MEAS HTU21D DIGITAL COMPONENT SENSOR (DCS) DRIVER FOR MicroZed

Digital Humidity and 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 relative humidity from 0% to 100%
- Measures temperature from -40°C to 125°C
- Console mode display
- Evaluate basic sensor functions
- Evaluate Dew Point computation from RH & T

Reference Material

- Detailed information regarding operation of the IC: HTU21D Datasheet
- Detailed information regarding the Peripheral Module: HTU21D Peripheral Module
- Complete software sensor evaluation kit for MicroZed: HTU21D_MicroZed.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	**htu21d_status { htu21d_status_ok, htu21d_status_i2c_transfer_error, htu21d_status_crc_error }**
enum	**htu21d_resolution { htu21d_resolution_t_14b_rh_12b, htu21d_resolution_t_12b_rh_8b, htu21d_resolution_t_13b_rh_10b, htu21d_resolution_t_11b_rh_11b }**
enum	**htu21d_battery_status { htu21d_battery_ok, htu21d_battery_low }**
enum	**htu21d_heater_status { htu21d_heater_off, htu21d_heater_on }**
Functions	
void	**htu21d_init (u32)** Initializes the AXI address of the AXI IIC Core and the internal resolution variable to t_14b_rh_12b to reflect the sensor's initial resolution value on reset.
Enum htu21d_status	**htu21d_reset (void)** Sends I^2C reset command to the HTU21D device.
Enum htu21d_status	**htu21d_set_resolution (enum htu21d_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 HTU21D device.
Enum htu21d_status	**htu21d_read_temperature_and_relative_humidity (float* t, float* rh)** Send the I^2C commands to start a temperature conversion, wait for completion, read the temperature value, start a relative humidity conversion, wait for completion, and read the relative humidity value.
Enum htu21d_status	**htu21d_get_battery_status (htu21d_battery_status* batt_stat)** Send I^2C command to read battery status.
enum htu21d_status	**htu21d_get_heater_status (htu21d_battery_status* heat_stat)** Send I^2C command to read heater status.
enum htu21d_status	**htu21d_enable_heater (void)** Send I^2C commands to perform a read/modify/write operation that will enable the on-chip heater.
enum htu21d_status	**htu21d_disable_heater (void)** Send I^2C commands to perform a read/modify/write operation that will disable the on-chip heater.
float	**htu21d_compute_dew_point (float Tamb, float RHamb)** Compute dew point temperature in degrees C.

Project Setup

This project is based on the MicroZed board with I/O carrier card. The FPGA hardware and the console application will be loaded via micro SD card.

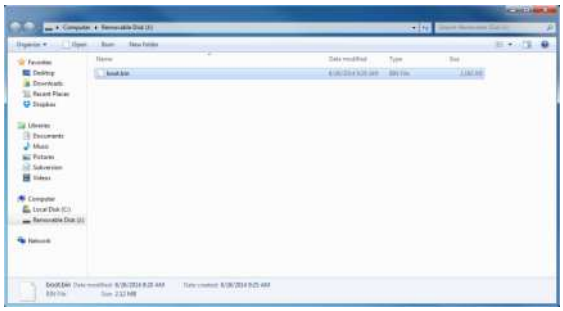
You will need:

- MicroZed board
- I/O carrier card
- HTU21D sensor for Digilent Pmod™ board
- Micro SD card
- I/O carrier card 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

