Application Report
SLAA292–February 2006

# TSC2100 WinCE 5.0 Drivers

*Wendy X. Fang* *DAP Group*

**ABSTRACT**

This application report discusses the TSC2100 touch screen and audio WinCE 5.0 drivers, running on an Intel™ PXA27x processor. The associated driver code was tested with a Texas Instruments TSC2100 evaluation module (EVM) on an Intel MainStone II platform.

**Contents**



**List of Figures**



## 1 Introduction

TSC2100 Windows™ CE 4.x drivers (SLAA198) have been upgraded under the Windows CE 5.0 operating system. By comparing the WinCE 5.0 drivers with the previous touch and audio drivers presented in the application report SLAA198, it can be seen that the principles are exactly the same, the hardware connections and the software driver code are similar, but the installation is different. This application note discusses only these differences.

The driver code was run and tested on a Texas Instruments TSC2100EVM evaluation module board (SLAU100) and an Intel™ MainStone II platform with the PXA270 Step B0 processor (see Reference 3).

## 2 Hardware Connections

Figure 1 shows the hardware connections between TSC2100 and the PXA27x processor and MainStone II platform. This illustration also shows two sets of digital serial interface buses : the SPI bus includes the SCLK, $\overline{SS}$, MOSI, and MISO lines, which is the control and touch data interface; the $I^2S$ bus includes the BCLK, WCLK, SDIN, and SDOUT, which is the audio data interface.

Note the directions of these bus lines. The TSC2100 is always an SPI slave device, whereas the host processor is the SPI master. The TSC2100 can be either an $I^2S$ master or a slave but, because PXA27x's $I^2S$ port can be used only as the master, the TSC2100 works in its $I^2S$ slave mode as is shown in Figure 1.

In addition to the connection lines displayed in Figure 1, two more digital pins of TSC2100 also can be connected to the host processor. The TSC2100 power-down control pin, $\overline{\text{PWD}}$, can be connected to one of the GPIO pins of the host, if desired. The TSC2100 hardware reset pin, $\overline{\text{RESET}}$, can be routed to the system $\overline{\text{RESET}}$ or a GPIO of the host processor. In this application report, both pins were pulled high.

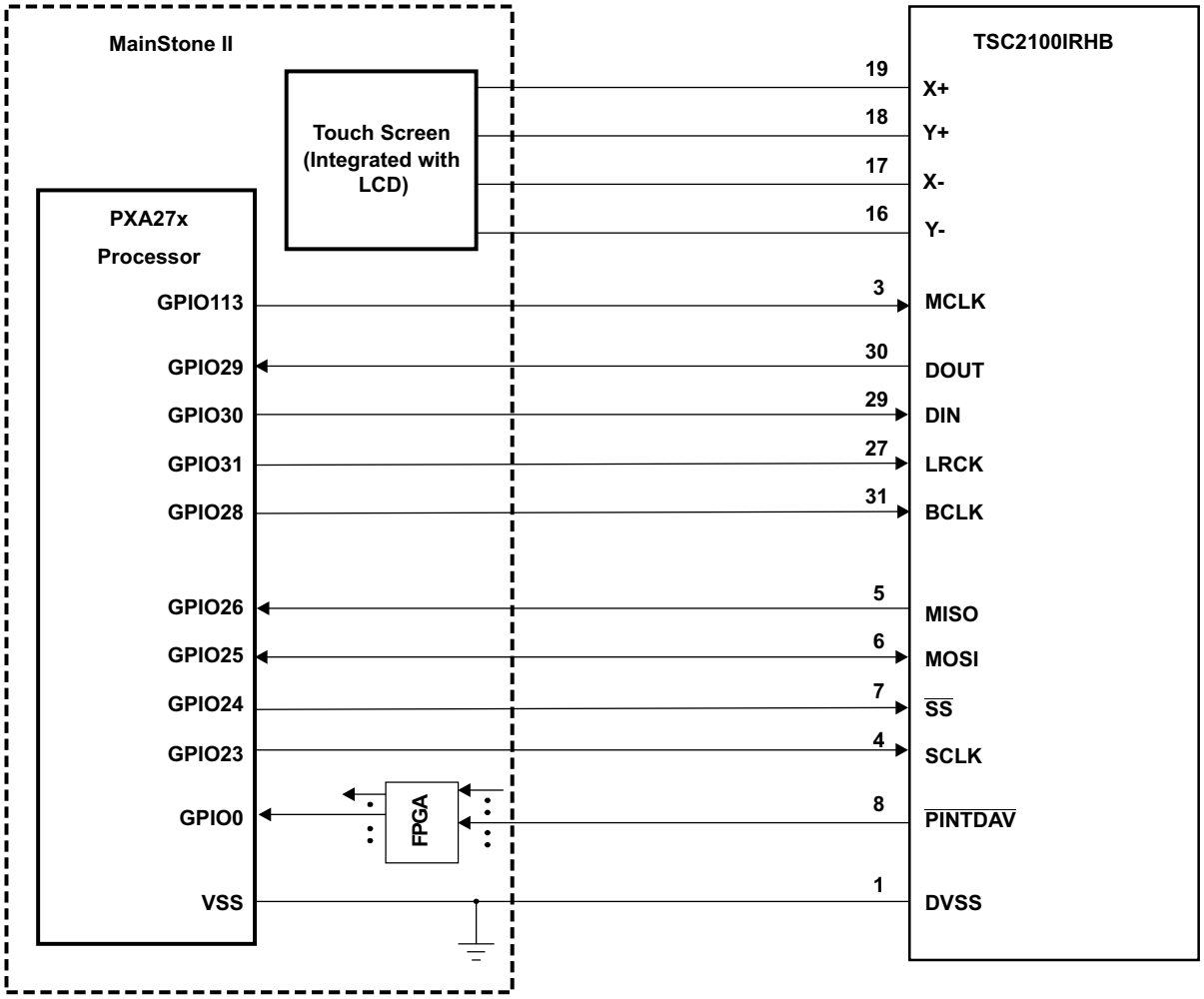


**Figure 1. TSC2100 Connections to MainStone II System**

In developing this application report, a TSC2100EVM board (SLAU100) was used, wired, and physically connected to the MainStone II platform.

On the TSC2100EVM board, the USB I2S, USB MCLK, and USB SPI of the SW1 were turned off so that the external connections from the host processor could be attached and interfaced to the TSC2100 device. See the SLAU100 user's guide for the schematic and other details of the EVM system.

On the MainStone II system, the original touch/audio module was removed and connections were replaced with those shown in Figure 1. See Reference 3 and other Intel documentation for further information of the MainStone II Platform.

## 3 Devices Drivers

Figure 2 lists the TSC2100 touch and audio device drivers' code files, where the files starting with *Host*… are the processor-dependent code or PDL, such as HostTouch.CPP or HostSPIComm.H.

