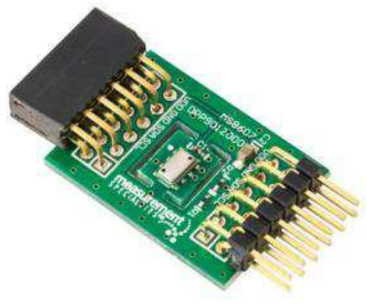


✓RoHS

MEAS MS8607 DIGITAL COMPONENT SENSOR (DCS) DRIVER FOR ZedBoard

Digital Pressure 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

- Operating pressure range: 300 to 1200 mbar
- Measures relative humidity from 0% to 100%
- Measures temperature from -40°C to 125°C
- Extended pressure range 10 to 2000 mbar
- Fast response time
- I^{2}C communication
- Very low power consumption

Reference Material

- Detailed information regarding operation of the IC:
  MS8607 Datasheet
- Detailed information regarding the Peripheral Module:
  MS8607 Peripheral Module
- Complete software sensor evaluation kit for Zedboard:
  MS8607_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	**ms8607_status { ms8607_status_ok, ms8607_status_i2c_transfer_error, ms8607_status_crc_error, ms8607_status_heater_on_error }**
enum	**ms8607_humidity_resolution { ms8607_humidity_resolution_12b, ms8607_humidity_resolution_8b, ms8607_humidity_resolution_10b, ms8607_humidity_resolution_11b }**
enum	**ms8607_pressure_resolution { ms8607_pressure_resolution_osr_256, ms8607_pressure_resolution_osr_512, ms8607_pressure_resolution_osr_1024, ms8607_pressure_resolution_osr_2048, ms8607_pressure_resolution_osr_4096, ms8607_pressure_resolution_osr_8192 }**
enum	**ms8607_battery_status { ms8607_battery_ok, ms8607_battery_low }**
enum	**ms8607_heater_status { ms8607_heater_off, ms8607_heater_on }**
Functions	
void	**ms8607_init (u32)** Initializes the AXI address of the AXI IIC Core, initializes the internal humidity resolution variable to 12b, and initializes the internal pressure resolution-oversampling rate to 8192 in order to reflect the sensor's initial resolution value on reset.
enum ms8607_status	**ms8607_reset (void)** Sends both I^{2}C reset commands to the MS8607 device.
enum ms8607_status	**ms8607_read_prom (void)** Reads the factory calibrated coefficients for use in temperature and pressure conversion.
enum ms8607_status	**ms8607_set_humidity_resolution (enum ms8607_humidity_resolution)** Read the user register from the device, modify its contents to reflect the resolution that is passed in to this function, and then write the updated user register value to the MS8607 device.
enum ms8607_status	**ms8607_set_pressure_resolution (enum ms8607_pressure_resolution_osr)** Set the over-sampling rate resolution for pressure measurements.
enum ms8607_status	**ms8607_read_temperature_pressure_humidity (float* temperature, float* pressure, float* relative_humidity)** Send the I^{2}C commands to measure temperature, pressure, and humidity.
enum ms8607_status	**ms8607_get_battery_status (ms8607_battery_status* batt_stat**) Send I^{2}C command to read battery status.
enum ms8607_status	**ms8607_get_heater_status (ms8607_battery_status* heat_stat)** Send I^{2}C command to read heater status.
enum ms8607_status	**ms8607_enable_heater (void)** Send I^{2}C commands to perform a read/modify/write operation that will enable the on-chip heater.
enum ms8607_status	**ms8607_disable_heater (void)** Send I^{2}C commands to perform a read/modify/write operation that will disable the on-chip heater.
float	**ms8607_compute_dew_point (float Tamb, float RHamb)** Compute dew point temperature in degrees C.

Digilent Pmod™ is a trademark.

