ECE 375 LAB 7

Lab session: 015

Time: 1200-1350

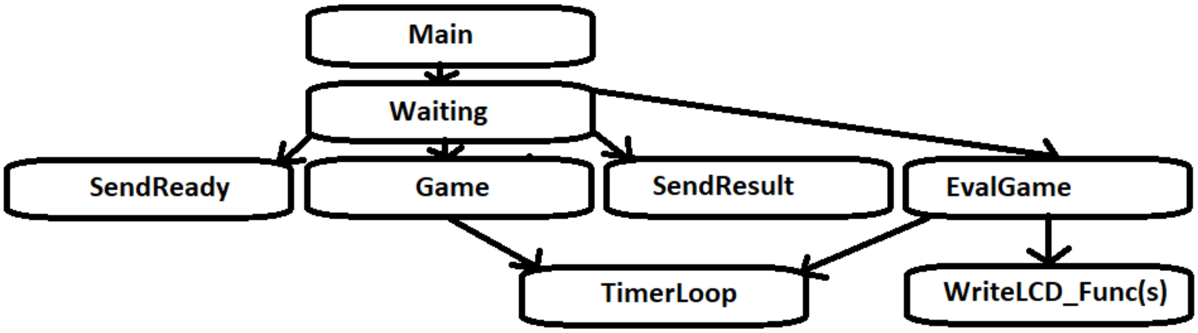
Author name: Paul Lipp

Programming partner name: Ryan Muriset

Introduction

In this lab we explored the USART communication protocol, and how it can be leveraged to have two AVR boards communicate with each other. This was explored in the form of a Rock-Paper-Scissors game, where the two boards send the played hand to each other and reveal the winner. It also reinforced concepts learned in previous labs, such as Timer/Counters, Interrupts, and Polling for PIN inputs.

Design



Program Overview

The Main flow of the program is explained in the design. We use a MAIN function to poll for the PD7 button press. Once that is pressed, we step into the Game. The WAITING Routine Writes the waiting prompt to the LCD and calls the GAME routine once both boards are ready. The GAME routine starts the game and initiates the TimerLoop, which is the LED countdown script responsible for counting down the LEDs every 1.5 seconds. During this time, interrupts are enabled, which means PD4 can be pressed to cycle the options on the LCD between rock - paper - scissors. After all LEDs are off, the Game routine returns, and Waiting calls the EvalGame, which prints both players hands on the LCD. It lastly evaluates the winner of the game, and prints the final result onto the LCD. We make use of various helper functions to help with writing to the LCD throughout the program.

Initialization Routine

We initialize the stack pointer first. Next the USART1 registers are configured according to the manual for this lab. We set the baud rate to 2400 and enable the RXC interrupt for later use. The data frame is formatted to 8 bits with 2 stop bits. We also set PortB for output (LEDs) and PortD for input (Pushdown Buttons). Timer Counter 1 is configured to have a 256-clk prescaler, so it can effectively generate a 1.5 second delay. EICRA is configured to enable INT0, and the LCD is initialized

Main Routine

The Main routine executes an infinite loop, during which it polls for PD7 input. Once PD7 is input, it calls the Waiting routine and doesn't return until the game ends.

Subroutines

1. Waiting

Waiting subroutine writes the waiting prompt to the LCD, and calls several functions responsible for running the game. The first function it calls is the SendReady function, which sends the ready signal to the other board as soon as PD7 has been pressed. Waiting also calls EvalGame and Game, during which the players make their choices and the final choices are compared and evaluated, respectively.

2. Game Routine

During the game routine the “Game Start” prompt is written to the LCD, and the player has the option to cycle through the LEDs via the Interrupt0. In the Game routine, the LEDs are being turned off every 1.5 seconds, to create a 6 seconds delay for the player to make a decision

3. SendResult

During the SendResult routine the final hand the player chose is sent to the other board via USART communication. UCSR1A is polled to ensure that the UDRE1 bit is set, meaning the data register is empty. If that’s the case, the data is loaded into the UDR1 register.

4. HandleRECX

This is the service routine for interrupt $0032, which is the USART1 RXC1 flag, which is raised when there’s unretrieved data in UDR1. This routine retrieves that data and loads it into opplay, the register that keeps track of the opponents play.

5. EvalGame

Both hands are printed onto the LCD during this routine, and the LED countdown is initiated

6. HandleINT0

This routine is the service routine for INT0. In this routine, the register keeping track of the players hand is incremented by 1, and the new value is printed onto the LCD. $00 = Rock, $01 = Paper, $02 = Scissors.

7. LCD Write Helper Functions

Various LCD Helper functions are used throughout the program. They all have the same logic, which is first the string is moved from program memory to data memory address $0100 for line 1, or $0110 for line 2, using the Z-pointer and the lpm command. Once that is done, a loop iterates over each character and moves it to the correct data memory address, and finally the LCDWrLnX function is called.