## 3.1 TSC2100 Control Registers

TSC2100 has the touch and audio control registers located on its internal memory space page1 and page 2, respectively. The header file, TSC2100Regs.H, defines these registers and their bits based on the TSC2100 data sheet (SLAS387) and provided to be used by the drivers described in Figure 2.

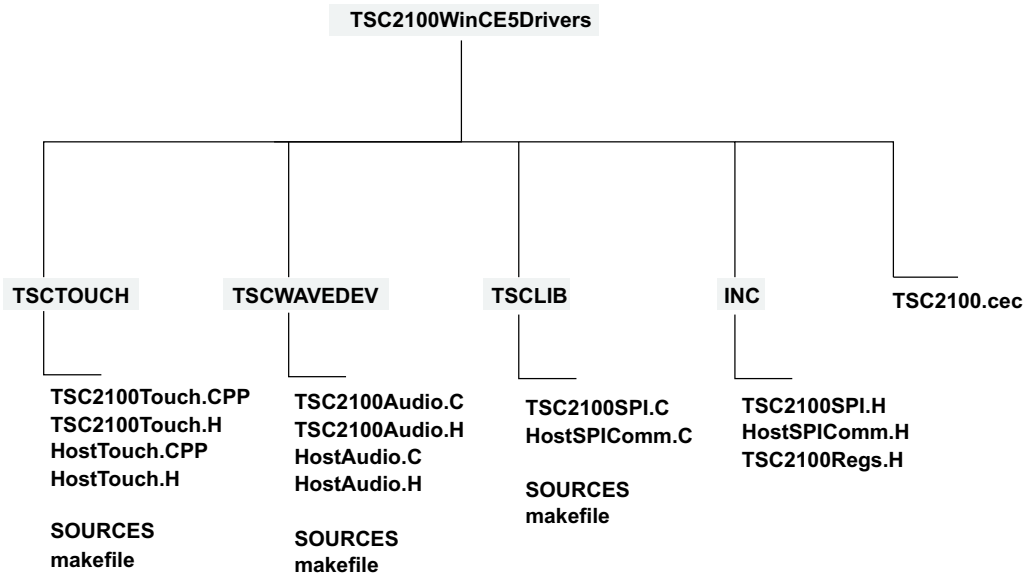


**Figure 2. TSC2100WinCE5 Drivers**

TSC2100 has the touch and audio control registers, located on its internal memory space page 1 and page 2, respectively. The header file, TSC2100Regs.H, defines these registers and their bits based on the TSC2100 data sheet (SLAS387) and provided to be used by the drivers, which can be found in the appendix of this application note.

## 3.2 SPI Interface Driver

The TSC2100 SPI driver is the key for the host processor to access both data (on page 0) and control registers (on page 1 and page 2) of the TSC2100. The driver code is located at the TSCLIB.

The host processor, PXA270, is the SPI master, and its synchronous serial port SSP1 was configured as the SPI master; the configuration was done using the routine HWSetupSPI():

```
//
//-----------------------------------------------------
// Function: void HWSetupSPI(BOOL InPowerHandle)
// Purpose:  This function must be called from the power handler of the respective drivers
//           using this library. This function will configure the GPIO pins according to
//           the functionality shown in the table below
//
//           Signals          Pin#          Direction       Alternate Function
//           SSPSCLK          GP23          output                  2
//           SSPSFRM          GP24          output                  2
//           SSPTXD           GP25          output                  2
//           SSPRXD           GP26          input                   1
//           SSPSCLKEN        GP27          input                   2
//-----------------------------------------------------
void HWSetupSPI(BOOL InPowerHandle)
{
     RETAILMSG(1,(TEXT("Setup Host GPIO & SSP for an SPI Interface... \r\n")));
     // disable Unit clock
     g_pClockRegs->cken &= ~XLLP_CLKEN_SSP1;
     // disable SSP1
     g_pSSPRegs->sscr0 &= ~SSE_ENABLE;
```

```
    // Set up the GPIO24=SFRM = 1 (GPSR0)
    g_pGPIORegs->GPSR0 |= ( GPIO_24_SFRM );
    // Program direction of the GPIOs (GPDR0)
    // (GPIO23/24/25 as outputs and GPIO26 as input)
    g_pGPIORegs->GPDR0 |= GPIO_23_SCLK;
    g_pGPIORegs->GPDR0 |= GPIO_24_SFRM;
    g_pGPIORegs->GPDR0 |= GPIO_25_MOSI;
    g_pGPIORegs->GPDR0 &= ~GPIO_26_MISO;
    // Program GPIO alternate function register (GAFR0_U)
    g_pGPIORegs->GAFR0_U &= 0xFFC03FFF;
    g_pGPIORegs->GAFR0_U |= GPIO_23_AF2_SSPSCLK;
    g_pGPIORegs->GAFR0_U |= GPIO_25_AF2_SSPTXD;
    g_pGPIORegs->GAFR0_U |= GPIO_26_AF1_SSPRXD;
    // Set up SSP registers (when disabled SSP)
    // set up SSP control register 0 and 1
    g_pSSPRegs->sscr0 = (SCR_590_KHZ | SSE_DISABLE |
                         ECS_INTERNAL |  FRF_MOTOROLA | DSS_16_BIT );
    g_pSSPRegs->sscr1 = (RFT_SEVEN |TFT_ZERO |
                         MWDS_16_BIT | SPH_HALF_DELAY |
                         SPO_IDLE_LOW |  LBM_DISABLE |
                         TIE_DISABLE | RIE_DISABLE );
    // Enable SSP last
    g_pSSPRegs->sscr0 |= SSE_ENABLE;
    // enable SPI1 Unit clock
    g_pClockRegs->cken |= XLLP_CLKEN_SSP1;
    // enable SPI1 Unit clock
    g_pClockRegs->cken |= XLLP_CLKEN_SSP1;
}
```

### 3.3 TSC2100 Touch Screen Driver

The touch screen driver handles and controls the TSC2100 touch screen function, which is located in the directory, TSCTOUCH, and was built on the WinCE standard device driver framework *Touch*.

Two major tasks of the touch driver are to set up/initialize the TSC2100 touch function; the routine is called InitTSC2100(); and reads the touch data from TSC2100.

```
// Initialize TSC2100 Touch Screen Registers for
// Normal X/Y TouchScreen Operation
void InitTSC2100Touch(BOOL bInPowerHandler)
{
     RETAILMSG(1, (TEXT("InitTSC2100Touch.\r\n")));
     TSC2100WriteReg(TSC2100_STATUS, STATUS_INT_DAV, bInPowerHandler);
     TSC2100WriteReg(TSC2100_REF, REF_SETUP_VALUE, bInPowerHandler);
     TSC2100WriteReg(TSC2100_CFG, CFG_SETUP_VALUE, bInPowerHandler);
     TSC2100WriteReg(TSC2100_ADC, ADC_SETUP_VALUE, bInPowerHandler);
     RETAILMSG(1, (TEXT("Done InitTSC2100Touch.\r\n")));
}
```

The setup values to TSC2100 touch control registers in the preceding routine was defined in the header file TSC2100Regs.H, shown in the appendix of this application report.

