

## Background Information

### AVR USART:

USART stands for “Universal Synchronous and Asynchronous Receive and Transmit.” It is a protocol for sending and receiving serial data. The protocol allows two machines to communicate by sending individual data *frames* from one to the other. The two machines must agree on several parameters, regarding both the makeup of the frames and the rate at which data will be transmitted.

For this laboratory, we will use the following settings.

- asynchronous transmission
- baud rate: 9600
- 8-bit data
- 1 stop bit
- odd parity
- clock parity bit = 0 (due to asynchronous transmission)

Several registers in the AVR's “I/O Register” space are involved with USART transmission, as follows. For more details, see pages 155-163 of the datasheet.

- UBRRH/UBRRL: This pair of registers is used to set the baud rate. For our purposes a slow baud rate is sufficient. To set up 9600 baud, set:
  - UBRRH = \$00
  - UBRRL = \$06
- UCSRC: Please use the following settings for both Tx (transmit) and Rx (receive).
  - bit7 = 1; because this register and UBRRH share space -- set to 1 to access UCSRC.
  - bit6 = 0; asynchronous
  - bit5 and bit4 = 1; odd parity
  - bit3=0; 1 stop bit
  - bit2 and bit1=1; 8-bit data
  - bit0=0; clock parity bit=0
- UCSRB:
  - bit7: should be set to 1 for Rx, and to 0 for Tx. When receiving data, this enables an interrupt which occurs when a complete frame has been received and loaded into the UDR register (discussed further below).
  - bit6: should be set to 0 for Rx and to 1 for Tx. When transmitting data, this enables an interrupt which occurs when a complete frame has been transmitted. Thus, the USART unit is ready for a new byte to be loaded into the UDR register for transmission.

- bit 4: should be set to 1 for Rx, and to 0 for Tx. This enables the receive functionality of the USART unit.
- bit 3: should be set to 0 for Rx, and to 1 for Tx. This enables the transmit functionality of the USART unit.
- all other bits should be set to 0
- UCSRA: There is only one flag we will need here, and it is only needed for Tx.
  - bit5=1; Initialize this bit to 1 for Tx only. It indicates that the UDR register is initially empty. Without this setting, values written to UDR will be ignored.
- UDR (USART I/O Data Register):
  - Rx: When an incoming byte is complete, it is placed in the UDR register automatically and an Rx Complete interrupt is generated. You can then read the data with a statement like "IN r4, UDR".
  - Tx: To send a byte of data, place it in the UDR register with an "OUT" statement. When the data transmission has been completed, a Tx Complete interrupt will be generated, which is your cue to place the subsequent byte.

*NOTE: The AVR datasheet indicates that the USART settings should be initialized BEFORE global interrupts are enabled.*

### **USART Interrupts:**

We will use the Rx Complete (RXC) and Tx Complete (TXC) interrupts in this laboratory. The instructions above indicate how to enable them. To use them, you must also include RJMP statements in the appropriate locations of the interrupt vector:

- RXC: address \$009 (i.e., the tenth position in the vector)
- TXC: address \$00B (i.e., the twelfth position)

### **AVR pins associated with USART Rx and Tx:**

The receive (RXD) and transmit (TXD) signals use the same pins as PD0 and PD1. When the receive and transmit functions are enabled, the RXD and TXD function overrides the normal function (as bits of PortD) for these pins.

These pins can be accessed on the STK500 development board as follows:

- RXD: Expansion header 1, pin 38
- TXD: Expansion header 1, pin 37

## Laboratory Exercises

1. In this exercise you will write a program that causes your AVR to receive data via the USART unit and display that data on the LEDs as it is received. To do this:
  - Set up an interrupt vector that recognizes and responds to the USART RXC interrupt, located at address \$009 of program memory
  - Initialize the USART for receiving data. (See background information above.)
  - Set up the stack and enable global interrupts. Per the AVR datasheet, global interrupts should be enabled AFTER initializing the USART.
  - Write a main loop that does nothing but spin its wheels while waiting for interrupts. You are writing an “event driven” AVR program!
  - Write an interrupt service routine that is triggered by the USART RXC interrupt. This routine should read the value just received in register UDR and output it to the LEDs for display. You do not need to introduce a delay here; that will be done by the sending program.
  - I will set up a test station where I can help you test your program. (I have already completed Program 2 below, which can send data via USART.)
2. Write a program that displays a recognizable repeating sequence of lights – preferably one that is different from your neighbor's. However, the structure of this program will differ from the similar one you wrote for the last laboratory. This time, you should:
  - Do the various set-up activities needed to control USART transmission via interrupts. (This part is similar to program 1 above, but some of the settings will differ. See the background information above.)
  - Write the initial value in your light sequence to the UDR register, and also display the value on your LEDs, as the last step of your set-up code. (This needs to be done to start the transmission process. Since the Tx Complete interrupt is generated when a byte *completes*, the first byte must be loaded without waiting for the first interrupt signal.)
  - Write a main loop that does nothing but spin its wheels while waiting for interrupts.
  - Write an interrupt service routine that is triggered by the USART TXC interrupt. This routine should: update the current byte in your light sequence, write that byte to the UDR register, display it on your LEDs, and cause a delay of an appropriate length to allow the light sequence to be seen.
  - Again, I can help you test your program when it is ready.