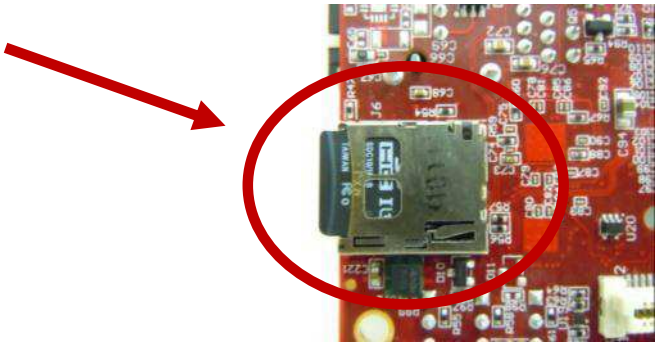
Project Setup

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

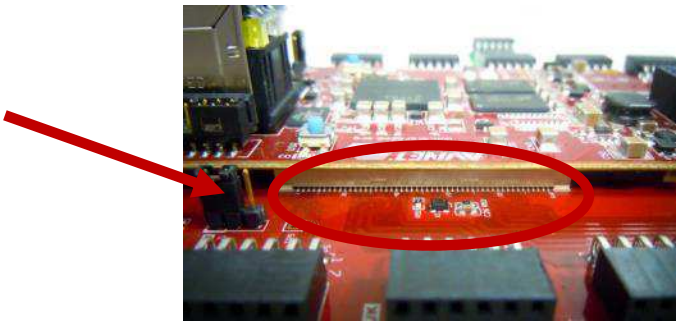
1. First, if you have not connected your computer to a ZedBoard or MicroZed device before, you will need to download and install the Silicon Labs CP2104 USB-to UART driver. The setup guide for installing the driver can be found at the address below: http://www.ZedBoard.org/sites/default/files/documentations/CP210x_Setup_Guide_1_2.pdf
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 HTU21D from the MicroZed software link and copy it onto the SD card so that it is the only file present on the file system.



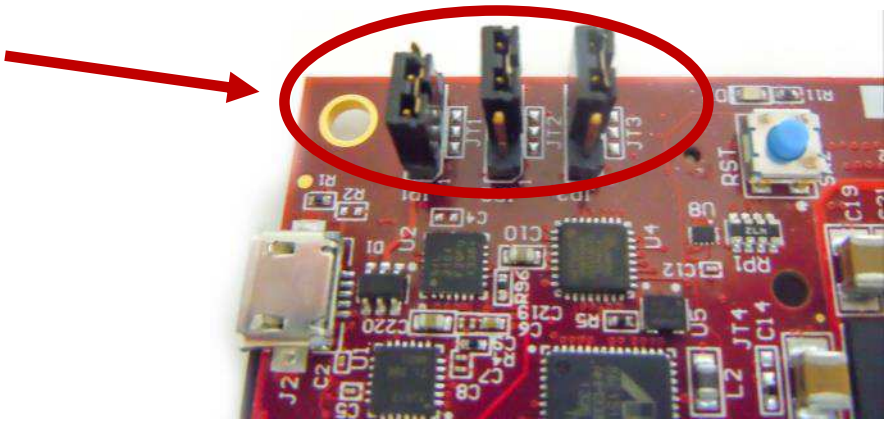
3. Safely eject the micro SD card from your computer. Insert the micro SD card into the card slot on the back of the MicroZed board.



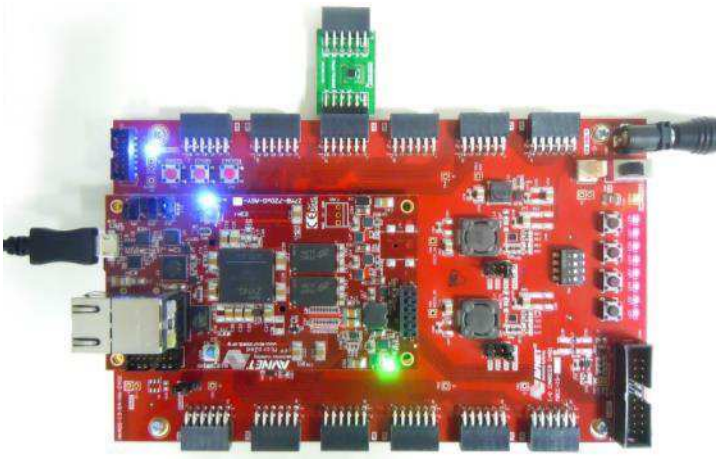
4. Carefully line up the MicroZed board with the I/O carrier card and push them together until snug.



5. Connect the HTU21D Digital Humidity and Temperature Sensor to the “JC” Digilent Pmod™ port of the I/O carrier card, ensure that jumpers J1, J2, and J3 are configure such that the MicroZed will boot from the SD card on start up, and connect the power adapter to the barrel jack on the I/O carrier card (shown on the bottom). Finally connect the micro-USB cable to the micro-USB port of the MicroZed (shown at the top). The USB cable will facilitate UART transmissions for the console application.



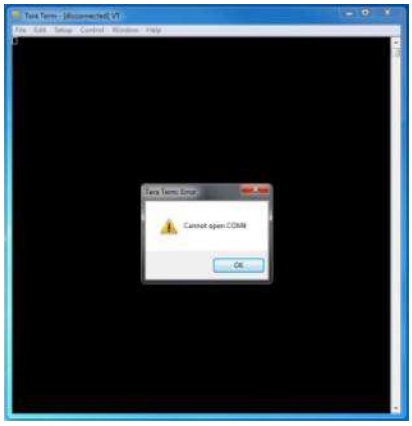
6. Turn on the power to the board with the switch next to the barrel jack. When the board powers up, the MicroZed will briefly illuminate a red LED, which will then turn off after less than a second. Once the FPGA has been successfully programmed by the boot image on the SD card, a blue “Done” LED will illuminate on both the MicroZed and the I/O carrier card. Your hardware should appear as shown below. If the board was powered on before this step, turn the power off and repeat this step.

