LAB #1 SERIAL COMMUNICATION (2 SESSIONS)

OBJECTIVE

- Learn about how to solder a null modem cable;
- Learn about the basic serial communication protocol;
- Learn how to send and receive serial communication packets from MATLAB;
- Practice MATLAB programing.

INTRODUCTION

I. Serial Communication

Please reference the class handout on serial communication. For a DB9 connector commonly used for RS-232 standards, the definition of each pin is explained in Figure 1.

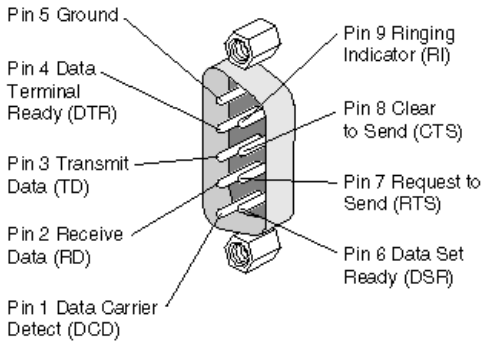

Fig.1. DB9 Connector for RS232 Serial Communication

II. Null Modem Cable

To connect two computers using a RS232 serial connection, a null modem cable will be needed. There are several ways to make a null modem cable. If hand shaking is needed, follow Figure 2; and if hand shaking is not needed, follow Figure 3.

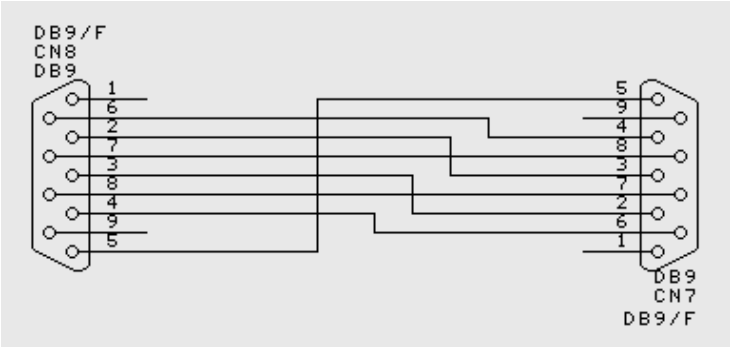

Fig.2. Null Modem Cable with Hand Shaking

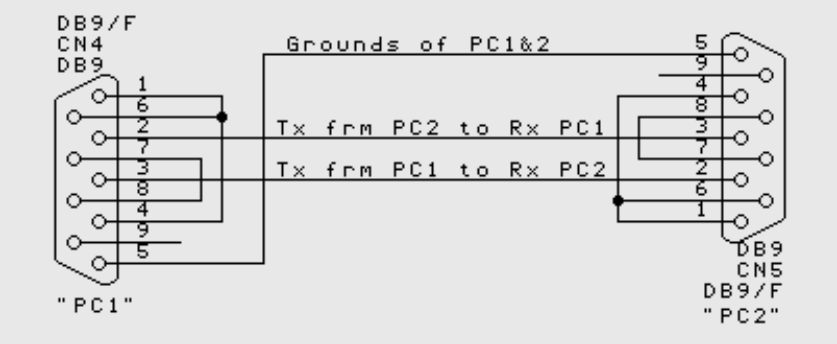

Fig.3. Null Modem Cable without Hand Shaking

III. Soldering

If you don't already have experience with soldering, watch this video first:

https://www.youtube.com/watch?v=I_NU2ruzyc4

IV. USB to Serial Adapter

Since most of the modern computers do not have RS232 ports anymore, you will sometime need to use a USB to serial adapter to create new serial ports. The one we are using for this class is made by Micro Connectors Inc. It actually converts one USB port to two RS232 serial ports. Once properly installed, the serial port will show up as 'COM 1' and 'COM 2', or other numbers: 'COM N' and 'COM N+1'.

Fig.4. Micro Connectors E07-162 USB Dual Serial Adapter

V. MATLAB Serial Function

Please reference the document linked below:

http://www.usna.edu/Users/weapsys/esposito/roomba.matlab/Serial Communication in Matlab V2.doc

VI. MATLAB Data Types and Data Type Conversion

MATLAB supports many different data types. When you define a variable, the default type is 'double'. 'Double' refers to the double-precision floating-point format, which ocupies 8 bytes and can repsent a real number with a large number of significant digits. If you want to represent a single byte (8-bit), the common types for that are uint8 (unsigned 8-bit integer) or int8 (signed 8-bit integer). The type 'char' is often used to represent ASCII characters. In MATLAB programing, you will often need to

perform data type conversions to handle varibles of different tyes. A list of MATLAB functions for the conversion can be found here: http://www.mathworks.com/help/matlab/numeric-types.html . For example, if you want to find the ASCII numbers associated with the string 'Hello world' you can do the following:

```
>> a='hello world';          % Creat a string
>> b=uint8(a)                % Convert to 8-bit numbers
b =  104  101  108  108  111   32  119  111  114  108  100
```