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:

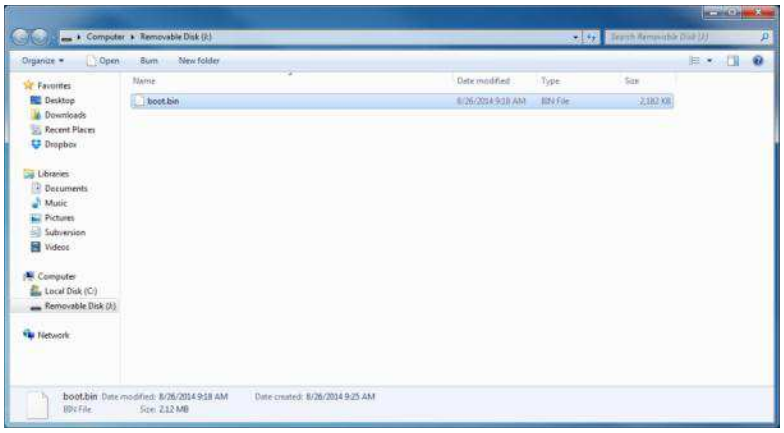
- ZedBoard
- MS8607 sensor for Digilent Pmod™ board
- SD card
- ZedBoard power adapter

- 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.

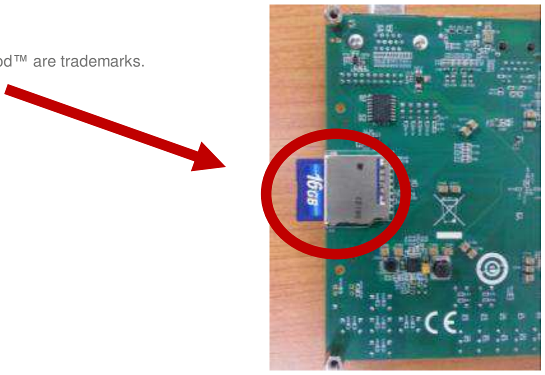


Name	Date modified	Type	Size
boot.bin	8/26/2014 9:18 AM	BIN File	2,182 KB

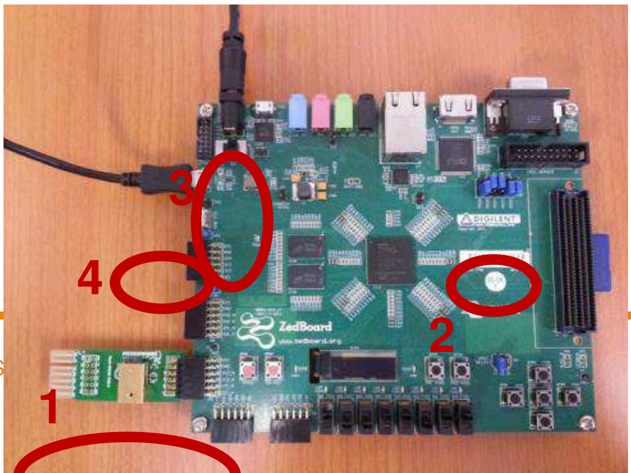
boot.bin Date modified: 8/26/2014 9:18 AM Date created: 8/26/2014 9:25 AM
BIN File Size: 2.12 MB

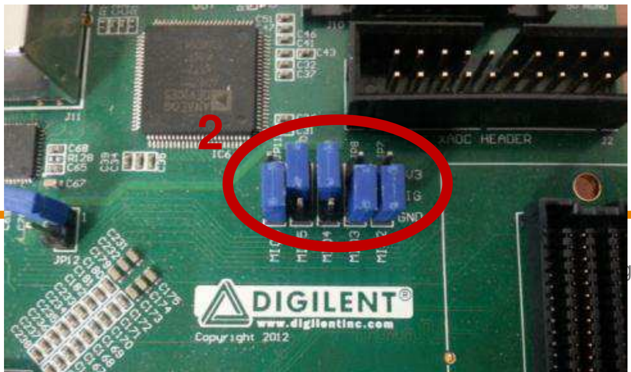
3. Safely eject the SD card from your computer. Insert the SD card into the card slot on the back of the ZedBoard.

