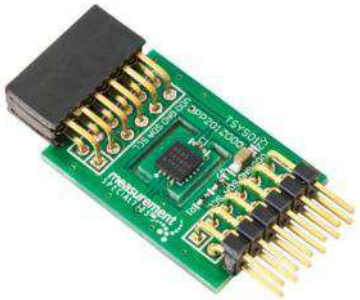


MEAS TSYS01 DIGITAL COMPONENT SENSOR (DCS) DRIVER FOR ZedBoard

Digital Temperature Sensor
Software Development Kit

Detailed example software and drivers are available that execute directly, without modification, on a number of development boards that support an integrated or synthesized microprocessor. The download contains several source files intended to accelerate customer evaluation and design. The source code is written in standard ANSI C format, and all development documentation including theory/operation, register description, and function prototypes are documented in the interface file.

Specifications

- Measures temperature from -40°C to 125°C
- I^2C communication
- Fully calibrated
- Fast response time
- Very low power consumption

Reference Material

- Detailed information regarding operation of the IC: TSYS01 Datasheet
- Detailed information regarding the Peripheral Module: TSYS01 Peripheral Module
- Complete software sensor evaluation kit for Zedboard: TSYS01_Zedboard.zip

Drivers & Software

Detailed example software and drivers are available that execute directly, without modification, on a number of development boards that support an integrated or synthesized microprocessor. The download contains several source files intended to accelerate customer evaluation and design. The source code is written in standard ANSI C format, and all development documentation including theory/operation, register description, and function prototypes are documented in the interface file.

Functions Summary

Enumerations	
enum	**tsys01_address { tsys01_i2c_address_csb_0, tsys01_i2c_address_csb_1 }**
enum	**tsys01_status { tsys01_status_ok, tsys01_status_i2c_transfer_error }**
Functions	
void	**tsys01_init (u32)** Initializes the AXI address of the AXI IIC Core and sets the default I^2C address to 0x77.
void	**tsys01_set_address (enum tsys01_address)** Sets the I^2C address of the device.
enum tsys01_status	**tsys01_reset (void)** Sends I^2C reset command to the TSYS01 device.
enum tsys01_status	**tsys01_read_prom (void)** Reads the factory calibrated coefficients for use in temperature conversion.
enum tsys01_status	**tsys01_read_temperature (float *)** Reads the temperature ADC value and computes the compensated value in degrees Celsius.

Project Setup

This project is based on a ZedBoard. The FPGA hardware and the console application will be loaded via SD card.

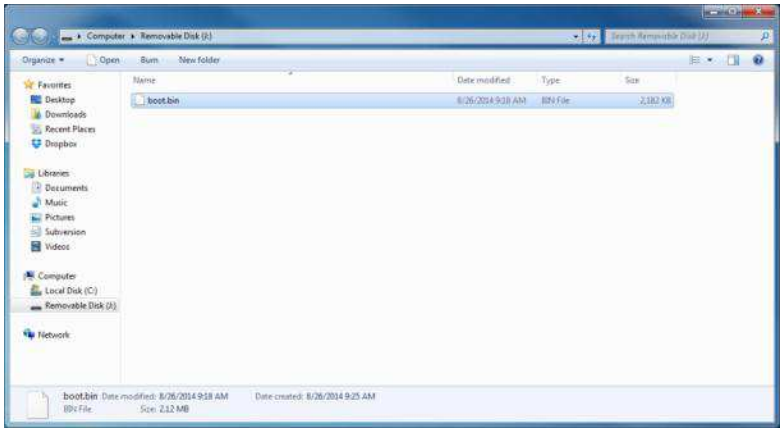
You will need:

- ZedBoard
- TSYS01 sensor for Digilent Pmod™ board
- SD card
- ZedBoard power adapter
- USB-to-MicroUSB cable for UART communications
- A computer with a card reader to write to the SD card and to host a terminal emulator

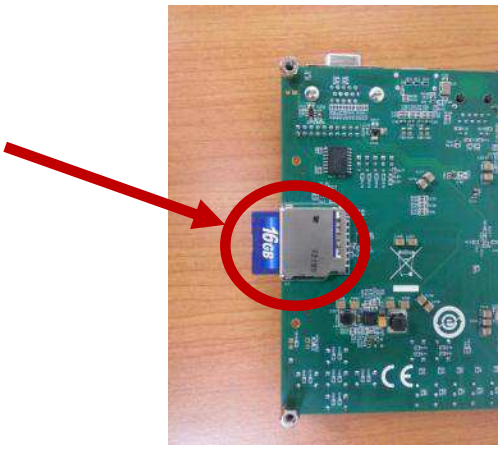
The following steps will guide you through setting up the hardware platform:

1. First, if you have not connected your computer to a ZedBoard or MicroZed device before, you will likely need to download and install the Silicon Labs CP2104 USB-to_UART driver. The setup guide for installing the driver can be found at the address below: http://www.zedboard.org/sites/default/files/documentations/CP210x_Setup_Guide_1_2.pdf

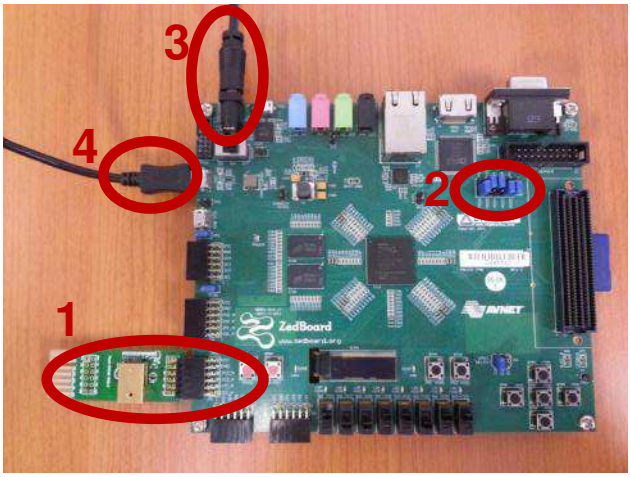
2. Next, attach the SD card to your computer via a card reader or through the built-in SD card slot. Download the “boot.bin” file that pertains to the MS5637 from the software link and copy it onto the SD card so that it is the only file present on the file system.



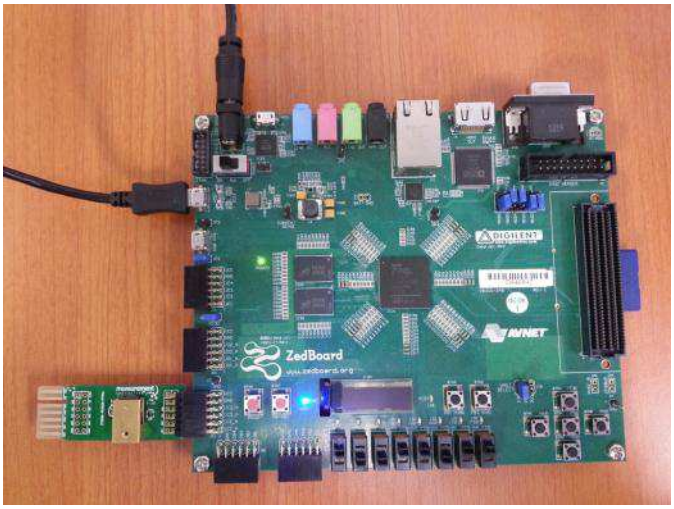
3. Safely eject the SD card from your computer. Insert the SD card into the card slot on the back of the ZedBoard.



4. Connect the TSYS01 digital pressure sensor to the “JC” Digilent Pmod™ port of the ZedBoard (1), ensure that jumpers JP7, JP8, JP9, JP10, and JP11 are configured such that the ZedBoard will boot from the SD card on start-up (2), and connect the power adapter to the barrel jack on the ZedBoard (3). Finally connect the micro-USB cable to the micro-USB port of the ZedBoard that is labeled “UART” (4). The USB cable will facilitate UART transmissions for the console application.



5. Turn on the power to the board with the switch next to the barrel jack. When the board powers up, the ZedBoard will illuminate a green power LED. After close to 30 seconds, the FPGA will be successfully programmed by the boot image on the SD card and a blue “Done” LED will illuminate on the ZedBoard. Your hardware should appear as shown below. If the board was powered on before this step, turn the power off and repeat this step.



