Requirements Specification for EE465 Lab Project 1: Real Time Clock and I2C

Lab project goal: Interface to a real time clock using the MSP430 and a real time clock.

This lab will introduce you to "bit banging" I2C on any microcontroller.

This lab will apply the input/output setup as well as the timing techniques we've explored in previous labs to use I2C protocol. The goal of the lab will be to communicate with an external device using only two pins, SCL as the clock line and SDA as your data line. To view these signals, we'll be using the Mixed Signal Oscilloscope's bus analysis feature. The MSO is capable of identifying I2C protocols in real time and outputting both the values it sees as well as how they are interpreted.

Outcomes:

After this lab you should be able to:

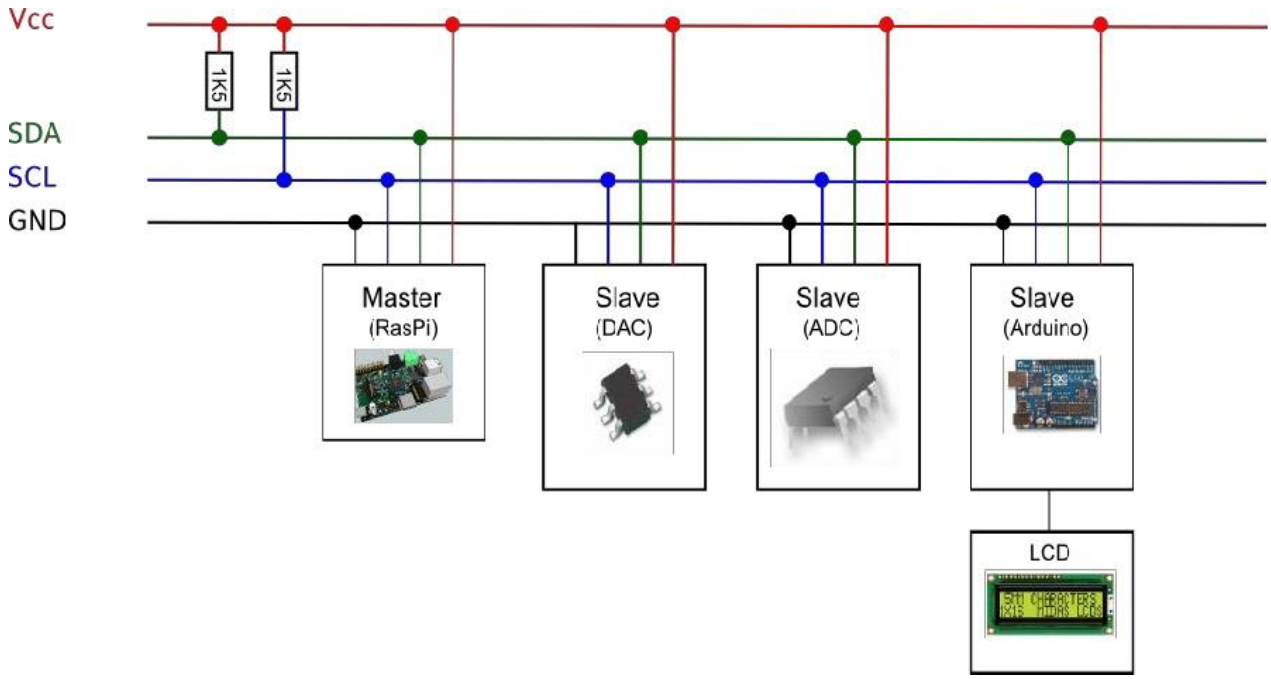
- Setup the MC9S08QG8 microcontroller for general purpose I/O (GPIO) on PORT XX
- Implement I2C addressing using "Bit Banging" to communicate with other devices in synchronous serial communication.
- Implement a counter and send the results over I2C for display on the oscilloscope.
- Implement a complete I2C packet including start, addressing, acknowledges, data and stop using "Bit Banging" to communicate with other devices in synchronous serial communication.
- Implement writes and reads to a DS3231 Real Time Clock
- Setup and Use the Mixed Signal Oscilloscope's (MSO) bus analysis feature to observe I2C signals coming out of the MSP430 Launchpad Board.