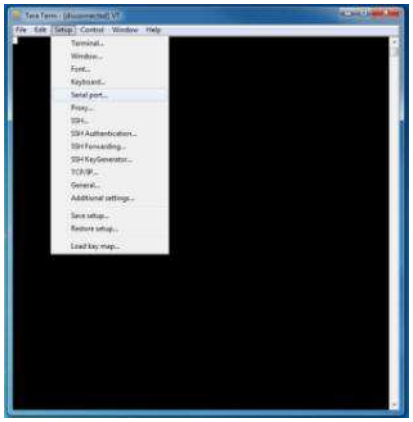


Launching the Console Application

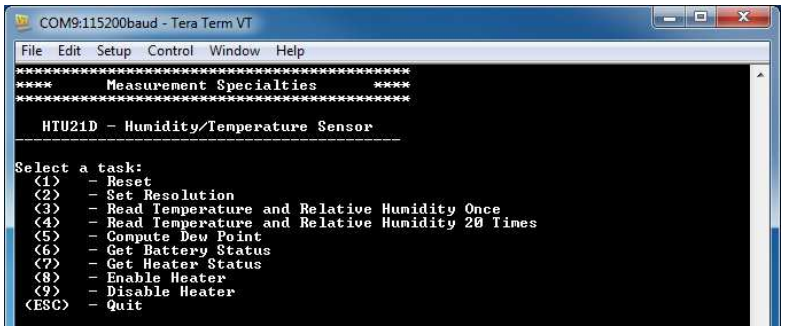
Now that you have successfully set up your hardware platform, you are ready to run the console application.

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 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 MicroZed 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 MicroZed. Press enter and the console application will display the main menu from which you can perform several tasks on the HTU21D digital temperature sensor.



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.

Now the sensor and the software are setup and ready to use. This first step only needs to be performed at power up.

b. The console application option (2) displays a menu that allows the user to select from the four possible resolution modes of the sensor.

c. The console application option (3) reads both the temperature and relative humidity values and displays each of them once.

d. The console application option (4) reads the temperature and relative humidity 20 times each at approximately two measurement pairs per second and displays them to the screen in real time.

e. The console application option (5) computes the dew point from the last measured temperature and relative humidity values.

f. The console application option (6) reads the HTU21D's battery status and displays it to the console.

g. The console application option (7) reads the HTU21D's heater status and displays it to the console.

h. The console application option (8) sends the I^2C command to the HTU21D device that enables the on-chip heater.

i. The console application option (9) sends the I^2C command to the HTU21D 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_HTU21D_Main.c: Console Application for Testing the HTU21D
 *
 * 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 "htu21d.h"

void htu21d_main_menu(void);

int main()
{
```

```
char key_input;
int i;
htu21d_status stat;
float temperature;
float relative_humidity;
float dew_point;
htu21d_battery_status       batt_stat;
htu21d_heater_status        heat_stat;

//Initialize the UART
init_platform();

printf("Hello World\n");

// Set the AXI address of the IIC core
htu21d_init(XPAR_AXI_IIC_JC_BASEADDR);

// Display the main menu
htu21d_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 HTU21D
        printf("\n");
        printf("Resetting HTU21D...\n");
        stat = htu21d_reset();

        // Display the status returned from the reset operation
        printf("HTU21D Reset Complete with status: ");
        if(stat==htu21d_status_ok)
            printf("Ok.\n");
        if(stat==htu21d_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);
        htu21d_main_menu();

