Introduction to I^2C Communication

Materials:

- 2 Series Mixed Series Oscilloscope (MSO)
- Arduino UNO
- Temperature Sensor
- Breadboard
- Jumper wires

Procedure:

1. I^2C (Inter-Integrated Circuit) is a synchronous, multi-device, serial communication bus. It uses two bidirectional lines: the Serial Data Line (SDA) and the Serial Clock Line (SCL). Devices connected to the I^2C bus can act as either controllers or receivers. The controller initiates communication by generating a start condition, followed by the address of the target device. Data is then transmitted in bytes, with each byte being acknowledged by the receiver. Communication ends with a stop condition. For this lab, the Arduino UNO will be the controller and a temperature sensor will be the receiver.

2. Connect the temperature sensor to the Arduino by connecting the following pins:
    - Vin on the temperature sensor to 5V on the Arduino UNO
    - GND on the temperature sensor to GND on the Arduino UNO
    - SCL on the temperature sensor to SCL on the Arduino UNO
    - SDA to the temperature sensor to SDA on the Arduino UNO

3. To decode the messages sent by the Arduino and temperature sensor connect the channel 1 probe on the 2 Series MSO to the clock signal (SCL) and the channel 2 probe to the data signal (SDA). Activate channels 1 and 2 by pressing the colored channel buttons on the right panel on the 2 Series MSO. Add a bus channel by pressing the “Bus” button. On the bus menu, select I2C under “Bus Type”. Follow the rest of the configuration settings shown in Figure 1. to set up the bus. Exit the menu and double tap on the screen to bring up the “WAVEFORM VIEW” menu. Select “Stacked” under “Display Mode” to see both input channels and the bus.

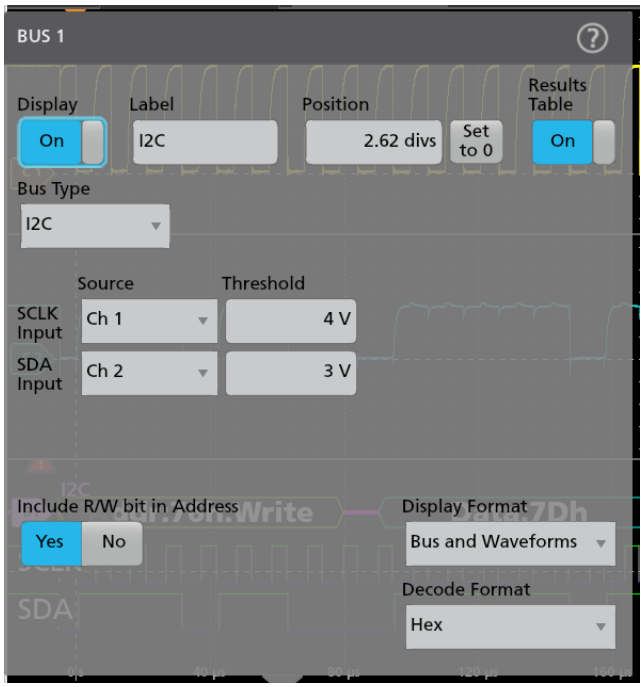

Figure 1. Bus configuration for I^2C input.

4. Write an Arduino script to read the temperature measurement from the sensor ever second. Make sure to first address the sensor, point to the correct register and then receive data back from the sensor. Note: refer to the temperature sensor's datasheet to calculate the temperature from the data received from the sensor. Add a serial output of the temperature to help validate the data received.

5. Use the 2 Series MSO to decode the messages sent to and from the temperature sensor. The oscilloscope and decode the messages in either hexadecimal or binary based on the "Decode Format" selected in the bus menu. Compare the data decoded on the oscilloscope to the temperature listed in the serial output. Record three sets of data in Table 1.

Data Type	Data Set 1	Data Set 2	Data Set 3
Hexadecimal			
Binary			
Serial Output			

Table 1. I^2C decoded readings from temperature sensor.

Instructor Notes:

Figure 2: Example of decorded I^2C signals on oscilloscope.

Example code for communication with temperature sensor via I^2C. The temperature sensor used is a BME280 sensor and has its own library available for download in the Arduino IDE.

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

Adafruit_BME280 bme;

void setup() {
 Serial.begin(9600);

 if (!bme.begin(0x76)) {
  Serial.println("Could not find a valid BME280 sensor");
  while (1);
 }
}

void loop() {
 Serial.print("Temperature = ");
 Serial.print(bme.readTemperature());
 Serial.println("*C");
 Serial.println();
 delay(1000);
}
```