Application Report
SBAA171–March 2009

TSC2008 Linux® Driver

Data Acquisition Products

ABSTRACT

This application report describes the TSC2008 touch driver for the Linux® operating system to help customers to implement designs using the TSC2008 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. The 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/SBAA171.

Contents

List of Figures

1 Description

The TSC2008 Linux driver acts as a standard input driver based on an SPI™ slave 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.

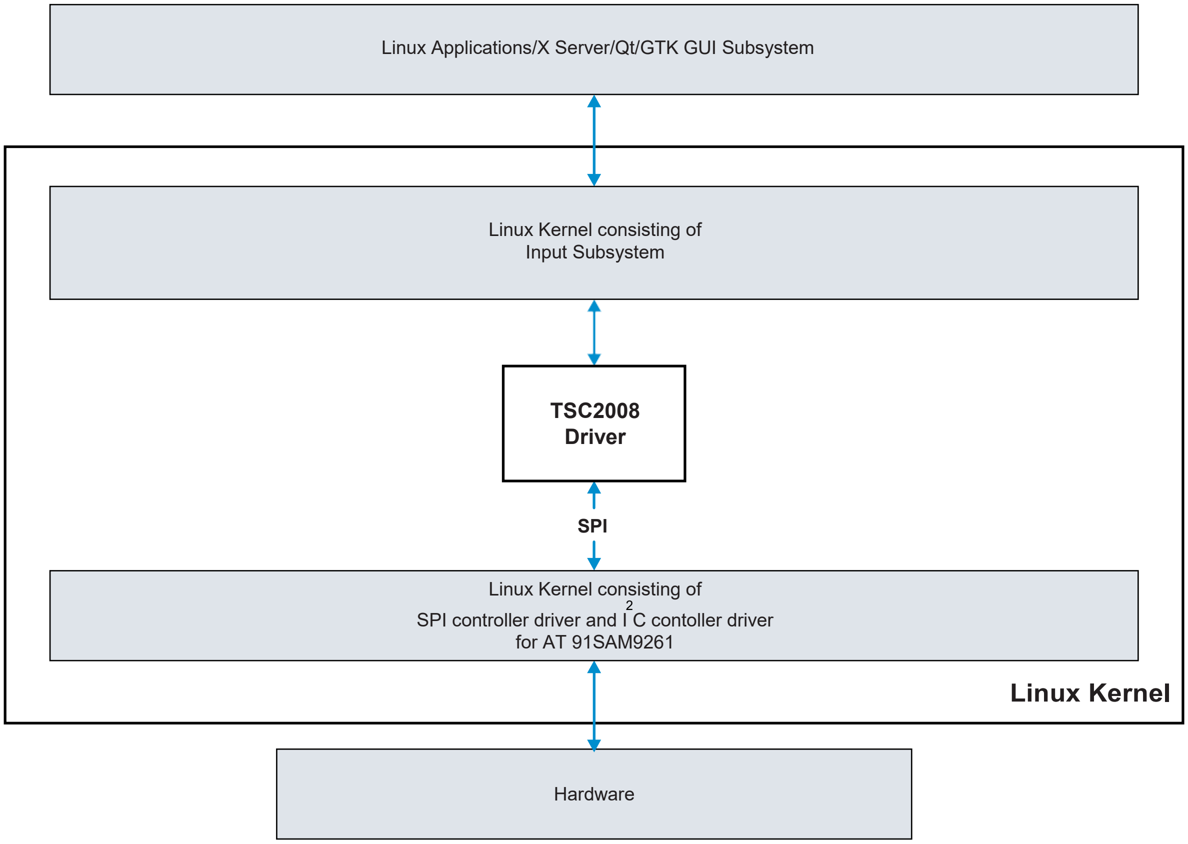
Linux Applications/X Server/Qt/GTK GUI Subsystem
Linux Kernel consisting of
Input Subsystem
TSC2008
Driver
SPI
Linux Kernel consisting of
SPI controller driver and I2C contoller driver
for AT 91SAM9261
Linux Kernel
Hardware

Figure 1. TSC2008 Linux Driver Architecture

Development of the Linux driver for the TSC2008 involves the following tasks:

- Developing a driver module that implements SPI data communications with the TSC2008.
- 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 SPI devices.