Requirements for lab project completion:

Setup the MSP-EXP430FR2355 Launchpad Eval Board and Code Composer on either a lab computer or your laptop. Program this lab in assembly.

For this lab use two of the I/Os (you pick) on the Launchpad board for the SCL and Data signals..

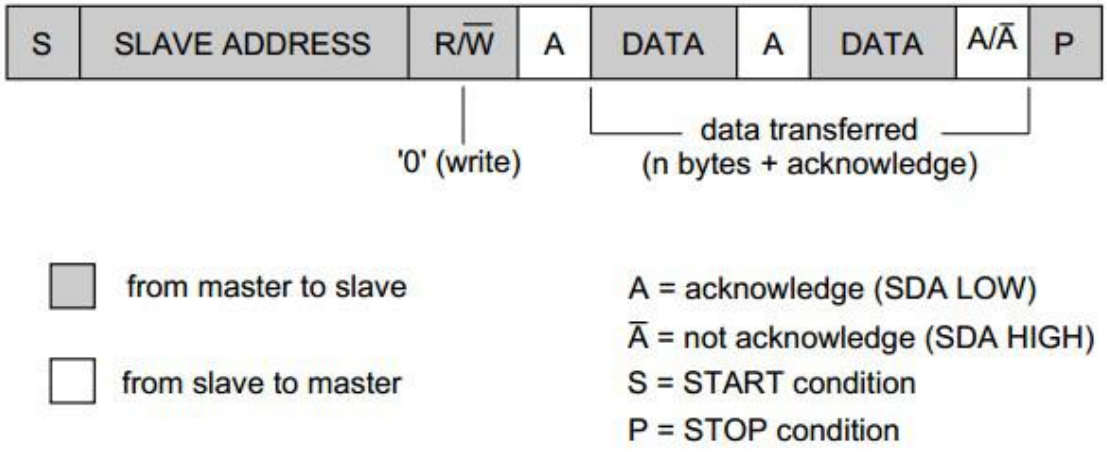
I2C protocol has some general guidelines that you must follow when sending data. It is not as strict as serial protocols such as CAN or Asynchronous serial protocols and is commonly used for short distance control systems. To use I2C you have to manipulate the SCL data lines and SDA data lines using a master device. The standard I2C system is shown below.



The master controls the SCL line of communication for all of the devices. The SCL line is a clock line that uses a square-wave to control the rate of data transfer to and from the master device. The master chooses which of the slave devices it is communicating with by first sending an address and then waiting for a response in the form of an acknowledge from the appropriate device. For nonprogrammable hardware, they will have an address assigned to them that is made up of 7 bits that can be found in the device's data sheets. For programmable hardware, you can program any address you wish into the device. The master always has control of the SCL line. The SDA line however will vary control based on the operation. If the master wants to send data to a device, it will drive the SDA line as an output. If the master needs to receive data from a slave, it will relinquish control of the line by switching SDA to an input for itself and wait to receive data. For example, if the master device had a slave thermometer and a slave heating device, the master would operate SDA as an input to receive the temperature of a device, then if the temperature needs to be changed, it will operate SDA as an output and tell the heating element to kick on.

Often master-slave systems will be made up of one master device and several slave devices.

When using I2C, there is a specific order signals must be sent and received, this lab will focus purely on sending signals in the proper format. The basic format is shown below.

