LM75 sensor and interrupts

Hardware interrupt

Read section 7.1 of the datasheet. The OS pin of the sensor is useful to let the device monitor temperature in the background, and send a signal when the temperature exceeds a threshold, Tos. A hysteresis temperature, Thyst, is also defined to avoid noisy signal in the interrupt pin. Fig 6 of the datasheet shows how the OS pin would be controlled for a certain temperature input. There are two different modes of operation for the interrupt in the device, the interrupt mode and the comparator mode. They are selected using the second bit (B1) of the Config register.

The code below illustrates the use of interrupts. We use here the interrupt mode, but the same task can be achieved with the comparator mode.

Start a new project, and paste this code in. The sensor needs to be connected as previously, with in addition the OS pin linked to the pin D7 of the microcontroller.

The code contains a couple of new elements:

- the address of the registers TOS and THYST are introduced, as well as some code to set the interrupt and hysteresis temperatures. The code essentially does the opposite of what we did to read the temperature. There are only 9 meaningful bits for these registers; the operation " & 0xFF80" is a bitwise AND operation on the 16-bit of data and the binary number "1111111110000000"; it essentially makes sure that we set to 0 the 7 least significant bits of the i16 variable.
- The interrupt pin is active when its value is low, so we should trigger the interrupt when OS goes from high to low. We therefore set the interrupt using the function "fall" rather than "rise" as introduced in the previous activity.

When the code is running, you should be able to raise the temperature enough with your fingers to trigger the interrupt and turn the blue led on. As the sensor cools down, as new interrupt is triggered once the temperature goes below 26 degree Celsius, turning the blue led off. Each time an interrupt is triggered, a red led should also flash on the sensor, indicating the state of the OS pin (led is on when OS is low).

```
#include "mbed.h"
#include "stdint.h" //This allow the use of integers of a known width
#define LM75_REG_TEMP (0x00) // Temperature Register
#define LM75_REG_CONF (0x01) // Configuration Register
#define LM75_ADDR     (0x90) // LM75 address

#define LM75_REG_TOS (0x03) // TOS Register
#define LM75_REG_THYST (0x02) // THYST Register



I2C i2c(I2C_SDA, I2C_SCL);

DigitalOut myled(LED1);
DigitalOut blue(LED2);

InterruptIn lm75_int(D7); // Make sure you have the OS line connected to D7

Serial pc(SERIAL_TX, SERIAL_RX);

int16_t i16; // This variable needs to be 16 bits wide for the TOS and THYST conversion to work
- can you see why?

void blue_flip()
{
        blue=!blue;
        // The instruction below may create problems on the latest mbed compilers.
        // Avoid using printf in interrupts anyway as it takes too long to execute.
        // pc.printf("Interrupt triggered!\r\n");
}

int main()
{
        char data_write[3];
        char data_read[3];

        /* Configure the Temperature sensor device STLM75:
           - Thermostat mode Interrupt
           - Fault tolerance: 0
           - Interrupt mode means that the line will trigger when you exceed TOS and stay
triggered until a register is read - see data sheet
        */
        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);
                }
        }

        float tos=28; // TOS temperature
        float thyst=26; // THYST tempertuare

        // This section of code sets the TOS register
        data_write[0]=LM75_REG_TOS;
        i16 = (int16_t)(tos*256) & 0xFF80;
        data_write[1]=(i16 >> 8) & 0xff;
        data_write[2]=i16 & 0xff;
        i2c.write(LM75_ADDR, data_write, 3, 0);
        //This section of codes set the THYST register
        data_write[0]=LM75_REG_THYST;
        i16 = (int16_t)(thyst*256) & 0xFF80;
        data_write[1]=(i16 >> 8) & 0xff;
        data_write[2]=i16 & 0xff;
        i2c.write(LM75_ADDR, data_write, 3, 0);

        // This line attaches the interrupt.
        // The interrupt line is active low so we trigger on a falling edge
        lm75_int.fall(&blue_flip);

        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
                int16_t i16 = (data_read[0] << 8) | data_read[1];
                // Read data as twos complement integer so sign is correct
                float temp = i16 / 256.0;

                // Display result
                pc.printf("Temperature = %.3f\r\n",temp);
                myled = !myled;
                wait(1.0);
        }

}
```

Task (optional)

Could you program the same behaviour using the comparator mode of the sensor, instead of the interrupt mode? Hint: You need to change the value of the configuration register (see section 7.4.2 of the datasheet). The code below would set its relevant bit to select the comparator mode. You will also need to set different behaviours for the rising and falling edges of the interrupt pin.

```
data_write[0] = LM75_REG_CONF;
data_write[1] = 0x00;
int status = i2c.write(LM75_ADDR, data_write, 2, 0);
```