Application Report
SBAA166–March 2009

TSC2006 WinCE® 5.0 Driver

Data Acquisition Products

ABSTRACT

The TSC2006 Microsoft® Windows® CE (WinCE) 5.0 touch driver has been developed with an SPI™ control interface; the code has been tested on a Samsung® SC32442 application processor. This application report discusses the TSC2006 driver, including the hardware connection between the TSC2006 and the platform, the WinCE 5.0 driver code structure, and the installation.

Contents

1 Introduction

The TSC2006 WinCE 5.0 driver was developed for helping users of the TSC2006 touch screen controller device from Texas Instruments to quickly set up, run, and use the device and to shorten software driver development time. The TSC2006 touch driver was coded on the standard WinCE touch device driver platform-dependent device (PDD) layer; the PDD layer was further split to have an additional processor-dependent layer (PDL) to make the TSC2006 driver easy to port into different host processors. See TI application report SLAA187 for additional details on both PDD and PDL. The driver was developed and tested using a TSC2006EVM board and the Samsung SMDK platform with an SC32442 application processor.

2 Connections

The TSC2006 device must be wired and connected to a host processor to which the device driver code is ported and executed. In developing the TSC2006 drivers for this application, the TI TSC2006EVM board and the Samsung platform with the SC32442A application processor were used.

The host processor controls the TSC2006 through an interface that consists of five digital signals:

- The four-wire SPI bus: MISO, MOSI, $\overline{SS}$, and SCLK;
- The touch pen-down and/or data ready interrupt, $\overline{PINTDAV}$.

See Figure 1 for the connections between the TSC2006 device and the SMDK2442 applications processor.

On the TSC2006EVM board, a connector to J2 was made in order to wire the five digital signals MISO, MOSI, $\overline{SS}$, SCLK, and $\overline{PINTDAV}$. For details on J2 and other aspects of the TSC2006EVM, see the TSC2006EVM User Guide.

On the Samsung SMDK2442 platform, the original touch module connected on the SMDK2442 main board was removed and replaced with the connections as shown in Figure 1. See Reference 4 and other relevant Samsung documentation for additional information about the Samsung SMDK2442 platform.

In addition to the five digital signal pins, the TSC2006 touch panel input signals, X+, X–, Y+ and Y–, are connected to the corresponding pins on the Samsung SMDK2442 touch panel, as Figure 1 indicates.

