EE443 - Embedded Systems

Experiment 8

Serial Data Transmission

Purpose: Establish serial communication between the MCU and a personal computer. Utilize the USART module and the related interrupt functions to perform serial I/O operations. Optimize timing of serial I/O functions by using a circular queue buffer to store the data to be transmitted.

Programming Hints

ATmega328 USART

In this experiment, attenuation of the PWM output will be controlled according to the commands received through the serial port instead of the switches used in the previous experiments. The text output displayed on the LCD will also be sent out through the serial port. The USART in ATmega328 can provide three interrupts to manage serial I/O operations in ISRs. SFR settings related to the USART functions are summarized below.

Register	Description	bit-7	bit-6	bit-5	bit-4	bit-3	bit-2	bit-1	bit-0
UDR0	USART I/O Data Register	USART buffer registers for receive and transmit data							
UBRR0H	USART Baud Rate Register High	---	---	---	---	UBRR0[11:8] - 4 MSBs of clock divider			
UBRR0L	USART Baud Rate Register Low	UBRR0[7:0] - 8 LSBs of clock divider setting							
UCSR0A	USART Control and Status Register A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
UCSR0B	USART Control and Status Register B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ0[2]	RXB80	TXB80
UCSR0C	USART Control and Status Register C	UMSEL0[1:0]		UPM0[1:0]		USBS0	UCSZ0[1:0]		UCPOL0

UDR0 – USART I/O Data Register

The data received through USART can be read from the receive buffer register after the Receive Complete flag is set. **UDR0** is also used to write transmit data to the USART Data Register.

UBRR0H and UBRR0L – USART Baud Rate Registers

The 12-bit number given by **UBRR0[11:0]** is used in a divide-by-N counter to set the baud rate. The baud rate can be calculated as a function of the main system clock and the number **N**, written to **UBRR0[11:0]**.

$$\text{Baud Rate} = \frac{F_{CPU}}{16\,(N+1)}$$

UCSR0A – USART Control and Status Register A

RXC0 : USART Receive Complete flag. Set when there are unread data in the receive buffer. Use this interrupt to read incoming data.

TXC0 : USART Transmit Complete flag. Set when Transmit Shift Register has been shifted out and there are no new data present in the transmit buffer.

UDRE0 : USART Data Register Empty flag. Set when the buffer is empty, and therefore ready to be written. Use this interrupt to send data out.

FE0, DOR0, UPE0 : Frame Error (invalid stop bit), Data OverRun (receive buffer cannot take incoming data), and USART Parity Error indicator flags. Always set these bits to zero when writing to **UCSR0A** register.

U2X0 : Doubles the baud rate for the asynchronous operation if set to 1.

MPCM0 : Multi-processor Communication Mode. Enables address detection in received data. Set to **0** for regular serial I/O.

UCSR0B – USART Control and Status Register B

RXCIE0, TXCIE0, and **UDRIE0 :** Interrupt enable bits corresponding to the **RXC0**, **TXC0**, and **UDRE0** flags in **UCSR0A** register.

RXEN0 : Writing this bit to one enables the USART Receiver.

TXEN0 : Writing this bit to one enables the USART Transmitter.

UCSZ0[2] : Combined with the **UCSZ0[1:0]** in **UCSR0C** register, selects the number of data bits in receive and transmit frames. Set **UCSZ0[2:0] = 011** for standard 8-bit data transmission.

RXB80 and **TXB80 :** The 9th data bit used in 9-bit receive and transmit frames.

UCSR0C – USART Control and Status Register C

UMSEL0[1:0] : Selects USART mode of operation. Set to **00** for asynchronous USART.

UPM0[1:0] : Selects parity mode. Set to **00** for no parity.

USBS0 : Selects the number of stop bits; **0** =>1 stop bit, **1** =>2 stop bits.

UCPOL0 : Selects the clock polarity in synchronous mode. Set to **0** for asynchronous mode.

Preliminary Work

1. How many data bits can be transmitted in one second at 9600 baud rate, when each data frame contains one start bit, 8 data bits, and one stop bit?

How many data bytes per second can be transmitted at 9600 baud rate?

How many ADC samples per second can be transmitted at 9600 baud rate, if each ADC sample takes 4 bytes of text (three decimal digits followed by a delimeter character, such as space or comma).

2. Write new enqueue and dequeue functions that imlement a circular queue for the data to be transmitted through the serial port. The dequeue function will write a byte to the ATmega328 USART data register instead of sending it to the LCD module.

Note that a new set of global variables should be declared for the USART circular queue buffer. Set the USART buffer size to 32 bytes.

3. Determine the USART SFR settings according to the following requirements for serial interface:

- Asynchronous receive/transmit mode.