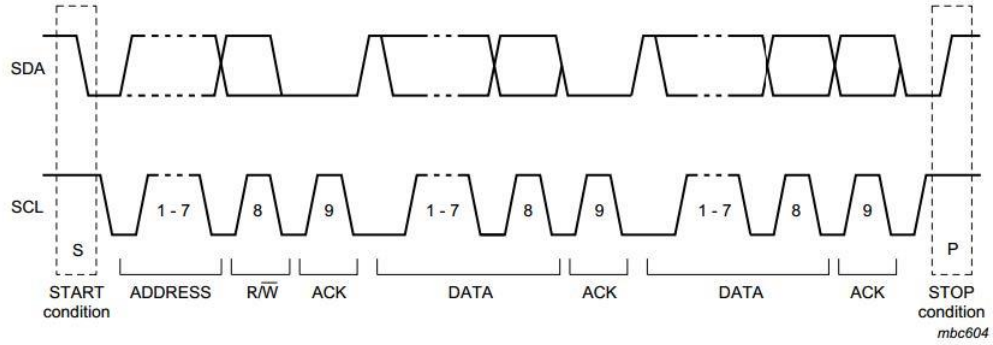


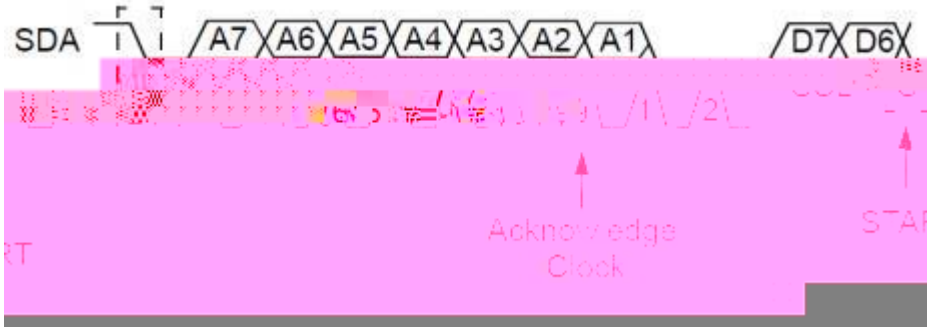
The general steps for using I2C:

1. **Start Condition** - This indicates to all of the devices that an address is about to be sent and to "wake-up" and watch the SDA and SCL lines. While SCL and SDA are high, SDA is pulled to a logic low.
2. **Address -** This tells the slave devices, which device the master is reaching out to communicate with. Addresses are 7 bits long and sent MSB first.
3. **R/W-** This bit tells the slave device whether it is to send or receive information. If a write is sent (0), the slave knows the master is sending information. If a read is sent (1) then the slave knows it is being asked to send data.
4. **Acknowledge-** This is sent from the slave device to the master device that indicates that the slave received and understood the information being sent. This is done after each byte of data is sent. If there is an issue with the data, the slave device will send a "not acknowledge" instead. Depending on how the system is written, a not acknowledge can cause different things to happen such as the previous byte being resent or the system stopping and starting over completely.
5. **Data-** This area is made up of eight bits of data. After each 8 bits sent, an acknowledge is expected.
6. **Stop Condition -** The slave will continue to expect to send or receive data until a stop condition is sent. A stop condition is caused by SDA being pulled from low to high while SCL is high.

Line Control and sending bits:

The default state for the SCL and SDA lines while they are not in use is SDA high and SCL high. When the device is in use, these lines will be pulled to low.