2 System Details

2.1 Architecture

Figure 2 shows a connection diagram for the TSC2008.

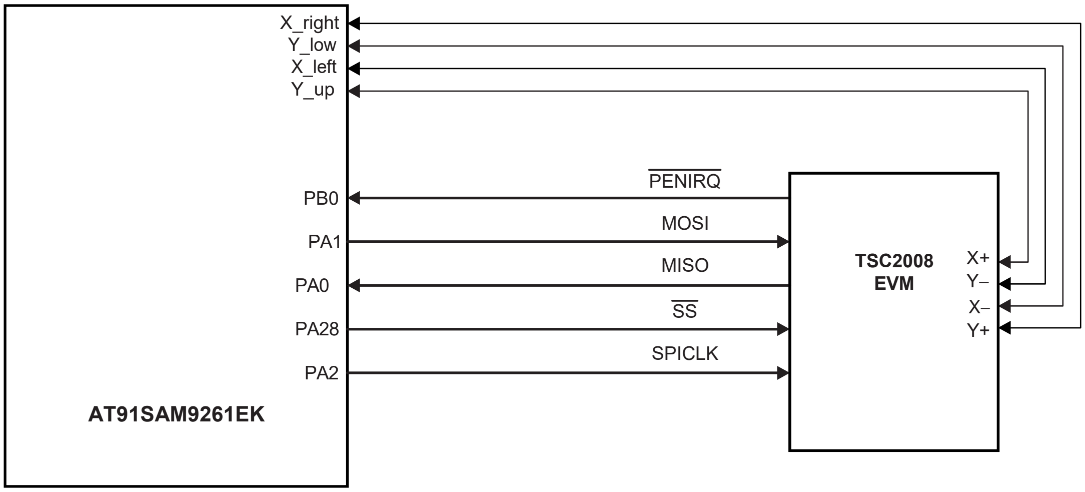

Figure 2. TSC2008 Connection Diagram

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 TSC2008 evaluation module (EVM) with the X+, Y+, X–, and Y– lines. The 3.3V power supply and ground required for the TSC2008 EVM were also provided by the onboard 3.3V source and ground.

2.2 Interface Details: Digital Interface

Table 1 summarizes the AT91SAM9261EK and TSC2008 digital hardware interface.

Table 1. AT91SAM9261EK and TSC2008 Digital Hardware Interface

Signal/Input	Host Processor Pin	TSC2008 Pin on EVM
SPI clock	PA2 / Pin 39	SPICLK
SPI MISO (Master In, Slave Out)	PA0 / Pin 37	MISO
SPI MOSI (Master Out, Slave In)	PA1 / Pin 38	MOSI
SPI /SS (Slave Select)	PA28 / Pin 65	$\overline{\text{SS}}$
$\overline{\text{PENIRQ}}$	PB0 /	$\overline{\text{PENIRQ}}$

2.3 Interface Details: Serial Peripheral Interface

The SPI bus on the TSC2008 is the primary hardware interface to the host processor. The host processor issues commands to the TSC2008 controller through the SPI.

2.3.1 SPI Driver

The TSC2008 SPI driver establishes the software interface between the processor and the TSC2008 device. It contains three files:

- TSC2008_core.c
- TSC2008.h
- TSC2008_platform.c

The third file, TSC2008_platform.c, is processor-dependent.

