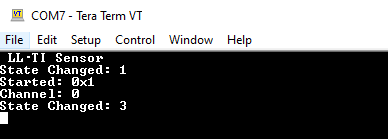
**Date Submitted: 12/11/19**

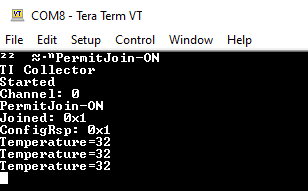
**Youtube:**

[**https://youtu.be/I2ssZnWopPY**](https://youtu.be/I2ssZnWopPY)

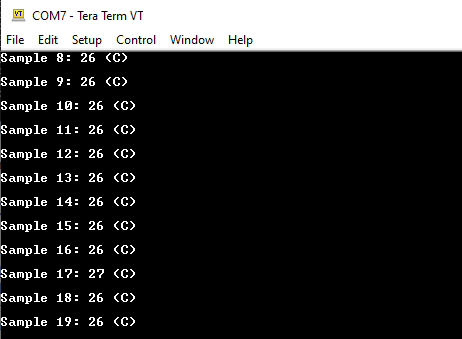
**Images:**

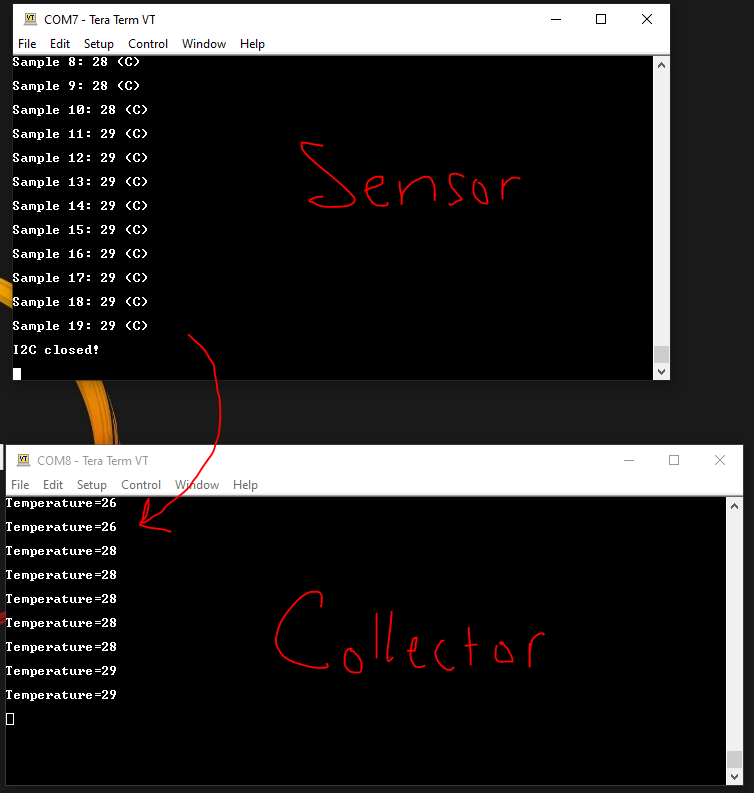
The sensor (COM 7) and collector (COM 8) is initialized on Tera Term – Task 3 on TI 15.4 Stack Project Zero:





The sensor portable app temperature samples (x20) – Task 1 on Using Stack and Portable App to Create a Remote Sensor:





**Temperature.c:**

/\*

\* Copyright (c) 2018-2019, Texas Instruments Incorporated

\* All rights reserved.

\*

\* Redistribution and use in source and binary forms, with or without

\* modification, are permitted provided that the following conditions

\* are met:

\*

\* \* Redistributions of source code must retain the above copyright

\* notice, this list of conditions and the following disclaimer.

\*

\* \* Redistributions in binary form must reproduce the above copyright

\* notice, this list of conditions and the following disclaimer in the

\* documentation and/or other materials provided with the distribution.

\*

\* \* Neither the name of Texas Instruments Incorporated nor the names of

\* its contributors may be used to endorse or promote products derived

\* from this software without specific prior written permission.

