```c
//lab8I2ConARM_templat.c

//This is a recommended template for Lab #8\
//lab8I2ConARM_???, where ??? is your name initial
//Summer 2021, HUST
//Lab #8 to read two tmp101 breakout boards from both PL and PS sides with ARM I2C modules
//Date: 1-14-2019
//separate I2C instance configuration and TMP101 configuration
//Revision to make the code more modular and portable

#include <stdio.h>
#include "xparameters.h"
#include <xiicps.h>

#define PS_I2C_DEVICE_ID XPAR_PS7_I2C_0_DEVICE_ID
#define PS_DATA_RATE 100000 // 100KHz
#define PS_TMP101_ADDRESS 0b1001001 //floating
#define PS_TMP101_RESOLUTION 0b01100000 //12 bit resolution for TMP101

#define PL_I2C_DEVICE_ID XPAR_PS7_I2C_1_DEVICE_ID
#define PL_DATA_RATE 200000 // 200KHz
#define PL_TMP101_ADDRESS 0b1001000 //ground
#define PL_TMP101_RESOLUTION 0b00100000 //10  bit resolution for TMP101

XIicPs i2c_ps,i2c_pl;   //device instances

#define DELAYLOOPCOUNT 80000000
//function prototype
int ReadTemperature(XIicPs * I2Cinstance, float * temperature, int I2C_address);

/*****************************************************************
* This function ConfigureTMP101() configures an I2C TMP101 instance.
*
* @param    I2Cinstance is a pointer to an I2C instance.
* @param    I2CAddress is the address of I2C TMP101 chip.
* @param    TEMPresolution is .the resolution of TMP101 temperature as 0b0RR00000
*
* @return
*       - XST_SUCCESS if everything went well.
*       - XST_FAILURE if failed.
*
*****************************************************************/

int ConfigureTMP101(XIicPs *I2Cinstance, u8 I2CAddress, u8 TEMPresolution) {
    u8 SetPointertoZero=0b00000000;
    u8 SetResolution[]={0b00000001,0b00000000};

    SetResolution[1]=TEMPresolution;

//Set resolution

//set pointer back to 0x00 to point to the temperature value

    return XST_SUCCESS;
} //end ConfigureTMP101()

/*****************************************************************
```

```
58 * This function ConfigureI2Cinstance() configures an I2C instance.
59 *
60 * @param    I2Cinstance is a pointer to an I2C instance.
61 * @param    I2CdeviceID is the ID of I2C device defined in xparamenters.h.
62 * @param    I2C_ClockRate is clock frequency of I2C TMP101 in Hz.
63 *
64 * @return
65 *       - XST_SUCCESS if everything went well.
66 *       - XST_FAILURE if failed.
67 *
68 ************************************************************/
69 int ConfigureI2Cinstance(XIicPs *I2Cinstance, int I2CdeviceID, int I2C_ClockRate) {
70
71     XIicPs_Config *I2C_config;
72 //Call lookup table to find I2C hardware module for I2CdeviceID
73         I2C_config = XIicPs_LookupConfig(I2CdeviceID);
74         if (NULL == I2C_config) {
75             printf("\r\n-- Failed to find I2CdeviceID %d", I2CdeviceID);    return
   XST_FAILURE;
76         }
77 //Initialize I2C instance. Return XST_FAILTURE if failed.
78
79
80 //Self test. Return XST_FAILTURE if failed
81
82
83 //Set I2C clock frequency. Return XST_FAILTURE if failed
84
85
86 //Wait when I2Cdevice is busy until bus is idle to start another transfer
87
88
89
90         return XST_SUCCESS;
91
92 } // end ConfigureI2Cinstance()
93
94 int main()
95 {
96     float temp_ps, temp_pl;
97     unsigned int loopcount;
98     int status=XST_FAILURE;
99     printf("\r\n\n -- Start Lab #8 I2C for TMP101 --\r\n");
100
101 //configure I2C instance and tmp101 board on PS port
102     ConfigureI2Cinstance();
103     if(status == XST_FAILURE) printf("\r\n Failed to configure I2C instance on PS.");
104     ConfigureTMP101();
105 //configure I2C instance and tmp101 board on PL port
106     ConfigureI2Cinstance();
107     if(status == XST_FAILURE) printf("\r\n Failed to configure I2C instance on PL.");
108     ConfigureTMP101();
109     while(1) {
110 //Read tmp101 board and calculate temperature value on PS port
111     status=ReadTemperature();
112     if(status == XST_FAILURE) printf("\r\n Failed to read TMP101 from PS on bottom row
   of Connector JF.");
113
```

```
114 //Read tmp101 board and calculate temperature value on PL port
115     status=ReadTemperature();
116     if(status == XST_FAILURE) printf("\r\n Failed to read TMP101 from PL on top row of
    Connector JB.");
117
118 //printing floating numbers is not supported by xil_printf().
119 //Use printf() from <stdio.h> to print floating points.
120 //Mixing Xil_printf() with printf() may cause some printf() being dropped.
121 //Display temperature in floating point number with 4 digits after decimal point
122         printf("\n\r");
123
124
125 //delay loop to pause for a while
126         for(loopcount=0; loopcount<DELAYLOOPCOUNT; loopcount++);    //delay time
127
128 } //while(1)
129     while(1);   // hold just in case
130 }   //end main()
131
132 /************************************************************
133 * This function ReadTemperature() reads temperature of an TMP101.
134 *
135 * @param    I2Cinstance is a pointer to an I2C instance.
136 * @param    temperature is the returned floaing point temperature value.
137 * @param    I2C_address is the address of I2C TMP101 chip.
138 *
139 * @return
140 *       - XST_SUCCESS if everything went well.
141 *       - XST_FAILURE if failed.
142 *
143 ************************************************************/
144 int ReadTemperature(XIicPs * I2Cinstance, float * temperature, int I2C_address) {
145     u8 temp[2]; //2 byte temperature
146     u8 SetPointertoZero=0b00000000;
147 //set pointer back to 0x00 to point to the temperature value
148
149
150 //Read temperature. Return XST_FAILURE if failed
151
152
153 //Convert temperature to floating number
154
155
156         return XST_SUCCESS;
157 }   //end of ReadTemperature()
158
```