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

       // Display resolution selection menu
        printf("\n");
        printf("Select a resolution:\n");
        printf("  (1)   - 14-Bit Temperature and 12-Bit Relative Humidity\n");
        printf("  (2)   - 12-Bit Temperature and  8-Bit Relative Humidity\n");
        printf("  (3)   - 13-Bit Temperature and 10-Bit Relative Humidity\n");
        printf("  (4)   - 11-Bit Temperature and 11-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 resolution to 12-bit RH and 14-bit Temp
            stat = htu21d_set_resolution(htu21d_resolution_t_14b_rh_12b);
            printf("\nSetting HTU21D Resolution to 14-bit Temperature and 12-bit Relative Humidity\n");
        }else if(key_input == '2'){     // If the '2' key is pressed
            // Set resolution to  8-bit RH and 12-bit Temp
            stat = htu21d_set_resolution(htu21d_resolution_t_12b_rh_8b);
            printf("\nSetting HTU21D Resolution to 12-bit Temperature and  8-bit Relative Humidity\n");
        }else if(key_input == '3'){     // If the '3' key is pressed
            // Set resolution to 10-bit RH and 13-bit Temp
            stat = htu21d_set_resolution(htu21d_resolution_t_13b_rh_10b);
            printf("\nSetting HTU21D Resolution to 13-bit Temperature and 10-bit Relative Humidity\n");
        }else if(key_input == '4'){     // If the '4' key is pressed
            // Set resolution to 11-bit RH and 11-bit Temp
            stat = htu21d_set_resolution(htu21d_resolution_t_11b_rh_11b);
            printf("\nSetting HTU21D Resolution to 11-bit Temperature and 11-bit Relative Humidity\n");
        }

        // Display the status returned from the set resolution operation
        printf("HTU21D Set Resolution Complete with status: ");
        if(stat==htu21d_status_ok)
            printf("Ok.\n");
        if(stat==htu21d_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);
        htu21d_main_menu();
```

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

   // Read Temperature and Relative Humidity once
    printf("\n");
    printf("Reading Temperature and Relative Humidity...\n");
    stat = htu21d_read_temperature_and_relative_humidity(&temperature, &relative_humidity);

    // Display the status returned from the read_temperature and relative humidity
    // operation and display the temperature and relative humidity if successful
    printf("Temperature and Relative Humidity Read Complete with status: ");
    if(stat==htu21d_status_ok){
        printf("Ok.\n");
        printf("Temperature : %5.2f%cC, \tRelative Humidity : %4.1f%%",temperature,248,relative_humidity);
    }else if(stat==htu21d_status_i2c_transfer_error){
        printf("Transfer Error.");
    }else if(stat==htu21d_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);
    htu21d_main_menu();

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

    // 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 = htu21d_read_temperature_and_relative_humidity(&temperature, &relative_humidity);
        printf("%2d: ",i+1);
        if(stat==htu21d_status_ok){
            printf("%5.2f%cC, \t%4.1f%%",temperature,248,relative_humidity);
        }else if(stat==htu21d_status_i2c_transfer_error){
            printf("Transfer Error.");
        }else if(stat==htu21d_status_crc_error){
              printf("CRC Error.");
        }
        printf("\n");
        usleep( (500-HTU21D_14B_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);
    htu21d_main_menu();

}else if(key_input == '5'){         //If the '5' 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");
    dew_point = htu21d_compute_dew_point(temperature, relative_humidity);

    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);
    htu21d_main_menu();

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

    // Get Battery Status
    printf("\n");
    printf("Getting Battery Status...\n");
    stat = htu21d_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==htu21d_status_ok)
        printf("Ok.\n");
              printf("Battery ");
              if(batt_stat == htu21d_battery_ok){
                        printf("Ok.\n");
              }else{
                        printf("Low.\n");
              }
    if(stat==htu21d_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);
    htu21d_main_menu();

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

```
            // Get Heater Status
            printf("\n");
            printf("Getting Heater Status...\n");
            stat = htu21d_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==htu21d_status_ok){
                printf("Ok.\n");
                      printf("Heater ");
                      if(heat_stat == htu21d_heater_on){
                                printf("On.\n");
                      }else{
                                printf("Off.\n");
                      }
            }else if(stat==htu21d_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);
            htu21d_main_menu();

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

            // Enable heater
            printf("\n");
            printf("Enabling Heater...\n");
            stat = htu21d_enable_heater();

            // Display the status returned from the enable heater operation
            printf("Enable Heater Operation Complete with status: ");
            if(stat==htu21d_status_ok)
                printf("Ok.\n");
            if(stat==htu21d_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);
            htu21d_main_menu();

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

            // Disable heater
            printf("\n");
            printf("Disabling Heater...\n");
            stat = htu21d_disable_heater();

            // Display the status returned from the disable heater operation
            printf("Disable Heater Operation Complete with status: ");
            if(stat==htu21d_status_ok)
                printf("Ok.\n");
            if(stat==htu21d_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);
            htu21d_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
            htu21d_main_menu();

        }
    }

    return 0;

}

void htu21d_main_menu(void){

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

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

    printf("\n");
    printf("   HTU21D - Humidity/Temperature Sensor   \n");
```

```
    printf("----------------------------------------\n");

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