Whenever the panel is touched, the TSC2100's $\overline{\text{PINTDAV}}$ pin, which was programmed into its *data available* or $\overline{\text{DAV}}$ mode, sends the host processor an interrupt after the new X and Y data have been converted, averaged, and put into the corresponding data registers.

Then, the following routine is called to read back the touch X and Y data:

```
// Sample touch data
TOUCH_PANEL_SAMPLE_FLAGS PDDSampleTouchScreen(INT *x,INT *y)
{
              UINT16 iReadX, iReadY;
              TOUCH_PANEL_SAMPLE_FLAGS TmpStateFlags = TouchSampleDownFlag;
              // read X and Y coord.
              TSC2100ReadXY(&iReadX, &iReadY, FALSE);
              RETAILMSG(1,(TEXT("TSC2100 samples - X=%d, Y=%d\r\n"),iReadX, iReadY));
              // check to ensure that the point is within 12 bit bounds
              if ( ((iReadX < MAX_X_DIGITIZER_COORD) && (iReadX >= MIN_X_DIGITIZER_COORD)) &&
              ((iReadY < MAX_Y_DIGITIZER_COORD) && (iReadY >= MIN_Y_DIGITIZER_COORD)))
              {
                       *x = (INT)(iReadX);
                       *y = (INT)(iReadY);
                       TmpStateFlags |= TouchSampleValidFlag;
              }
              else
              {
                       TmpStateFlags |= TouchSampleIgnore;
                       RETAILMSG(1,(TEXT("TSC2100 samples error: X/Y data outside range!\r\n")));
              }
              return(TmpStateFlags);
}
```

## 3.4 TSC2100 Audio Driver

The TSC2100 audio function is controlled by the TSC2100 audio driver, TSCWAVEDEV, and was built on the WinCE standard audio driver framework *WaveDev*.

TSC2100 Audio driver needs to use two serial buses: the SPI bus for writing/reading the TSC2100 audio control registers and the $I^2S$ bus for audio data streaming. Section 3.2 discusses the SPI interface; this section discusses the $I^2S$ setup.

PAX270's $I^2S$ interface can only be used as the $I^2S$ master; the setup can be found at the routine HWEnableI2S():

```
//
//-----------------------------------------------------------------------
// Function: HWEnableI2S()
//-----------------------------------------------------------------------
//
void HWEnableI2S(void)
{
       RETAILMSG(1,(TEXT("Setup Host GPIO & I2S Interface... \r\n")));
       //Basic Outline:
       // configure the GPIO registers and set to I2S mode
       // Set up I2S control registers at default condition
       // insert reset for I2S
       v_pI2SRegs->sacr0 |= 0x00000008;
// un-insert the reset
       v_pI2SRegs->sacr0 = 0x00007700;
// disable I2S unit clock
       v_pClockRegs->cken &= ~XLLP_CLKEN_I2S;
// setup GPIO direction regs
       v_pGPIORegs->GPDR0 |= XLLP_GPIO_BIT_I2SBITCLK     |
                             XLLP_GPIO_BIT_I2S_SYNC      |
                             XLLP_GPIO_BIT_I2S_SDATA_OUT;
       v_pGPIORegs->GPDR0 &= ~XLLP_GPIO_BIT_I2S_SDATA_IN;
       v_pGPIORegs->GPDR3 |= XLLP_GPIO_BIT_I2S_SYSCLK;     // SYSCLK as output
// configure GPIO alternate function regs
       v_pGPIORegs->GAFR0_U &= 0x00FFFFFF;
       v_pGPIORegs->GAFR0_U |= XLLP_GPIO_AF_BIT_I2SBITCLK_OUT |
```

```
                                XLLP_GPIO_AF_BIT_I2S_SDATA_IN  |
                                XLLP_GPIO_AF_BIT_I2S_SDATA_OUT |
                                XLLP_GPIO_AF_BIT_I2S_SYNC;
        v_pGPIORegs->GAFR0_U &= ~XLLP_GPIO_AF_BIT_I2S_SYSCLK_MASK;
        v_pGPIORegs->GAFR3_U |= XLLP_GPIO_AF_BIT_I2S_SYSCLK;
// configure I2S reg sacr0 but not enable I2S yet
      v_pI2SRegs->sacr0 = 0x00001104;
// configure system for I2S mode
      v_pI2SRegs->sacr1 = 0x00000000;
// configure clock divider
       v_pI2SRegs->sadiv = I2SRATE_44_1;    // divider for 44.1kHz audio
// enable I2S
       v_pI2SRegs->sacr0 |= 0x00000001;
// enable Unit clock
      v_pClockRegs->cken |= XLLP_CLKEN_I2S;
//            DumpRegsI2S();
              return ;
}
```

To program the TSC2100's audio function, this application report initially set up the following routine InitTSC2100Audio:

```
//   Initialize TSC2100 Audio Register at Default
     void InitTSC2100Audio(BOOL bInPowerHandler)
{
     RETAILMSG(1, (TEXT("InitTSC2100Audio.\r\n")));
     TSC2100WriteReg(TSC2100_AUDCTL1, AUDCTL1_SETUP_VALUE, bInPowerHandler);
     TSC2100WriteReg(TSC2100_ADCVOL, ADCVOL_SETUP_VALUE, bInPowerHandler);
     TSC2100WriteReg(TSC2100_DACVOL, DACVOL_SETUP_VALUE, bInPowerHandler);
     TSC2100WriteReg(TSC2100_BPVOL, BPVOL_SETUP_VALUE, bInPowerHandler);
     TSC2100WriteReg(TSC2100_AUDCTL2, AUDCTL2_SETUP_VALUE, bInPowerHandler);
     TSC2100WriteReg(TSC2100_AUDCTL3, AUDCTL3_SETUP_VALUE, bInPowerHandler);
     TSC2100WriteReg(TSC2100_PLL1, PLL1_SETUP_VALUE, bInPowerHandler);
     TSC2100WriteReg(TSC2100_PLL2, PLL2_SETUP_VALUE, bInPowerHandler);
     TSC2100WriteReg(TSC2100_AUDCTL4, AUDCTL4_SETUP_VALUE, bInPowerHandler);
     TSC2100WriteReg(TSC2100_AUDCTL5, AUDCTL5_SETUP_VALUE, bInPowerHandler);
     TSC2100WriteReg(TSC2100_AUDPD, AUDPD_SETUP_VALUE, bInPowerHandler);
     RETAILMSG(1, (TEXT("Done InitTSC2100Audio.\r\n")));
}
```

The same as the touch function initialization, the setup values for audio in the preceding routine also can be found at the header file "TSC2100Regs.H", in the Appendix.

During the preceding initialization, the TSC2100 ADC and DAC were not powered up or unmuted. The power up and unmute for the audio path is done when recording or playing back. Some requests are on the audio paths' power up/down; for more details, see SLAA230.