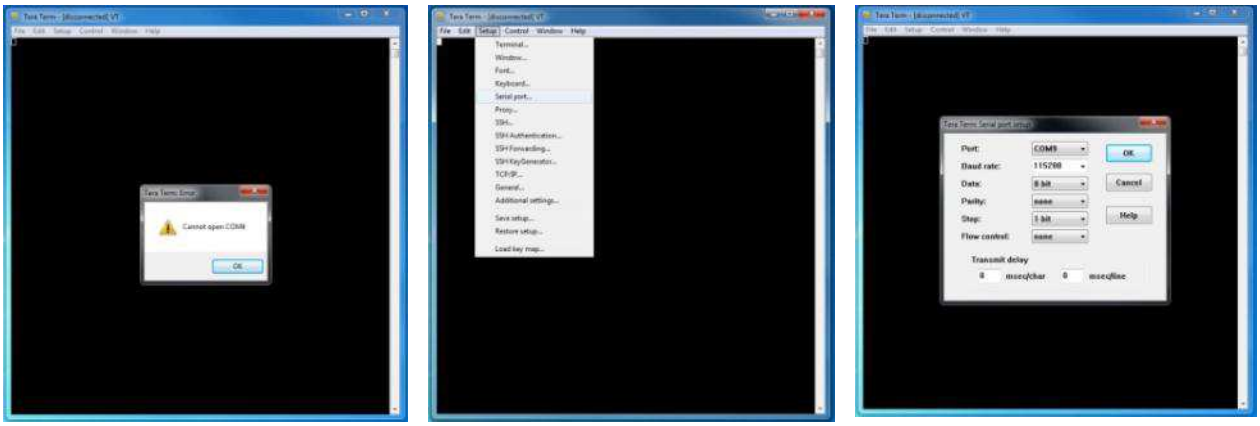
Launching the Console Application

Now that you have successfully set up your hardware platform, you are ready to run the console application.

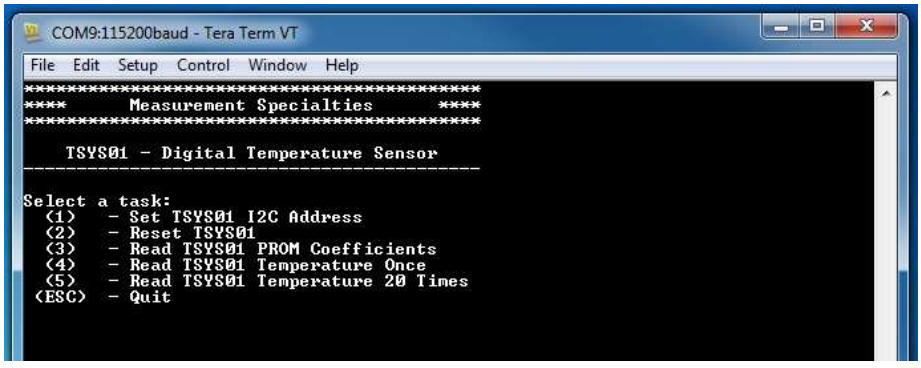
1. Upon power-on, the console application should already be running. It will be necessary to open a terminal and configure a serial connection to interact with the console application. Do this by opening tera term (which can be downloaded from http://en.sourceforge.jp/projects/ttssh2/releases/) or a similar terminal emulation software package.

2. Tera term may display an error when it starts up if it tries to connect to a COM port where no device is present. It is safe to ignore this warning, so click OK. Next, open the "Setup" menu and click the "Serial Port…" option.

3. Now select the appropriate COM port that your ZedBoard setup is connected to. If you are not sure which this is, refer to the Device Manager. Configure your serial connection with 115200 Baud, 8 bit data, no parity, 1 stop bit, and no flow control, and then click OK.



4. You should now have a live connection open to the console application running on the ZedBoard. Press enter and the console application will display the main menu from which you can perform several tasks on the MS8607 digital pressure sensor.



Running the Console Application

The console application is intended to demonstrate the required operations when using the sensor.

a. The TSYS01 software must have an I²C address set or it may not function. Do this by selecting (1) BEFORE performing any other options.

b. Second, after startup, it is a good idea to reset the sensor. This puts it in a known state. Do this by selecting (2) in the console application.

c. Each sensor is tested at the factory to determine the variation of the sensor due to fabrication. Calibration coefficients are stored in the device at that time for later use in calculating the correct output. These coefficient values must be read from the device and stored in software variables before temperature measurements can be taken. Do this by selecting (3) in the console application.

