

# TSC2004 Linux® Driver

*Data Acquisition Products*

## ABSTRACT

This application report describes the TSC2004 Linux® operating system driver to help customers to implement designs using the [TSC2004 touch-screen controller](#) from Texas Instruments. It also discusses the driver and associated code. The Linux driver code can be integrated into a customer's software system under different host processors. This driver has been tested and used on the Atmel [AT91SAM9261EK](#) platform. Project collateral discussed in this application report can be downloaded from the following URL: <http://www.ti.com/lit/zip/SBAA164>.

---

## Contents

1	Description.....	2
2	System Details .....	3
3	TSC2004 Touch Driver .....	4
4	Environment Details .....	8
5	Board Power-Up .....	8
6	Configuration Parameters .....	9
7	References .....	9

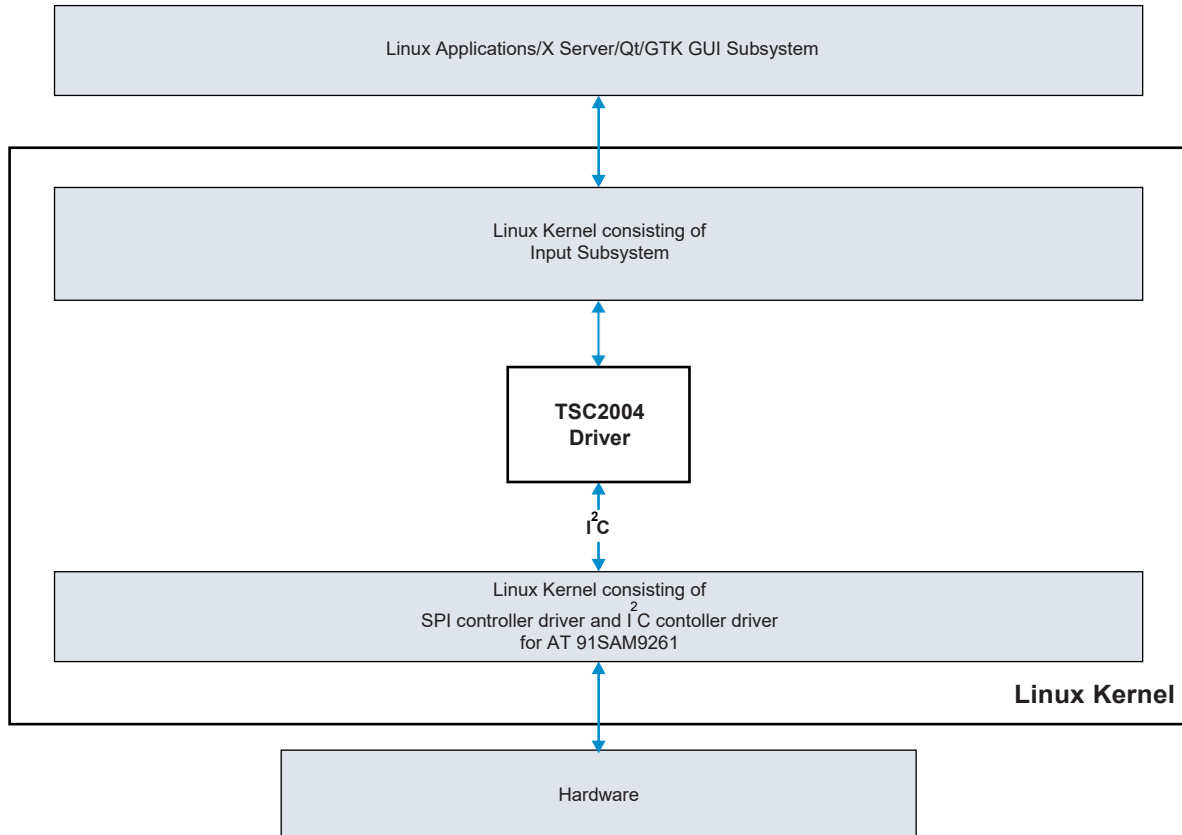
## List of Figures

1	TSC2004 Linux Driver Architecture .....	2
2	TSC2004 Connection Diagram .....	3
3	TSC2004 Touch Driver Code Organization .....	4
4	TSC2004 Touch Driver Initialization .....	5

Linux is a registered trademark of Linus Torvalds.  
Microsoft, Windows are registered trademarks of Microsoft Corporation.  
I<sup>2</sup>C is a trademark of NXP Semiconductors.  
All other trademarks are the property of their respective owners.