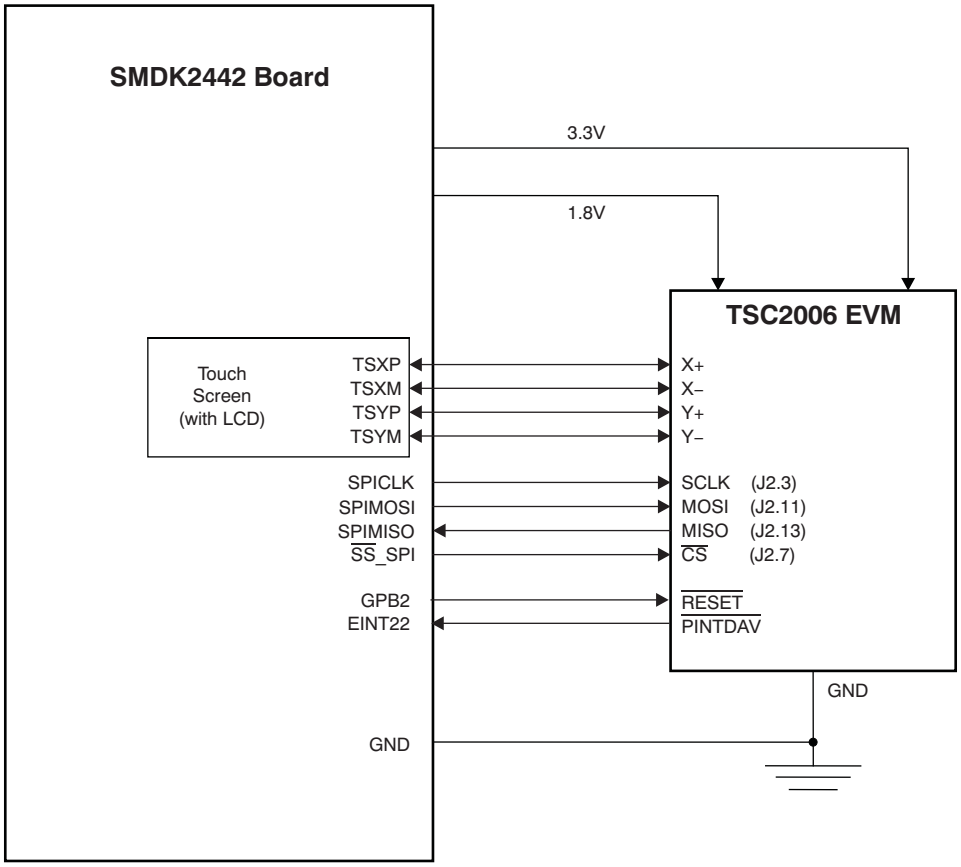

Figure 1. TSC2006 Connection to SC32442 Application Processor

3 Device Driver

Figure 2 shows the details of the TSC2006 touch device driver code file structure.

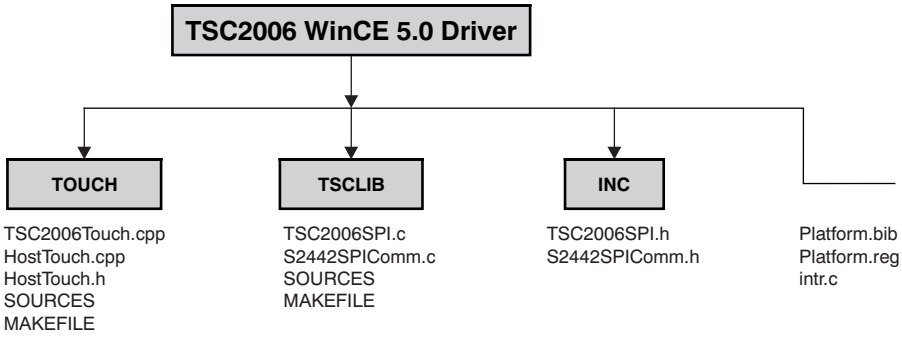

Figure 2. TSC2006 WinCE 5.0 Driver Files with SPI Control Interface

3.1 Serial Peripheral Interface

The SPI bus is the control and data bus through which the host processor sends address and control commands to the TSC2006 and reads the touch screen coordinates or other data back from the device. The SPI communication code was developed as a library and is available in the directory TSCLIB.

The host configuration for the SPI is summarized in Table 1.

Table 1. Host Configuration for SPI

Signal/Input	GPIO Pin	SPI Function
1	GPE13	SPICLK
2	GPE12	SPIMOSI0
3	GPE11	SPIMISO0
4	GPG2	$\overline{SS}$_SPI

On the hardware side, there are four TSC2006 SPI bus pins. On the software side, the Samsung SC32442, SPI, and clock management control registers are set up to communicate with the TSC2006 through the SPI. This setup is implemented at the routine, *HWInitSPI( )*. This method also includes the configuration of the TSC2006 configuration registers and the initialization routine for the TSC2006. The routine *HWInitSPI( )* is available in the file *S2442SPIComm.C*.

```
///////
// Function: HWInitSPI
// Purpose:  Setup and Initialize S3C2442 SPI Channel0 Module
///////
void HWInitSPI(BOOL InPowerHandle)
{
    int i=0,Value=0;


//    GPIO PIN Configuration    //


// enable SPI unit clock (the clock should be enabled first)
//g_pClockRegs->CLKCON |=  S3C_CLK_EN_SPI;
    g_pClockRegs->CLKCON |=  (0x1 << 18);

// set up GPE
g_pGPIORegs->GPEDN = (g_pGPIORegs->GPEDN & ~(7<<11))|(1<<13)|(0x1 << 12) | (0x1 << 11);
g_pGPIORegs->GPECON = ((g_pGPIORegs->GPECON&0xf03fffff)
| (0x2 << 26)         //Configure GPE13=> SPICLK ,
       | (0x2 << 24)         //GPE12=>SPIMOSI0
       | (0x2 << 22) );         //GPE11=>SPIMISO0    */



// set up GPG
    g_pGPIORegs->GPGCON=((g_pGPIORegs->GPGCON&0xffffffcf)|0x10); // Master(GPIO_Output)
    g_pGPIORegs->GPGDAT &= ~(0x1<<2); // De-Activate nSS



//    SPI Module Configuration  //
//Configure Rate Prescaler Register (SPPRE0).
    g_pSPIRegs->SPPRE0 = 0x00;     //if PCLK=50Mhz,SPICLK=25Mhz

//Configure SPI CONTROL REGISTER (SPCON0)
g_pSPIRegs->SPCON0 = (0<<6)|(0<<5)        //SMOD = 00     (SPI Mode Select=Polling)
|(1<<4)    //ENSCK  = 1    (Enable SPI Clock)
    |(1<<3)    //MSTR = 1     (Master Mode)
    |(0<<2)    //CPOL =0    (Clock Polarity Select )
    |(1<<1)    //CPHA=0     (Clock Phase Select )
    |(0<<0);    //TAGD=0     (normal)                        *??????*
```

```
    //SPI PIN CONTROL REGISTER (SPPIN0)
g_pSPIRegs->SPPIN0 = (0<<2)
//ENMUL =0    (Disable Multi Master error detect )
|(0<<1)    //CSn=0     (Deactivate CSn)
    |(0<<0);   //KEEP=0

HWStartFrame();

//Writing 0xff  10 times
for(i=0;i<10;i++)
{

    while(!(g_pSPIRegs->SPSTA0 & 0x1));               //Check for Rx Ready
    g_pSPIRegs->SPTDAT0=0xff;
    Value=g_pSPIRegs->SPRDAT0;
    HWSPIWait(1);
}
HWStopFrame();

return;
}
```