\*

\* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

\* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,

\* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR

\* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR

\* CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,

\* EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,

\* PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;

\* OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,

\* WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR

\* OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,

\* EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

\*/

/\*

\* ======== i2ctmp116.c ========

\*/

**#include** <stdint.h>

**#include** <stddef.h>

**#include** <unistd.h>

/\* Driver Header files \*/

**#include** <ti/drivers/GPIO.h>

**#include** <ti/drivers/I2C.h>

**#include** <ti/display/Display.h>

/\* Example/Board Header files \*/

**#include** "Board.h"

**#define** TASKSTACKSIZE 640

/\*

\* ======== TMP Registers ========

\*/

**#define** Si7021\_TMP\_REG 0xE3

**#define** Si7021\_HUM\_REG 0xE5

**#define** Si7021\_ADDR 0x40;

**static** Display\_Handle display;

/\*

\* ======== mainThread ========

\*/

**void** \***mainThread**(**void** \*arg0)

{

uint16\_t sample;

uint16\_t temperature,temperaturef;

uint8\_t txBuffer[1];

uint8\_t rxBuffer[2];

I2C\_Handle i2c;

I2C\_Params i2cParams;

I2C\_Transaction i2cTransaction;

/\* Call driver init functions \*/

Display\_init();

**GPIO\_init**();

**I2C\_init**();

/\* Configure the LED and if applicable, the TMP116\_EN pin \*/

**GPIO\_setConfig**(Board\_GPIO\_LED0, GPIO\_CFG\_OUT\_STD | GPIO\_CFG\_OUT\_LOW);

**#ifdef** Board\_GPIO\_TMP116\_EN

**GPIO\_setConfig**(Board\_GPIO\_TMP116\_EN, GPIO\_CFG\_OUT\_STD | GPIO\_CFG\_OUT\_HIGH);

/\* 1.5 ms reset time for the TMP116 \*/

sleep(1);

**#endif**

/\* Open the HOST display for output \*/

display = Display\_open(Display\_Type\_UART, NULL);

**if** (display == NULL) {

**while** (1);

}

/\* Turn on user LED \*/

**GPIO\_write**(Board\_GPIO\_LED0, Board\_GPIO\_LED\_ON);

Display\_printf(display, 0, 0, "Starting the i2ctmp example.");

/\* Create I2C for usage \*/

**I2C\_Params\_init**(&i2cParams);

i2cParams.bitRate = *I2C\_400kHz*;

i2c = **I2C\_open**(Board\_I2C\_TMP, &i2cParams);

**if** (i2c == NULL) {

Display\_printf(display, 0, 0, "Error Initializing I2C\n");

**while** (1);

}

**else** {

Display\_printf(display, 0, 0, "I2C Initialized!\n");

}

/\* Common I2C transaction setup \*/

i2cTransaction.writeBuf = txBuffer;

i2cTransaction.writeCount = 1;

i2cTransaction.readBuf = rxBuffer;

i2cTransaction.readCount = 2;

/\* Try Si7021 \*/

txBuffer[0] = Si7021\_TMP\_REG;

i2cTransaction.slaveAddress = Si7021\_ADDR;

**if** (!**I2C\_transfer**(i2c, &i2cTransaction)) {

/\* Could not resolve a sensor, error \*/

Display\_printf(display, 0, 0, "Error. No TMP sensor found!");

**while**(1);

}

**else** {

Display\_printf(display, 0, 0, "Detected Si7021 sensor.");

}

/\* Take 20 samples and print them out onto the console \*/

**for** (sample = 0; sample < 20; sample++) {

**if** (**I2C\_transfer**(i2c, &i2cTransaction)) {

/\*

\* Extract degrees C from the received data;

\* see Si7021 datasheet

\*/

temperature = (rxBuffer[0] << 8) | (rxBuffer[1]);

temperaturef = (((175.72 \* temperature)/ 65536) - 46.85);

Display\_printf(display, 0, 0, "Sample %u: %d (C)",

sample, temperaturef);

}

**else** {

Display\_printf(display, 0, 0, "I2C Bus fault.");

}

/\* Sleep for 1 second \*/

sleep(1);

}

**I2C\_close**(i2c);

Display\_printf(display, 0, 0, "I2C closed!");

**return** (NULL);

}