SY202 – Cyber Systems Engineering

Due Date: **7 March 2019**

LABORATORY INVESTIGATION #06: Serial Communications

Objectives:

i) To establish asynchronous serial communication links between multiple devices.
ii) To establish a MATLAB's serial communication interface with a microcontroller, collect data, and conduct rudimentary analysis.
iii) To understand the effect of mismatched serial settings between the receiver and transmitter.

Discussion: Despite the increasing prevalence of IP connected devices, asynchronous serial communication remains a commonly used technique to connect sensors, actuators, and microcontrollers. This lab seeks to improve the students understanding of the settings associated with asynchronous serial communications through practical experience. Two serial ports on a single mbed NXP LPC1768 microcontroller will be used in Part A to demonstrate the effect of mismatched serial settings between receiving (Rx) and transmitting (Tx) devices. The student should understand that this is not a practical use of the microcontroller's multiple serial ports, but is being studied to gain an understanding of the importance of correct serial port set-up. Similarly, in Part B and C two mbed NXP LPC1768 microcontrollers are arranged to communicate using an asynchronous serial link. The communications are simultaneously being monitored on a PC terminal connected to each microcontroller.

Procedure:

Serial communications are often specified using the following notation:

9600N81 (*xxxxxx* Baud/*x* Parity/*x* Data Bits/*x* Stop Bits)

Part A: Using the instructor provided mbed code on a single mbed connected as shown in Figure 1, investigate the effect from mismatched asynchronous serial communication parameters.

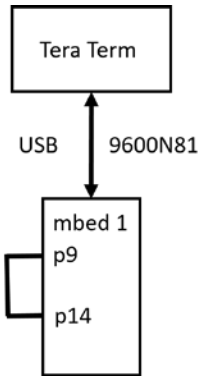

Figure 1 Part A wiring diagram.

1) Review the provided SY202_Lab6_PartA_SerialCommsSingle.txt source code file to gain an understanding of how to establish asynchronous serial communications.
2) After reviewing the code, take the opportunity to review the mbed OS documentation for the Serial Class Reference that can be found at https://os.mbed.com/docs/mbed-os/v5.11/apis/serial.html. Note that library functions exist for the user to alter the serial communications settings for baud rate, data bits, parity, and stop bit length.
3) Connect the mbed p9 (Serial #1 Tx) and p14 (Serial #2 Rx) on a single mbed using the provided wire as shown in Figure 1. Ensure that you have the correct pins connected or you will not successfully complete this section. It is recommended that the connection be made with power secured to the mbed (center blue LED off) to avoid damaging the circuit if an incorrect pin is connected. See Figure 2 for the pin locations.

4) Connect the mbed to a PC and load the compiled SY202_Lab6_PartA_SerialCommsSingle.bin file provided by your instructor onto the mbed flash drive.
5) Establish a terminal window using Tera Term. Note that the default serial communication settings (9600N81, 9600 Baud/None Parity/8 Data Bits/1 Stop Bit) are used for this example. Be sure that the terminal uses an LF or AUTO for New-line (TeraTerm->Setup->Terminal->Receive) so that the terminal output is easier to read.