- 9600 baud rate.
- 8 data bits, 1 stop bit, no parity check.
- Enable USART "Receive Complete" and "Data Register Empty" interrupts.

4. Write an ISR (**`ISR(USART_RX_vect)`**) that processes the characters received from serial port. The ISR sets the attenuation control variable (i.e. **Atten**) according to the received characters instead of the push-button switch inputs. The receiver ISR should perform the following operations:

- Detect two valid commands:
  - If the received character is '**u**' or '**U**' then decrement **Atten** (if Atten > 1).
  - If the received character is '**d**' or '**D**' then increment **Atten** (if Atten < 11).

  The ISR ignores all other characters. Note that lowercase letters can be converted to uppercase letters by clearing the bit-5 of the received character. For example,

```
'A' = 0x41,  'a' = 0x61,  0x41 = 0x61 & 0xDF
'D' = 0x44,  'd' = 0x64,  0x44 = 0x64 & 0xDF
```

- Display the new **Atten** setting on the second line of LCD after receiving a valid command. LCD commands will be sent through the LCD circular queue as in the previous experiment.
- Send back the new **Atten** display data using the USART circular queue:
  - Send a carriage return (**0x0D**) to start a new line on the receiver terminal.
  - Send the text output (i.e. "**`Atten= 3`**") .

5. Write another ISR (**`ISR(USART_UDRE_vect)`**) that sends out the queued USART data.

- Call the dequeue function you wrote for the USART circular buffer.
- Set the pin-5 of port-D at the beginning of the ISR and clear the pin before the ISR returns.

Procedure

1. Create a new project directory and a new AVR project for Experiment 8. Copy the main program and the ISR code you wrote in Experiment 7 and add the files **LCDmodule.c** and **LCDmodule.h** to the new AVR project in Code::Blocks.

In the main program, add the statements for USART initialization and delete the statements in the **`while(1)`** loop. All tasks will be performed in the ISRs.

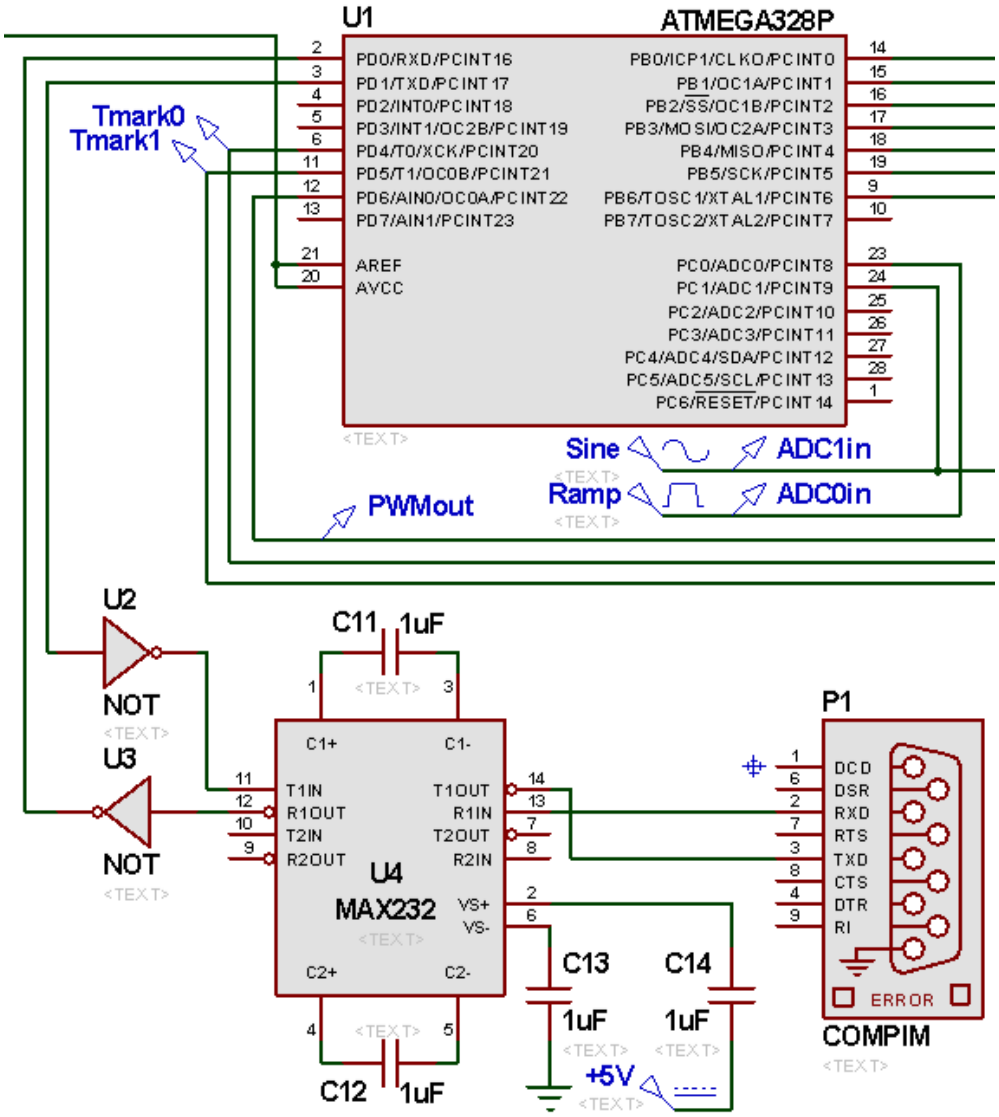
Add the enqueue and dequeue functions and the two ISRs you wrote for the USART in the preliminary work.