## 1 Description

The TSC2004 Linux driver acts as a standard input driver based on an I<sup>2</sup>C™ protocol driver. This configuration is described in the block diagram shown in [Figure 1](#), which depicts the position of the driver in the Linux kernel and the various interfaces it uses and feeds.



**Figure 1. TSC2004 Linux Driver Architecture**

Development of the Linux driver for the TSC2004 involves the following tasks:

- Developing a driver module that implements I<sup>2</sup>C data communications with the TSC2004.
- Developing a driver module that initializes and captures the general-purpose input/output (GPIO) data from the host processor.
- Inclusion of the device entry into the Linux kernel list of I<sup>2</sup>C devices.

## 2 System Details

### 2.1 Architecture

Figure 2 shows a connection diagram for the TSC2004.

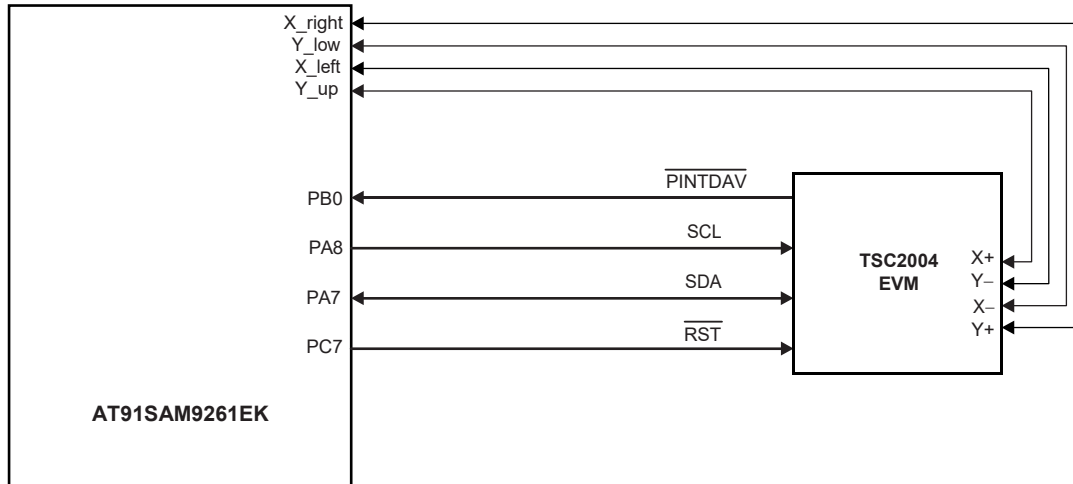


Figure 2. TSC2004 Connection Diagram

The Atmel AT91SAM9261EK board has an onboard [ADS7843 touch-screen controller](#). The digital and analog lines to the touch-screen controller device are isolated to provide the TSC2004 evaluation module (EVM) with the X+, Y+, X-, and Y- lines. The 3.3V power supply and ground required for the TSC2004 EVM are also provided by the onboard 3.3V source and ground.

### 2.2 Interface Details: Digital Interface

Table 1 summarizes the AT91SAM9261EK and TSC2004 digital hardware interface.

Table 1. AT91SAM9261EK and TSC2004 Digital Hardware Interface

Signal/Input	Host Processor Pin	TSC2004 Pin on EVM
TWCK (I <sup>2</sup> C Clock)	PA8 / Pin 45	SCL
TWD (I <sup>2</sup> C Data)	PA7 / Pin 44	SDA
PINTDAV	PB0 /	PINTDAV
RESET	PC7 /	RST

### 2.3 Interface Details: I<sup>2</sup>C Interface

The I<sup>2</sup>C bus on the TSC2004 is the primary hardware interface to the host processor. The host processor issues commands to the TSC2004 controller through the I<sup>2</sup>C interface.

