ECE 375: Computer Organization and Assembly Language Programming

Lab 7 – Remotely communicated Rock Paper Scissors

SECTION OVERVIEW

Complete the following objectives:

- Write an assembly program (for two separate `mega32u4` boards) and have them interact.
- Learn how to configure and use the *Universal Synchronous/Asynchronous Receiver/Transmitter* (USART) module on the ATmega32U4 microcontroller.
- Learn how to configure and use the 16-bit Timer/Counter1 module to generate a 1.5-sec delay.

PRELAB

If you consult any online sources to help answer the prelab questions, you **must** list them as references in your prelab.

1. In this lab, you will be utilizing Timer/Counter1, which can make use of several 16 bit timer registers. The datasheet describes a particular manner in which these registers must be manipulated. To illustrate the process, write a snippet of assembly code that configures OCR1A with a value of 0x1234. For the sake of simplicity, you may assume that no interrupts are triggered during your code's operation.
2. Each ATmega32U4 USART module has two flags used to indicate its current **transmitter** state: the Data Register Empty (UDRE) flag and Transmit Complete (TXC) flag. What is the difference between these two flags, and which one always gets set first as the transmitter runs? You will probably need to read about the *Data Transmission* process in the datasheet (including looking at any relevant USART diagrams) to answer this question.
3. Each ATmega32U4 USART module has one flag used to indicate its current **receiver** state (not including the error flags). **For USART1 specifically**, what is the name of this flag, and what is the interrupt vector address for the interrupt associated with this flag? This time, you will probably need to read about *Data Reception* in the datasheet to answer this question.

BACKGROUND

As previous labs have demonstrated, microcontrollers are well-suited for tasks such as: receiving user input and generating output via general-purpose I/O (GPIO) ports, performing arithmetic and logic computations, and accurately measuring intervals of time. However, microcontrollers are also frequently used for tasks that require communication with the "outside world", which for our purposes means anything that exists outside of the `mega32U4` board.

When using a microcontroller to interact with the outside world, it is very convenient to have a structured, *standardized* way of exchanging information, as opposed to manually implementing data transfer using the GPIO ports. For this reason, microcontrollers often implement a variety of *communication protocols*.

Communication protocols come in two varieties: *parallel* and *serial*. Parallel communication protocols are typically used for internal microcontroller communication, such as a shared data bus. Serial communication protocols are more commonly used for external interactions. Some common examples of serial protocols are *Serial Peripheral Interface* (SPI) and *Two-wire Serial Interface* (TWI).

The ATmega32U4 microcontroller, in addition to providing built-in modules for both SPI and TWI, also comes with *Universal Synchronous/Asynchronous Receiver/Transmitter* (USART) modules. USART is not a communication protocol; instead, it is a highly-configurable hardware module that can be setup to implement point-to-point serial communication (with various parity, data rate, and frame format settings). If two microcontrollers each have a USART module, then they can perform *full-duplex* serial communication as long as their respective USART modules are configured with the same settings, and as long as there is a physical interface (wired) that links their respective TX/RX pins. **Specifically, TX and RX pins in a board need to be wired with RX and TX pins in the other board, respectively. GND pins in both boards also need to be connected.**

Board 1		Board 2
PD2	↔	PD3
PD3	↔	PD2
PD.gnd	↔	PD.gnd

IMPLEMENTATION

In this lab, there is a challenge part to earn extra credits. For students who have decided to implement the challenge part, you can just submit the challenge code to Canvas. Challenge code will replace the requirement of the source code.

In this project you will be designing a system to battle of rock paper scissors between two ECE375 boards. We will use the LCD screen on our board to prompt messages (i.e., ready, start, result, etc.) and choices of your hands (i.e., rock, paper and scissors). For a smooth process, we will use the 4 LEDs, which are connected to PB4-7, as a countdown timer.

Functionality

When the CPU firsts boots, the LCD screen should show the following content to the user:

```
Welcome!
Please press PD7
```

This content will remain indefinitely until the user presses the button which is connected to Port D, pin 7. After the button is pressed, the user's board should transmit a ready signal to the opponent so that it knows the user is ready. Additionally, this information will be displayed on the user's screen:

```
Ready. Waiting
for the opponent
```

This content will remain indefinitely until the opponent player presses the PD7 on their board and sends a ready signal to the user's board.

When the user's board transmits/receives the ready signal to/from the opponent, both the user and the opponent's 4 LEDs are on and start counting down by turning off one by one at each 1.5-second. Simultaneously, their screens will also display:

```
Game start
```

The second line of the LCD is where the user is able to select the choice among three gestures (i.e., rock, paper, and scissors) that they want to send.

- First pressing PD4 selects the Rock gesture.
- Pressing PD4 changes the current gesture and iterates through the thee gestures in order. For example, Rock → Paper → Scissor → Rock → ...
- Pressing PD4 immediately displays the choice on the user's LCD.