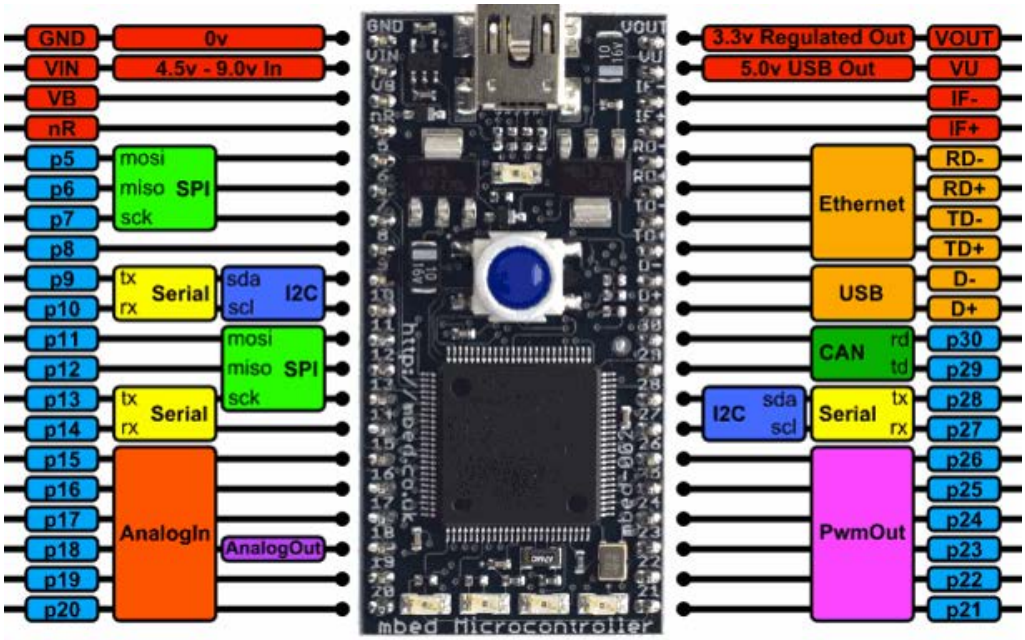

Figure 2 mbed NXP LPC1768 pinout.

6) Conduct the four demonstrations in SY202_Lab6_PartA_SerialCommsSingle by following the on-terminal instructions. Be sure to copy down the transmitted and received characters for each case and convert the decimal equivalent to binary (or use the ASCII table if the character is defined). If nothing is received, record "Buffer # Empty" in the Rx row:

NOTE: You may send a break command (press 'alt-b' together) in Tera Term to reset the mbed. This has the same effect as pressing the reset button.

a. Identical settings: Tx/Rx 600 Baud, Tx/Rx 8 data bits, Tx/Rx 1 stop bit

Tx 600N81/Rx 600N81 (1st Transmission)					
Serial Device	#	Character	Hexadecimal	Decimal	Binary
Tx	1				
	2				
Rx	1				
	2				

b. Mismatched baud rates: Tx **300 Baud**/Rx 600 Baud, Tx/Rx 8 data bits, Tx/Rx 1 stop bit

Tx 300N81/Rx 600N81 (1st Transmission)					
Serial Device	#	Character	Hexadecimal	Decimal	Binary
Tx	1				
	2				
Rx	1				
	2				

Tx 300N81/Rx 600N81 (2nd Transmission)					
Serial Device	#	Character	Hexadecimal	Decimal	Binary
Tx	1				
	2				
Rx	1				
	2				

c. Mismatched data bit length: Tx/Rx 600 Baud, Tx 8 data bits /Rx **5 data bits**, Tx/Rx 1 stop bit

Tx 600N81/Rx 600N51 (1st Transmission)					
Serial Device	#	Character	Hexadecimal	Decimal	Binary
Tx	1				
	2				
Rx	1				
	2				

Tx 600N81/Rx 600N51 (2nd Transmission)					
Serial Device	#	Character	Hexadecimal	Decimal	Binary
Tx	1				
	2				
Rx	1				
	2				

d. Mismatched stop bit length: Tx/Rx 600 Baud, Tx/Rx 8 data bits, Tx 1 stop bit/Rx **2 stop bit**

Tx 600N81/Rx 600N82					
Serial Device	#	Character	Hexadecimal	Decimal	Binary
Tx	1				
	2				
Rx	1				
	2				

7) Discuss your results for each case (6.a – 6.d) and explain why the transmit and receive characters are the same or different in each case. The binary values may be useful to develop your explanation. Consider that serial communications of each character entails sending a binary string comprised of: **start bit - data bits - parity bit (if used) - stop bit(s)**. Also, note that the "Buffer # empty" line(s) between Transmitting and Receiving provides valuable information.