Modify the Timer-1 ISR, so that an ADC sample will be sent out through the USART buffer once in every **10 ms**. Send all ADC samples to PWM output and display them on the LCD as it is done in Experiment 7. To obtain the serial port output, skip 9 samples, and send the text output for the 10th sample to the USART circular buffer. Place a delimeter character such as a space (**0x20**) or a comma (**0x2C**) before sending out each ADC sample.

Compile your program and debug if necessary.

2. Open the design file for Experiment 7 in **ISIS**, and save it with a different name in the new project directory. Select the **.hex** file generated by the compiler for Experiment 8 in the **ATMEGA328P** properties window.

Add the serial port components provided in **Exp8_SerialPort.sec** file by using the **Import Section...** option in the **File** menu. Disconnect the **RXD** and **TXD** pins from the switches and connect these pins to the serial port as shown below. Enter proper reference labels for all components in the imported section.



The I/O voltage levels on the processor side of MAX232 are compatible with 5 V TTL logic. On the connector side, signal levels are +/-10 V as required for the RS-232 interface. MAX232 has two charge pumps to convert +5 V supply input to +/-10 V for the RS-232 driver. The four capacitors are necessary for the charge pumps.

Apply a **10 Hz** sinusoidal waveform varying between **0 V** and **5 V** to the **ADC1** input.

Keep all oscilloscope probe connections as they were in the previous experiment.

3. You will use the program, **Terminal.exe**, to monitor the data transmitted by the simulated circuit. Proteus and Terminal.exe access two virtual COM ports on your computer. Another program, **Virtual Serial Ports Emulator (VSPE)**, connects these two virtual COM ports to each other as if there is a RS-232 data cable between the ports. The link between Proteus and Terminal.exe can be established as follows.

- Extract the contents of **VSPE.zip** archive, and run **SetupVSPE.msi** to install the emulator.
- Run **VSPE** and select the menu item **Device/Create**. Select the **Pair** device type, choose two different COM ports as virtual serial port 1 and port 2.
- Start Proteus, open your design file, and edit the **COMPIM** port properties. Select the first virtual port determined in VSPE. Enter other communication options.
- Run Terminal.exe and select the menu item **File/New session**. Select the second virtual port determined in VSPE. Activate the port using **File/Open port** option and select **View/Terminal view** to see the received data in text format.

4. Start the simulation on Proteus and monitor the data received at Terminal.exe. Compare the period of received data with the period of ADC input waveform.

Test the USART receiver ISR by typing command characters in Terminal.exe to change the **Atten** setting. Check that the new **Atten** value is displayed properly on the LCD and on the terminal window.

5. Activate the "**Send Window**" in Terminal.exe by using the **View/Toolbars/Send Window** menu item. Type five or more successive command characters, such as "**DDDDD**" and click on the **Send** button. Check the attenuation values displayed on the terminal window.

Question 1: What is the cause of display errors on the terminal window? What can be done to prevent these errors?

6. Eliminate the unnecessary "Data Register Empty" interrupts generated by USART. Disable the interrupt when there are no bytes left in the USART circular queue. Enable the interrupt after writing data into the USART queue.

Question 2: How can you determine the CPU load using the following idle loop in the main program?

```
while(1)
{ PORTD |= _BV(PORTD4);   // Set timing marker.
  _delay_ms(50);
  PORTD &= ~_BV(PORTD4);  // Clear timing marker.
  _delay_ms(50);
}
```

Measure the CPU time spent for execution of the idle loop in one second. How much CPU time is spent for execution of the ISRs in one second?

Hint: Each execution of **`_delay_ms(50)`** function requires **50 ms** CPU time. If the ISRs use **X ms** of CPU time in **50 ms**, then it takes **50+X ms** to execute the **`_delay_ms(50)`** function.