TSC device-specific initialization is present in the *TSC2006SPI.C* routine. Here is a snippet of the initialization code.

```
///////
// Function: BOOL InitSPI(BOOL bInPowerHandler)
// Purpose:  Initialize Processor for SPI Interface.
/////////
BOOL InitSPI(BOOL bInPowerHandler)
{

int i=0,Value=0,V[3];

// Allocate SPI Control Resources.
if (!HWAllocateSPIResources( ))
    return(FALSE);

// Setup SPI Interface at Host
HWInitSPI(bInPowerHandler);

//Software Reset
HWSPIConvertFunction(CON_FN_SW_RESET);
HWSPIConvertFunction(CON_FN_STOP);

if(TSC_Controlled_Mode == 1 )
{

HWSPIWrite(CTRL_BYTE_WRITE|CFR0_ADDR,
(CFR0_TSC_CONV | CFR0_STS_NRML |CFR0_RM_12BIT | CFR0_CLK_2MHz |
 CFR0_PVS_5mS| CFR0_PRE_1044|CFR0_SNS_2080| CFR0_LSM_ON),
 0);    //Configure CFR0

HWSPIWrite(CTRL_BYTE_WRITE|CFR1_ADDR, (CFR1_BTD_2mS),0);//Configure CFR1

HWSPIWrite(CTRL_BYTE_WRITE|CFR2_ADDR,
        (CFR2_DAV1 | CFR2_ZONE_DIS | CFR2_MAVE_DIS)
        ,0);    //Configure CFR2


//PSM=1 start conversion function 0001 (read xy ).

HWSPIConvertFunction(CON_FN_12_BIT | XY_SCAN_FN);
}
else
{
//Configure CFR0 Register
HWSPIWrite(CTRL_BYTE_WRITE | CFR0_ADDR,
(CFR0_HOST_CONV| CFR0_STS_NRML |CFR0_RM_12BIT |             CFR0_CLK_2MHz | CFR0_PVS_1mS  |
CFR0_DTW_ON), 0);
```

```
//Configure CFR1 Register
HWSPIWrite(CTRL_BYTE_WRITE | CFR1_ADDR, CFR1_BTD_2mS, 0);

//Configure CFR2 Register
HWSPIWrite(CTRL_BYTE_WRITE | CFR2_ADDR, (CFR2_PENIRQ | CFR2_MAVE_XYZ), 0);
}
return(TRUE);
}
```