Part B: Develop and test mbed code to pass serial data to a PC. Note that two mbed microcontrollers connected as shown in Figure 3 are required to complete this section. If desired, the two mbed microcontrollers could be connected to different PCs. If this option is selected, be sure to connect the mbed ground pins together to avoid spurious signals. See your instructor for assistance. Otherwise, it is recommended to stick to a single computer.

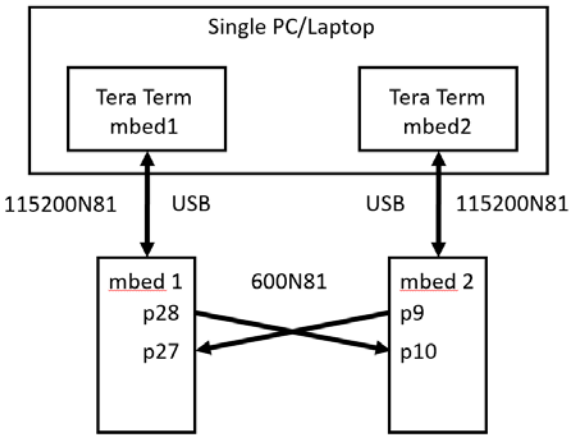

Figure 3 Schematic layout of Part B.

1) Disconnect the mbed from the USB port.
2) Connect p28 (mbed1, Serial Tx) on one mbed to p10 (mbed2, Serial Rx) on the second mbed. Connect p27 (mbed1, Serial Rx) on the first mbed to p9 (mbed2, Serial Tx) on the second mbed.
3) Connect the first mbed (mbed1) to the PC using a USB cable. Note that this mbed should have wires connected to p27 and p28.
4) Upload the compiled SY202_Lab6_PartB_SerialCommsRandGen.bin file provided by your instructor onto the first mbed (mbed1) flash drive. This program transmits two floats ("%f,%f\n") that consist of a time stamp (first float) and a random number (second float) separated by a comma. This format (Comma Separated Values or csv) is commonly used to store data for input into other software such as MS Excel or MATLAB. The separating comma is called the "delimiter" and this is one form of "delimited" data.
5) Connect the first mbed (mbed1) to Tera Term (TeraTerm->Setup->Serial Port…) using the following serial settings: **115200N81**. Be sure that the terminal uses an LF or AUTO for New-line (TeraTerm->Setup->Terminal->Receive). mbed1 will flash LED1 to indicate that it is transmitting data.

NOTE: The required Tera Term settings are necessary for the mbed to connect because of options set in the compiled file. The default Tera Term terminal setting of 9600 Baud is incompatible with a serial device using 115200 Baud. Before proceeding to develop the code for mbed2 ensure that you are able to see data from the first mbed in the terminal window.

6) Develop mbed code to:
   a. Establish a serial connection with the first mbed (mbed1) using the following serial settings: **600N81**.
   b. Establish a serial connection with the PC using the following serial settings: **115200N81**.
   c. Light LED2 to indicate that it is the second mbed (mbed2).
   d. Pass any data received from the first mbed (mbed1) to the PC. Consider using the getc/putc commands.
7) Connect the second mbed (mbed2) to the PC using a USB cable. Note that this mbed should have wires connected to p9 and p10.
8) The mbeds should now be connected to the same PC using two available USB ports.
9) Connect the second mbed (mbed2) to Tera Term using the following serial settings: **115200N81**. Be sure that the terminal uses an LF or AUTO for New-line (TeraTerm->Setup->Terminal->Receive). Your setup should appear similar to Figure 3.
10) Verify that the data on both Tera Term windows are identical.
11) Capture a simultaneous screen shot of both Tera Term windows (mbed1/mbed2) as shown in Figure 4 for the lab report.

