EE333 Introduction to Microcontrollers

Lab#2 Serial Communication

Allan A. Douglas, Associate Professor

**Lab Overview**

In this lab, the student will learn how to connect a microcontroller to a standard serial port using the built-in UART and communication to other external serial devices using SPI and I2C protocols.

**Required Materials**

|  |  |
| --- | --- |
| Breadboard | Various resistors |
| Arduino UNO (or Spark Fun RedBoard equivalent) | Jumper wires |
| ATmega328P Microcontroller Datasheet | Oscilloscope |
| IC I/O EXPANDER I2C 8B 18DIP | Arduino ISE Software |
| IC I/O EXPANDER SPI 8B 18DIP |  |
| LEDs |  |

**Required Work**

*\*\* All lab report requirements are in bold and italicized below.*

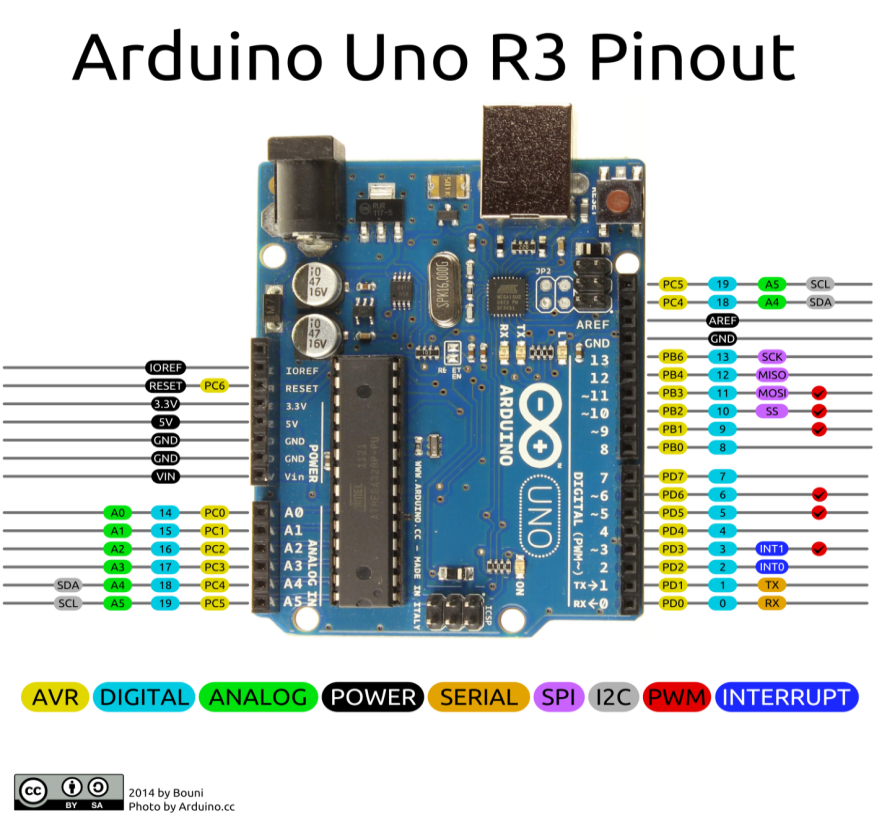
Part 1a: RS232 Serial Port (Transmit Only)

Connect your microcontroller to the PC USB port. The UNO uses a special chip to create a bridge between USB and the RS232 serial port on the microcontroller. In Lab#1, this chip was on the FTDI board. We used the chip to download our compiled code over USB into the microcontroller memory for execution. In this lab, we are going to use the same connection to interact with our programs while they are running.

Load example sketch called DigitalReadSerial. This sketch reads a digital input on pin 2, prints the result to the serial monitor. Change the delay command to delay (100). This will cause a 100ms delay between each println command. Download the sketch. Open the Serial Monitor under the Tools menu in the Arduino IDE. Connect pin 2 to ground. You should see a “0” printed to the serial monitor. Then, connect pin 2 to 5V power. You should see a “1” printed to the serial monitor.

