Lab 5: Asynchronous communication — the USART

Goals:

The main goal of this lab is to become familiar with the USART port, and with asynchronous data communication. The C standard I/O functions will also be used.

Asynchronous communication — USART

The USART, particularly in asynchronous mode, is a common interface to peripheral devices. It uses a small number of pins, and can maintain a reasonable data transfer rate. Its most popular communication interface, the RS232 port, is the classical interface to terminal devices, dating back to the teletype terminals.

Communication between an RS232 port and the the SPI port

In this lab, you will input characters through the RS232 port of the STK-500 display the character on the LEDs, and output the character through the SPI port.

The value returned from the SPI input will be echoed back to the terminal through the RS232 port.

The function can be tested by connecting the output (the `MOSI` pin) to the input (the `MISO` pin) and displaying the output on the LEDs.

The STK500 has a RS232 driver built into the board. Recall that the RS232 standard requires positive and negative voltages for 0 and 1 respectively.

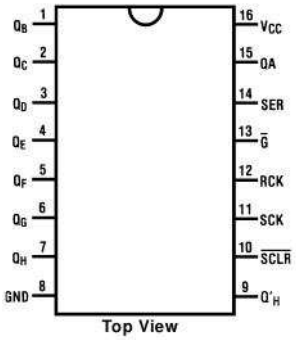
The two pins labeled `RXD` and `TXD` at `RS232 SPARE` (between LEDs 4 and 5) can be used to connect to the RS232 driver. They can be connected directly to the USART pins `RXD0` and `TXD0` directly (pins `PD0` and `PD1`).

In order to exercise this program, there needs to be a terminal connected to the device, configured to run at 9600 baud. You can use `putty`, `Hyperterminal`, or any other terminal emulator for this.

The code supplied in class sets up the terminal output as `stdin` and the terminal input as `stdout`. Try configuring the terminal input as `stderr`.

Configuring a shift register (74HC595) as a SPI slave

We will add a shift register and LED array to display the value output from the SPI port. The pin configuration for the 74HC595 shift register is in the following figure.



To use it as an SPI device, we need to determine the connections for `MOSI`, `MISO`, `SCK`, and the device select `SS`.

From the data sheet, the pin `RCK` enables transfer of the input to the output on its rising edge, so we can use it as SS. Serial input (`SER`) can be connected to `MOSI`. Serial output (Q'_H) can be connected to `MOSI`. The clock input (`SCK`) is connected to `SCK` from the SPI port.

There are two other inputs, `SCLR`, and `G`. The `G` input is a kind of "device select" which enables the outputs when it is set low, so we can tie it to ground. The `SCLR` input clears the shift register when it is set low, so we can tie it high.

The outputs can then be connected to the LED array through a resistor array, as for the 7 segment display.

What is the output on the terminal in this configuration? Explain.

Try configuring the SPI port as `stdout`, with the terminal as `stdin` and `stderr`. Is this a good configuration for this example? Why or why not?

Optional:

Replace the SPI port with the second serial port, in SPI mode. Describe in detail the configuration (settings of the register values) required for this.