Two other important SPI interface routines are the *HWSPIWrite( )* and *HWSPIRead( )*. These routines allow the S3C2442 to control the TSC2006, performing touch data acquisition and reading the data back from the TSC2006. The complete SPI write and read transmission routines are defined in the TSC2006 product data sheet (Reference 1).

```
///////
// Function: HWSPIWrite Routine
// Purpose:  This routine allows the SMDK2442 to write to TSC2006
//           control register(s) using SPI bus.
///////
BOOL HWSPIWrite(UINT8 CtrlByte,  UINT16 RegValue, BOOL InPowerHandle)
{


    UINT8 TxByte, MSB, LSB, i;
    UINT16 temp;




    if (!InPowerHandle)
    {

        HWStartFrame();

        while(!(g_pSPIRegs->SPSTA0 & 0x1));        //Send the Register Address
        g_pSPIRegs->SPTDAT0=CtrlByte;

        Delay(10);

        temp =  RegValue >> 8 ;
        MSB = (UINT8) temp;
        while(!(g_pSPIRegs->SPSTA0 & 0x1));        //Send the MSB
        g_pSPIRegs->SPTDAT0=MSB;

        Delay(10);


           LSB = 0x00;   //clear the variable
           LSB = (UINT8)( RegValue & 0x00ff);

            while(!(g_pSPIRegs->SPSTA0 & 0x1));    //Send the LSB
            g_pSPIRegs->SPTDAT0=LSB;

        HWStopFrame();
    }
    return (TRUE);
}
```

```
///////
// Function: HWSPIRead Routine
// Purpose:  This routine allows the SMDK2442 to read from TSC2006
//           control register(s) using SPI bus.
///////
UINT16 HWSPIRead(UINT8 CtrlByte,BOOL InPowerHandle)
{


    UINT8  MSB, LSB, Dummy;
    UINT16 finalValue;



    if (!InPowerHandle)
    {
        HWStartFrame();

        while(!(g_pSPIRegs->SPSTA0 & 0x1));                                  //Send the Control
Byte to read the register
        g_pSPIRegs->SPTDAT0=CtrlByte;


        Delay(1);

        while(!(g_pSPIRegs->SPSTA0 & 0x1));
        //Read Dummy data
        g_pSPIRegs->SPTDAT0=0x00;
        Dummy=g_pSPIRegs->SPRDAT0;

        Delay(1);

        while(!(g_pSPIRegs->SPSTA0 & 0x1));                                  //Read LSB
        g_pSPIRegs->SPTDAT0=0x00;
        MSB=g_pSPIRegs->SPRDAT0;


        Delay(1);

        while(!(g_pSPIRegs->SPSTA0 & 0x1));
        //Read LSB
        g_pSPIRegs->SPTDAT0=0x00;
        LSB=g_pSPIRegs->SPRDAT0;

        Delay(1);


        finalValue= (UINT16) MSB<<8;
        finalValue |= (UINT16) LSB;
        HWStopFrame();

        return finalValue;

    }
    else
    {
        RETAILMSG(DebugMsg, (TEXT("HW Tx Error...\r\n")));
        return(FALSE);
    }



}
```

3.2 Touch Device Driver

In the Samsung SMDK2442 system, the interrupt $\overline{\text{PINTDAV}}$ pin has been connected to the external interrupt EINT22, where the $\overline{\text{PINTDAV}}$ is fed to the S3C2442 GPG14 (J2.2) pin. The touch device driver is in the directory TOUCH, developed on the PDD layer of the standard touch screen device driver structure.

In the TSC2006 touch driver, the TSC2006 $\overline{\text{PINTDAV}}$ is enabled to detect any touch on the screen. $\overline{\text{PINTDAV}}$ triggers the *DdsiTouchPanelGetPoint( )* routine on the PDD layer whenever $\overline{\text{PINTDAV}}$ becomes active, as shown here:

```
///////
// DDSI Implementation
//
// @func void | DdsiTouchPanelGetPoint |
//Returns the most recently acquired point and its associated tip state
//        information.
//
// @parm PDDSI_TOUCHPANEL_TIPSTATE | pTipState |
//Pointer to where the tip state information will be returned.
// @parm PLONG | pUnCalX |
//Pointer to where the x coordinate will be returned.
// @parm PLONG | pUnCalY |
//Pointer to where the y coordinate will be returned.
//
// @comm
//Implemented in the PDD.
///////
void DdsiTouchPanelGetPoint(TOUCH_PANEL_SAMPLE_FLAGS *pTipStateFlags,
                            INT *pUncalX, INT *pUncalY)
{
    static INT PrevX=0;
    static INT PrevY=0;
    static bool fPenDown = TRUE;

    if(g_pIORegs->EINTPEND & (1<<22))
    {
        g_pIORegs->EINTPEND = (1<<22);
        fPenDown = TRUE;
    }
    else
        fPenDown = FALSE;

    if (g_pINTregs->INTMSK & (1<<IRQ_TIMER3))
    {
        if(fPenDown)
        {
            if (!GetCoordinate(&PrevX, &PrevY))
                *pTipStateFlags = TouchSampleIgnore;
            else
            {
                TransCoordinate(&PrevX, &PrevY);
                *pTipStateFlags = TouchSampleValidFlag
| TouchSampleDownFlag;
                *pTipStateFlags &= ~TouchSampleIgnore;

                *pUncalX = PrevX;
                *pUncalY = PrevY;

                TouchTimerStart();

            }
            InterruptDone(gIntrTouchChanged);
        }
        else
        {
            *pTipStateFlags = TouchSampleValidFlag;

            *pUncalX = PrevX;
            *pUncalY = PrevY;
```

```
            TouchTimerStop();
            g_pIORegs->EINTPEND = (1<<22);
            g_pIORegs->EINTMASK &= ~(1<<22);

            InterruptDone(gIntrTouch);
            InterruptDone(gIntrTouchChanged);
        }

    }
    else
    {
        if (!GetCoordinate(&PrevX, &PrevY))
            *pTipStateFlags = TouchSampleIgnore;
        else
            TransCoordinate(&PrevX, &PrevY);

        *pTipStateFlags = TouchSampleValidFlag;
        *pTipStateFlags |= TouchSampleIgnore;

        *pUncalX = PrevX;
        *pUncalY = PrevY;

        *pTipStateFlags |= TouchSampleDownFlag;

        if (g_pINTregs->INTMSK & (1<<IRQ_TIMER3))
            InterruptDone(gIntrTouchChanged);

        TouchTimerStart();
        InterruptDone(gIntrTouch);
    }
}
```