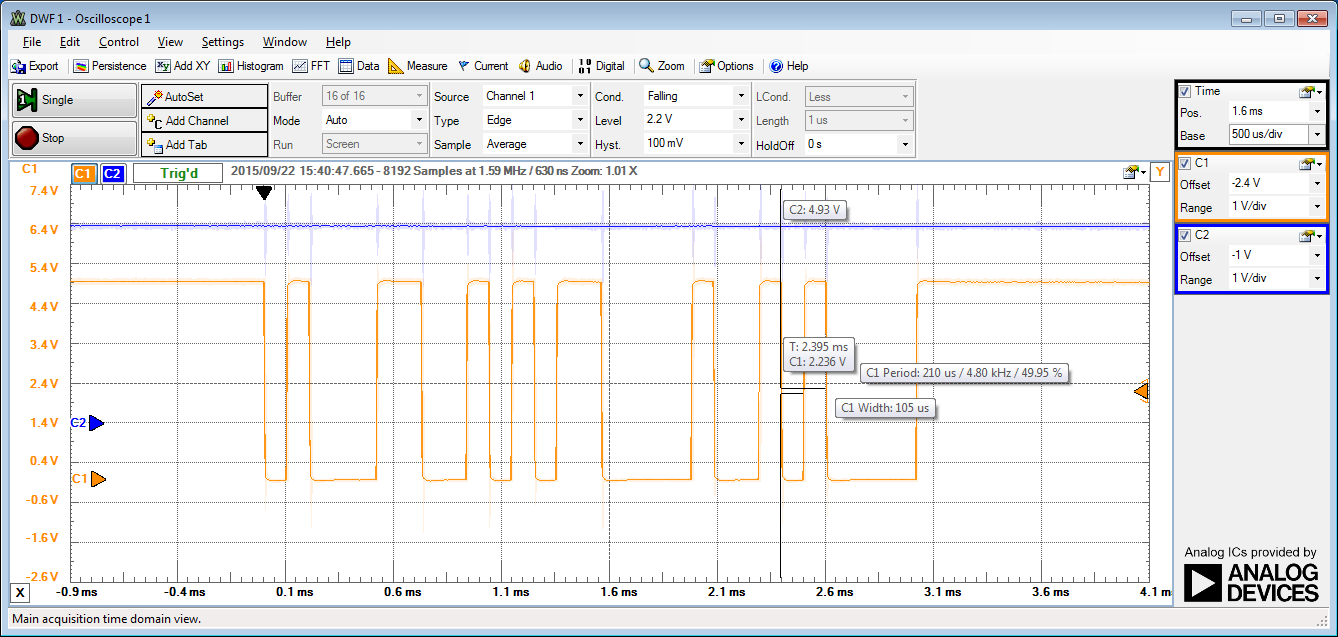
***Explain the setup() and loop() functions in the DigitalReadSerial sketch.***

Next, connect an oscilloscope to the to the Arduino TX and RX pins.



You should see that the TX signal is changing and the RX signal is a constant at about 5V. The changes in the TX signal correspond to the “0” or “1” ASCII character being sent by the Arduino to your PC. Since you are not sending any data from the PC to the Arduino, we expect the RX signal to be a constant.

***Measure the bit width in ms and confirm that this matches the serial port baud rate of 9600. Include a screenshot from the oscilloscope in your report and the baud rate measurement.***