#### 2.3.1 I<sup>2</sup>C Driver

The TSC2004 I<sup>2</sup>C driver establishes the software interface between the processor and the TSC2004 device. It contains three files:

- tsc2004\_core.c
- tsc2004.h
- tsc2004\_platform.c

The third file, tsc2004\_platform.c, is processor-dependent.

The fundamental routines for the I<sup>2</sup>C driver and platform routines are summarized in [Table 2](#) and [Table 3](#), respectively.

**Table 2. I<sup>2</sup>C Driver Routines**

Routine Name	Function
tsc2004_handle_penirq	The Interrupt handler function which is invoked when the PINTDAV line goes low. This routine is the Linux top half; therefore, a bottom-half handler must also be called.
tsc2004_detect	The detect function checks for the I <sup>2</sup> C functionalities of the underlying I <sup>2</sup> C master driver. It also registers the client driver, sets up the controller mode, and then registers the driver as an input driver. It requests the IRQ by invoking the tsc2004_register driver.
tsc2004_timer	The bottom-half handler for the Linux Interrupt handler, which performs I <sup>2</sup> C transactions to read X, Y, Z <sub>1</sub> , and Z <sub>2</sub> coordinates from the controller and report to the input subsystem.

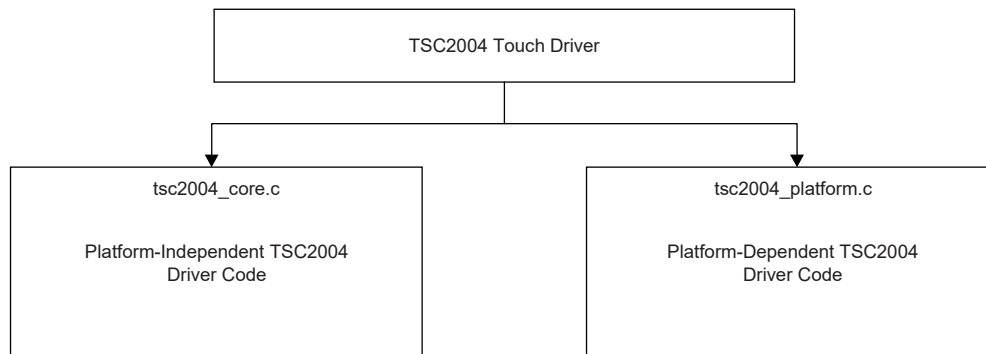
**Table 3. Platform Routines**

Routine Name	Function
tsc2004_detect_irq	To set up the IRQ line that is connected to the TSC2004. This function expects the IRQ number/ID to be returned. If connected on a GPIO, this function performs the GPIO initialization, such as setting the GPIO into input mode and enabling glitch filters on availability.
get_pendown_state	<u>This function</u> returns the current line status of the PINTDAV line. This action is required in order to determine the current state of the pen-touch, so as to continue reporting the coordinates to the Input subsystem.

### 3 TSC2004 Touch Driver

#### 3.1 Driver Code Organization

[Figure 3](#) illustrates the TSC2004 touch driver code structure.


**Figure 3. TSC2004 Touch Driver Code Organization**

The TSC2004 touch driver is an I<sup>2</sup>C client driver that is a kernel module in Linux. The entry point of this driver is *tsc2004\_module\_init*. This routine is called upon the insertion of the module. This driver makes use of the I<sup>2</sup>C subsystem, which provides kernel APIs to read and write from and to an I<sup>2</sup>C slave device.

The kernel application programming interfaces (APIs) provided are:

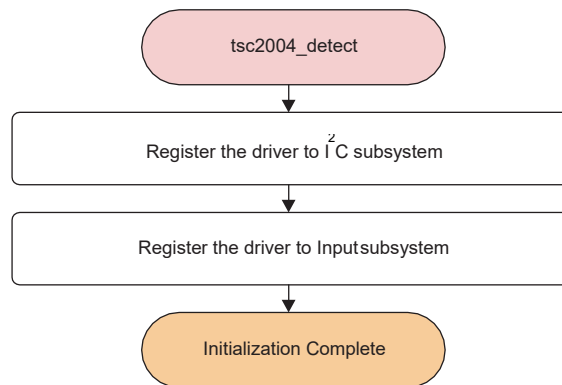
- `i2c_master_send`
- `i2c_master_recv`

Upon module initialization, the driver is registered as an I<sup>2</sup>C client driver through the `i2c_add_driver` API. Upon detection of an I<sup>2</sup>C slave device from the I<sup>2</sup>C subsystem, the `tsc2004_detect` function (which is an I<sup>2</sup>C detect function) is called.