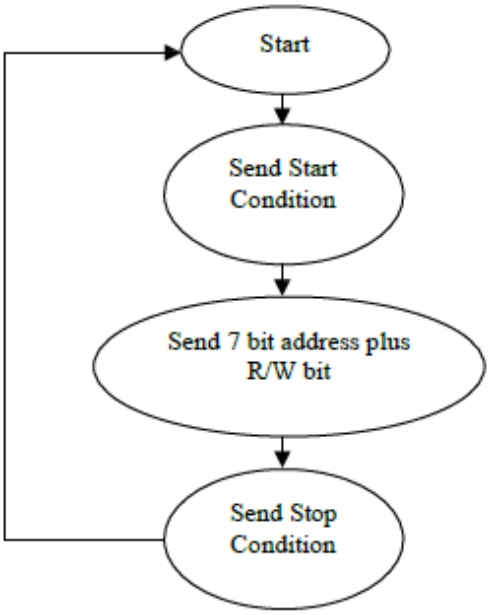




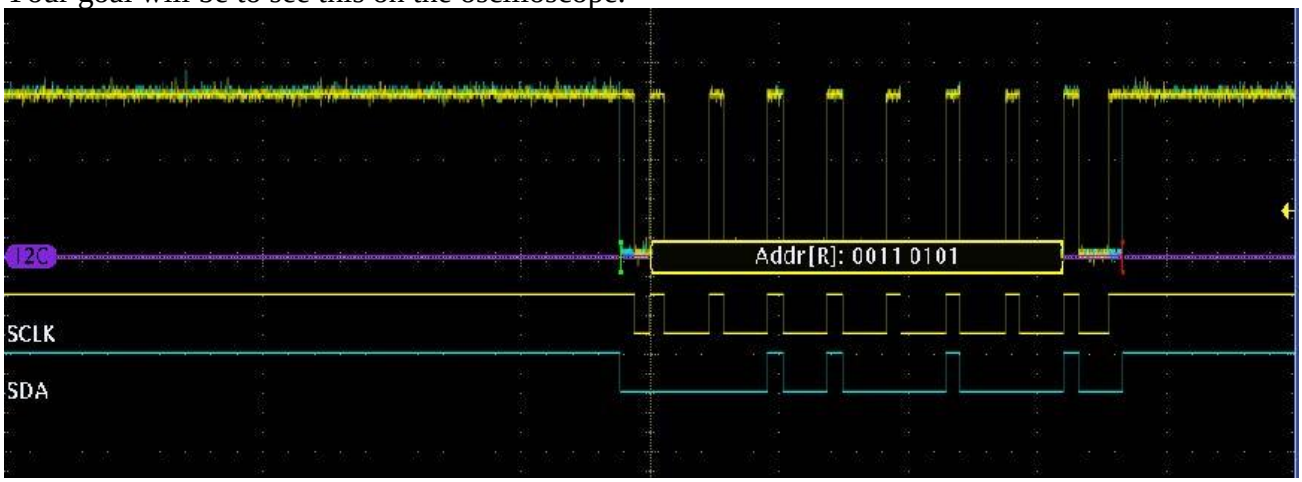
This bit-by-bit figure shows the line status when sending data. The clock line (SCL) is used to indicate a sampling period for the slave or master devices. Once SDA has set the appropriate value, SCL will move from low to high. After a short delay (1ms or less) SCL will move back low and indicate the end of sending a bit. Because of this process, the clock pulses will be smaller than the SDA pulses, this is to be expected. The screenshot below shows the oscilloscope readout of this lab.
You can see that the address and data chunks are identified by the oscilloscope if the I2C protocol is sent correctly. If you do not see the I2C translation on the oscilloscope, then data is not being sent in the correct format.

For this lab, you will be sending only an address using Port X SCL as the clock line and Port X SDA as your data line. Port X will be setup in such a way that only those bits are affected. You will not need to use pullup resistors in this I2C simulation. Once the ports are set to outputs, we will send just an address to the oscilloscope accompanied with start and stop conditions. If done properly, you will get the I2C translation showing on the oscilloscope. A walkthrough on using the bus reader on the oscilloscope will be in the appendix of this lab.

Use Application Note AN 1820 as your starting point for this lab. Read thru the application note and the code given at the end of the application note.
Here is a flowchart to get you started.