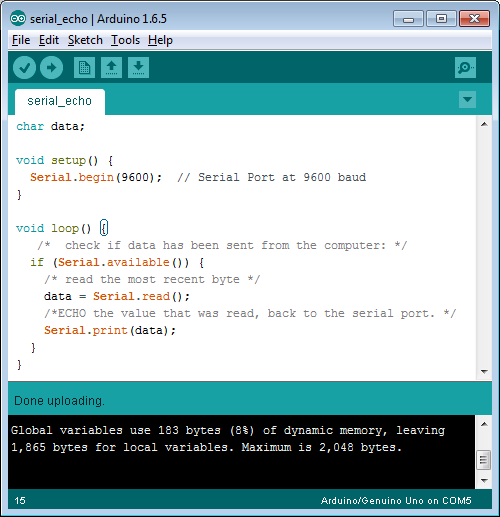


***Research the RS232 protocol on the internet and determine the bit pattern that represents a “0” or a “1”. Confirm the pattern on the oscilloscope matches the expected bit pattern in your report.***

***There appears to be extra characters transmitted. What are these extra ASCII characters?***

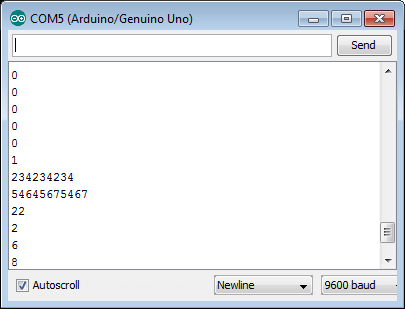
Part 1b: RS232 Serial Port (Transmit and Receive)

Create a new sketch. Enter the following code.

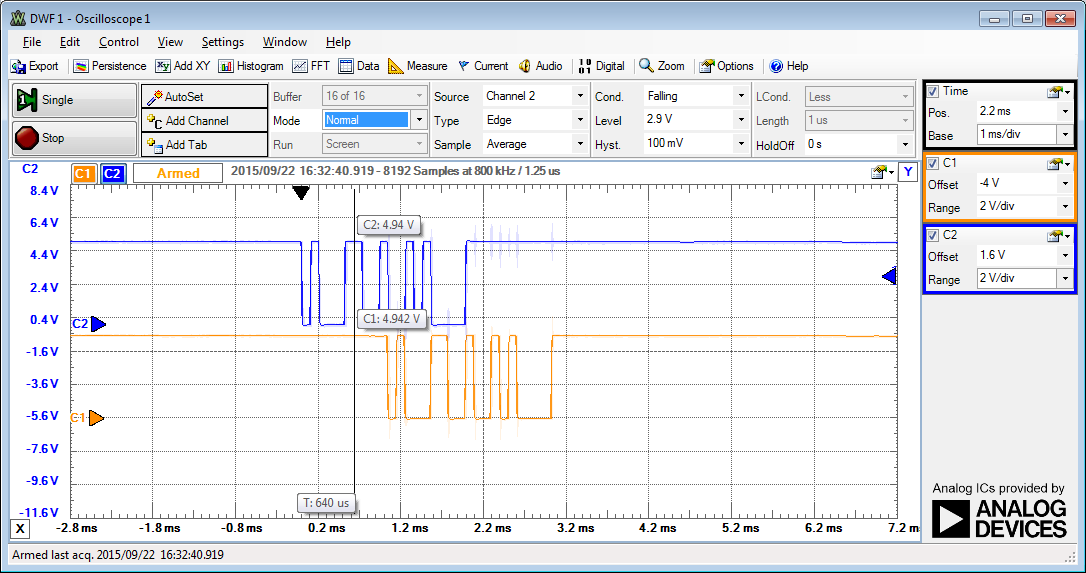


Download the code and open the serial monitor. Type a character into the upper entry box. When you hit the Enter key, this character is transmitted to the Arduino. The Arduino receives the character on the RX connection and then transmits the same character on the TX connection back to the PC.

