

SCC.369 Assessment 3 – Comms – Temperature Monitor

Aim of the Exercise

The aim of this exercise is to familiarize you with communications protocols and apply the I²C protocol to a realistic scenario. This exercise starts in WK5 and demonstrated in your Week 7 practical. If you have viewed all the blended lectures provided for WK1-5 and read the PIC16F1507 datasheet and instruction sheet, you have are in a good position to start. You will just need to familiarize yourself with one additional datasheet for the NCT75-D temperature sensor.

Software Configuration

You can develop the exercise initially in the MPLAB X simulator but remember that the simulator doesn't necessarily replicate all behaviour of a PIC. It definitely doesn't support a peripheral like the NCT75-D temperature sensor

You will need to use the default configuration for Configuration Words

Hardware

You are also expected to demo the code on the actual PIC in Week 7. So be prepared to wire-up the circuits correctly during development and demo phases.

Components

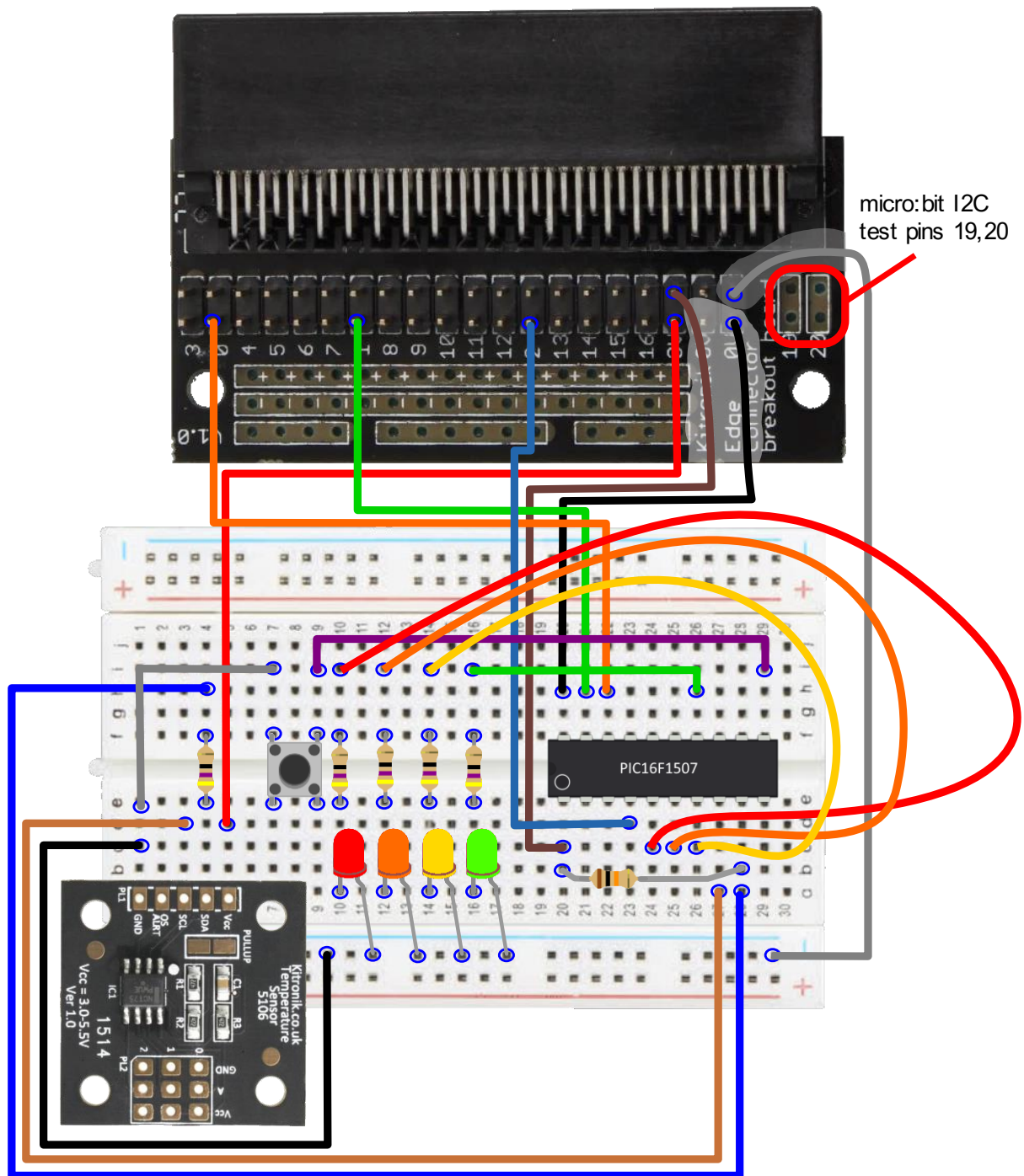
- 1 each of Red/Green/Yellow/Orange LEDs.
- 5 47R and one 10K resistors
- M2M, M2F Jumper wires and breadboard
- Micro:bit to flash program to PIC
- Kitronik Temperature Sensor breakout board #5106
- 1 Pushbutton

