5 Two Wire Interface I2C Bus
Multiplexed displays on Arduino

1.Objective

Increasing familiarity and experimenting with two wire interface, and the concept of multiplexing.

2.Overview

2.1 I2C

The I2C bus was designed by Philips in the early '80s to allow easy communication between components which reside on the same circuit board. Philips Semiconductors migrated to NXP in 2006.
The name I2C translates into "Inter IC". Sometimes the bus is called IIC or I²C bus.
The original communication speed was defined with a maximum of 100 kbit per second and many applications don't require faster transmissions. For those that do there is a 400 kbit fast mode and - since 1998 - a high speed 3.4 Mbit option available. Recently, fast mode plus a transfer rate between this has been specified.
I2C is not only used on single boards, but also to connect components which are linked via cable. Simplicity and flexibility are key characteristics that make this bus attractive to many applications.
Most significant features include:

- Only two bus lines are required
- No strict baud rate requirements like for instance with RS232, the master generates a bus clock
- Simple master/slave relationships exist between all components
  Each device connected to the bus is software-addressable by a unique address
- I2C is a true multi-master bus providing arbitration and collision detection

2.2 Multiplexed displays

Multiplexed displays are electronic displays where the entire display is not driven at one time. Instead, sub-units of the display (typically, rows or columns for a dot matrix display or individual characters for a character orientated display, occasionally individual display elements) are multiplexed, that is, driven one at a time, but the electronics and the persistence of vision combine to make the viewer believe the entire display is

continuously active.
A multiplexed display has several advantages compared to a non-multiplexed display:

- Fewer wires (often, far fewer wires) are needed
- Simpler driving electronics can be used
- And both lead to reduced cost
- Reduced power consumption

3.Theory

3.1 I2C

- two-wired bus
  - serial data line (SDA)
  - serial clock line (SCL)
- bit transfer
  - level triggered
  - one clock pulse per data bit
  - stable data during high clocks
  - data change during low clocks

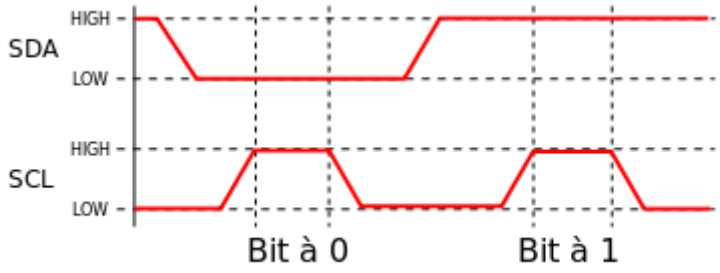

Figure 1- one clock pulse per data bit

3.2 Wire Library

This library allows you to communicate with I2C / TWI devices. On the Arduino boards with the R3 layout, the SDA (data line) and SCL (clock line) are A4 (SDA), A5 (SCL).

3.3 LM75B temperature sensor

The LM75B is a temperature-to-digital converter using an on-chip band gap temperature sensor and Sigma-Delta A-to-D conversion technique. The result of conversion is available via I2C serial connection. The principle of the sensor is that the forward voltage

VBE of a silicon diode is temperature-dependent. If VBE is measured for 2 different values IC1 and IC2 of the current, the variation DVBE is related to the temperature by:

$$\Delta V_{BE} = \frac{KT}{q} \cdot \ln\left(\frac{I_{C1}}{I_{C2}}\right)$$

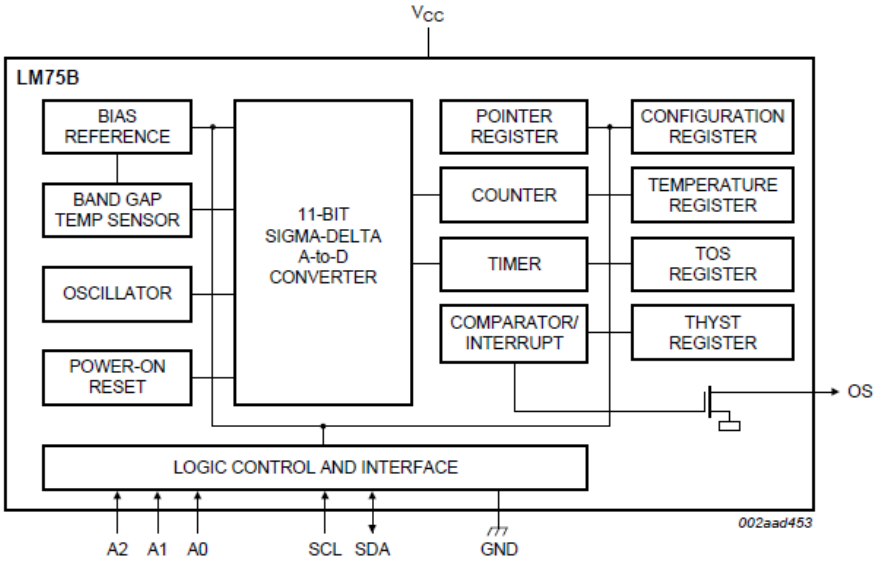

Figure 2 Address Clock Data

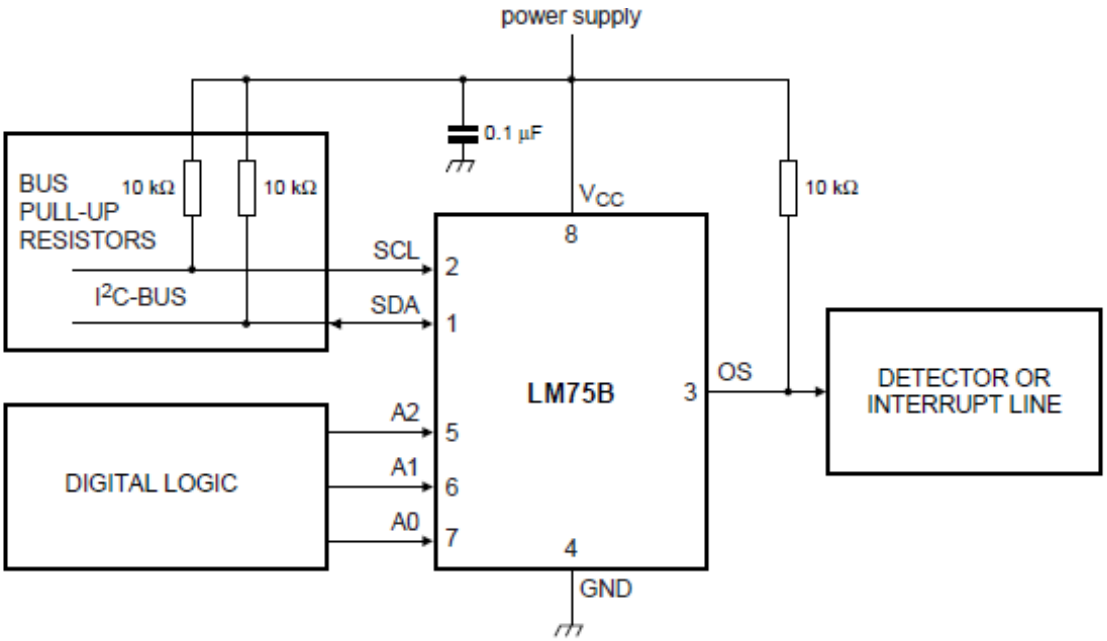

Figure 3 LM75B schematic

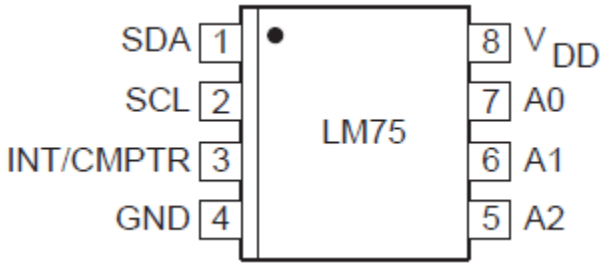

Figure 4 LM75B Pins

Pin No.	Symbol	Description
1	SDA	Bidirectional Serial Data
2	SCL	Serial Data Clock Input
3	INT/CMPTR	Interrupt or Comparator Output
4	GND	System Ground
5	A_2	Address Select Pin (MSB)
6	A_1	Address Select Pin
7	A_0	Address Select Pin (LSB)
8	V_{DD}	Power Supply Input

Figure 5 LM75B Pin Description

1	0	0	1	A2	A1	A0
MSB						LSB

Figure 6 LM75B Address

Address (A2, A1, A0) Inputs. Sets the three least significant bits of the LM758–bit address. A **match between the LM75's address and the address specified in the serial bit stream** must be made to initiate communication with the LM75.

3.4 7 seg display

A seven-segment display (SSD), or seven-segment indicator, is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot-matrix displays. Seven-segment displays are widely used in digital clocks, electronic meters, and other electronic devices for displaying numerical information.