## 4 Installation

The following procedure serves as an example of how to install the TSC2100 drivers on the MainStone II platform .

**Step I: Copy –**

- Copy \TSC2100WinCE5Drivers\TSC2100.cec file into:
  - C:\WINCE500\PUBLIC\COMMON\OAK\CATALOG\CEC\
- Copy all files inside \TSC2100WinCE5Drivers\INC\ into:
  - C:\WINCE500\PLATFORM\MAINSTONEII\SRC\INC\

- Copy the directories TSCLIB, TSCTouch, and TSCWaveDev into:
  - C:\WINCE500\PLATFORM\MAINSTONEII\SRC\DRIVERS\

**Step II: Set Up –**

This step sets up the catalog to include the TSC2100 device drivers.

Run *Platform Builder 5.0* , and the Platform Builder IDE appears.

- At the Platform Builder 5.0 IDE, open **Manage Catalog Items** from the menu **File\Manage Catalog Items** …**\.** When the **Manage Catalog Items** window appears, click on the **Import** button on the right side of the window, navigate, find, and select **TSC2100.cec** in the directory:
  - C:\WINCE500\PUBLIC\COMMON\OAK\CATALOG\CEC\,
- and then click on **Open** so that the item is ported in.
- Click and drag to select all *.cec files in the **Manage Catalog Items** window, and then click on the **Refresh** button to ensure that the new item is loaded.
- Close the **Manage Catalog Items** window by clicking on its **OK** button.

**Step III: Open –**

This step opens a new or existing MainStone II workspace in the Platform Builder 5.0 IDE per the application. However, the procedure is ignored here.

**Step IV: Add –**

This step adds the TSC2100 device drivers from the **Catalog** into the existing **OS design**.

In the **Catalog** window of the **Platform Builder 5.0 IDE**, find **TI TSC2100 Touch Controller Driver**, right-click on it, and select **Add to OS Design** to add the touch controller driver to the OS.

Similarly, find **TI TSC2100 Audio CODEC Driver**, right-click on it, and select **Add to OS Design** to add the audio driver to the OS.

As a result, both device drivers should appear under the **Device Drivers** section at the **OSDesignView** window of the WorkSpace.

**Step V: Modify –**

This step modifies the building device drivers so as to include the TI TSC2100 drivers.

- Open the **dirs** file in the directory:
  - C:\WINCE500\PLATFORM\MAINSTONEII\SRC\DRIVERS\
- Eliminate the original **touch** from the list, and add on the **TSCLIB, TSCTOUCH** and **TSCWAVEDEV**. For example, the **dirs** file could be:

```
DIRS=\
   TSCLIB \
   TSCTOUCH \
   TSCWAVEDEV \
# @CESYSGEN IF CE_MODULES_POINTER
#   touch \
# @CESYSGEN ENDIF CE_MODULES_POINTER
# @CESYSGEN IF CE_MODULES_DEVICE
# @CESYSGEN IF CE_MODULES_USBD
  hcd   \
# @CESYSGEN ENDIF CE_MODULES_USBD
# @CESYSGEN IF CE_MODULES_SERIAL
  serial \
# @CESYSGEN ENDIF CE_MODULES_SERIAL
......
......
```

- Save and close the modified **dirs** file.

**Step VI: Update –**

This step updates the Hardware Specific Files, so that the OS uses TSC2100 device drivers.

- Open the existing **platform.reg** file from **Hardware Specific** section of the **ParameterView** window of the workspace.

- Edit the **platform.reg** file such that the old audio **dll** is deleted and the new **dll** is added into the TSC2100 audio:

```
; ------------------------------------------------------------------------------
; @CESYSGEN IF CE_MODULES_WAVEAPI
IF BSP_NOAUDIO !
[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\WaveDev]
   "Prefix"="WAV"
;  "Dll"="pxa27x_wavedev.dll"
   "Dll"="wavedev.dll"
   "Index"=dword:1
   "Order"=dword:0
   "Priority256"=dword:95
   "Sysintr"=dword:19
. . .
```

- Save and close the updated **platform.reg** file.
- Similarly, edit the **platform.bib** file so that:

```
; ------------------------------------------------------------------------------
; @CESYSGEN IF CE_MODULES_WAVEAPI
IF BSP_NOAUDIO !
;  pxa27x_wavedev.dll     $(_FLATRELEASEDIR)\pxa27x_wavedev.dll    NK  SH
   wavedev.dll     $(_FLATRELEASEDIR)\wavedev.dll    NK  SH
ENDIF BSP_NOAUDIO !
; @CESYSGEN ENDIF CE_MODULES_WAVEAPI
; ------------------------------------------------------------------------------
```

- Save and close the updated **platform.bib** file.

**Step VII: Change –**

This step changes one secondary interrupt of the GPIO0 from the AC97 to $\overline{\text{PENIRQ}}$ (TSC2100 $\overline{\text{PINTDAV}}$).

- At the menu **File\Open**…, navigate, find, and open the software code file **intr.c** inside the directory:
  - C:\WINCE500\PLAFORM\MAINSTONEII\SRC\KERNEL\OAL\
- Change the line in the **BSPIntrInit()** routine from:
  - OALIntrStaticTranslate(SYSINTR_TOUCH, IRQ_GPIO0_UCB1400)
    to:
  - OALIntrStaticTranslate(SYSINTR_TOUCH, IRQ_GPIO0_PENIRQ);
- Save and close the **intr.c** code file.

## 5 WinCE 5.0 TSC2100 Driver Code

To obtain the WinCE 5.0 TSC2100 driver code, contact the TI DAP Application Support Group at E-mail address

dataconvapps@list.ti.com

## 6 References


1. *TSC2100 WinCE Generic Drivers* application report (SLAA198)
2. *TSC2100 EVM, Touch Screen Controller Evaluation Module* user's guide (SLAU100)
3. *Intel PXA27x Processor Developer's Kit,* Order #: 278827-005
4. *TSC2100, Programmable Touch Screen Controller with Integrated Stereo Audio CODEC and Headphone/Speaker Amplifier* data sheet (SLAS378)
5. *Programming Audio Power Up/Down on TSC210x and TLV320AIC26/28* application report (SLAA230)