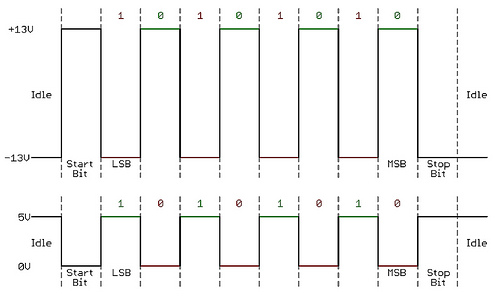
***Research the commands in the Serial Echo sketch. Explain how the sketch works in your own words.***



***Record waveforms with an oscilloscope of both the Rx and Tx pins of the microcontroller.***



***Decode the received bit pattern keeping in mind that the serial pattern is framed with a Start bit and a Stop bit.***



***Does the bit pattern match the character you transmitted? Is there more than one character? If so, what are the extra characters?***

***Measure the amount of delay between the received character and the re-transmitted character? What causes this delay?***

Part 2: I2C Port Expander

For this part of the lab, you will need the Arduino Wire library: <https://www.arduino.cc/en/Reference/Wire>

Follow the guide found on this link to learn how to add an Arduino library https://www.arduino.cc/en/Guide/Libraries.

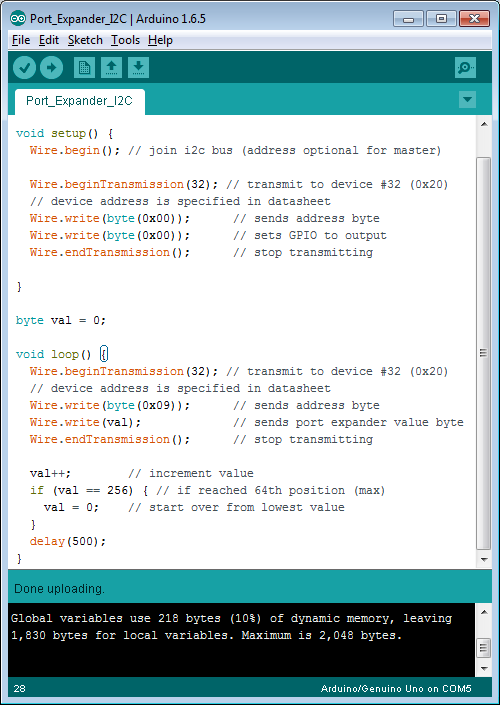
Add the MCP23008 I2C Port Expander chip to your breadboard. Connect 5V and ground from the Arduino UNO to the breadboard power rails. Connect the Port Expander to 5V power and ground.

The datasheet for the Port Expander can be found at: <http://ww1.microchip.com/downloads/en/DeviceDoc/21919e.pdf>

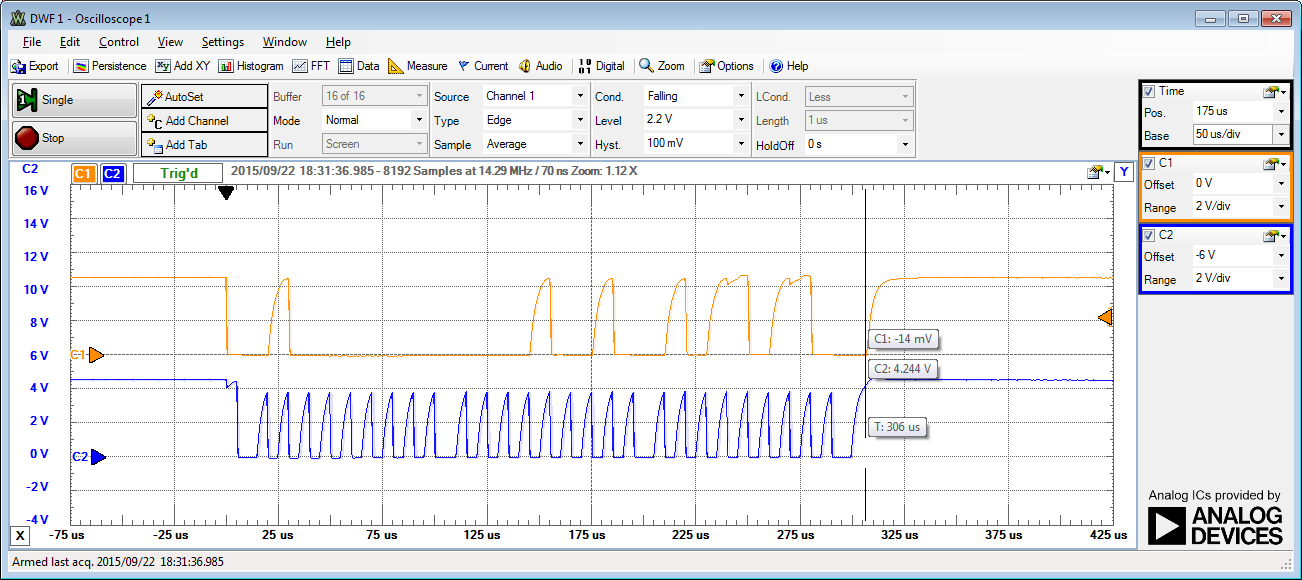
Connect the I2C SCL and SDA pins on the Port Expander to the Arduino. Connect each of the 8 GPIO pins on the Port Expander to a 330 Ohm resistor in series with an LED. All LED cathodes should be connected to ground.

