)
- Task resources
  - Digital Output Pins
    - `PIN_SENSOR_0`
    - `PIN_SENSOR_1`
    - `PIN_SENSOR_2`
  - Timer 2
  - System CPU Alert (optional)
  - Timer 2 Event Trigger

### A.1.1 Initialization Code

```
timer2SetClockSource(TIMER2_CLOCK_SRC_HFDIV2);
timer2WaitForClockSource();

evhSetupTimer2Trigger(0,3,1,EVH_TIMER2_TRIG_ON_MATCH);

//Set enable pins as outputs
gpioCfgMode(AUXIO_O_PIN_SENSOR_0,GPIO_MODE_OUTPUT);
gpioCfgMode(AUXIO_O_PIN_SENSOR_1,GPIO_MODE_OUTPUT);
gpioCfgMode(AUXIO_O_PIN_SENSOR_2,GPIO_MODE_OUTPUT);

// Time = nCycles*period = nCycles/Frequency
// nCycles = Time*Frequency, [ms]*[kHz]=[Clock ticks]

//Set up timer comparison values
timer2SetInitCmpValue(0,(cfg.period - cfg.SensorSetupTimes[0])<<5);// <<5 = *32 (32KHz)
timer2SetInitCmpValue(1,(cfg.period - cfg.SensorSetupTimes[1])<<5);
timer2SetInitCmpValue(2,(cfg.period - cfg.SensorSetupTimes[2])<<5);
timer2SetInitCmpValue(3,(cfg.period)<<5);

//Set cc channels, Set events when timer reaches comparator value (Remember to clear events when
in active mode)
timer2CfgCcChannel(0,TIMER2_CC_MODE_SET_ON_CMP,0x01); //Pin 0 event
timer2CfgCcChannel(1,TIMER2_CC_MODE_SET_ON_CMP,0x02); //Pin 1 event
timer2CfgCcChannel(2,TIMER2_CC_MODE_SET_ON_CMP,0x04); //Pin 2 event
timer2CfgCcChannel(3,TIMER2_CC_MODE_SET_ON_CMP,0x08); //Event handler event

//Connect timer events to sensor enable pins
timer2ConnectEventToGpio(0,AUXIO_O_PIN_SENSOR_0);
timer2ConnectEventToGpio(1,AUXIO_O_PIN_SENSOR_1);
timer2ConnectEventToGpio(2,AUXIO_O_PIN_SENSOR_2);

timer2SetInitCounterTarget(cfg.period<<5);
timer2SetClockSource(TIMER2_CLOCK_SRC_LF);
timer2WaitForClockSource();
timer2StartWithTarget(TIMER2_CNTR_MODE_UP_REP);
```

### A.1.2 Event Handler A Code

```
//=====
// STEP 1: Gather Data
//=====

//=====
// STEP 2: Set pins low
//=====

//Clear all events, this causes enable pins to go low
timer2ModifyEvents(TIMER2_ALL_CLEAR);

//=====
// STEP 3: Process data
//=====

//=====
// STEP 4: Set up next wakeup
//=====

//Set up new trigger for event 3
evhSetupTimer2Trigger(0,3,1,EVH_TIMER2_TRIG_ON_MATCH);

//Wake up main CPU for data processing if desired
//fwGenAlertInterrupt();
```

### A.1.3 Termination Code

```
//Stop the timer
timer2Stop();

//Reset cc channels
timer2ResetCcChannel(0);
timer2ResetCcChannel(1);
timer2ResetCcChannel(2);
timer2ResetCcChannel(3);

//Clear all generated events
timer2ModifyEvents(TIMER2_ALL_CLEAR);

//Disable the clock
timer2SetClockSource(TIMER2_CLOCK_SRC_NONE);
timer2WaitForClockSource();
```

## A.2 Code Composer Studio Code

The following code should be added to your Code Composer Studio™ project. The code sets up three sensors with wake-up times of 5, 7, and 10 ms and a Sensor Controller wake up every 100 ms. Remember to register a task alert callback if the Sensor Controller generates an ALERT signal – `fwGenAlertInterrupt()`; – upon waking up.

### Includes

```
#include "scif.h"  
#define BV(x)    (1 << (x))
```

### Thread setup

```
// Initialize the Sensor Controller  
scifOsalInit();  
//scifOsalRegisterCtrlReadyCallback(scCtrlReadyCallback);  
//scifOsalRegisterTaskAlertCallback(scTaskAlertCallback); //If ALERT is used  
scifInit(&scifDriverSetup);  
  
scifTaskData.sensorSequencing.cfg.SensorSetupTimes[0] = 5; //Sensor 0  
scifTaskData.sensorSequencing.cfg.SensorSetupTimes[1] = 7; //Sensor 1  
scifTaskData.sensorSequencing.cfg.SensorSetupTimes[2] = 10; //Sensor 2  
scifTaskData.sensorSequencing.cfg.period = 100; //Desired period (ms)  
  
// Start Sensor Controller task  
scifStartTasksNbl(BV(SCIF_YOUR_TASK_NAME_TASK_ID));
```

## IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ("TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products (<http://www.ti.com/sc/docs/stdterms.htm>), [evaluation modules](#), and [samples](http://www.ti.com/sc/docs/sampterm.htm) (<http://www.ti.com/sc/docs/sampterm.htm>).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2018, Texas Instruments Incorporated