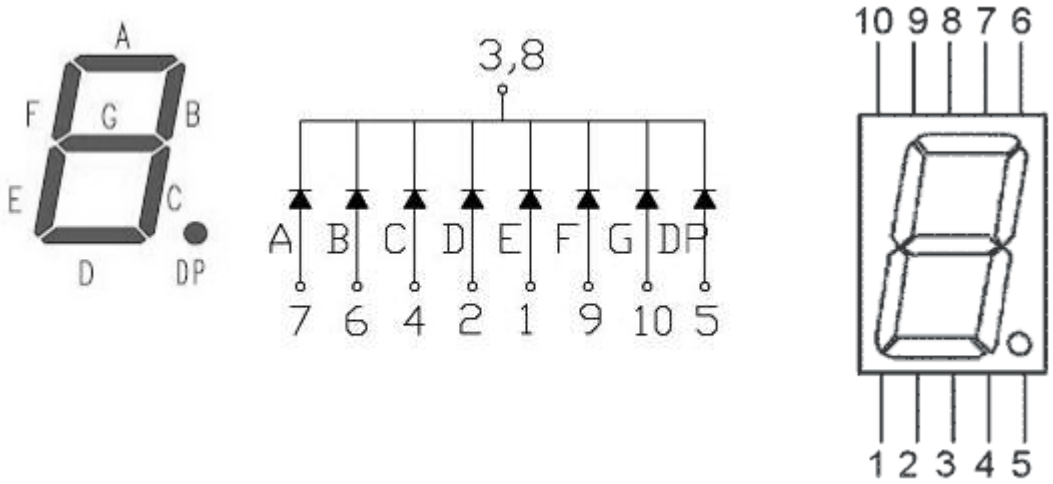

Figure 7 pin mappings of the display

4.Procedure

4.1 LM75B :

In this part you have to test the following example for the LM75B temperature sensor. Note that the LM75B Address not given and you have to find it using the theory information.

```
//Simple code for the LM75B, simply prints temperature via serial
#include <Wire.h>
int LM75BAddress = ??????;

void setup(){
  Serial.begin(9600);
  Wire.begin();  //communication start
}
```

```
void loop(){
  float celsius = getTemperature();
  Serial.print("Celsius: ");
  Serial.println(celsius);

  float fahrenheit = (1.8 * celsius) + 32;
  Serial.print("Fahrenheit: ");
  Serial.println(fahrenheit);

  delay(200); //just here to slow down the output. You can remove this
}

float getTemperature(){
  Wire.requestFrom(LM75BAddress,2);

  byte MSB = Wire.read();
  byte LSB = Wire.read();

  //it's a 12bit int, using two's compliment for negative
  int TemperatureSum = ((MSB << 8) | LSB) >> 4;

  float celsius = TemperatureSum*0.0625;
  return celsius;
}
```

4.2 7SEG

In this part you have to test the following example for the 7Seg. Note that not complete and you have to complete it.

```
// Arduino 7 segment display example software

// Define the LED digit patters, from 0 - 9
// Note that these patterns are for common anode displays
// Arduino pin: 2,3,4,5,6,7,8
```

```
byte seven_seg_digits[4][7] = {         { 0,0,0,0,0,0,1 },  // = 0
                                         { 1,0,0,1,1,1,1},  // = 1
                                       Complete it
                                              };
byte count=0;
void setup() {
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  writeDot(0);  // start with the "dot" off
}

void writeDot(byte dot) {
  digitalWrite(9, dot);

}

void sevenSegWrite(byte digit) {
  byte pin = 2;
  for (byte segCount = 0; segCount < 7; ++segCount) {
    digitalWrite(pin, seven_seg_digits[digit][segCount]);
    ++pin;
  }
}

void loop() {
  for (byte count = 10; count > 0; --count) {
   delay(1000);
   sevenSegWrite(count - 1);
  }
  delay(4000);
}
```

4.3 **TODO**

Try to build your own project, which will be able to read the temperature and print out the result on three 7Seg as the following format **dd.d** "**digit digit dot digit**".

Lab Report

1. Report as described in the Lab Policies.
2. Screen Shots of the "Block Diagram" and "Front Panel" windows where only these windows and their title bars and nothing else are visible. Points will be deducted if the components are not clearly visible in these windows or of any of the Windows screen components like the "Start Button" or "Task Bar" are visible.