ZedBoard, MicroZed and Digilent Pmod™ are trademarks.



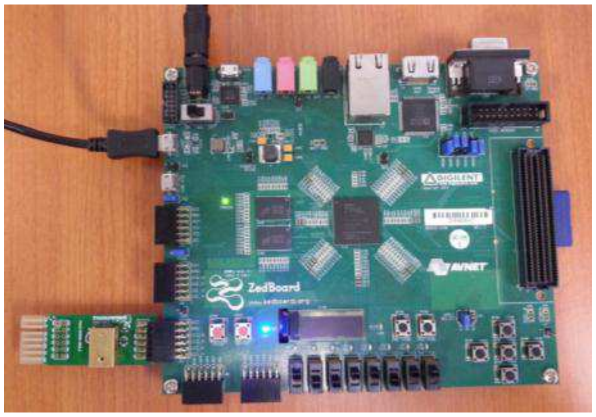
4. Connect the MS8607 digital pressure sensor to the "JC" Digilent Pmod™ port of the ZedBoard (1), ensure that jumpers JP7, JP8, JP9, JP10, and JP11 are configured such that the ZedBoard will boot from the SD card on start-up (2), and





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.



ZedBoard and Digilent Pmod™ are trademarks.

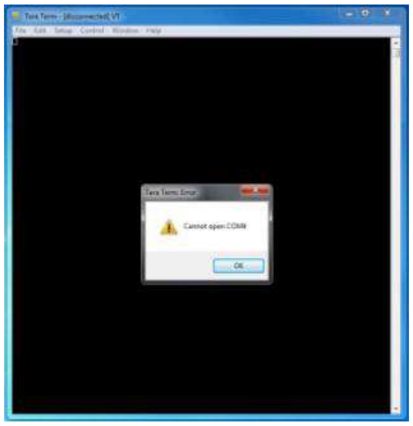
Launching the Console Application

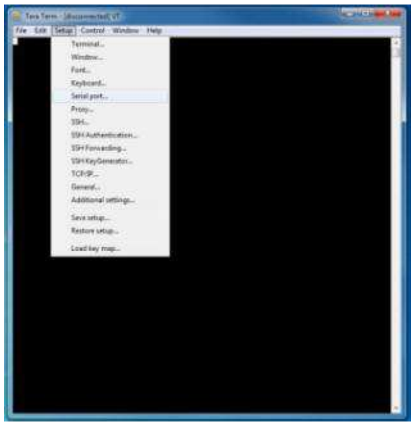
Now that you have successfully set up your hardware platform, you are ready to run the console application.

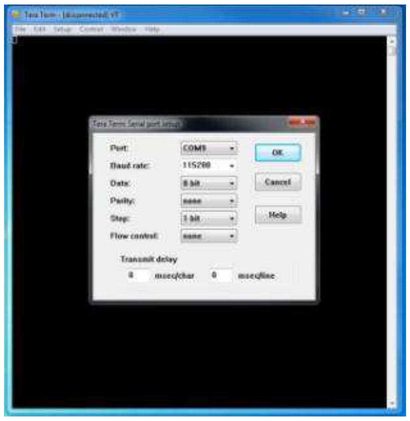
1. Upon power-on, the console application should already be running. It will be necessary to open a terminal and configure a serial connection to interact with the console application. Do this by opening tera term (which can be downloaded from http://en.sourceforge.jp/projects/ttssh2/releases/) or a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

