TSC2007 Linux® Driver

Data Acquisition Products

ABSTRACT

This application report describes the TSC2007 Linux® operating system driver to help customers to implement designs using the TSC2007 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.

Contents

List of Figures

1 Description

The TSC2007 Linux driver acts as a standard input driver based on the I²C™ protocol. 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.

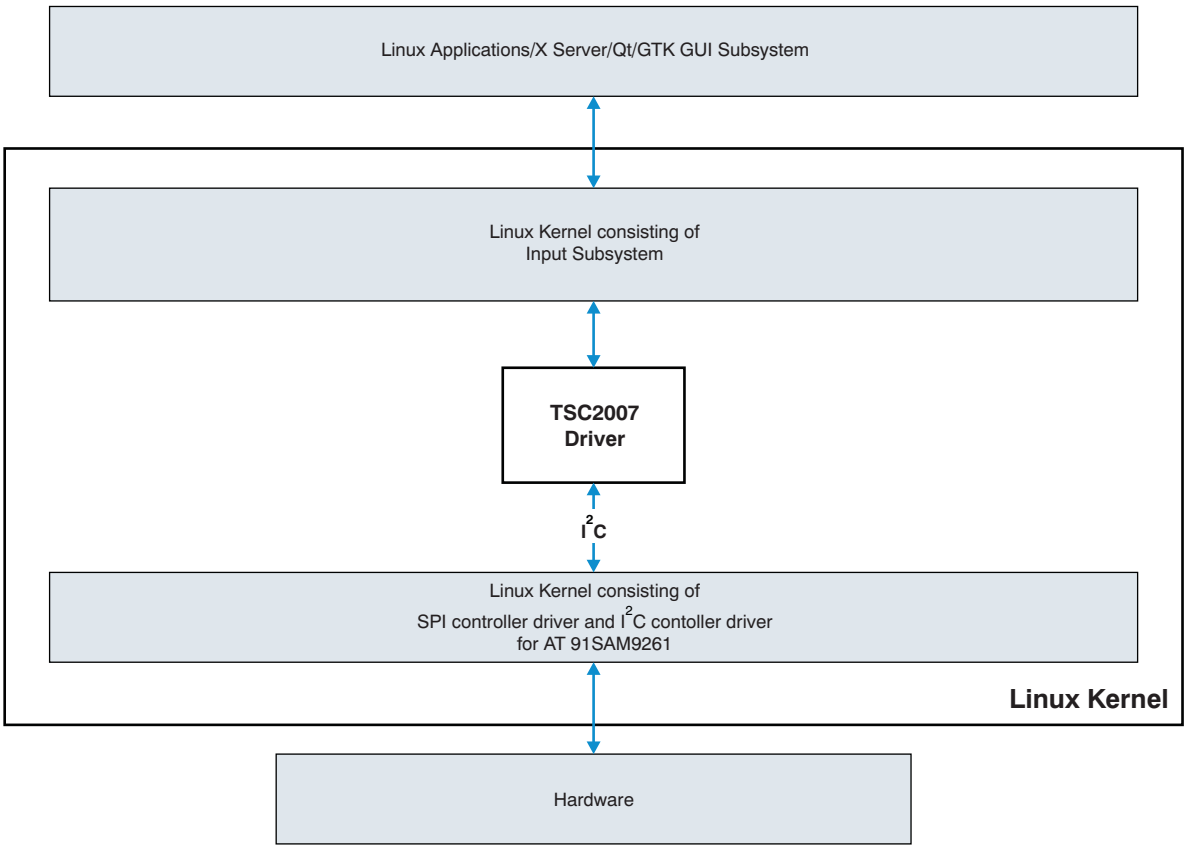

Figure 1. TSC2007 Linux Driver Architecture

Development of the Linux driver for the TSC2007 involves the following tasks:

- Developing a driver module that implements I²C data communications with the TSC2007.
- 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 I²C devices.

2 System Details

2.1 Architecture

Figure 2 shows a connection diagram for the TSC2007.