Testing

Case	**Expected**	**Actual meet expected**
Send Scissors	Sent Scissors	Yes

Send Paper	Sent Paper	Yes
Send Rock	Sent Rock	Yes
Receive Scissors	Receive Scissors	Yes
Receive Paper	Receive Paper	Yes
Receive Rock	Receive Rock	Yes
Rock Vs Rock	Draw	Yes
Rock Vs Paper	Paper Wins	Yes
Rock Vs Scissors	Rock Wins	Yes
Paper Vs Paper	Draw	Yes
Paper Vs Scissors	Scissors Wins	Yes

Source Code

```
;***********************************************************
;*
;*    This is the TRANSMIT skeleton file for Lab 7 of ECE 375
;*
;*      Rock Paper Scissors
;*     Requirement:
;*     1. USART1 communication
;*     2. Timer/counter1 Normal mode to create a 1.5-sec delay
;***********************************************************
;*
;*     Author: Paul Lipp and Ryan Muriset
;*     Date: 2023-03-11
;*
;***********************************************************

.include "m32U4def.inc"      ; Include definition file

;***********************************************************
;*  Internal Register Definitions and Constants
;***********************************************************
.def   mpr = r16             ; Multi-Purpose Register
.def    counter = r19
.def    play = r23
.def    opplay = r18

; Use this signal code between two boards for their game ready
.equ   ReadyComp = $FF
.equ    PD_seven = 7
.equ    PD_four = 4
.def    waitcnt = r17                        ; Wait Loop Counter
.def    ilcnt = r25                          ; Inner Loop Counter
.def    olcnt = r24                          ; Outer Loop Counter

.equ    WTime = 15

;***********************************************************
```

```
;*  Start of Code Segment
;***********************************************************
.cseg                        ; Beginning of code segment

;***********************************************************
;*  Interrupt Vectors
;***********************************************************
.org   $0000                 ; Beginning of IVs
       rjmp   INIT                   ; Reset interrupt

.org    $0002
        rcall HandleINT0
        reti

.org    $0032
        rcall HandleRECX
        reti

.org   $0056                 ; End of Interrupt Vectors

;***********************************************************
;*  Program Initialization
;***********************************************************
INIT:
    ;Stack Pointer (VERY IMPORTANT!!!!)

    ; Initialize the Stack Pointer

        ldi    mpr, low(RAMEND)    ; Initialize Stack Pointer
        out SPL, mpr
        ldi mpr, high(RAMEND)
        out SPH, mpr

    ;I/O Ports
        ldi mpr, $00                 ; Initialize Port D for input
        out DDRD, mpr
        ldi mpr, $FF
        out PORTD, mpr

        ldi mpr, $FF
        out DDRB, mpr
        ldi mpr, $00
        out PORTB, mpr

    ;USART1
        ;Set baudrate at 2400bps
        ;Enable receiver and transmitter
        ;Set frame format: 8 data bits, 2 stop bits
        ;Set baudrate at 2400bps
        ldi mpr, $00
        sts    UBRR1H, mpr
        ldi mpr, $CF
        sts UBRR1L, mpr

        ldi mpr, (1<<UDRE1)
        sts UCSR1A, mpr

        ;Enable receiver and transmitter
        ldi mpr, (1<<RXCIE1)|(1<<RXEN1)|(1<<TXEN1)
        sts    UCSR1B, mpr

        ;Set frame format: 8 data bits, 2 stop bits
        ldi mpr, (1<<USBS1)|(3<<UCSZ10)
        sts UCSR1C, mpr

    ;TIMER/COUNTER1
        ;Set Normal mode, 256 pre-scaler

        ldi mpr, 0b00000000
        sts TCCR1A, mpr
        ldi mpr, 0b00000100
        sts TCCR1B, mpr

        ldi    mpr, $02
        sts    EICRA, mpr


    ;Other
    ;Initialize LCD

        rcall LCDInit
        rcall LCDClr

        ldi ZL, low(Init_START<<1)
        ldi ZH, high(Init_START<<1)
        ldi YL, $00
        ldi YH, $01
        ldi counter, 8

        rcall InitWriteL1

        ldi ZL, low(InitL2_START<<1)
        ldi ZH, high(InitL2_START<<1)
        ldi YL, $10
        ldi YH, $01
        ldi counter, 16

        rcall InitWriteL2

    ;Write initial welcome to LCD

        ldi play, $00

        ldi mpr, $01
        out EIMSK, mpr

        ldi opplay, $00

        ;sei
    ;Initialize the LCD



;***********************************************************
;*  Main Program
;***********************************************************
MAIN:
        in mpr, PIND                 ;Polling PIND
        andi mpr, (1<<7)      ;Check for PD7 press
        cpi mpr, (1<<7)
        breq NEXT