Wiring Details:

NOTE: Perform this step carefully. For sake of simplicity we provide a wiring list and a layout recommendation. DO NOT connect the wrong things in the wrong way.

Connector Type	Connector Colour	Connection (TS: Temperature Sensor)	Use
M2M	Brown	TS SCL to PIC PC6 (Pin 8) [in test mode to micro:bit 19]	I2C SCL
	Red	Red LED +ve to 47R to PIC RC5 (Pin 5)	LED
	Orange	Orange LED +ve to 47R to PIC RC4 (Pin 6)	LED
	Yellow	Yellow LED +ve to 47R to PIC RC3 (Pin 7)	LED
	Green	Green LED +ve to 47R to PIC RC2 (Pin 14)	LED
	Blue	TS SDA to 47R to PIC RC7 (Pin 9) [in test mode, TS SDA direct to micro:bit 20]	I2C SDA
	Violet	PBttn to PIC RB6 (Pin 11)	Push Button
	Grey	PBttn to TS GND	GND
	Black	TS GND to Breadboard -ve rail	GND
	Brown	micro:bit 3V to PIC VCC (Pin 1)	VCC
	Red	micro:bit 3V to TS Vcc	VCC
	Orange	micro:bit 0 to PIC ICSPCLK (Pin 18)	ICSP
	Green	micro:bit 1 to PIC ICSPDAT (Pin 19)	ICSP
	Blue	micro:bit 2 to PIC MCLR (Pin 4)	ICSP
	Grey	micro:bit 0V to Breadboard -ve rail	GND
	Black	micro:bit 0V to PIC GND (Pin 20)	GND

Wiring Diagram



Task Description

The task is somewhat simple. Monitor the temperature of the surroundings by querying the temperature sensor breakout board. Based on the values read, indicate the temperature using the four LEDs provided.

As before, style and comment your code properly as you go along. Remember - separate code into semantic blocks and comment every block of code and subroutine. We'll be marking on how elegant, clean, and efficient your code is. Do your best to make it perfect. Have fun, and don't forget to document everything you discover in your lab book!

Task 0 : I2C and NCT75-D Basics

- 1) Read up on the I2C lecture slides and watch the lecture video.
- 2) Read the annotated datasheet provided on Moodle for the NCT-75D (the actual chip on the Kitronick 5106 breakout board).
- 3) Test the sensor operation using the micro:bit as a test system (see instructions at end of this document).
- 4) Sensor responds to I2C address 0x48. So, the PIC asserts a start condition and then the first byte written by the PIC is the address plus the R/W bit. This will be followed by two bytes of read, MSB first and in each case, a 9th ACK bit sent by the PIC. Finally, the PIC needs to assert a stop condition.