## Appendix A Header File for Defining TSC2100 Registers

```
/*************************************************************/
//
// Copyright ©) Texas Instruments 2005. All rights reserved.
//
/*************************************************************
THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT
LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR
FITNESS FOR A PARTICULAR PURPOSE.
Module Name:
                    TSC2100Regs.H
Abstract:
                    This header file contains the definition of the TSC2100
                    device's control registers.
Functions:

Revision History:
Rev 0.0             Original Release    WXF    11-30-2005
*************************************************************/
#ifndef __TSC2100Regs_H__
#define __TSC2100Regs_H__
#define TSC2100_READ    0x8000
#define TSC2100_WRITE    0x0000
/*********************************
/*  TSC2100 Registers
*********************************/
// Data Registers (page0)
#define TSC2100_X        0x0000
#define TSC2100_Y        0x0020
#define TSC2100_Z1        0x0040
#define TSC2100_Z2        0x0060
#define TSC2100_BAT1     0x00A0
#define TSC2100_BAT2    0x00C0
#define TSC2100_AUX        0x00E0
#define TSC2100_TEMP1    0x0120
#define TSC2100_TEMP2    0x0140
// TouchScreen Control Registers (page1)
#define TSC2100_ADC        0x0800
#define TSC2100_STATUS    0x0820
#define TSC2100_REF        0x0860
#define TSC2100_RESET    0x0880
#define TSC2100_CFG        0x08A0
// Audio Control Registers (page2)
#define TSC2100_AUDCTL1        0x1000
#define TSC2100_ADCVOL        0x1020
#define TSC2100_DACVOL        0x1040
#define TSC2100_BPVOL        0x1060
#define TSC2100_AUDCTL2        0x1080
#define TSC2100_AUDPD        0x10A0
#define TSC2100_AUDCTL3        0x10C0
#define TSC2100_BASSLN0        0x10E0
#define TSC2100_BASSLN1        0x1100
#define TSC2100_BASSLN2        0x1120
#define TSC2100_BASSLN3        0x1140
#define TSC2100_BASSLN4        0x1160
#define TSC2100_BASSLN5        0x1180
#define TSC2100_BASSLD1        0x11A0
#define TSC2100_BASSLD2        0x11C0
#define TSC2100_BASSLD4        0x11E0
#define TSC2100_BASSLD5        0x1200
#define TSC2100_BASSRN0        0x1220
```

```
#define TSC2100_BASSRN1        0x1240
#define TSC2100_BASSRN2        0x1260
#define TSC2100_BASSRN3        0x1280
#define TSC2100_BASSRN4        0x12A0
#define TSC2100_BASSRN5        0x12C0
#define TSC2100_BASSRD1        0x12E0
#define TSC2100_BASSRD2        0x1300
#define TSC2100_BASSRD4        0x1320
#define TSC2100_BASSRD5        0x1340
#define TSC2100_PLL1        0x1360
#define TSC2100_PLL2        0x1380
#define TSC2100_AUDCTL4        0x13A0
#define TSC2100_AUDCTL5        0x13C0
/*********************************
/*  TSC2100 Register Definitions
*********************************/
/*
** Touch Screen
*/
//////////
// A/D Converter Control Register: TSC2100_ADC
//////////
// ControlMode
#define ADC_PSM_HOST    0x0000
#define ADC_PSM_TSC        0x8000
#define ADC_PSM_PENUP    0x0000
#define ADC_PSM_PENDOWN    0x8000
// ConversionControl
#define    ADC_STS_NORMAL    0x0000
#define    ADC_STS_STOP    0x4000
// A/D Function
#define ADC_AD_MASK            0x3C00
#define ADC_AD_NOSCAN        0x0000
#define    ADC_AD_XY_SCAN        0x0400
#define ADC_AD_XYZ_SCAN        0x0800
#define ADC_AD_X_SCAN        0x0C00
#define ADC_AD_Y_SCAN        0x1000
#define ADC_AD_Z_SCAN        0x1400
#define ADC_AD_BAT1_CONV    0x1800
#define ADC_AD_BAT2_CONV    0x1C00
#define ADC_AD_AUX_CONV        0x2000
#define ADC_AD_AUX_SCAN        0x2400
#define ADC_AD_TEMP1_CONV    0x2800
#define ADC_AD_PORT_SCAN    0x2C00
#define ADC_AD_TEMP2_CONV    0x3000
#define ADC_AD_X_DRIVER        0x3400
#define ADC_AD_Y_DRIVER        0x3800
#define ADC_AD_Z_DRIVER        0x3C00
// A/D Resolution Control
#define ADC_RS_8        0x0100
#define ADC_RS_10        0x0200
#define ADC_RS_12        0x0000
// A/D Convert Averaging Control
#define ADC_AV_0        0x0000
#define ADC_AV_4        0x0040
#define ADC_AV_8        0x0080
#define ADC_AV_16        0x00C0
// A/D Convert Clock Control
#define ADC_CL_8MHz        0x0000
#define ADC_CL_4MHz        0x0010
#define ADC_CL_2MHz        0x0020
#define ADC_CL_1MHz        0x0030

// Panel Voltage Stabilization Time Control
#define ADC_PV_0uS        0x0000
#define ADC_PV_100uS    0x0002
```

```
#define ADC_PV_500uS    0x0004
#define ADC_PV_1mS        0x0006
#define ADC_PV_5mS        0x0008
#define ADC_PV_10mS        0x000A
#define ADC_PV_50mS        0x000C
#define ADC_PV_100mS    0x000E
// Mean vs Median average mode
#define ADC_AVG_Mean    0x0000
#define ADC_AVG_Median  0x0001
//////////
// Status Control Register: TSC2100_STATUS
//////////
// PENIRQ/DAV pin function
#define STATUS_INT_PEN        0x0000
#define STATUS_INT_DAV        0x4000
#define STATUS_INT_BOTH        0x8000
// SAR ADC status
#define STATUS_PWRDN_DOWN    0x2000
// SAR ADC (TSC or Host) mode status
#define STATUS_HCTLM_TSC    0x1000
// SAR ADC data available status
#define STATUS_DAV_DAV        0x0800
#define STATUS_XSTAT_DAV    0x0400
#define STATUS_YSTAT_DAV    0x0200
#define STATUS_Z1STAT_DAV    0x0100
#define STATUS_Z2STAT_DAV    0x0080
#define STATUS_B1STAT_DAV    0x0040
#define STATUS_B2STAT_DAV    0x0020
#define STATUS_AXSTAT_DAV    0x0010
#define STATUS_T1STAT_DAV    0x0004
#define STATUS_T2STAT_DAV    0x0002
//////////
// Reference Control Register: TSC2100_REF
//////////
// Internal Reference Mode
#define REF_INT_EXTERNAL 0x0000
#define REF_INT_INTERNAL 0x0010
// Power-Up Delay Reference
#define REF_DL_0uS        0x0000
#define REF_DL_100uS    0x0004
#define REF_DL_500uS    0x0008
#define REF_DL_1mS        0x000C
// Power Down Reference - Ref PowerDown between Conversions