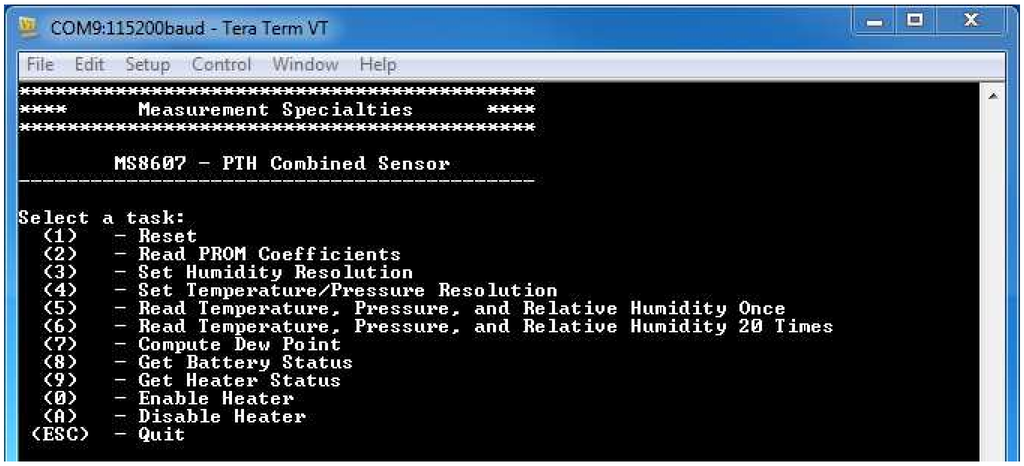
3. Now select the appropriate COM port that your ZedBoard setup is connected to. If you are not sure which this is, refer to the Device Manager. Configure your serial connection with 115200 Baud, 8 bit data, no parity, 1 stop bit, and no flow control, and then click OK.


Tera Term - [disconnected] VT
File Edit Setup Control Window Help
Tera Term: Error
OK
Cannot open COM8




Tera Term - [disconnected] VT
File Edit Setup Control Window Help
Tera Term: Serial port setup
OK
Cancel
Help
Port: COM9
Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none
Transmit delay
0 msec/char 0 msec/line

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.


COM9:115200baud - Tera Term VT
File Edit Setup Control Window Help
```
*******************************************
****       Measurement Specialties     ****
*******************************************

        MS8607 - PTH Combined Sensor
-------------------------------------------

Select a task:
  (1)    - Reset
  (2)    - Read PROM Coefficients
  (3)    - Set Humidity Resolution
  (4)    - Set Temperature/Pressure Resolution
  (5)    - Read Temperature, Pressure, and Relative Humidity Once
  (6)    - Read Temperature, Pressure, and Relative Humidity 20 Times
  (7)    - Compute Dew Point
  (8)    - Get Battery Status
  (9)    - Get Heater Status
  (0)    - Enable Heater
  (A)    - Disable Heater
 (ESC)   - Quit
```

Running the Console Application

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

a. After startup, it is a good idea to reset the sensor. This puts it in a known state. Do this by selecting (1) in the console application.

b. 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 measurements can be taken. Do this by selecting (2) in the console application.

Now the sensor and the software are setup and ready to use. These first two steps only need to be performed at power up.

c. The console application option (3) displays a menu that allows the user to select from the four possible humidity resolution modes of the sensor.

d. The console application option (4) displays a menu that allows the user to select from the six possible pressure over-sampling resolution modes of the sensor.

e. The console application option (5) reads the temperature, pressure, and relative humidity values and displays each of them once.

f. The console application option (6) reads the temperature, pressure, and relative humidity 20 times each at approximately two measurement triplets per second and displays them to the screen in real time.

g. The console application option (7) computes the dew point from the last measured temperature and relative humidity values.

h. The console application option (8) reads the MS8607's battery status and displays it to the console.

i. The console application option (9) reads the MS8607's heater status and displays it to the console.

j. The console application option (0) sends the I^{2}C command to the MS8607 device that enables the on-chip heater.

k. The console application option (A) sends the I^{2}C command to the MS8607 device that disables the on-chip heater.