Task 1: Read and write

- 1) Read the temperature from the sensor. You will have to write the code required to setup the pins for I2C communication and then request the data from the sensor.
- 2) The returned data (2-bytes) will need to be stored in two registers and translates to the temperature as per the table below (and note the bit positions in the table heading):

Table 6. 12-BIT TEMPERATURE DATA FORMAT

Temperature	Binary Value D15 to D4	Hex Value
-55°C	1100 1001 0000	0xC90
-25°C	1110 0111 0000	0xE70
-0.0625°C	1111 1111 1111	0xFFF
0°C	0000 0000 0000	0x000
+0.0625°C	0000 0000 0001	0x001
+25°C	0001 1001 0000	0x190
+75.25°C	0100 1011 0100	0x4B4
+100°C	0110 0100 0000	0x640
+125°C	0111 1101 0000	0x7D0

- 3) Display the read value in a suitable way, using the following LED statuses:

Temperature	Green	Yellow	Orange	Red
< 15°C	Blink	Off	Off	Off
15.0625-20	On	Off	Off	Off
20.0625-25	On	On	Off	Off
25.0625-30	Off	On	Off	Off
30.0625-35	Off	On	On	Off
35.0625-37			On	
37.0625-39			On	On
39.0625-40				On
> 40°C				Blink

- 4) Task 1 is default mode at startup. The readings should be taken 4 times a second. You are free to select a blink rate for under-temp and over-temp situations (<15 and >40) as long as individual blinks are identifiable and the behaviour is identifiable as blinking (i.e. not too fast and not too slow).

Task 2: Store min and max values of temperature

- 1) You need to store the max and min values observed since start.
- 2) At each reading, compare if the reading is higher than previously noted max value. If so, save the current reading as the new max value. If the reading is lower than previously noted min value, save the current reading as the new min value.

Task 3: Display min, max and current value on demand

Use the push-button on PB6 to display the min, max and current values as follows:

- 1) When PB6 is pressed and held down for 2s, all LEDs blink once to indicate display mode is altered.
- 2) If PB6 continues to be held down for another 2s, the Green LED blinks once and then after a suitable delay, the stored min-value is displayed (following the display logic used in Task 1).
- 3) If PB6 continues to be held down for another 2s, the Red LED blinks once and then after a suitable delay, the stored max-value is displayed (following the display logic used in Task 1).
- 4) If PB6 continues to be held down for another 2s, all LEDs blink once and then current reading is displayed.
- 5) If PB6 is held down even longer, steps 2 to 4 above are repeated.
- 6) If PB6 is released at any point, 2s after the release, all LEDs blink once and then the program returns to Task 1 (without losing the previously saved min-max values).

Assessment

To gain marks for this assessment, you need to:

- a. Complete the individual tasks up to specification.
- b. Demonstrate quality of implementation through the code produced the comments written.
- c. Submit the code by the Week 6 deadline (WK6 Friday 4pm).
- d. Present a working practical demonstration in Week 7 during your own lab session.
- e. Demonstrate understanding of PIC fundamentals through code walkthrough of your work.

Remember: NO THIRD-PARTY CODE ALLOWED. (But it's okay to use the examples from previous weeks as a starting point)

Assessment Weights

Task	Task Description	Weights
1	Read and write	60%
2	Store min and max values of temperature	10%
3	Display min, max and current value on demand	30%

In all cases a grade descriptor (A, B, C, D, F) will be used to mark your work in each category.

NOTE:

Both code submission (due WK6 Friday 4pm) and demo/attendance during marking slot in WK7 are essential for grading. Missing either of these without a legitimate extension will result in a 0 or F4 for this piece of coursework.

IMPORTANT:

There aren't enough spares of the Temperature Sensor to replace them easily. Follow the provided wiring diagram and other information CAREFULLY before wiring the sensor up. If in doubt, ASK FIRST! If you burn out the sensor, we may not be able to provide alternative options.