        rcall WAITING                ;Start Game Routines
        ret

NEXT:
        rjmp MAIN

;***********************************************************
;*    Functions and Subroutines
;***********************************************************



WAITING:
        ldi ZL, low(PressedL1_START<<1)
        ldi ZH, high(PressedL1_START<<1)
        ldi YL, $00
        ldi YH, $01
        ldi counter, 14
        rcall ReadyWrLn1

        ldi ZL, low(PressedL2_START<<1)
        ldi ZH, high(PressedL2_START<<1)
        ldi YL, $10
        ldi YH, $01
        ldi counter, 16

        rcall ReadyWrLn2
        ;Write Ready Lines 1 and 2

        rcall SendReady       ;Send Ready USART1
        rcall LCDClr
        rcall Game            ;Call Game Routine
        sei
        rcall SendResult    ;SendResult to USART1
        ldi waitcnt, 50       ;Wait to allow interrupt to trigger
        rcall Wait
        cli
        rcall EvalGame        ;Write both plays to LCD
        rcall LCDClr
        rcall WritePlay1
        rcall PrintResult    ;Write Result

        ;LEDs Countdown Script
        ldi mpr, (1<<7|1<<6|1<<5|1<<4)
        out PORTB, mpr
        rcall TimerLoop
        ldi mpr, (0<<7|1<<6|1<<5|1<<4)
        out PORTB, mpr
        rcall TimerLoop
        ldi mpr, (0<<7|0<<6|1<<5|1<<4)
        out PORTB, mpr
        rcall TimerLoop
        ldi mpr, (0<<7|0<<6|0<<5|1<<4)
        out PORTB, mpr
        rcall TimerLoop
        ldi mpr, (0<<7|0<<6|0<<5|0<<4)
        out PORTB, mpr

        ret

;***********************************************************
;*    Called when PD7 is pressed. Writes Waiting for player
;*  and calls SendReady. Starts Game once SendReady returns
;***********************************************************

SendReady:
        ldi mpr, $FF          ;Send Ready ($FF) to USART1
        sts UDR1, mpr

        lds mpr, UDR1         ;Poll for Ready from USART1
        cpi mpr, $FF
        brne SendReady
        ret

;***********************************************************
;*    Sends FF to USART UDR1, then polls UDR1 to wait for Ready
;*  from other board. Returns to Waiting
;***********************************************************

HandleINT0:
        cli
        inc play              ;Service INT0, Cycle Play Options on LCD
        cpi play, $03
        breq Handle2
        rcall WritePlay1
        sbi EIFR, 0
        ldi waitcnt, WTime
        rcall Wait
        sei
        ret
Handle2:
        rcall Reset
        ret

;**********************************************************
;*    Interrupt Service, Increments play reg
;**********************************************************


Game:
        sei                                         ;Interrupts   are   enabled   during   this
routine to
                                                           ;allow   for   cycling   of   line   2
options
        rcall WriteLine1Start        ;Write "game start" prompt Line 1
        rcall WritePlay1             ;Write option Line 2
        ldi mpr, (1<<7|1<<6|1<<5|1<<4)       ;LED Countdown Script
        out PORTB, mpr
        rcall TimerLoop
        ldi mpr, (0<<7|1<<6|1<<5|1<<4)
        out PORTB, mpr
        rcall TimerLoop
        ldi mpr, (0<<7|0<<6|1<<5|1<<4)
        out PORTB, mpr
        rcall TimerLoop
        ldi mpr, (0<<7|0<<6|0<<5|1<<4)
        out PORTB, mpr
        rcall TimerLoop
        ldi mpr, (0<<7|0<<6|0<<5|0<<4)
        out PORTB, mpr
        ldi mpr, $00
        out EIMSK, mpr

        cli
        ret

;**********************************************************
;*  TimerCounter1 Loop
;*  Has TC1 Count to 18660, which is 65535 - 46875
;*    8/256 = 0.03125 = 32000ns, 32000*x = 1.5s
;*  1.5s/32000 = x; x = 46875
;**********************************************************

TimerLoop:
        ldi mpr, $48
        sts TCNT1H, mpr
        ldi mpr, $E5
        sts TCNT1L, mpr
TimerLoopHelper:
        sbis TIFR1, 0
        rjmp TimerLoopHelper
        sbi TIFR1, 0
        ret

Reset:
        ldi play, $00
        rcall LCDClrLn2
        rcall WritePlay1
        ret

;**********************************************************
;*    Main Game Routine
;**********************************************************

SendResult:
        lds mpr, UCSR1A
        sbrs mpr, UDRE1
        rjmp SendResult
        sts UDR1, play
        ret

HandleRECX:                                         ;USART1  RXC1  Interrupt  Service  Routine.
Retrieves Data from UDR1