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

Table 2. SPI Driver Routines

Routine Name	Function
TSC2008_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.
TSC2008_probe	The probe function probes for the device, initializes the SPI slave device structures, registers the touch screen as an input device, and requests for the IRQ.
TSC2008_timer	The bottom-half handler for the Linux Interrupt handler, which performs SPI 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
TSC2008_detect_irq	To set up the IRQ line that is connected to the TSC2008. 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 TSC2008 Touch Driver

3.1 Driver Code Organization

Figure 3 illustrates the TSC2008 touch driver code structure.

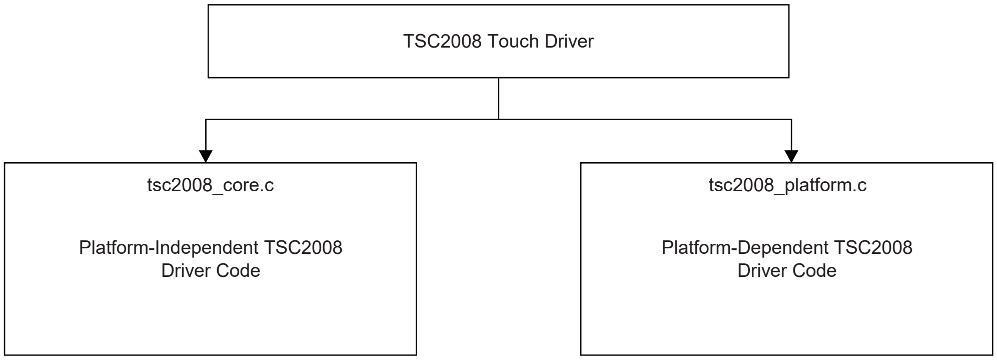
TSC2008 Touch Driver
tsc2008_core.c
Platform-Independent TSC2008 Driver Code
tsc2008_platform.c
Platform-Dependent TSC2008 Driver Code

Figure 3. TSC2008 Touch Driver Code Organization

The TSC2008 touch driver is a SPI slave driver that is a kernel module in Linux. The entry point of this driver is *TSC2008_module_init*. This routine is called upon the insertion of the module. The TSC2008 touch driver architecture adheres to the SPI protocol and the input subsystem of the Linux kernel.

On insertion of the module TSC2008_init function is invoked, which registers TSC2008_driver as an SPI slave device. If a matching entry is found in the list of SPI slave devices found, the TSC2008_probe function is called, which is the probe function invoked by the SPI subsystem, this function registers the TSC2008_driver as an input driver and performs the relevant initialization.

```
input_dev->name = "TI TSC2008 TouchScreen Controller";
err = input_register_device(input_dev);
if (err) {
goto err_free_mem;
}
```

The probe function also sets up the SPI transfers by invoking the function TSC2008_spi_setup.

```
/*
* Setup SPI transfers here
*/
if (TSC2008_spi_setup(ts) != 0) {
/*
* Setup SPI transfers here
*/
if (TSC2008_spi_setup(ts) != 0) {
}
```

The probe function also requests for the IRQ, which is a GPIO pin connected to the PENIRQ line from the TSC controller.

```
/*
* Request IRQ here,
* ts->penirq should hold the IRQ upon which the TSC shall
* Interrupt
*/
ts->penirq = TSC2008_detect_irq();
if (request_irq(ts->penirq, TSC2008_irq, IRQF_TRIGGER_FALLING,
spi->dev.driver->name, ts)) {
dev_dbg(&spi->dev, "irq %d busy?\n", spi->irq);
err = -EBUSY;
goto err_free_irq;
}
return 0;
```

3.2 *Initialization and Code Routines*

3.2.1 Touch Driver Initialization

Figure 4 shows the TSC2008 touch driver initialization process.