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_MS8607_Main.c: Console Application for Testing the MS8607
 *
 * 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 "ms8607.h"

#define               XPAR_AXI_IIC_JC_BASEADDR        XPAR_IIC_0_BASEADDR

void ms8607_main_menu(void);

int main()
{

    char key_input;
    int i;
    ms8607_status stat;
    u8     prom_read_flag = 0;
    float temperature;
    float pressure;
    float relative_humidity;
    float dew_point;
    ms8607_battery_status        batt_stat;
    ms8607_heater_status         heat_stat;

    //Initialize the UART
    init_platform();

    printf("Hello World\n");

    // Set the AXI address of the IIC core
    ms8607_init(XPAR_AXI_IIC_JC_BASEADDR);

    // Display the main menu
    ms8607_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

            // Send the reset command to the MS8607
            printf("\n");
            printf("Resetting MS8607...\n");
            stat = ms8607_reset();

            // Display the status returned from the reset operation
            printf("MS8607 Reset Complete with status: ");
            if(stat==ms8607_status_ok)
                printf("Ok.\n");
            if(stat==ms8607_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);
            ms8607_main_menu();

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

            // Read the PROM coefficients from the MS8607
            printf("\n");
            printf("Reading PROM Coefficients...\n");
            stat = ms8607_read_prom();

            // Display status returned from read_prom operation
            // and display prom values if successful
            printf("Read PROM Complete with status: ");
            if(stat==ms8607_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<7;i++){
                    printf("|\t%d\t|     %5d\t|     0x%4X\t|\n",i,(unsigned int)ms8607_prom_coeffs[i],(unsigned int)ms8607_prom_coeffs[i]);
                }
            }else if(stat==ms8607_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);
            ms8607_main_menu();

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

           // Display humidity resolution selection menu
            printf("\n");
            printf("Select a humidity resolution:\n");
            printf("  (1)   - 12-Bit Relative Humidity\n");
            printf("  (2)   - 11-Bit Relative Humidity\n");
            printf("  (3)   - 10-Bit Relative Humidity\n");
            printf("  (4)   -  8-Bit Relative Humidity\n");

            // Get keyboard input ignoring keypresses that are not or '1' or '2' or '3' or '4'
            read(1, (char*)&key_input, 1);
            while(key_input!='1' && key_input!='2' && key_input!='3' && key_input!='4'){
                read(1, (char*)&key_input, 1);
            }

            if(key_input == '1'){           // If the '1' key is pressed
                // Set humidity resolution to 12-bits
                stat = ms8607_set_humidity_resolution(ms8607_humidity_resolution_12b);
                printf("\nSetting MS8607 Humidity Resolution to 12-bits\n");
            }else if(key_input == '2'){     // If the '2' key is pressed
                // Set humidity resolution to 11-bits
                stat = ms8607_set_humidity_resolution(ms8607_humidity_resolution_11b);
                printf("\nSetting MS8607 Humidity Resolution to 11-bits\n");
            }else if(key_input == '3'){     // If the '3' key is pressed
                // Set humidity resolution to 10-bits
                stat = ms8607_set_humidity_resolution(ms8607_humidity_resolution_10b);
                printf("\nSetting MS8607 Humidity Resolution to 10-bits\n");
            }else if(key_input == '4'){     // If the '4' key is pressed
                // Set humidity resolution to  8-bits
                stat = ms8607_set_humidity_resolution(ms8607_humidity_resolution_8b);
                printf("\nSetting MS8607 Humidity Resolution to 8-bits\n");
            }

            // Display the status returned from the set resolution operation
            printf("MS8607 Set Humidity Resolution Complete with status: ");
            if(stat==ms8607_status_ok)
                printf("Ok.\n");
            if(stat==ms8607_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);
            ms8607_main_menu();

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

           // Display resolution selection menu
            printf("\n");
            printf("Select a pressure/temperature resolution (over-sampling rate):\n");
            printf("  (1)   - 256\n");
            printf("  (2)   - 512\n");
```

```
            printf("  (3)   - 1024\n");
            printf("  (4)   - 2048\n");
            printf("  (5)   - 4096\n");
            printf("  (6)   - 8192\n");