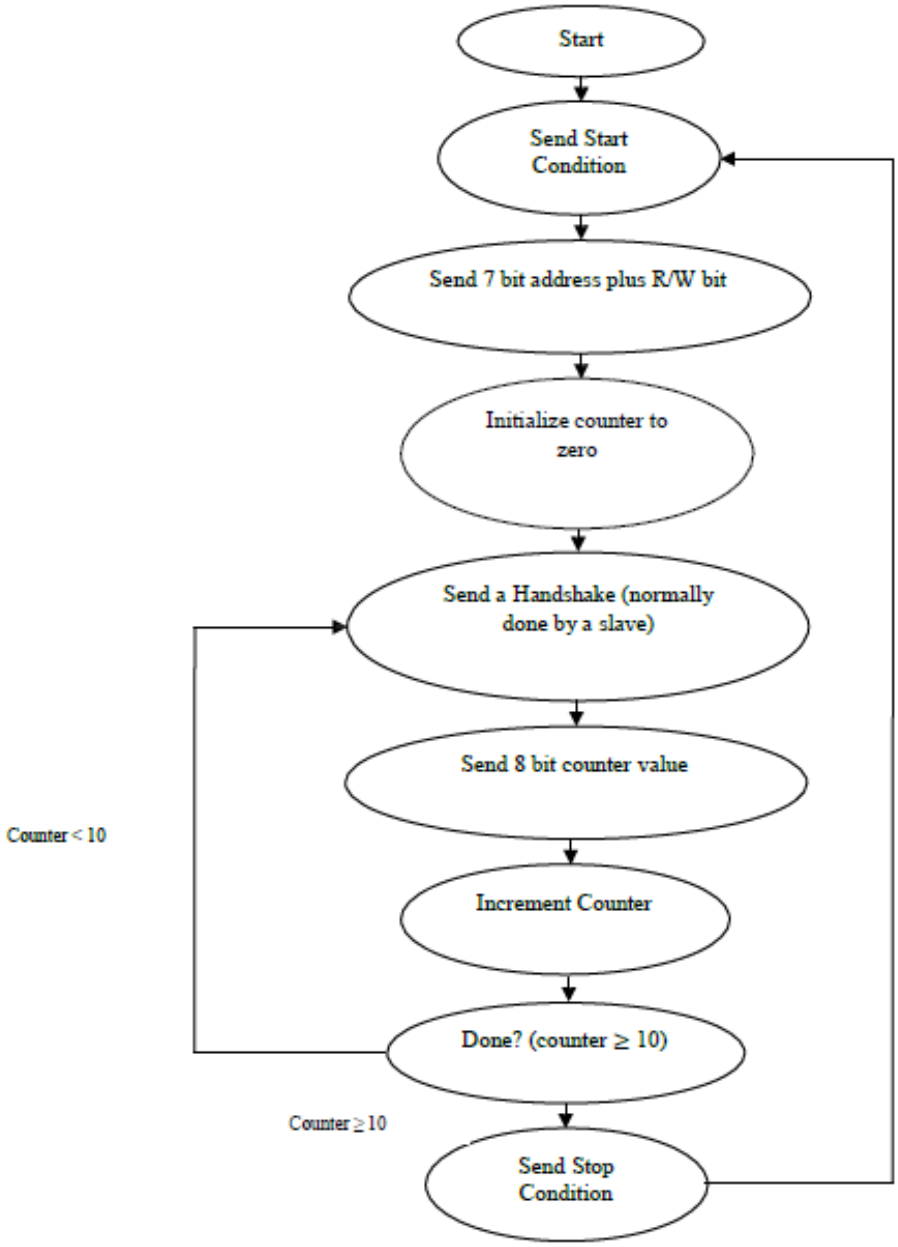
Your goal will be to see this on the oscilloscope.