#define REF_PDN_ON        0x0000
#define REF_PDN_OFF        0x0002
// Reference Voltage Control
#define REF_RFV_125        0x0000
#define REF_RFV_250        0x0001
//////////
// Reset Control Register: TSC2100_RESET
//////////
// Reset
#define RESET_RESET        0xBB00
//////////
// Configuration Control Register: TSC2100_CFG
//////////
// Precharge Time (in uSecond)
#define CFG_PRE_20        0x0000
#define CFG_PRE_84        0x0008
#define CFG_PRE_276        0x0010
#define CFG_PRE_340        0x0018
#define CFG_PRE_1044    0x0020
#define CFG_PRE_1108    0x0028
#define CFG_PRE_1300    0x0030
#define CFG_PRE_1364    0x0038
// Sense Time (in uSecond)
```

```
#define CFG_SNS_32        0x0000
#define CFG_SNS_96        0x0001
#define CFG_SNS_544        0x0002
#define CFG_SNS_608        0x0003
#define CFG_SNS_2080    0x0004
#define CFG_SNS_2144    0x0005
#define CFG_SNS_2592    0x0006
#define CFG_SNS_2656    0x0007
/*
** Audio
*/
//////////
// Audio Control Register: TSC2100_AUDCTL1
//////////
// ADC High-Pass Filter
#define AUDCTL1_ADCHPF_DISABLED    0x0000
#define AUDCTL1_ADCHPF_00045Fs    0x4000
#define AUDCTL1_ADCHPF_00125Fs    0x8000
#define AUDCTL1_ADCHPF_0025Fs    0xC000
// ADC Input Mux
#define AUDCTL1_ADCIN_MIC        0x0000
#define AUDCTL1_ADCIN_AUX        0x1000
#define AUDCTL1_ADCIN_DIFF        0x2000
// Codec word length
#define AUDCTL1_WLEN_16Bit        0x0000
#define AUDCTL1_WLEN_20Bit        0x0400
#define AUDCTL1_WLEN_24Bit        0x0800
#define AUDCTL1_WLEN_32Bit        0x0C00
// Data format
#define AUDCTL1_DATFM_I2S        0x0000
#define AUDCTL1_DATFM_DSP        0x0100
#define AUDCTL1_DATFM_RJUST        0x0200
#define AUDCTL1_DATFM_LJUST        0x0300
// DAC Sample rate
#define AUDCTL1_DACFS_1            0x0000
#define AUDCTL1_DACFS_1_5        0x0008
#define AUDCTL1_DACFS_2            0x0010
#define AUDCTL1_DACFS_3            0x0018
#define AUDCTL1_DACFS_4            0x0020
#define AUDCTL1_DACFS_5            0x0028
#define AUDCTL1_DACFS_5_5        0x0030
#define AUDCTL1_DACFS_6            0x0038
// ADC Sample rate
#define AUDCTL1_ADCFS_1            0x0000
#define AUDCTL1_ADCFS_1_5        0x0001
#define AUDCTL1_ADCFS_2            0x0002
#define AUDCTL1_ADCFS_3            0x0003
#define AUDCTL1_ADCFS_4            0x0004
#define AUDCTL1_ADCFS_5            0x0005
#define AUDCTL1_ADCFS_5_5        0x0006
#define AUDCTL1_ADCFS_6            0x0007
//////////
// Gain Control for Headset/Aux: TSC2100_ADCVOL
//////////
// ADC Mute
#define ADCVOL_ADMUT_ACTIVE        0x0000
#define ADCVOL_ADMUT_MUTE        0x8000
// ADC PGA Settings
#define ADCVOL_ADPGA_MASK        0x7F00
// AGC Target Gain for ADC input.
#define ADCVOL_AGCTG_05_5        0x0000
#define ADCVOL_AGCTG_08            0x0020
#define ADCVOL_AGCTG_10            0x0040
#define ADCVOL_AGCTG_12            0x0060
#define ADCVOL_AGCTG_14            0x0080
#define ADCVOL_AGCTG_17            0x00a0
```

```
#define ADCVOL_AGCTG_20            0x00c0
#define ADCVOL_AGCTG_24            0x00e0
// AGC Time constant for Headset/Aux Input
#define ADCVOL_AGCTC_8_100        0x0000
#define ADCVOL_AGCTC_11_100        0x0002
#define ADCVOL_AGCTC_16_100        0x0004
#define ADCVOL_AGCTC_20_100        0x0006
#define ADCVOL_AGCTC_8_200        0x0008
#define ADCVOL_AGCTC_11_200        0x000a
#define ADCVOL_AGCTC_16_200        0x000c
#define ADCVOL_AGCTC_20_200        0x000e
#define ADCVOL_AGCTC_8_400        0x0010
#define ADCVOL_AGCTC_11_400        0x0012
#define ADCVOL_AGCTC_16_400        0x0014
#define ADCVOL_AGCTC_20_400        0x0016
#define ADCVOL_AGCTC_8_500        0x0018
#define ADCVOL_AGCTC_11_500        0x001a
#define ADCVOL_AGCTC_16_500        0x001c
#define ADCVOL_AGCTC_20_500        0x001e
// AGC Enable for Headset/Aux Input
#define ADCVOL_AGCEN_OFF        0x0000
#define ADCVOL_AGCEN_ON            0x0001
//////////
// DAC Volume Control Register: TSC2100_DACVOL
//////////
// Left DAC Mute
#define DACVOL_DALMU_ACTIVE        0x0000
#define DACVOL_DALMU_MUTE        0x8000
//////////
// Right DAC Mute
//////////
#define DACVOL_DARMU_ACTIVE        0x0000
#define DACVOL_DARMU_MUTE        0x0080
//////////
// BPVOL PGA Control Register: TSC2100_BPVOL
//////////
// Analog Sidetone Mute Control
#define BPVOL_ASTMU_ACTIVE        0x0000
#define BPVOL_ASTMU_MUTE        0x8000
// Analog sidetone gain setting
#define BPVOL_ASTG_MASK            0x7F00
#define BPVOL_ASTG_0DB            0x4500
// Digital Sidetone Mute
#define BPVOL_DSTMU_ACTIVE        0x0000
#define BPVOL_DSTMU_MUTE        0x0080
// Digital sidetone gain setting
#define BPVOL_DSTG_MASK            0x007E
#define BPVOL_DSTG_0DB            0x0000
// Analog sidetone PGA Flag
#define BPVOL_ASTGF_SOFT        0x0000
#define BPVOL_ASTGF_DONE        0x0001
//////////
// Audio Control 2 Register: TSC2100_AUDCTL2
//////////
// Keyclick enable
#define AUDCTL2_KCLEN_OFF        0x0000
#define AUDCTL2_KCLEN_ON        0x8000
// Keyclick amplitude
#define AUDCTL2_KCLAC_MASK        0x7000
#define AUDCTL2_KCLAC_MED        0x4000
// Headset/Aux of Handset PGA Soft-stepping
#define AUDCTL2_APGASS_ONE        0x0000
#define AUDCTL2_APGASS_TWO        0x0800
// Keyclick Frequency
#define AUDCTL2_KCLFRQ_MASK        0x0700
#define AUDCTL2_KCLFRQ_1KHZ        0x0400
```

```
// Keyclick length
#define AUDCTL2_KCLLN_32        0x00F0
// DAC Left/Right PGA Flag
#define AUDCTL2_DLGAF_DONE        0x0008
#define AUDCTL2_DRGAF_DONE        0x0004
// DAC Channel PGA Soft-stepping control
#define AUDCTL2_DASTC_ONE        0x0000
#define AUDCTL2_DASTC_TWO        0x0002
// ADC PGA Flag
#define AUDCTL2_ADGAF_DONE        0x0001
//////////
// Codec Power Control Register: TSC2100_AUDPD
//////////
// Audio CODEC Power Down