Micro:bit I2C

To perform an initial test of the Temperature sensor (and to check if it has been damaged subsequently), please follow these steps:

Wiring:

- 1) Disconnect the micro:bit from the PC to power off the circuit.
- 2) Remove the wiring to SDA and SCL (if already wired)
- 3) Pin 19 on Micro:bit breakout board (holes on the side where the 0V pins are) to SCL of sensor using male-to-male connector
- 4) Pin 20 on Micro:bit breakout board (same as above), to SDA of sensor using male-to-male connector

Program the Micro:bit

1. Download the hex file from Moodle into your Micro:bit. (You can view the block-code for it here: https://makecode.microbit.org/_hfb2TfUXe8w6)

2. If download was successful, the LED matrix on the Micro:bit should show a heart-beat display.

3. Open up Terminal (Linux only) and enter the command:

```
screen /dev/ttyACM[number] 115200
```

This should start displaying a repeating message "Press 'A' to read temp".

4. For Windows/Mac, you can try the default Serial application, selecting the correct port and 115200 baud rate. For e.g. Arduino IDE comes with a Serial monitor which can help look at the serial data.

5. When you press 'A' on the Micro:bit, a value (something between 10-30 depending on how cold it is) should be displayed in the Serial feed. Note this value will appear as a decimal number as it reads the two bytes from the sensor (MSB first, hence BigEndian), divides by 256 (total 8-bit shift, 4 for insignificant LSBs of lower byte, then another 4 bit shifts to convert number to actual temperature in decimal form)

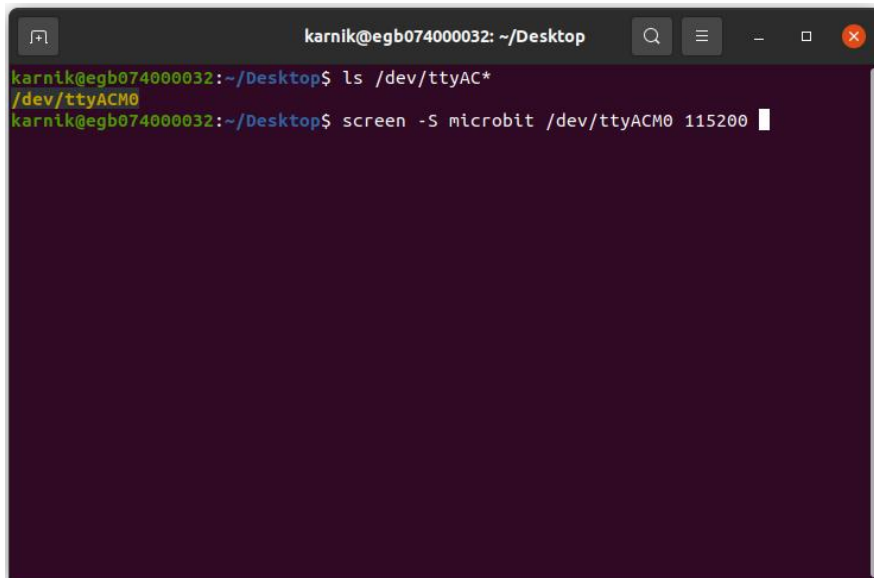
6. If you hold the sensor over a heat source (like radiator) or touch it (take care that finger is not wet/damp and static-free [touch floor before touching chip]), it should gradually show increase in temperature readings.

IMPORTANT:

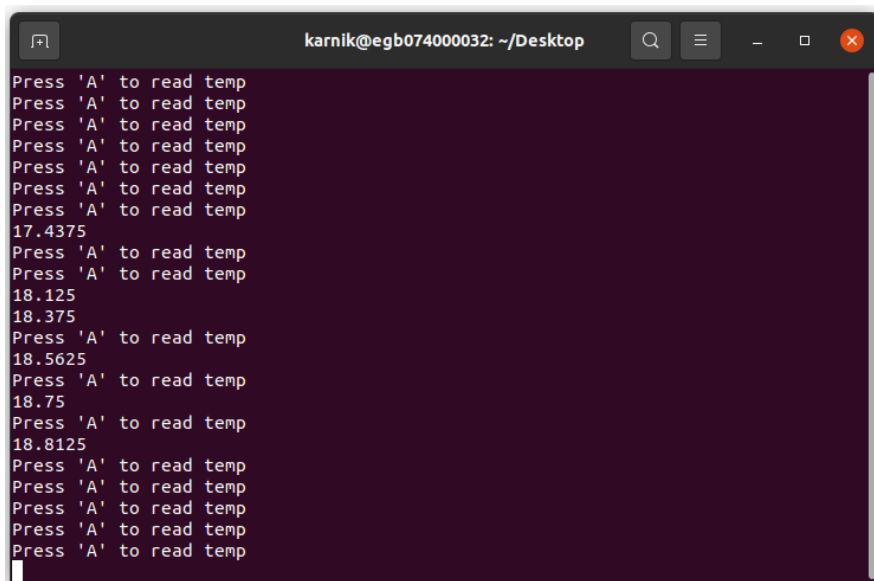
For your actual assignment, you will need to write your own I2C implementation using the chip. The above steps are just a demonstration for confirming that the sensor is still working fine.