## 3.2 Initialization and Code Routines

### 3.2.1 Touch Driver Initialization

The touch driver initializes when the function `tsc2004_init` is called while loading the driver module. This function adds the I<sup>2</sup>C driver using `i2c_add_driver`, which in turn calls the `tsc2004_detect` function that is registered to be called upon the detection of an I<sup>2</sup>C slave device. Figure 4 shows the TSC2004 touch driver initialization process.



**Figure 4. TSC2004 Touch Driver Initialization**

The registration of the driver into the Input subsystem is done by calling `tsc2004_register_driver` from the `tsc2004_detect` function.

```

/*Device found. Attach it to the I2C bus */
if ((err = i2c_attach_client(new_client))){
dev_err(&new_client->dev,
"%s: unable to attach device on : %#x\n",
__FUNCTION__, address);
goto err_exit;
}
err = tsc2004_register_driver(data)
  
```

This code fragment shows the initialization of our driver as both an I<sup>2</sup>C client and as an input device.

### 3.2.2 Interrupt Routine

Upon an interrupt request, or IRQ (interfaced through a GPIO pin connected to the PINTDAV line from the TSC controller), the *tsc2004\_handle\_penirq* routine is called. This routine checks for the line state, starts a timer if the line is down, reads the data, and reports touch coordinates to the input subsystem.

```
static irqreturn_t tsc2004_handle_penirq(int irq, void *v)
{
    struct tsc2004_data *data = (struct tsc2004_data *)v;
    disable_irq(data->pen_irq);
    /*Start timer when the touch is detected */
    if (pen_down_state(data->pen_irq)) {
        mod_timer(&data->timer, jiffies + START_TIMER_DELAY);
    } else {
        /*Pen off. Report to the input subsystem */
        input_report_key(data->idev, BTN_TOUCH, TOUCH_FALSE);
        input_report_abs(data->idev, ABS_PRESSURE, PRESSURE_FALSE);
        input_sync(data->idev);
        enable_irq(data->pen_irq);
    }
    return IRQ_HANDLED;
}
```

### 3.2.3 Reading Touch Data

Upon an IRQ, the read command is sent to the TSC2004 controller. The following functions are then called to read the touch data. Received valid touch data are then reported to the input subsystem.

```
/*Send command to read X register */
tx_buffer = TSC2004_CMD0(X_REG, PND0_FALSE, READ_REG);
stat = i2c_master_send(client, &tx_buffer, TX_BUFFER_BYTES_1);
if (stat != TX_BUFFER_BYTES_1) {
    dev_warn(&client->dev, "Unable to read X command\n");
}
/* *
 * Use the sequential read cycle of the TSC2004 to read
 * back the registers
 * */
stat = i2c_master_recv(client, (u8 *) reg_values, REG_BUF_BYTES);
```

The data received are byte-swapped and modified according to the LCD coordinates before being reported to the Input subsystem.

```
/*Check for prolonged touches i.e drags and drawings */
/* Check the IRQ line to avoid false clicks*/
if (pen_down_state(data->pen_irq)) {
    /*Report the X,Y,Z1,Z2 inside so as to avoid false clicks*/
    input_report_abs(data->idev, ABS_X, kernel_x);
    input_report_abs(data->idev, ABS_Y, kernel_y);
    input_report_abs(data->idev, ABS_PRESSURE, pressure_val);
    input_report_key(data->idev, BTN_TOUCH, TOUCH_TRUE);
    input_sync(data->idev);
    mod_timer(&data->timer, jiffies + RESTART_TIMER_DELAY);
} else {
    enable_irq(data->pen_irq);
}
```

The sequence of steps outlined here repeat for a prolonged touch event (such as drags or draws across the surface of the touch screen).