(Challenge part should meet all of the above features as well.)

For example, if the display currently shows:

```
Game start
```

then pressing PD4 will immediately update the screen to display:

```
Game start
Rock
```

pressing PD4 again will display:

```
Game start
Paper
```

At this point, the opponent's play should not appear in the user's board even if it receives the opponent's signals.

After 4 LEDs are off, which means 6 seconds timer is done, the final choice of the opponent's gesture and the user's will be displayed in the first line and the second line of the LCD respectively. For example, if the opponent's gesture is scissor and the user's one is paper, the user's LCD will display:

```
Scissor
Paper
```

Meanwhile, the opponent's LCD will display:

```
Paper
Scissor
```

Simultaneously, 4 LEDs on both boards will turn on in order to start counting down for the 6 seconds delay. After timeout, the outcome will be displayed on the first line of the screens and start a new timer. For example, the user's LCD will display:

```
You lost
Paper
```

On the other hand, the opponent's LCD will display:

```
You won!
Scissor
```

if both players have chosen the same gesture, the first line should be updated with a draw message.

After the new timer is done, the code will return to the prompt as shown:

```
Welcome!
Please press PD7
```

The user can now enter another game round, exactly as before.

Specifications

- You will need to configure the USART1 module on the boards. Also, although the ATmega32U4's USART modules can communicate at rates as high as 2×10^6 bits per second (2 Mbps), you will use the (relatively) slow baud rate of **2400 bits per second** with **double data rate**.
- Packets, which consist of **8-bit data frame with 2-stop bits and no parity bit**, will be sent back-to-back by the USART modules. One of packets will be a "send ready" byte, which indicates the sender is ready to start a game. The others include a choice of their gestures.
- The LCD display needs to be updated immediately whenever the user provides input.
- Single button press must result in a single action.
- You **must use the Timer/Counter1 module with NORMAL mode** to manage the countdown unit timing. You may design your code to use polling or you may use interrupts (either approach is fine). You may not utilize any busy loop for the code delay, although it is allowed to loop if you are monitoring an interrupt flag.
- Do not include switch debouncing delays of more than 150ms. A busy loop for debouncing is okay.
- The LCD screen must never display symbols, gibberish, or other undesired output.

Hints

- Students often ask about the behavior of the LEDs which are connected to PB1-PB3. In some cases the LCD library will illuminate one or more of those LEDs (even if you don't want them to be enabled). You do not need to worry about this. The project guidelines do not dictate the behavior of any LEDs which aren't mentioned in the project instructions. However, overwriting values to PB1-PB3 (e.g., during 4 LEDs countdown implementation) can result in the failure of using the LCD. You will need to avoid overwriting unused bits (PB0-PB3).

STUDY QUESTIONS / REPORT

A full lab write-up is required for this lab. When writing your report, be sure to include a summary that details **what you did and why, and how you configured USART and Timer/Counter 1 including any calculations.** Your write-up and code must be submitted prior to your final (Week 10) lab session. As usual, NO LATE WORK IS ACCEPTED.

Study Questions

No additional questions for this lab assignment.

CHALLENGE

Rock, Paper, Scissors and Get one out! is a variant of the game rock paper scissors. Two players first shoot gestures in both right and left hands. After checking opponents' hands, each player shoots either their right or left hand to win the game following the base rule of rock paper and scissors. In order to earn extra credit, incorporate all of the following features:

Once the game starts and a 6-sec timer begins to countdown, the user is at the default prompt as follows:

```
Game start
```

then pressing PD4 will choose the gesture of **right hand** and iterates through the three gestures as before. For example, the user's LCD is updated as following:

```
Game start
        |Rock
```

On top of this, pressing PD5 works the same as PD4 functionality but it will select/change the current gesture of **left hand**. Suppose that the user presses PD5 twice, then the LCD prompt will change to the following:

```
Game start
Paper    |Rock
```

After the timer is done, the final choice of the opponent's hands and the user's will be displayed in the first line and the second line of the LCD respectively. For example, if the opponent's gestures are rock and scissor, the user's LCD will display:

```
Rock     |Scissor
Paper    |Rock
```

Meanwhile, the opponent's LCD will display:

```
Paper    |Rock
Rock     |Scissor
```

Simultaneously, 4 LEDs on both boards will turn on in order to start counting down for the 6 seconds delay. During the delay, each player choose which hand they want to shoot. Pressing PD4 and PD5 will decide right and left side, respectively. For example, if the user presses PD5 (left side) while the opponent presses PD4 (right side), the outcome will be displayed after timeout and start a new timer. For example, the user's LCD will display:

```
Scissor
Paper
```

After timeout, a new timer will start again and the game result will be displayed on the first line of the screens:

```
You lost
Paper
```

After the new timer is done, the code will return to the prompt as shown:

```
Welcome!
Please press PD7
```

The user can now enter another game round, exactly as before.