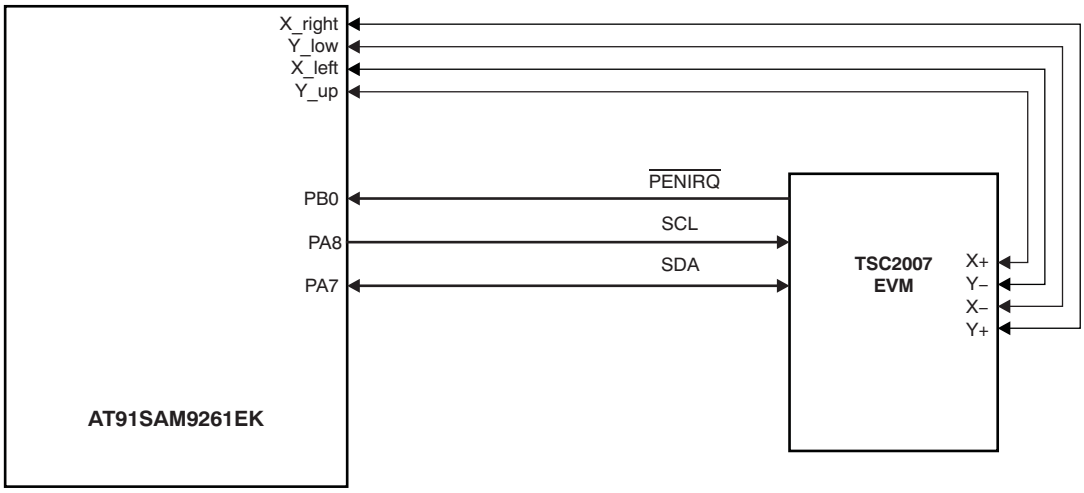

Figure 2. TSC2007 Connection Diagram

The Atmel AT91SAM9261EK board has an onboard ADS7843 touch-screen controller. The digital and analog lines to the touch-screen controller device were isolated to provide the TSC2007 evaluation module (EVM) with the X+, Y+, X–, and Y– lines. The 3.3V power supply and ground required for the TSC2007 EVM were also provided by the onboard 3.3V source and ground.

2.2 Interface Details: Digital Interface

Table 1 summarizes the AT91SAM9261EK and TSC2007 digital hardware interface.

Table 1. AT91SAM9261EK and TSC2007 Digital Hardware Interface

Signal/Input	Host Processor Pin	TSC2007 Pin on EVM
TWCK (I^2C clock)	PA8 / Pin 45	SCL
TWD (I^2C data)	PA7 / Pin 44	SDA
$\overline{\text{PENIRQ}}$	PB0 /	$\overline{\text{PENIRQ}}$

2.3 Interface Details: I^2C Interface

The I^2C bus on the TSC2007 is the primary hardware interface to the host processor. The host processor issues commands to the TSC2007 controller through the I^2C interface.

2.3.1 I^2C Driver

The TSC2007 I^2C driver establishes the software interface between the processor and the TSC2007 device. It contains three files:

- TSC2007_core.c
- TSC2007.h
- TSC2007_platform.c

The third file, TSC2007_platform.c, is processor-dependent.

The fundamental routines for the I²C driver and platform routines are summarized in Table 2 and Table 3, respectively.

Table 2. I²C Driver Routines

Routine Name	Function
TSC2007_handle_penirq	The Interrupt handler function which is invoked when the PENIRQ line goes low. This routine is the Linux top half; therefore, a bottom-half handler must also be called.
TSC2007_detect	The detect function checks for the I²C functionalities of the underlying I²C master driver, registers the client driver, sets up the controller mode and then registers the driver as an input driver, and requests for the IRQ by invoking the tsc2007_register driver.
TSC2007_timer	The bottom-half handler for the Linux Interrupt handler, which performs I²C transactions to read X, Y, Z_1, and Z_2 coordinates from the controller and report to the input subsystem.

Table 3. Platform Routines

Routine Name	Function
TSC2007_detect_irq	To set up the IRQ line that is connected to the TSC2007. 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	This function returns the current line status of the PENIRQ 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 TSC2007 Touch Driver

3.1 Driver Code Organization

Figure 3 illustrates the TSC2007 touch driver code structure.

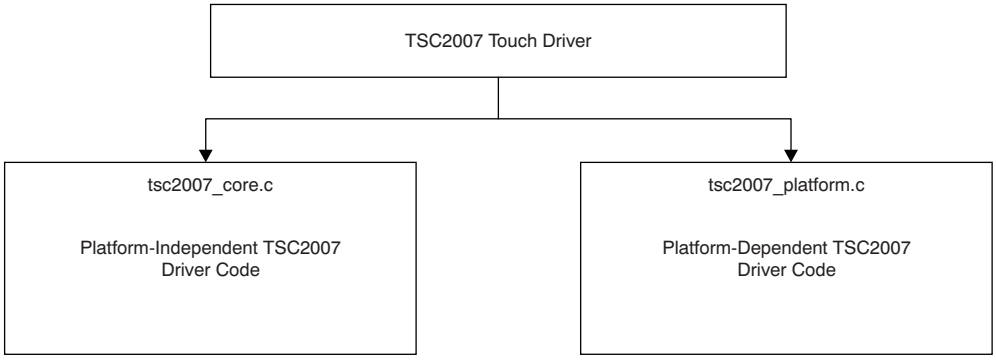

Figure 3. TSC2007 Touch Driver Code Organization

The TSC2007 touch driver is an I^{2}C client driver that is a kernel module in Linux. The entry point of this driver is *TSC2007_module_init*. This routine is called upon the insertion of the module. This driver makes use of the I^{2}C subsystem, which provides kernel APIs in order to read from and write to an I^{2}C slave device.