4 Installation

This section presents the installation steps required to run the TSC2006 WinCE5.0 drivers on the Samsung SMDK2442 platform. The SC32442 application processor BSP can be obtained from Samsung and should be installed. After the application processor BSP installation, it will be located on your PC in a standard location within the WinCE directory; for example, at C:\WinCE500\PLATFORM\ as *SMDK2442*.

To install the TSC2006 WinCE 5.0 driver into one of the SMDK2442 workspace, execute the following steps.

Step 1. Install the SMDK2442 BSP by following the procedures provided with the BSP.

Step 2. Replace the **Touch** folder in \WINCE500\PLATFORM\SMDK2442\Src\Drivers with the folder given in TSC200x\SMDK2442\Src\Drivers\.

Step 3. Copy the **TSCLIB** folder from TSC200x\SMDK2442\Src\Drivers\ to \WINCE500\PLATFORM\SMDK2442\Src\Drivers\.

Step 4. Edit the **Dirs** file located at \WINCE500\PLATFORM\SMDK2442\Src\Drivers\, by adding the entry *TSCLIB*\ just before *Touch*\.

Step 5. Copy the header files from TSC200x\SMDK2442\Src\Inc\ file to the \WINCE500\PLATFORM\SMDK2442\Src\Inc\ file.

Step 6. Replace the **intr.c** file in \WINCE500\PLATFORM\SMDK2442\Src\Common\Intr\ with the intr.c file present in TSC200x\SMDK2442\Src\Common\Intr\.

Step 7. Replace this code:

```
; @CESYSGEN IF CE_MODULES_POINTER
;IF BSP_NOTOUCH !
touch.dll $(_FLATRELEASEDIR)\s3c2440a_touch.dll NK SH
;ENDIF BSP_NOTOUCH !
; @CESYSGEN ENDIF CE_MODULES_POINTER
```

in the Platform.bib with the stub provided in the Platform.bib file at TSC200x\SMDK2442\Files\.

Step 8. In the \WINCE500\PLATFORM\SMDK2442\Files\Platform.reg file, replace the following code:

```
; @CESYSGEN IF CE_MODULES_POINTER
IF BSP_NOTOUCH !
[HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\TOUCH]
"MaxCalError"=dword:7
; portrait
"CalibrationData"="476,602 123,99 123,1097 830,1102 828,96 "
; "CalibrationData"="466,627 137,165 146,1098 805,1088 799,159 "
; "CalibrationData"="471,605 115,105 132,1101 825,1108 822,102 "
; "CalibrationData"="466,623 149,163 145,1101 798,1097 792,160 "
; "CalibrationData"="500,512 762,268 758,760 244,758 241,266 "
; Landscape
; "CalibrationData"="515,503 763,748 258,749 269,255 764,255 "
ENDIF BSP_NOTOUCH !
; @CESYSGEN ENDIF CE_MODULES_POINTER
```

with the stub present in the \SMDK2442\Files\platform.reg file.

5 References

The following documents are available for download through the Texas Instruments web site (www.ti.com), except where noted.

- TSC2006: Nano-power touch screen controller with SPI interface. Product data sheet SBAS415.
- Chammings, Y. and Fang, W. (2003.) TSC2301 WinCE Generic Drivers. Application report SLAA187.
- TSC2006EVM and TSC2006EVM-PDK. User's guide SLAU200.
- Samsung SC32442A Processor Developer's Kit. User guide. Available at www.samsung.com.

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