### 3.2.4 Exit/Cleanup

The exit point of this driver is the *tsc2004\_exit* function, which is invoked during the removal of the kernel module from the system. This function removes the entry of the driver as the I<sup>2</sup>C client driver, during which the subsystem invokes the *tsc2004\_detach* function, which frees the IRQ, deregisters the device as an input driver, and frees up the memory that has been allocated.

### 3.3 Development System Connection

These items are required in order to connect the development system:

- 9-pin serial COM cable from J15 of AT91SAM board to an available PC COM port connector
- Ethernet cable from Ethernet plug under the LCD/touch screen to an available PC Ethernet connector

### 3.4 Setup and Download Sequence

#### **For Microsoft® Windows®-based PC installations with a serial download:**

1. Install WinSCP.
2. Unzip the files, then move the Kernel Image and TSC200x driver files to the proper directory.
3. Start HyperTerminal. Build and configure a connection with these settings:
  - Baud rate: 115200
  - Data bits: 8
  - Parity: None
  - Stop bits: 1
  - Flow control: None
4. Power up the AT91SAM9261EK, and wait for the **u-boot>** prompt to appear.
5. At the hyperterminal window, key in this command sequence:
  - loadb 0x22200000
6. Find the ulmage kernel binary file to load using Kermit protocol.
7. After downloading the file, key in this command sequence:
  - bootm 22200000
8. Login using *root* as the username.
9. Configure the Ethernet connection using the following command:
  - ifconfig eth0 <Some IP> up
10. Send the tsc200x.ko to the AT91SAM board through the winSCP protocol. An ssh server is already running on the board.
11. Key in this command sequence:
  - insmod tsc200x.ko
  - pkill x

#### **For Linux system installations with an Ethernet download:**

1. Install a terminal emulator such as Kermit.
2. Open Kermit with the same settings as shown [for Windows installations](#).
3. Set up tftp server on the Linux host using this sequence:
  - vi /etc/xinetd.d/tftp
  - Delete the line that contains *disable = yes*
  - /etc/init.d/xinetd restart
4. Copy the ulmage binary in /tftpboot.
5. On the terminal emulator window, at the **u\_boot>** prompt, key in the following commands:
  - setenv ipaddr <SAM IP>
  - setenv serverip <Host IP>
  - saveenv
  - tftp 22200000 ulmage
  - bootm 22200000
6. At the login prompt, login as *root*.
7. On the Atmel board, issue this command:
  - ifconfig eth0 <SAM IP> up
8. From the Linux host, key in the following command:
  - scp tsc200x.ko root@<SAM IP>:~

9. On the Atmel board, key in the following commands:
  - insmod tsc200x.ko
  - pkill X

## 4 Environment Details

### 4.1 Setting Up the Development Environment

The development environment consists of a host (the development system) and a target (the board). The host system should have all the software discussed in this section and the correct tools for a proper setup of the environment.

---

**Note:** Some of these requirements may be optional, depending on the selection of the specific development environment.

---

- A Linux-based machine, with a kernel of 2.6 or higher installed. (Any Linux flavor such as CentOS-el5 is suitable.)
- The cross-compiler tool *arm-linux-gnueabi* installed. (Review the code source repository for details of the same tool to be installed on a Windows-based machine, or locate these details at <http://www.linux4sam.org/twiki/bin/view/Linux4SAM/SoftwareTools>.)
- TFTP server, NFS server, and Secure shell utilities. (These utilities should be available with a normal Linux distribution.)
- For a Windows-based machine (with the Windows XP operating system), the SAM boot assistant (SAM-BA) should be installed. The PC should also have available serial port and USB interfaces. Locate the SAM-BA through the following link, [http://www.linux4sam.org/twiki/bin/view/Linux4SAM/SoftwareTools#SAM\\_BA](http://www.linux4sam.org/twiki/bin/view/Linux4SAM/SoftwareTools#SAM_BA).
- A serial terminal program such as hyperterminal or TeraTerm, running on a Windows/Linux PC that is connected to the board via a serial cable.

The target system here is an Atmel AT91SAM9261EK evaluation board, which comes equipped with a USB cable, serial cable, Ethernet cable, and an LCD + touch screen interface (an ADS7846 device).