Another example, you can convert a 16-bit signed integer, -2,589, to two bytes in a communication packet. These two bytes can then be recovered as the signed integer after received by another communication device:

```
>> a=-2589;                                % the initial value
>> b=typecast(int16(-2589),'uint8')        % converts to two unsigned 8-bit integers
b =  227  245
>> c=typecast(uint8(b),'int16')            % recover the originial value
c =  -2589
```

PROCEDURE

I. Solder a Null Modem Cable and Make Connections

1. The first step is to make a null modem cable. Two female DB9 connectors and wires will be supplied to each team. Solder a null modem cable using the configuration shown in Figure 2 that supports full hand shaking;
2. Ask the instructor to inspect the cable you soldered;
3. Connect the two serial ports on the USB to serial adapter using the null modem cable;
4. Open two copies of the serial communication program MTTTY on your computer;
5. Configure the MTTTYs for two different serial ports with the same parameters (e.g. baud rate, data bit, parity, stop bit, etc.);
6. Type in one of the MTTTY window and see if the characters show up in another window. If it works in both ways, your cable is soldered correctly for the transmitting and receiving lines (but not necessarily for other lines).

II. Test MATLAB Serial Communication Functions

1. Turn off one of the MTTTY window;
2. Initialize the unused serial port in MATLAB using the 'serial' function;
3. Use the 'set' function to set the serial port parameters to match the one in MTTTY;
4. Use the 'fopen' function to open the serial port;
5. Send a character (e.g. 'A') from MTTTY and receive it from the MATLAB using the 'fread' function. Compare the output value with the ASCII table;
6. Send a series of values from MATLAB using the fwrite function so that it will be displayed as 'Mobile Robotics' on MTTTY;
7. Send the same string using the MATLAB 'fprintf' function;
8. Use the MATLAB function 'fscanf' so that when you type 'Mobile Robotics' in MTTTY, it will be displayed in MATLAB;
9. Ask the instructor to check what you did so far;
10. Close the serial port properly with the 'delete' and 'clear' functions.

III. Assemble and Transmit Communication Packets

1. Properly initialize a serial port and configure the baud rate to 115200;
2. Write a MATLAB code that can continuously send out 1,000 communication packets in the following format:

Byte	Content	Comment
0	Header 1 (65)	ASCII 'A'
1	Header 2 (90)	ASCII 'Z'
2	Counter	An 8-bit counter that ranges from 0 to 255
3,4	Time t	Use the 'tic' and 'toc' functions to determine the elapse time from the beginning of the program and report it using two bytes in each data packet. **Hint**: two bytes can represent a number between 0-65535. Think about a way that you can send a high-resolution time measurement through the communication link using only two bytes.
5,6	Data Ch. #1: sin(t)	Again, think about how to preserve the maximum amount of information.
7,8	Data Ch. #2: sin(20t)	
9	Check Sum	Add up bytes 0-8 and keep the last 8-bit

Table 1. Serial Communication Packet

3. Put 'pause(0.02)' between every two consecutive transmissions;
4. Receive the data from MTTTY and explain what you see in the window;
5. Turn the MTTTY off and initialize another serial port in MATLAB;
6. Write the second part of the MATLAB code that can receive the packets. Save (use the 'save' function) and plot the before and after (communication) values of the counter, elapse time, data channels, and check sum;
7. Verify if the packets are transmitted correctly using the counter and check sum;
8. Change the parameter of the 'pause' function from 0.02 to 0.01. Save and plot the data again;
9. Change the size of the input buffer for the receiving side serial port to 10 and run the program again (set(SerialPort, 'InputBufferSize',10);).

DELIVERABLE

An abbreviated lab report of your experiments and answer the following questions:

1. Why did we need a null modem cable to connect between two computers (or two ports on the same computer)?
2. If you want to display '25.32' on MTTTY, what binary values do you need to send from the serial port?
3. What is the function of the serial buffer?
4. What are the percentages of data loss under different conditions (different pause values and buffer size)? How many of them are missing packets and how many are received but corrupted packets? How can you tell? How would you minimize the packet loss?
5. What are the communication delays under different communication parameter? How would you minimize the delay?