            // Get keyboard input ignoring keypresses that are not '1' or '2' or '3' or '4' or '5' or '6'
            read(1, (char*)&key_input, 1);
            while(key_input!='1' && key_input!='2' && key_input!='3' && key_input!='4' && key_input!='5' && key_input!='6'){
                read(1, (char*)&key_input, 1);
            }

            if(key_input == '1'){           // If the '1' key is pressed
                // Set OSR to 256
                ms8607_pressure_res = ms8607_pressure_resolution_osr_256;
                printf("\nSet MS8607 Over-Sampling Rate to 256\n");
            }else if(key_input == '2'){     // If the '2' key is pressed
                // Set OSR to 512
                ms8607_pressure_res = ms8607_pressure_resolution_osr_512;
                printf("\nSet MS8607 Over-Sampling Rate to 512\n");
            }else if(key_input == '3'){     // If the '3' key is pressed
                // Set OSR to 1024
                ms8607_pressure_res = ms8607_pressure_resolution_osr_1024;
                printf("\nSet MS8607 Over-Sampling Rate to 1024\n");
            }else if(key_input == '4'){     // If the '4' key is pressed
                // Set OSR to 2048
                ms8607_pressure_res = ms8607_pressure_resolution_osr_2048;
                printf("\nSet MS8607 Over-Sampling Rate to 2048\n");
            }else if(key_input == '5'){     // If the '5' key is pressed
                // Set OSR to 4096
                ms8607_pressure_res = ms8607_pressure_resolution_osr_4096;
                printf("\nSet MS8607 Over-Sampling Rate to 4096\n");
            }else if(key_input == '6'){     // If the '6' key is pressed
                // Set OSR to 8192
                ms8607_pressure_res = ms8607_pressure_resolution_osr_8192;
                printf("\nSet MS8607 Over-Sampling Rate to 8192\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);
            ms8607_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, pressure, and humidity yet
                printf("PROM Coefficients have not yet been read. Cannot complete temperature/pressure read.\n");
            }else{                          // PROM has been read--continue on to read temperature, pressure, and humidity

                                            // Read Temperature, Pressure, and Relative Humidity once
                                            printf("\n");
                                            printf("Reading Temperature, Pressure, and Relative Humidity...\n");
                                            stat = ms8607_read_temperature_pressure_humidity(&temperature, &pressure, &relative_humidity);

                                            // Display the status returned from the read_temperature_pressure_humidity
                                            // operation and display the temperature, pressure, and humidity if successful
                                            printf("Temperature, Pressure, and Relative Humidity Read Complete with status: ");
                                            if(stat==ms8607_status_ok){
                                                    printf("Ok.\n");
                                                    printf("Temperature : %5.2f%cC, \tPressure : %5.1fhPa, \tRelative Humidity :
%4.1f%%",temperature,248,pressure,relative_humidity);
                                            }else if(stat==ms8607_status_i2c_transfer_error){
                                                    printf("Transfer Error.");
                                            }else if(stat==ms8607_status_crc_error){
                                                    printf("CRC Error.");
                                            }
                                            printf("\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);
            ms8607_main_menu();

        }else if(key_input == '6'){         // If the '6' key is pressed

            if(prom_read_flag==0){          // PROM was not yet read--cannot read temperature, pressure, and humidity
                printf("PROM Coefficients have not yet been read. Cannot complete temperature/pressure/humidity read.\n");
            }else{                          // PROM has been read--continue on to read temperature, pressure, and humidity

                                            // Read 20 temperature and relative humidity values at ~2 per second
                                            printf("\n");
                                            printf("Reading 20 Temperature and Relative Humidity Values...\n");
                                            for(i=0;i<20;i++){
                                                    stat = ms8607_read_temperature_pressure_humidity(&temperature, &pressure, &relative_humidity);
                                                    printf("%2d: ",i+1);
                                                    if(stat==ms8607_status_ok){
                                                            printf("%5.2f%cC, \t%5.1fhPa, \t%4.1f%%",temperature,248,pressure,relative_humidity);
                                                    }else if(stat==ms8607_status_i2c_transfer_error){
                                                            printf("Transfer Error.");
                                                    }else if(stat==ms8607_status_crc_error){
                                                            printf("CRC Error.");
                                                    }
                                                    printf("\n");
                                                    usleep( (500-MS8607_OSR_8192_CONV_DELAY_MS)*1000 );
                                            }