The kernel APIs provided are:

- i2c_master_send
- i2c_master_recv

On module initialization, the driver is registered as an I^{2}C client driver through i2c_add_driver API. On detection of an I^{2}C slave device from the I^{2}C subsystem, an i2c_detect function (tsc2007_detect) is called.

3.2 Initialization and Code Routines

3.2.1 Touch Driver Initialization

Figure 4 shows the TSC2007 touch driver initialization process. The touch driver initializes when the function *tsc2007_init* is called while loading the driver module. This function adds the I^{2}C driver using *i2c_add_driver*; this operation, in turn, calls the *tsc2007_detect* function, which is registered to be called upon detection of an I^{2}C slave device.

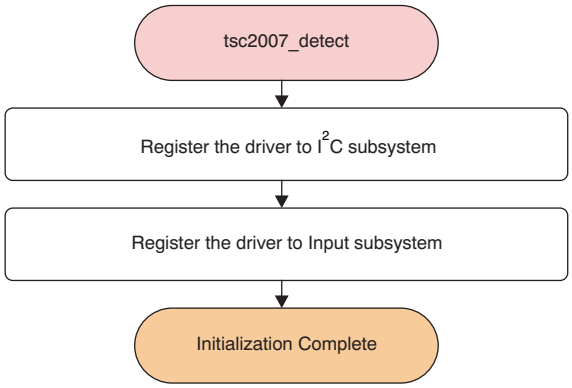

Figure 4. TSC2007 Touch Driver Initialization

The registration of the driver into the input subsystem is done by calling *tsc2007_register_driver* from the tsc2007_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 = tsc2007_register_driver(data)
```

This code fragment shows initialization of the driver as both an I^{2}C client and as an input device. As an option, the tsc2007_driver may also validate all the commands by invoking the *tsc2007_read* function for each command.

3.2.2 Interrupt Routine

On an interrupt request, or IRQ (interfaced through a GPIO pin connected to the $\overline{\text{PENIRQ}}$ line from the TSC controller), the routine *tsc2007_handle_penirq* is called, which checks for line state, starts a timer if the line is down, reads the data, and reports touch coordinates to the input subsystem.

```
static irqreturn_t tsc2007_handle_penirq(int irq, void *v)
{
struct tsc2007_data *data = (struct tsc2007_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 TSC2007 controller. The following functions are then called to read the touch data. Received valid touch data are then reported to the input subsystem.

```
/* Try to read Y values */
stat = i2c_master_send (client, &measure_y, BYTES_TO_TX);
if (unlikely (stat != BYTES_TO_TX))
alert_tsc ( "unable to send Y read command: %s\n", __FUNCTION__);
stat = i2c_master_recv (client,(u8*)&y_value, BYTES_TO_RX);
if (unlikely (stat != BYTES_TO_RX))
alert_tsc ("unable to receive data: %s\n", __FUNCTION__);

/*Try to read the X values */
stat = i2c_master_send (client, &measure_x, BYTES_TO_TX);
if (unlikely (stat != BYTES_TO_TX))
alert_tsc ("unable to send X read command: %s\n", __FUNCTION__);
stat = i2c_master_recv (client,(u8*)&x_value, BYTES_TO_RX);
if (unlikely (stat != BYTES_TO_RX))
alert_tsc ("unable to receive data: %s\n", __FUNCTION__);
```

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 (such as drags or draws across the surface of the touch screen).

3.2.4 Exit/Cleanup

Exit point of this driver is the *TSC2007_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^2C client driver, during which the subsystem invokes the *tsc2007_detach* function, which frees the IRQ, deregisters 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 uImage 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 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 uImage 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 uImage
   - 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-none-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.
- 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 flashing images on the AT91SAM board a tool called SAM-BA is required. (This tool is available from the Atmel website at www.atmel.com.) The tool is driven by a USB connection from the PC to the board.

The following sequence of steps is 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).

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 AT91xxxxx 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), 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 uImage 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 uImage; bootm 22200000'

   #savenv

   On rebooting the board, the uImage 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_8	Enables 12-bit or 8-bit resolution (that is, the number of valid bits read for each coordinate)
SETUP_MAV_RIRQ	Enables the configuration options to enable or bypass the MAV filter and the R_{IRQ} internal pull-up resistance to 90kΩ or 50kΩ.
REPORT_ACTUAL_PRESSURE	Enables reporting of actual pressure using Z_1, Z_2 values
VALIDATE_ALL_COMMANDS	Validates all the TSC2007 commands

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
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless

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