Now the sensor and the software are setup. These first three steps only need to be performed once at power up.

d. The console application option (4) reads the temperature once.

e. The console application option (5) reads the temperature 20 times at approximately two measurements per second and displays them to the screen in real time.

Application Code

This section is intended to provide a basic example of functionality.

```
/*
 * Copyright (c) 2009-2012 Xilinx, Inc.  All rights reserved.
 *
 * Xilinx, Inc.
 * XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS" AS A
 * COURTESY TO YOU.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION AS
 * ONE POSSIBLE   IMPLEMENTATION OF THIS FEATURE, APPLICATION OR
 * STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS IMPLEMENTATION
 * IS FREE FROM ANY CLAIMS OF INFRINGEMENT, AND YOU ARE RESPONSIBLE
 * FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE FOR YOUR IMPLEMENTATION.
 * XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO
 * THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO
 * ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE
 * FROM CLAIMS OF INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.
 *
 */

/*
 * MEAS_TSYS01_Main.c: Console Application for Testing the TSYS01
 *
 * This application configures UART 16550 to baud rate 9600.
 * PS7 UART (Zynq) is not initialized by this application, since
 * bootrom/bsp configures it to baud rate 115200
 *
 * ------------------------------------------------
 * | UART TYPE   BAUD RATE                        |
 * ------------------------------------------------
 *   uartns550   9600
 *   uartlite    Configurable only in HW design
 *   ps7_uart    115200 (configured by bootrom/bsp)
 */

#include <stdio.h>
#include <unistd.h>
#include "platform.h"
#include "xparameters.h"
#include "tsys01.h"

#define               XPAR_AXI_IIC_JC_BASEADDR         XPAR_IIC_0_BASEADDR

void tsys01_main_menu(void);

int main()
{

    char key_input;
    char prom_read_flag=0;
    int i;
    tsys01_status stat;
    float temperature;
```

```
//Initialize the UART
init_platform();

// Set the AXI address of the IIC core and
// initialize the i2c address to 0x77
tsys01_init(XPAR_AXI_IIC_JC_BASEADDR);

// Display the main menu
tsys01_main_menu();

// Infinite loop
while(1){

    // Get keyboard input
    read(1, (char*)&key_input, 1);

    if(key_input == '1'){              //If the '1' key is pressed

        // Display address selection menu
        printf("\n");
        printf("Select an address:\n");
        printf("  (0)   - CSB is set low  (Address=0x77)\n");
        printf("  (1)   - CSB is set high (Address=0x76)\n");

        // Get keyboard input ignoring keypresses that are not '0' or '1'
        read(1, (char*)&key_input, 1);
        while(key_input!='0' && key_input!='1'){
            read(1, (char*)&key_input, 1);
        }

        if(key_input == '0'){          // If the '0' key is pressed
            // Set i2c address to 0x77
            tsys01_set_address(tsys01_i2c_address_csb_0);
            printf("Set TSYS01 I2C Address to 0x77 (CSB low)\n");
        }else if(key_input == '1'){    // If the '1' key is pressed
            // Set i2c address to 0x76
            tsys01_set_address(tsys01_i2c_address_csb_1);
            printf("Set TSYS01 I2C Address to 0x76 (CSB high)\n");
        }

        // Wait for another key press and then display the main menu again
        printf("\nPress any key to continue...\n");
        read(1, (char*)&key_input, 1);
        tsys01_main_menu();

    }else if(key_input == '2'){        //If the '2' key is pressed

        // Send the reset command to the TSYS01
        printf("\n");
        printf("Resetting TSYS01...\n");
        stat = tsys01_reset();

        // Display the status returned from the reset operation
        printf("TSYS01 Reset Complete with status: ");
        if(stat==tsys01_status_ok)
            printf("Ok.\n");
        if(stat==tsys01_status_i2c_transfer_error)
            printf("Transfer Error.\n");

        // Wait for another key press and then display the main menu again
        printf("\nPress any key to continue...\n");
        read(1, (char*)&key_input, 1);
        tsys01_main_menu();

    }else if(key_input == '3'){        // If the '3' key is pressed

        // Read the PROM coefficients from the TSYS01
        printf("\n");
        printf("Reading PROM Coefficients...\n");
        stat = tsys01_read_prom();

        // Display status returned from read_prom operation
        // and display prom values if successful
        printf("Read PROM Complete with status: ");