            }

            // 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);
            ms8607_main_menu();

        }else if(key_input == '7'){         //If the '7' key is pressed

            // Compute Dew Point from last read Temperature and Relative Humidity values
            printf("\n");
            printf("Computing Dew Point from last read Temperature and Relative Humidity values...\n");
            stat = ms8607_compute_dew_point(temperature, relative_humidity, &dew_point);
            if(stat == ms8607_status_i2c_transfer_error){
                      printf("Could not read heater status.\n");
            }else if(stat == ms8607_status_heater_on_error){
                      printf("Cannot calculate dew point while heater is on.\n");
            }else if(stat == ms8607_status_ok){
                printf("Dew Point : %5.2f%cC",dew_point,248);
            }

            // 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);
            ms8607_main_menu();

        }else if(key_input == '8'){         //If the '8' key is pressed

            // Get Battery Status
            printf("\n");
            printf("Getting Battery Status...\n");
            stat = ms8607_get_battery_status(&batt_stat);

            // Display the status returned from the battery status check operation
            printf("Get Battery Status Check Complete with status: ");
            if(stat==ms8607_status_ok){
                printf("Ok.\n");
                      printf("Battery ");
                      if(batt_stat == ms8607_battery_ok){
                                 printf("Ok.\n");
                      }else{
                                 printf("Low.\n");
                      }
            }
            if(stat==ms8607_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);
            ms8607_main_menu();

        }else if(key_input == '9'){         //If the '9' key is pressed

            // Get Heater Status
            printf("\n");
            printf("Getting Heater Status...\n");
            stat = ms8607_get_heater_status(&heat_stat);

            // Display the status returned from the heater status check operation
            printf("Get Heater Status Check Complete with status: ");
            if(stat==ms8607_status_ok){
                printf("Ok.\n");
                      printf("Heater ");
                      if(heat_stat == ms8607_heater_on){
                                 printf("On.\n");
                      }else{
                                 printf("Off.\n");
                      }
            }else if(stat==ms8607_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);
            ms8607_main_menu();

        }else if(key_input == '0'){         //If the '0' key is pressed

            // Enable heater
            printf("\n");
            printf("Enabling Heater...\n");
            stat = ms8607_enable_heater();
```

```
            // Display the status returned from the enable heater operation
            printf("Enable Heater Operation Complete with status: ");
            if(stat==ms8607_status_ok)
                printf("Ok.\n");
            if(stat==ms8607_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);
            ms8607_main_menu();

        }else if(key_input == 'A' || key_input == 'a'){         //If the 'A' key is pressed

            // Disable heater
            printf("\n");
            printf("Disabling Heater...\n");
            stat = ms8607_disable_heater();

            // Display the status returned from the disable heater operation
            printf("Disable Heater Operation Complete with status: ");
            if(stat==ms8607_status_ok)
                printf("Ok.\n");
            if(stat==ms8607_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);
            ms8607_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
            ms8607_main_menu();

        }
    }

    return 0;

}

void ms8607_main_menu(void){

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

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

    printf("\n");
    printf("        MS8607 - PTH Combined Sensor       \n");
    printf("-------------------------------------------\n");

    printf("\n");
    printf("Select a task:\n");
    printf("  (1)   - Reset\n");
    printf("  (2)   - Read PROM Coefficients\n");
    printf("  (3)   - Set Humidity Resolution\n");
    printf("  (4)   - Set Temperature/Pressure Resolution\n");
    printf("  (5)   - Read Temperature, Pressure, and Relative Humidity Once\n");
    printf("  (6)   - Read Temperature, Pressure, and Relative Humidity 20 Times\n");
    printf("  (7)   - Compute Dew Point\n");
    printf("  (8)   - Get Battery Status\n");
    printf("  (9)   - Get Heater Status\n");
    printf("  (0)   - Enable Heater\n");
    printf("  (A)   - Disable Heater\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