Connecting and testing the device

Connecting the device

You were previously asked to identify the default I2C pins on your micro-controller, called I2C1_SCL and I2C1_SDA in the pin layout; they are names D14 and D15 on the board, on the top-right section of the board.

While your board is disconnected from the computer, use the male end of the jumper wires to connect VCC to the 3.3V pin of the board, and GND to GND. Then connect SDA and SCL to the pins D14 and D15, respectively, in order to use the default I2C pins.

Testing that the device is able to communicate and check its address

The code below will help you test communications with your device. It scans all I^{2}C addresses on the bus connected to the default I^{2}C pins. For each address, it tries to read something. If it manages to find a device, it returns the address on the serial port so that you can catch it on your terminal. It also uses led flashes to communicate visually the address.

Task

Paste the code below in a new project on the mbed compiler. Compile it and place it on your board. Check the that device address returned matches to address set to the device after soldering the pads.

```
#include "mbed.h"

// Create an I2C object on the defeault pins D15 and D14
I2C i2c(I2C_SDA, I2C_SCL);
// There are other pins that can be used for I2C the above is the default bus
// See the board pinout on the mbed webpage for more details.


// Use the built in serial bridge to output
Serial pc(USBTX, USBRX);


// Some flashy lights
DigitalOut green(LED1);
DigitalOut blue(LED2);
DigitalOut red(LED3);


// Create a timer so we can time the bus scan
Timer t;

// Buffer for read data
char read_data[2];

// A variable for counting
unsigned int i=0;

// For recording the address of the last device found
unsigned int address=0;

int main()
{
    green=0;
    blue=0;


        // Make sure you set your serial terminal to match this
    pc.baud(9600);

    // Most I2C devices can cope with 400 kHz bus speed.
    // If you have any problems reduce it to 100 kHz.
    // Try changing the I2C speed and seeing what the effect on the speed of the program is.

    i2c.frequency(400000);
    pc.printf("Starting Bus Scan....\r\n");

    // Reset and start timer
    t.reset();
    t.start();

    // Address '0' is all call and it is undefined how this would work
    // so we run from address 1 to 127

    for(i = 1; i < 128 ; i++)

        {
        // Read one byte from whatever the default read register is from every I2C address.
        // i2c.read returns the I2C ACK bit sent by the slave, if anyone answered the call:
        // 0 on success (ack), non-0 on failure (nack)

        // Note that while I2C addresses are from 1-127 we need to left-shift one bit
        // as the address sent on the I2C bus is 8bits
        // with the lowest but indicating if this is a write or read transaction.
```

```
        // The operation i<<1 does the bit shifting.

        if(i2c.read(i<<1, read_data, 1)==0)
            {

            // Print the address at which we found a device as a hex and as a decimal number
            pc.printf ("I2C device found at address Hex: %x Decimal: %d\r\n",i,i);

            // If we find one device at least light the green LED and save its address
            green=1;
            address=i;
            }
        // Flash the blue LED to show we are scanning - only slow if no devices connected
        blue=!blue;
        }

    // Stop the timer and report time to scan
    t.stop();
    pc.printf("Bus scanned in %d ms\r\n",t.read_ms());

    // If device not found flash both red & blue LEDs
    if (address==0)
        {
            red=0;green=0;blue=1;
            while(1)
                {
                red=!red;
                blue=!blue;
                wait(0.25);
                }
            }

    // If we find at least one device
    // Flash address using LEDs:
    // Red flashes first digit and blue second

    red=0;      blue=0;

    while(1)
        {
        wait(2);
        for (i=0;i<(address/16);i++)
            {
            wait(0.25);
            red=1;
            wait(0.25);
            red=0;
            }
        wait(0.5);
        for (i=0;i<(address%16);i++)
            {
            wait(0.25);
            blue=1;
            wait(0.25);
            blue=0;
            }
        }
}
```

Getting your first temperature measurements

Task

Start a new project, and select the basic template *"mbed OS Blinky LED Hello World"*. **Replace the content of main.cpp with the code below. Compile it and try it on your board. Hold the sensor between your fingers, and monitor the evolution of the temperature.**

You will need to catch the serial output to read the temperature, as explained in the corresponding tutorial section: Communications between the computer and the microcontroller

Use this code as it is, without necessarily trying to understand it at this point. The next section will describe in detail how to communicate with the sensor and extract relevant information from the data sheet, if you are not familiar with this yet.

```
#include "mbed.h"

#define LM75_REG_TEMP (0x00) // Temperature Register
#define LM75_REG_CONF (0x01) // Configuration Register
#define LM75_ADDR     (0x90) // LM75 address

I2C i2c(I2C_SDA, I2C_SCL);

DigitalOut myled(LED1);

Serial pc(SERIAL_TX, SERIAL_RX);

volatile char TempCelsiusDisplay[] = "+abc.d C";

int main()
{

        char data_write[2];
        char data_read[2];

        /* Configure the Temperature sensor device STLM75:
        - Thermostat mode Interrupt
        - Fault tolerance: 0
        */
        data_write[0] = LM75_REG_CONF;
        data_write[1] = 0x02;
        int status = i2c.write(LM75_ADDR, data_write, 2, 0);
        if (status != 0) { // Error
                while (1) {
                        myled = !myled;
                        wait(0.2);
                }
        }

        while (1) {
                // Read temperature register
                data_write[0] = LM75_REG_TEMP;
                i2c.write(LM75_ADDR, data_write, 1, 1); // no stop
                i2c.read(LM75_ADDR, data_read, 2, 0);

                // Calculate temperature value in Celcius
                int tempval = (int)((int)data_read[0] << 8) | data_read[1];
                tempval >>= 7;
                if (tempval <= 256) {
                        TempCelsiusDisplay[0] = '+';
                } else {
                        TempCelsiusDisplay[0] = '-';
                        tempval = 512 - tempval;
                }

                // Decimal part (0.5°C precision)
                if (tempval & 0x01) {
                        TempCelsiusDisplay[5] = 0x05 + 0x30;
                } else {
                        TempCelsiusDisplay[5] = 0x00 + 0x30;
                }

                // Integer part
                tempval >>= 1;
                TempCelsiusDisplay[1] = (tempval / 100) + 0x30;
                TempCelsiusDisplay[2] = ((tempval % 100) / 10) + 0x30;
                TempCelsiusDisplay[3] = ((tempval % 100) % 10) + 0x30;
```

```
                // Display result
                pc.printf("temp = %s\n", TempCelsiusDisplay);
                myled = !myled;
                wait(1.0);
        }

}
```