        if(stat==tsys01_status_ok){
            prom_read_flag=1;
            printf("Ok.\n");
            printf("\n");
            printf("_____________________________________________\n");
            printf("|   PROM Addr   |Coeff (Base 10)|  Coeff (Hex)  |\n");
            printf("|---------------+---------------+---------------|\n");
            for(i=0;i<5;i++){
                printf("|\t%d\t|     %5d\t|     0x%4X\t|\n",i,(unsigned int)tsys01_prom_coeffs[i],(unsigned int)tsys01_prom_coeffs[i]);
            }
        }else if(stat==tsys01_status_i2c_transfer_error){
            printf("Transfer Error.\n");
        }

        // Wait for another key press and then display the main menu again
        printf("\nPress any key to continue...\n");
        read(1, (char*)&key_input, 1);
```

```
            tsys01_main_menu();

        }else if(key_input == '4'){         // If the '4' key is pressed

            if(prom_read_flag==0){          // PROM was not yet read--cannot read temperature yet
                printf("PROM Coefficients have not yet been read. Cannot complete temperature read.\n");
            }else{                          // PROM has been read--continue on to read temperature
                // Read one temperature value
                printf("\n");
                printf("Reading Temperature Value...\n");
                stat = tsys01_read_temperature(&temperature);

                // Display the status returned from the read_temperature
                // operation and display the temperature if successful
                printf("Temperature Read Complete with status: ");
                if(stat==tsys01_status_ok){
                    printf("Ok.\n");
                    printf("Temperature : %5.2f%cC\n",temperature,248);
                }else if(stat==tsys01_status_i2c_transfer_error){
                    printf("Transfer Error.\n");
                }
            }

            // Wait for another key press and then display the main menu again
            printf("\nPress any key to continue...\n");
            read(1, (char*)&key_input, 1);
            tsys01_main_menu();

        }else if(key_input == '5'){         // If the '5' key is pressed

            if(prom_read_flag==0){          // PROM was not yet read--cannot read temperature yet
                printf("PROM Coefficients have not yet been read. Cannot complete temperature reads.\n");
            }else{                          // PROM has been read--continue on to read temperatures
                // Read 20 temperature values at ~2 per second
                printf("\n");
                printf("Reading 20 Temperature Values...\n");
                for(i=0;i<20;i++){
                    stat = tsys01_read_temperature(&temperature);
                    if(stat==tsys01_status_ok){
                        printf("%2d: %5.2f%cC\n",i+1,temperature,248);
                    }else{
                            printf("Transfer Error.\n");
                    }
                    usleep(490000);
                }
            }

            // Wait for another key press and then display the main menu again
            printf("\nPress any key to continue...\n");
            read(1, (char*)&key_input, 1);
            tsys01_main_menu();

        }else if(key_input == 27){    // If the 'ESC' key is pressed

            // Print done and exit.
            printf("Done.\n");
            break;

        }else{                      // If some other key is pressed

            // Redisplay the main menu
            tsys01_main_menu();

        }

    }

    return 0;

}

void tsys01_main_menu(void){

    //Clear the screen
    printf("\033[2J");

    //Display the main menu
    printf("*****************************************\n");
    printf("****      Measurement Specialties      ****\n");
    printf("*****************************************\n");

    printf("\n");
    printf("    TSYS01 - Digital Temperature Sensor    \n");
    printf("-------------------------------------------\n");

    printf("\n");
    printf("Select a task:\n");
    printf("  (1)   - Set TSYS01 I2C Address\n");
    printf("  (2)   - Reset TSYS01\n");
    printf("  (3)   - Read TSYS01 PROM Coefficients\n");
    printf("  (4)   - Read TSYS01 Temperature Once\n");
```

```
    printf("  (5)   - Read TSYS01 Temperature 20 Times\n");
    printf(" (ESC)  - Quit\n");
    printf("\n");

    return;
}
```


te.com/sensorsolutions

.

PRODUCT SHEET

MEAS France SAS,
a TE Connectivity company.
Impasse Jeanne Benozzi CS 83 163
31027 Toulouse Cedex 3, FRANCE
Tel:+33 (0) 5 820 822 02
Fax: +33 (0) 5 820 821 51
customercare.tlse@te.com