Experiment 8

Communicating with External Devices

Introduction

In this lab, we will be extending the Simon board's functionality. In the first part of the lab, you will be asked to create code for transmitting signals mapped to the Simon Board's pushbuttons through to the RS232 Port. In the second part of the lab, you will verify your design by then reading the signals out using a terminal application called PuTTY. The skills learned will be very valuable in future work.

Objectives

1. Familiarization with RS232 Port
2. Familiarization with communication between two devices
3. Dynamically utilize serial capture with a computer terminal

MATERIALS REQUIRED

- Computer with RS232 Port
- Keil Microvision 3, or later
- Flash Magic Software
- PuTTY Software
- RS232 Female to Female

Background

The Simon board has an 8051 on it. With this 8051 you can output to the RS232 port for communication with outside sources. The RS-232 protocol was the main way to transmit serially for many years, and is still a staple for many embedded applications in industry. It consists of either 9 pins or 25 pins. In this lab we will be using the 9 pin model. The pins are set up like in Figure 1.

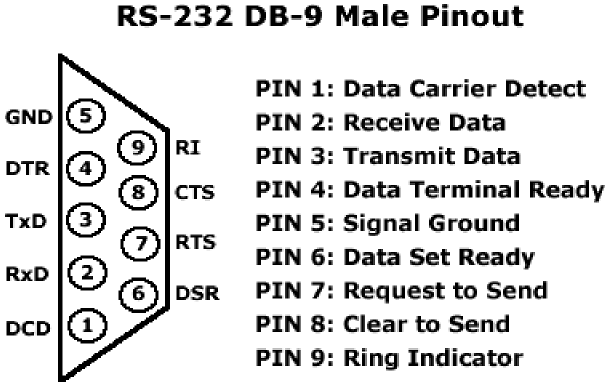


Figure 1: RS-232 Pinout

In this lab we'll be using the transmit Data pin and the Signal Ground to print ASCII characters over the RS232 port to be captured by a PC's serial terminal program (PuTTY). If you want to read more about the RS232 Protocol, do a Google Search on it.

Procedure:

Part 1.

1. Write the C code needed to output any arbitrary letter, i.e., the first four letters of your name, for each of the 4 buttons when pressed to the RS232 port. Start like usual, by creating a new project in Microvision and specifying the Device ID as NxP/P89LPC932, as well as remembering to set the create output Hex file option for the Target 1. Like Lab 7, some useful header information and includes are necessary:

   uart.c – which is available at: http://eceweb.mst.edu/courses/Intro/uart.c
   uart.h – which is available at: http://eceweb.mst.edu/courses/Intro/uart.h
   reg932.h – also available at: http://eceweb.mst.edu/courses/Intro/reg932.h

   Copy these three files to your project's directory, then add them into the project by right-clicking on Source Target 1-> Add Files.

   Then for your main C file, simply include the following header information:

   ```
   #include "uart.c"
   sbit switch1=P2^1;   // green button
   sbit switch2=P2^2;   // amber button
   sbit switch3=P2^0;  // yellow button
   sbit switch4=P2^3;  // red button
   P2M1=0x00;           // port 2 (buttons) set to bidirectional
   void main();         // main prototype
   ```

   Lastly, we need to configure the baud rate of 9600 for our particular Simon board. The internal RC oscillator is running at 7.3728 MHz. The formula for calculating baud rate for a particular oscillation frequency is given by:

   $$BAUD = \frac{F_{oscillator}}{16(BRGR+1)}$$

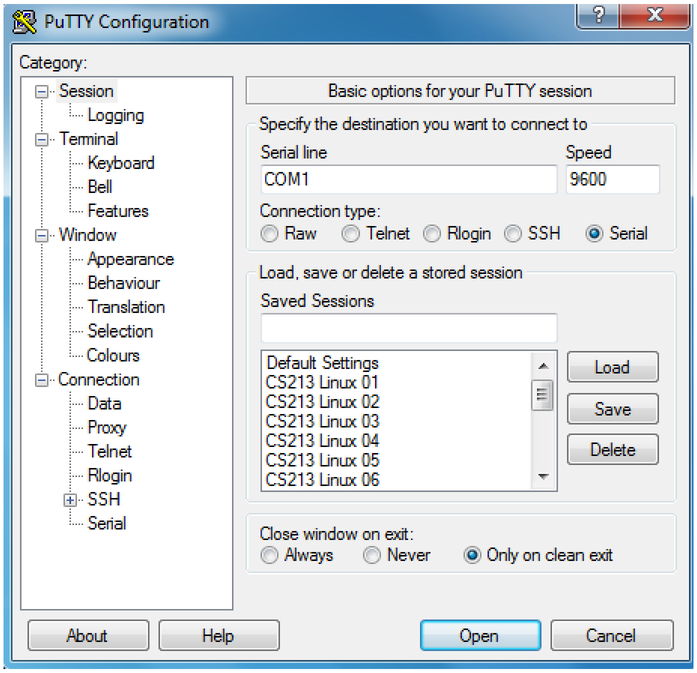
   Note that since we are using an 8-bit architecture, BRGR will have to be split into an upper and lower nibble. You will need to locate the following segment of code in uart.c, within the uart_init() function and then change BRGR0 and BRGR1 to reflect the values listed below:

```
// configure baud rate generator
BRGCON = 0x00;
BRGR0 = 0xF0;
BRGR1 = 0x02;
BRGCON = 0x03;
```

2. Create the HEX file for this code, just like Lab 7.

3. Use the Flash Magic Software to flash the HEX file on the Simon Board, again like Lab 7.

4. Open up PuTTY and choose serial port COM1. The setup should look like Figure 2.



Serial line: COM1

Speed: 9600

Connection type: ○ Raw ○ Telnet ○ Rlogin ○ SSH ● Serial

Saved Sessions:

Default Settings
CS213 Linux 01
CS213 Linux 02
CS213 Linux 03
CS213 Linux 04
CS213 Linux 05
CS213 Linux 06

Close window on exit: ○ Always ○ Never ● Only on clean exit

Figure 2: PuTTY terminal configuration.

Part 2.

1. Observe what happens when a particular button is pressed on the Simon board. Do you see the desired result? If so, show your TA. If not, recheck your work to find the error(s).

Questions

1. Describe in a brief summary how one would go about setting up the FPGA to receive the transmitted characters instead of the PC’s COM1.
2. Assuming the FPGA can receive the data, what sort of data structure would be convenient to help display all of the received characters from buttons 1-4 presses on the LCD at once?