## 5 Board Power-Up

In order to flash images on the AT91SAM board, a tool called SAM-BA is required. (This tool is available from the Atmel website at [www.atmel.com](http://www.atmel.com).) The tool is driven by a USB connection from the PC to the board.

The following steps are required to download the bootstrap loader and the other images onto the SAM9261 evaluation board (a detailed explanation of this procedure is also available on the linux4sam.org web site,

[http://www.linux4sam.org/twiki/bin/view/Linux4SAM/GettingStarted#Flashing\\_a\\_demo\\_on\\_AT91\\_boards](http://www.linux4sam.org/twiki/bin/view/Linux4SAM/GettingStarted#Flashing_a_demo_on_AT91_boards)).

Follow these steps to flash the images into the AT91SAM board.

- Step 1. Connect a USB cable to the board.
- Step 2. Jumper J4 on the AT91SAM board must be opened (BMS = 1) to boot from the on-chip Boot ROM.
- Step 3. Remove the data flash jumper (J21).
- Step 4. Power up the board.
- Step 5. Verify that the USB connection is established. (If *Atmel AT91xxxx Test Board* appears in the taskbar notification area, this message indicates that the tool must be installed.)
- Step 6. Plug the Data Flash Jumper (J21) back into position 1-2.
- Step 7. Now the user can proceed with downloading a *demo* routine, which has the set of images available from the linux4sam site.
- Step 8. The demo directory also has a .bat file that can be executed, which self-installs all the images on the board.



- Step 9. A logfile.log is created upon completion of the steps outlines above; the USB cable then can be removed from the board.
- Step 10. On power-cycling the board, the LCD should display the Angstrom desktop and other applications.

### 5.1 Downloading and Customizing a Kernel for the Board

The demo images do not provide all the necessary modules to develop the TSC200x drivers, so a new Linux-2.6.24 image is built and loaded instead of the demo image provided. Download the copy of the Linux-2.6.24 image provided by the website shown here ([linux4sam.org](http://linux4sam.org)), which also provides the latest copy of the patch file for the AT91SAM9261EK, 2.6.24-at91.patch.gz. (The procedure to extract the Image and apply the patch is also given at the linux4sam.org website at: <http://www.linux4sam.org/twiki/bin/view/Linux4SAM/LinuxKernel#Build>.)

### 5.2 Installation of the Linux Kernel Image

Loading an image onto the board can be carried out in two ways:

1. Load an image by replacing the Linux kernel image file provided in the demo.  
This method requires the newly created image to have the same name as that of the older one, and follows the same procedures discussed earlier in loading the demo applications.
2. Loading an image through u-boot.

The Linux kernel ulmage can also be loaded through u-boot, by first setting the following environment variables in u-boot, running a *tftp* server on one of the Linux/Windows machines, and providing network access to the board in order to connect to the machine that is running the TFTP server.

```
#setenv serverip 172.22.1.119
#setenv ipaddr 172.22.1.118
#setenv bootcmd='tftp 22200000 ulmage; bootm 22200000'
#saveenv
```

On rebooting the board, the ulmage created and transferred onto the tftp server should be loaded onto the board at address 22200000, and should have started to execute.

## 6 Configuration Parameters

Table 4 summarizes the required configuration parameters.

**Table 4. Configuration Parameters**

Macro Name	Feature
BIT_MODE_12 / BIT_MODE_10	Enables 12- or 10-bit resolution (that is, the number of valid bits read for each coordinate)
TSMODE1 / TSMODE2	Configures the TSC2004 for either TSC-controlled or Host-controlled mode
REPORT_ACTUAL_PRESSURE	Enables reporting of actual pressure using Z <sub>1</sub> , Z <sub>2</sub> values
BATCH_DELAY_ON	Enables the Batch Delay mode

## 7 References

The following documents are available for download through the Texas Instruments web site ([www.ti.com](http://www.ti.com)).

- [TSC2004](#): Nano-power touch screen controller with I<sup>2</sup>C serial interface. Product data sheet [SBAS408E](#).
- TSC2004EVM and TSC2004EVM-PDK. User's guide [SLAU215](#).

## IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale ([www.ti.com/legal/termsofsale.html](http://www.ti.com/legal/termsofsale.html)) or other applicable terms available either on [ti.com](http://ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2019, Texas Instruments Incorporated