Sample screenshots for Step 3 from “Program the Microbit”

Note the commands required to identify which port the Microbit is on and then actually attach a screen session to it

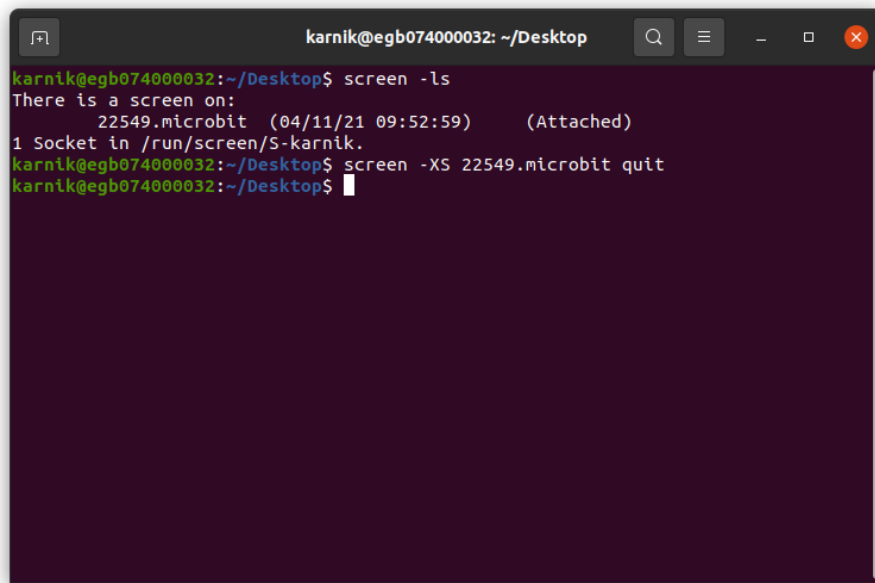


```
karnik@egb074000032: ~/Desktop
karnik@egb074000032:~/Desktop$ ls /dev/ttyAC*
/dev/ttyACM0
karnik@egb074000032:~/Desktop$ screen -S microbit /dev/ttyACM0 115200
```



```
Press 'A' to read temp
Press 'A' to read temp
Press 'A' to read temp
Press 'A' to read temp
Press 'A' to read temp
Press 'A' to read temp
Press 'A' to read temp
17.4375
Press 'A' to read temp
Press 'A' to read temp
18.125
18.375
Press 'A' to read temp
18.5625
Press 'A' to read temp
18.75
Press 'A' to read temp
18.8125
Press 'A' to read temp
Press 'A' to read temp
Press 'A' to read temp
Press 'A' to read temp
Press 'A' to read temp
```

Also, to terminate an old screen session, or the current one, you will need to use the syntax as shown here:



```
karnik@egb074000032: ~/Desktop
karnik@egb074000032:~/Desktop$ screen -ls
There is a screen on:
      22549.microbit  (04/11/21 09:52:59)  (Attached)
1 Socket in /run/screen/S-karnik.
karnik@egb074000032:~/Desktop$ screen -XS 22549.microbit quit
karnik@egb074000032:~/Desktop$
```

Sample screenshots for Step 4 from “Program the Microbit”
Select the correct port and correct baud rate for this to work.