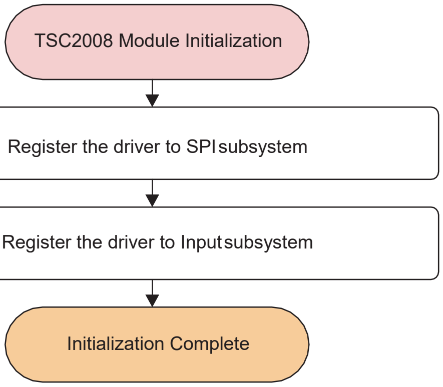
TSC2008 Module Initialization
Register the driver to SPI subsystem
Register the driver to Input subsystem
Initialization Complete

Figure 4. TSC2008 Touch Driver Initialization

3.2.2 Interrupt Routine

On an Interrupt, the *TSC2008_irq* function routine 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 TSC2008_irq(int irq, void *handle)
{
struct TSC2008 *tsc = handle;
struct input_dev *ip = tsc->input_dev;
unsigned long flags = 0;
spin_lock_irqsave(&tsc->lock, flags);
disable_irq(irq);
if (likely(!(tsc->get_pendown_state()))) {
mod_timer(&tsc->timer, jiffies + ON_IRQ /*HZ*1 */ );
} else {
input_report_key(ip, BTN_TOUCH, 0);
input_report_abs(ip, ABS_PRESSURE, 0);
input_sync(ip);
enable_irq(irq);
}
spin_unlock_irqrestore(&tsc->lock, flags);
return IRQ_HANDLED;
}
```

3.2.3 Reading Touch Data

Upon an IRQ, the read command is sent to the TSC2008 controller. The following functions are then called to read the touch data. Additionally, a timer is started in order to report drags and drawings. This reporting is done by submitting the SPI transactions that were setup during the initialization.

```
static void tsc2008_timer (unsigned long arg)
{
struct tsc2008_data *data = (struct tsc2008_data*)arg;
int stat;
spin_lock_irq (&data->lock);
/*Check whether pen has gone up*/
if ( !pen_down_state (data->pen_irq) )
{
input_report_abs (data->idev,ABS_PRESSURE,0);
input_report_key (data->idev,BTN_TOUCH,0);
input_sync (data->idev);
enable_irq (data->pen_irq);
}
else
{
data->msg_idx = 0;
stat = spi_async (data->spi,&data->msg[0]);
if (stat)
dev_err (&data->spi->dev,"spi_async----> %d\n",stat);
}
spin_unlock_irq (&data->lock);
return;
}
```

Valid touch data are reported to the input subsystem upon receipt.

The routines responsible for the above activities are:

- TSC2008_complete: To report the received data
- tsc2008_again_submit: To receive the touch data and submit the next SPI transaction.

3.2.4 Reporting Coordinates

In the TSC2008_complete function, the data received are byte-swapped and modified according to the LCD coordinates before being reported to the input subsystem.

```
/*
* Restart Timer if pendown, else enable_irq which was disabled in
* the IRQ handler
*/
if (!ts->get_pendown_state())
{
input_report_abs(ip, ABS_X, x);
input_report_abs(ip, ABS_Y, y);
#ifdef report_actual_pressure
input_report_abs(ip, ABS_PRESSURE, touch_pressure);
#else
input_report_abs(ip, ABS_PRESSURE, 7500);
#endif /* report_actual_pressure */
input_report_key(ip, BTN_TOUCH, 1);
input_sync(ip);
mod_timer(&ts->timer, jiffies + BETWEEN_READS);
} else {
input_report_abs(ip, ABS_PRESSURE, 0);
input_report_key(ip, BTN_TOUCH, 0);
input_sync(ip);
enable_irq(ts->penirq);
}
```

3.2.5 Exit/Cleanup

The exit point of this driver is the *TSC2008_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 SPI slave driver, during which the subsystem invokes the *TSC2008_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 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)
DISABLE_MAV_FILTER	Disable or enable the MAV filter
USE_90K_RIRQ	The driver configures the TSC to use a 90kΩ R_{IRQ} pull-up resistor or a default 51kΩ resistor

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) or other applicable terms available either on 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