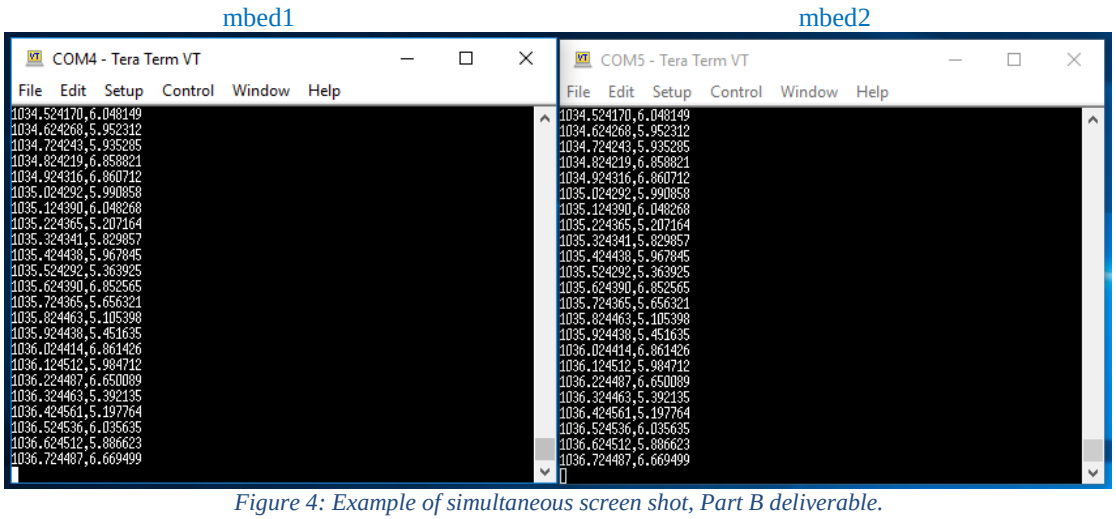

Figure 4: Example of simultaneous screen shot, Part B deliverable.

Part C: Analyzing data collected using serial communications. The following instructions detail how to transfer data from the mbed (using the terminal program Tera Term) into MATLAB for analysis. Plot the received data and perform basic mathematical analysis in MATLAB. Note the physical set-up should remain the same as that used in Part B and shown in Figure 3.

1) Using the set-up and your code from Part B, data is transmitted from mbed1 to mbed2 over a 600 baud serial connection and from mbed2 to the PC over the USB serial communication link at 115200 baud. On the PC, a terminal program (Tera Term) is used to receive the information.
2) Data from the mbed2 can be stored into a file using the Tera Term logging function (File -> Log…). Be sure that **only** the "Plain Text" block is checked when starting the log. This will ensure that the data is in the proper format to import into MATLAB and will avoid appending data onto the file (if it already exists). Collect at least 1000 data points (although you are free to collect more).

NOTE: Consider how long do you expect it will take to collect 1000 data points? Include your answer in your lab write-up.

3) Stop the Tera Term data log by closing the data log (File->Show log dialog…) and selecting "Pause" then "Close".
4) Delimited data can be imported into MATLAB and the information stored into variables. There are multiple ways to accomplish this task. One approach is to use the "Import Tool" in MATLAB. Review the

documentation detailing its use on the Mathworks site, Matlab Data Import and Export, or by watching the Import Data Tool tutorial, https://www.mathworks.com/videos/importing-data-from-text-files-interactively-101489.html. For this lab, two variables are required: one variable for "time", another for "output". Be sure to select "Output Type" of "Column Vectors" and label the first column variable as "time" and the second column variable as "output" then import.

NOTE: You may need to perform error checking to ensure that the data is not corrupted prior to using the received data for plotting or calculations. When logging data, the first and/or last entries commonly only partially record the data and must be removed from the data set. A simple error checking routine for this data would verify that time is increasing with each successive data point. Consider other possibilities by reviewing the mbed1 Tera Term window as shown in Figure 4.

5) Plot Time (s) vs Output (V). Be sure to label the axes and use the 'x' marker to show the value at each time. Do **not** connect each data point with a line since each measurement is independent of its neighbors.
6) Calculate the mean, maximum, and minimum values of the Output and include these values in your report. Use MATLAB help files to find the proper syntax for each command.

Deliverables:

Follow the lab report template and the general lab guidelines for SY202 lab reports. Refer to the lab rubric for the grading of the lab. Be sure to include the required analysis of results in your lab report.