#define AUDPD_PWDNC_ON            0x0000
#define AUDPD_PWDNC_OFF            0x8000
// Audio Bypass/sidetone Power Down
#define AUDPD_ASTPWD_ON            0x0000
#define AUDPD_ASTPWD_OFF        0x2000
// Audio Output Driver Enable
#define AUDPD_DAODRC_OFF        0x0000
#define AUDPD_DAODRC_ON            0x1000
// Audio Bypass/sidetone Power Down Flag
#define AUDPD_ASTPWF            0x0800
// DAC power down
#define AUDPD_DAPWDN_ON            0x0000
#define AUDPD_DAPWDN_OFF        0x0400
// ADC power down
#define AUDPD_ADPWDN_ON            0x0000
#define AUDPD_ADPWDN_OFF        0x0200
// Virtual ground power down
#define AUDPD_VGPWDN_ON            0x0000
#define AUDPD_VGPWDN_OFF        0x0100
// ADC and DAC power down Flag
#define AUDPD_ADPWDF            0x0080 // ADC power down flag
#define AUDPD_DAPWDF            0x0040 // DAC power down flag
// ADWS Pin Function Select
#define AUDPD_ADWSF_HWPWDN        0x0000
#define AUDPD_ADWSF_ADCW        0x0020
// VBIAS Voltage
#define AUDPD_VBIAS_25V            0x0000
#define AUDPD_VBIAS_20V            0x0010
// Digital Audio Effects Filter control
#define AUDPD_EFFCTL_DISABLE    0x0000
#define AUDPD_EFFCTL_ENABLE        0x0002
// De-emphasis filter enable
#define AUDPD_DEEMPF_DISABLE    0x0000
#define AUDPD_DEEMPF_ENABLE        0x0001
//////////
// Audio Control 3 Register: TSC2100_AUDCTL3
//////////
// DAC Channel Master Volume Control
#define AUDCTL3_DMSVOL_INDEP    0x0000
#define AUDCTL3_DMSVOL_RIGHT    0x4000
#define AUDCTL3_DMSVOL_LEFT        0x8000
// Reference Sampling Rate
#define AUDCTL3_REFFS_48        0x0000
#define AUDCTL3_REFFS_44_1        0x2000
// Master transfer mode
#define AUDCTL3_DAXFM_CONT        0x0000
#define AUDCTL3_DAXFM_256S        0x1000
// Codec master/slave selection
#define AUDCTL3_SLVMS_SLAVE        0x0000
#define AUDCTL3_SLVMS_MASTER    0x0800
// DAC Max Output Signal Swing
#define AUDCTL3_DAPK2PK_2000    0x0000
```

```
#define AUDCTL3_DAPK2PK_2192    0x0200
#define AUDCTL3_DAPK2PK_2402    0x0400
#define AUDCTL3_DAPK2PK_2633    0x0600
// AGC Noise Threshold
#define AUDCTL3_AGCNL_60        0x0000
#define AUDCTL3_AGCNL_70        0x0010
#define AUDCTL3_AGCNL_80        0x0020
#define AUDCTL3_AGCNL_90        0x0030
// AGC clip stepping disable
#define AUDCTL3_CLPST_DISABLE    0x0000
#define AUDCTL3_CLPST_ENABLE    0x0008
#define AUDCTL3_REVID_MASK        0x0007
//////////
// PLL Program Register: TSC2100_PLL1
//////////
// PLL Enable
#define PLL1_PLLSEL_DISABLE        0x0000
#define PLL1_PLLSEL_ENABLE        0x8000
// PLL Program Q, P, I
#define PLL1_QVAL_MASK            0x7800 // Q Value
#define PLL1_PVAL_MASK            0x0700 // P Value
#define PLL1_IVAL_MASK            0x00fc // I Value
//////////
// PLL Program Register: TSC2100_PLL2
//////////
// PLL Program D
#define PLL2_DVAL_MASK          0xfffc // D Value
//////////
// Audio Control 4 Register: TSC2100_AUDCTL4
//////////
// ADC PGA Soft_Stepping Control
#define AUDCTL4_ADSTPD_ENABLE    0x0000
#define AUDCTL4_ADSTPD_DISABLE    0x8000
// DAC PGA Soft_Stepping Control
#define AUDCTL4_DASTPD_ENABLE    0x0000
#define AUDCTL4_DASTPD_DISABLE    0x4000
// Analog Sidetone PGA soft stepping
#define AUDCTL4_ASSTPD_ENABLE    0x0000
#define AUDCTL4_ASSTPD_DISABLE    0x2000
// Digital Sidetone Zero Cross Control
#define AUDCTL4_DSTPD_ENABLE    0x0000
#define AUDCTL4_DSTPD_DISABLE    0x1000
// AGC Hysteresis selection
#define AUDCTL4_AGCHYS_1        0x0000
#define AUDCTL4_AGCHYS_2        0x0200
#define AUDCTL4_AGCHYS_4        0x0400
#define AUDCTL4_AGCHYS_0        0x0600
// Short Circuit Detected Enable
#define AUDCTL4_SHCKT_Enable    0x0000
#define AUDCTL4_SHCKT_Disable    0x0100
// Short Circuit Detected Mode (auto PWDN if SC detected)
#define AUDCTL4_SHCKTPD_NOPWD    0x0000
#define AUDCTL4_SHCKTPD_AUTOPWD    0x0080
// Short circuit Detection flag
#define AUDCTL4_SHCKTF            0x0040
// DAC POP Reduction Enable
#define AUDCTL4_PRON_Disable    0x0000
#define AUDCTL4_PRON_Enable        0x0020
// DAC POP Reduction Clock Setting
#define AUDCTL4_PRCL_Fast        0x0000
#define AUDCTL4_PRCL_Slow        0x0010
// DAC POP Reduction Duration Setting
#define AUDCTL4_PRRT_Long        0x0000
#define AUDCTL4_PRRT_Median1    0x0004
#define AUDCTL4_PRRT_Median2    0x0008
#define AUDCTL4_PRRT_Short        0x000C
```

```
//***** Audio Control Register 5: TSC2100_AUDCTL5
// Max ADC PGA for AGC
//   Unsigned Integer * 0x200
// AGC Debounce Time for Detecting Noise
#define AUDCTL5_MNDB_0mS        0x0000
#define AUDCTL5_MNDB_05mS        0x0040
#define AUDCTL5_MNDB_1mS        0x0080
#define AUDCTL5_MNDB_2mS        0x00C0
#define AUDCTL5_MNDB_4mS        0x0100
#define AUDCTL5_MNDB_8mS        0x0140
#define AUDCTL5_MNDB_16mS        0x0180
#define AUDCTL5_MNDB_32mS        0x01C0
// AGC Debounce Time for Detecting Signal
#define AUDCTL5_MSDB_0mS        0x0000
#define AUDCTL5_MSDB_05mS        0x0008
#define AUDCTL5_MSDB_1mS        0x0010
#define AUDCTL5_MSDB_2mS        0x0018
#define AUDCTL5_MSDB_4mS        0x0020
#define AUDCTL5_MSDB_8mS        0x0028
#define AUDCTL5_MSDB_16mS        0x0030
#define AUDCTL5_MSDB_32mS        0x0038
// Driver POP Reduction Enable
#define AUDCTL5_DPOP_Enable        0x0000
#define AUDCTL5_DPOP_Disable    0x0004
// Driver POP Reduction Duration Setting
#define AUDCTL5_DPRT_Normal        0x0000
#define AUDCTL5_DPRT_Long        0x0002
//*************************************************************
//
//            TSC2100 Initialization Value Definitions
//
//*************************************************************
// Values for Initializing TSC2100 Touch Screen Function
#define ADC_SETUP_VALUE            (ADC_PSM_TSC | \