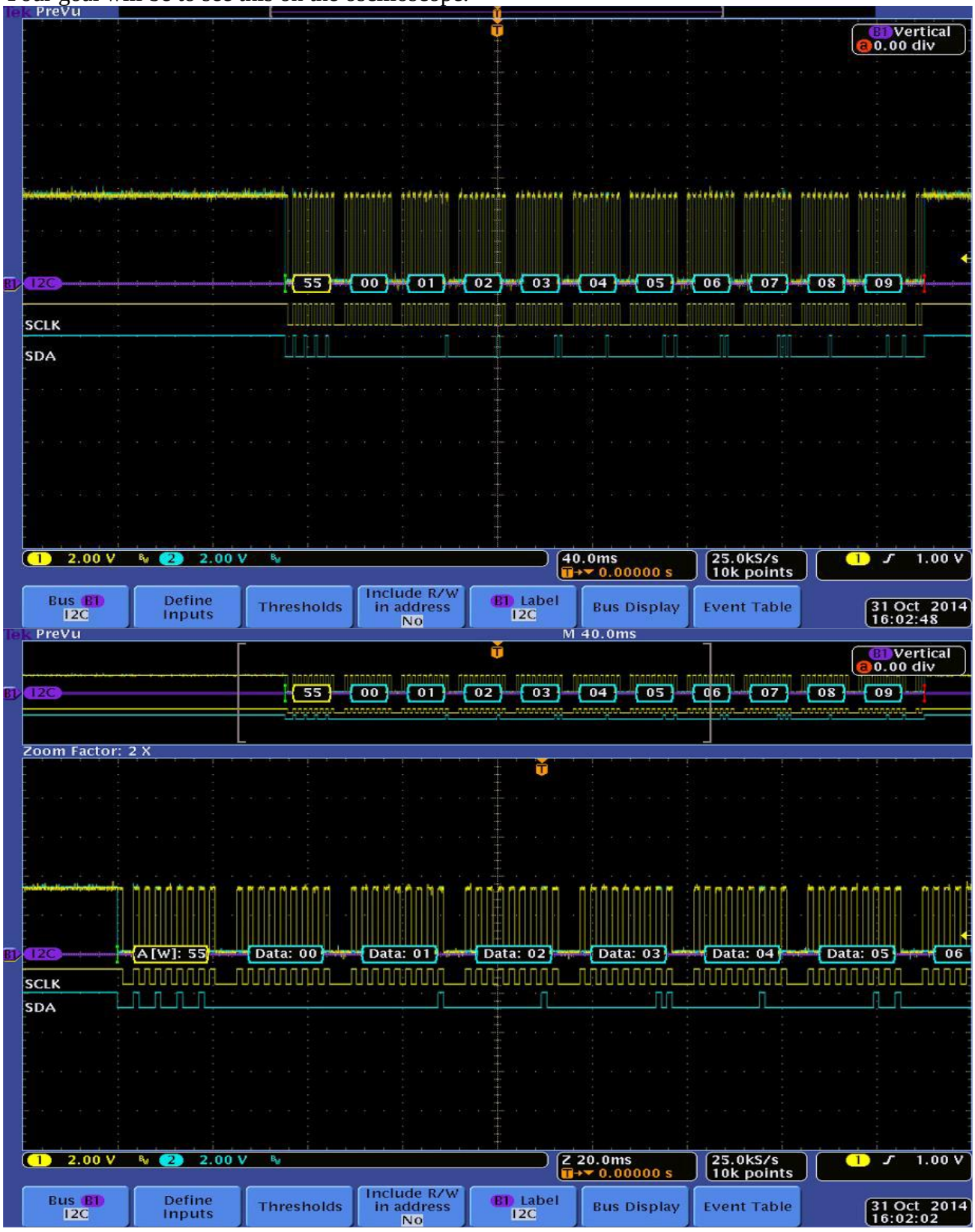
DEMO #1 - Choose an address (other than $55 or $53) to send in I2C to the oscilloscope. Demo your waveform to your lab instructor. Use a flash-drive to save your screenshot so you can upload it to Dropbox.

As you did in demo 1, you will send a made up address. The difference now will be you must follow the address with an acknowledge or "handshake". Since there is no slave device in this lab, the master will fake the "handshake" for the sake of the MSO. You will have to generate a "handshake" subroutine in your code. Use a jump to subroutine to call this routine after you have sent the address. Now that you have sent an address along with a write bit, followed that with a "handshake", you must now write your outgoing data. For the outgoing data we will be writing a counter that starts at zero and counts up to nine. The implementation of this code will take some thinking but the flowchart below will outline the general process involved.

Here is a flowchart to get you started.



Your goal will be to see this on the oscilloscope.



DEMO #2 - Demonstrate your I2C protocol to your lab instructor showing both the address (address must not be $55 or 53), and an 8-bit counter from zero to nine. Save your screenshots onto a flash-drive and upload a screenshot to D2L. Note: Make sure your LSB of your address byte is a '0' to indicate a write.

This part of the lab will allow you to use the skills that you have developed in Demo 1 and Demo 2 to interface to an actual I2C device. Connect the DS3231 to the MSP430 Launchpad board. You will want to use the DS3231 datasheet posted on D2L for information needed for this lab. Be sure you are sending the correct address of the DS3231 in your I2C waveforms. The difference now will be that the DS3231 is capable of generating the appropriate acknowledge responses for your I2C string of data. It is also expecting the master to send the appropriate acknowledge signals at the correct location in the string of data. Remember that the data direction register must be set to be an input on the master side when the slave is sending data or an acknowledge. The data direction register must be set to be an output on the master side when the master is sending data or an acknowledge.
You might generate a flowchart to get you started.

Let's start by reading the seconds register from the DS3231 and showing this on the scope. You should see that the seconds are counting up. Don't worry about initializing the registers yet.
DEMO #3 - Demonstrate your I2C protocol on the scope to your lab instructor showing the seconds value counting up as you read from the device.