Make sure to connect the Port Expander I2C address pins A2, A1, and A0. For the example below, these pins are each connected to ground. Also, be sure to connect the Port Expander /RESET pin to 5V power.

Power up the microcontroller and load the following sketch.



***Connect an oscilloscope to the SCL and SDA pins on the microcontroller and observe the waveforms. Include a snapshot of the waveform in your report. Your oscilloscope output should be similar to the following figure***.



***Verify that your LEDs are lighting up in like and follow the pattern of a binary counter. The output value should correspond to the value of the variable called val in the sketch.***

***Write a description of the code in the Arduino sketch. Make sure to describe the bits being sent over I2C and relate them to the code and to the internal registers in the I2C Port Expander. Make sure to describe what is happening on the I2C bus in the setup() function and what is happening during the loop() function.***

Part 3: SPI Port Expander

Replace your I2C Port Expander with the SPI Port Expander.

The datasheet for the Port Expander can be found at: <http://ww1.microchip.com/downloads/en/DeviceDoc/21919e.pdf>

Note, that the pinout of the MCP23S08 is different from MCP23008. You will need to connect SCK, SI, SO, and /CS pins to the microcontroller (Arduino UNO). There are only 2 address pins. You can connect these to ground.

For the SPI implementation, you will need to include the SPI.h library at the top of your code. (#include SPI.h).

Modify the I2C sketch to create the same behavior with the SPI device.

***Connect an oscilloscope to the SCK, MOSI, and MISO pins on the microcontroller and observe the waveforms. Include a snapshot of the waveform in your report.***

***Write a description of your SPI sketch. Make sure to describe the bits being sent over SPI and relate them to the code and to the internal registers in the SPI Port Expander. Make sure to describe what is happening on the SPI bus in the setup() function and what is happening during the loop() function.***

**Required Work Checklist**

□ Part 1a.

Include all measured waveforms

Answer all questions stated in the section

Include copies of you working sketches.

Oscilloscope waveforms showing pin 9 toggling on and off at both 0.5Hz, 10Hz and 100Hz

□ Part 1b.

Include all measured waveforms

Answer all questions stated in the section

Include copies of you working sketches.

□ Part 2.

Include all measured waveforms

Answer all questions stated in the section

Include copies of you working sketches.

□ Part 3.

Include all measured waveforms

Answer all questions stated in the section

Include copies of you working sketches.

**Extra Credit Work**

1. Using your breadboard version of the ATmega328P and the Arduino UNO implement the Master Writer/Slave Receiver example on the following webpage <https://www.arduino.cc/en/Tutorial/MasterWriter>
2. Include a complete description of both sketches and waveforms from the oscilloscope that demonstrate the working design.

**Extra Credit Checklist**

□ Copy of both sketches.

□ Oscilloscope waveforms.

□ Explanation of the sketches explaining all the functions used for the example.