                                   ADC_STS_NORMAL | \
                                   ADC_AD_XY_SCAN | \
                                   ADC_RS_12 | \
                                   ADC_AV_16 | \
                                   ADC_CL_2MHz | \
                                   ADC_PV_1mS | \
                                   ADC_AVG_Median)
#define ADC_STOP_CONVERSIONS    (ADC_PSM_TSC | \
                                   ADC_STS_STOP | \
                                   ADC_AD_XY_SCAN | \
                                   ADC_RS_12 | \
                                   ADC_AV_16 | \
                                   ADC_CL_2MHz | \
                                   ADC_PV_1mS | \
                                   ADC_AVG_Median)
#define STS_SETUP_VALUE            (STATUS_INT_DAV)
#define REF_SETUP_VALUE            (REF_INT_INTERNAL | REF_DL_100uS | \
                                   REF_PDN_ON | REF_RFV_250)
#define CFG_SETUP_VALUE            (CFG_PRE_84 | CFG_SNS_608)
// Values for Initializing TSC2100 Audio Function
#define AUDCTL1_SETUP_VALUE        (AUDCTL1_ADCHPF_DISABLED | \
                                   AUDCTL1_ADCIN_MIC    | \
                                   AUDCTL1_WLEN_16Bit   | \
                                   AUDCTL1_DATFM_I2S    | \
                                   AUDCTL1_DACFS_1      | \
                                   AUDCTL1_ADCFS_1)
#define ADCVOL_SETUP_VALUE        (ADCVOL_ADMUT_MUTE  | \
                                   ADCVOL_AGCTG_05_5  | \
                                   ADCVOL_AGCTC_8_100 | \
                                   ADCVOL_AGCEN_OFF)
#define ADCVOL_UNMUTE_VALUE        (ADCVOL_ADMUT_ACTIVE | \
                                   ADCVOL_AGCTG_05_5   | \
```

```
                                    ADCVOL_AGCTC_8_100  | \
                                    ADCVOL_AGCEN_OFF)
#define DACVOL_SETUP_VALUE           (DACVOL_DALMU_MUTE | \
                                    DACVOL_DARMU_MUTE)
#define BPVOL_SETUP_VALUE           (BPVOL_ASTMU_MUTE | \
                                    BPVOL_DSTMU_MUTE | 0x4500 )
#define AUDCTL2_SETUP_VALUE           (AUDCTL2_KCLEN_ON    | \
                                    AUDCTL2_KCLAC_MED   | \
                                    AUDCTL2_APGASS_ONE  | \
                                    AUDCTL2_KCLFRQ_1KHZ | \
                                    AUDCTL2_KCLLN_32    | \
                                    AUDCTL2_DASTC_ONE)
// this is for capless mode // NOTE: the VGND is powered ON for capless mode output
#define AUDPD_SETUP_VALUE           (AUDPD_PWDNC_ON     | \
                                    AUDPD_ASTPWD_OFF   | \
                                    AUDPD_DAODRC_ON    | \
                                    AUDPD_DAPWDN_OFF   | \
                                    AUDPD_ADPWDN_OFF   | \
                                    AUDPD_VGPWDN_ON    | \
                                    AUDPD_ADWSF_HWPWDN | \
                                    AUDPD_VBIAS_25V    | \
                                    AUDPD_EFFCTL_DISABLE | \
                                    AUDPD_DEEMPF_DISABLE )
#define AUDCTL3_SETUP_VALUE           (AUDCTL3_DMSVOL_INDEP | \
                                    AUDCTL3_REFFS_44_1   | \
                                    AUDCTL3_DAXFM_256S   | \
                                    AUDCTL3_SLVMS_SLAVE  | \
                                    AUDCTL3_DAPK2PK_2000 | \
                                    AUDCTL3_AGCNL_60     | \
                                    AUDCTL3_CLPST_DISABLE)
// Using I2S/SYSCLK which is 11.343Mhz and, with Q=2, gives 44.3kHz.
// which is what the Intel processor is using for the sample rate.
#define PLL1_SETUP_VALUE           (PLL1_PLLSEL_DISABLE | \
                                    (2 << 11) | \
                                    (1 << 8) | \
                                    (8 << 2))
#define PLL2_SETUP_VALUE           (0)
#define AUDCTL4_SETUP_VALUE           (AUDCTL4_ADSTPD_ENABLE | \
                                    AUDCTL4_DASTPD_ENABLE | \
                                    AUDCTL4_ASSTPD_ENABLE | \
                                    AUDCTL4_DSTPD_ENABLE  | \
                                    AUDCTL4_AGCHYS_1      | \
                                    AUDCTL4_SHCKT_Enable  | \
                                    AUDCTL4_SHCKTPD_NOPWD | \
                                    AUDCTL4_PRON_Enable   | \
                                    AUDCTL4_PRCL_Slow     | \
                                    AUDCTL4_PRRT_Long )
#define AUDCTL5_SETUP_VALUE           (0xFE00 | \
                                    AUDCTL5_MNDB_0mS    | \
                                    AUDCTL5_MSDB_0mS    | \
                                    AUDCTL5_DPOP_Enable | \
                                    AUDCTL5_DPRT_Long )
#endif  //__TSC2100Regs_H__
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////
```

## 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 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.

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.

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 |
| DSP | dsp.ti.com | Broadband | www.ti.com/broadband |
| Interface | interface.ti.com | Digital Control | www.ti.com/digitalcontrol |
| Logic | logic.ti.com | Military | www.ti.com/military |
| Power Mgmt | power.ti.com | Optical Networking | www.ti.com/opticalnetwork |
| Microcontrollers | microcontroller.ti.com | Security | www.ti.com/security |
| | | Telephony | www.ti.com/telephony |
| | | Video & Imaging | www.ti.com/video |
| | | Wireless | www.ti.com/wireless |

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