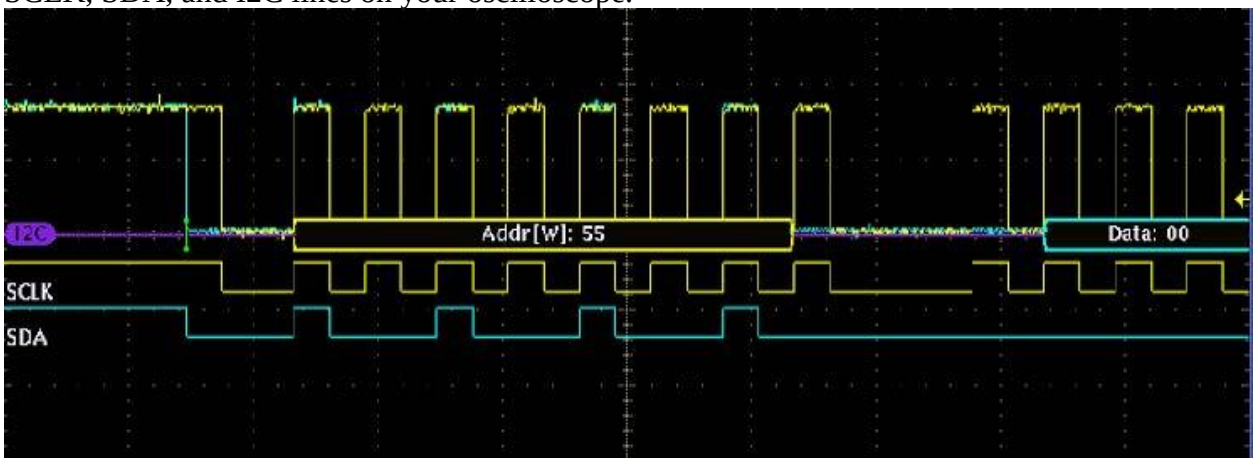
Now let's read the internal temperature register of the DS3231 device.
DEMO #4 – Demonstrate your I2C protocol on the scope to your lab instructor showing the temperature value (in hex) as you read from the device. Save a screenshot onto a flash-drive and upload a screenshot to D2L. For fun you might place your finger on the DS3231 IC and see if you can get the temperature to increase.

Now let's initialize the first three registers of the DS3231 to a time that you select and then read back the first three registers (hours, minutes and seconds) using your I2C protocol.
DEMO #5 – Demonstrate your I2C protocol on the scope to your lab instructor showing at least two sets of read time data (in hex) as you read from the device. The data should show the seconds counting up. Save a screenshot onto a flash-drive and upload a screenshot to D2L.

TA Sign Off_____

Appendix Using the Oscilloscope –

The use of the bus reader function of the oscilloscope will be covered this lab by connecting channel 1 probe of the oscilloscope to your clock pin, Port PTB7-SCL, and by connecting channel 2 probe to the data line pin, Port PTB6-SDA. Channel 1, Channel 2, and B1 should be all turned on in lab. If B1 is off, have your lab instructor show you how to set up this function to take in I2C. If properly set up, you should be able to see the SCLK, SDA, and I2C lines on your oscilloscope.



Your project grade will be based on the memo report that you hand in during this or subsequent lab sessions and your demonstration of your code written for this lab.

Your **Memo Report** must include:

a. A memo report summarizing the methods you used to solve the problem for each section above.. Your memo report should include a flow chart for your program. See the "Example Lab Report" folder for an example. Please upload a single pdf to the Lab1 Dropbox on D2L that includes your memo report and flow charts.

b. Each student should upload their commented code to the Lab1 Dropbox on D2L. Upload the appropriate screen shots. For this lab only upload the code for Demo 5.

Memo Report Date: Tuesday, February 4, 2020 (by 6 PM)

Code Demonstration:

c. A sign-off from the instructor or a TA indicating that your program performed as required and the required circuit modifications were completed. **Each lab team member must build and demo a hardware circuit to receive a sign off for their own circuit**. A sign-off sheet will be kept by the instructor and TA indicating completion of the lab.

1. Demo 1

2. Demo 2

3. Demo 3

4. Demo 4

5. Demo 5

Demo Due Date: Thursdday, Jan 30, 2020 (by end of your lab time)