        lds mpr, UCSR1A
        sbrs mpr, RXC1
        rjmp HandleRECX
        lds opplay, UDR1

        ret


;**********************************************************
;*    Send Result to other board
;**********************************************************

EvalGame:
        rcall LCDClr
        rcall WriteOpPlay1          ;opplay - Write Opponents Play to Line2
        rcall WritePlay1Ln1         ;play   - Write Play to line1

        ldi mpr, (1<<7|1<<6|1<<5|1<<4)    ; LED Countdown Script
        out PORTB, mpr
        rcall TimerLoop
        ldi mpr, (0<<7|1<<6|1<<5|1<<4)
        out PORTB, mpr
        rcall TimerLoop
        ldi mpr, (0<<7|0<<6|1<<5|1<<4)
        out PORTB, mpr
        rcall TimerLoop
        ldi mpr, (0<<7|0<<6|0<<5|1<<4)
        out PORTB, mpr
        rcall TimerLoop
        ldi mpr, (0<<7|0<<6|0<<5|0<<4)
        out PORTB, mpr
        ret

;**********************************************************
;*    Print both hands
;**********************************************************

PrintResult:                                                ; Print Winner of Game
        cp play, opplay
        brne Print2
        rcall WriteResultDraw
        ret
Print2:
        cpi play, $00                                ; 00 01 02     00  beats  02  ||  01
beats 00 || 02 beats 01
        brne Print3
        rcall PrintResultHelper1
        ret
Print3:
        cpi play, $01
        brne Print4
        rcall PrintResultHelper2
        ret
Print4:
        cpi opplay, $01
        breq PrintWin
        rcall WriteResultLoss
        ret
PrintWin:
        rcall WriteResultWin
        ret

PrintResultHelper1:
        cpi opplay, $02
        brne PrintLossHelper1
        rcall WriteResultWin
        ret
PrintLossHelper1:
        rcall WriteResultLoss
        ret

PrintResultHelper2:
        cpi opplay, $00
        brne PrintLossHelper2
        rcall WriteResultWin
        ret
PrintLossHelper2:
        rcall WriteResultLoss
        ret

;***********************************************************
;*    Print Result
;***********************************************************

;*********************************************************************************************
*************************
;*********************************************************************************************
*************************


;***********************************************************
;*    Write Functions
;***********************************************************

WriteLine1Start:
        ldi ZL, low(Start_START<<1)
        ldi ZH, high(Start_START<<1)
        ldi YL, $00
        ldi YH, $01
        ldi counter, 10

        rcall WriteLine1Helper
        ret

WriteLine1Helper:
        lpm mpr, Z+
        st Y+, mpr
        dec counter
        brne WriteLine1Helper

        rcall LCDWrLn1
        ret

;****************************************************

WritePlay1:
        cpi play, $00
        brne WritePlay2
        ldi ZL, low(Rock_START<<1)
        ldi ZH, high(Rock_START<<1)
        ldi YL, $10
        ldi YH, $01
        ldi counter, 4

        rcall WriteRock
        ret
WritePlay2:
        cpi play, $01
        brne WritePlay3
        ldi ZL, low(Paper_START<<1)
        ldi ZH, high(Paper_START<<1)
        ldi YL, $10
        ldi YH, $01
        ldi counter, 5
        rcall WritePaper
        ret
WritePlay3:
        ldi ZL, low(Scissor_START<<1)
        ldi ZH, high(Scissor_START<<1)
        ldi YL, $10
        ldi YH, $01
        ldi counter, 8
        rcall WriteScissors
        ret

WritePlay4:
        ret

;*********************************************************

WriteOpPlay1:
        cpi opplay, $00
        brne WriteOpPlay2
        ldi ZL, low(Rock_START<<1)
        ldi ZH, high(Rock_START<<1)
        ldi YL, $10
        ldi YH, $01
        ldi counter, 4

        rcall WriteRock
        ret
WriteOpPlay2:
        cpi opplay, $01
        brne WriteOpPlay3
        ldi ZL, low(Paper_START<<1)
        ldi ZH, high(Paper_START<<1)
        ldi YL, $10
        ldi YH, $01
        ldi counter, 5

        rcall WritePaper
        ret
WriteOpPlay3:
        cpi opplay, $02
        brne WriteOpPlay4
        ldi ZL, low(Scissor_START<<1)
        ldi ZH, high(Scissor_START<<1)
        ldi YL, $10
        ldi YH, $01
        ldi counter, 8
        rcall WriteScissors
        ret

WriteOpPlay4:
        ldi YL, $10
        ldi YH, $01

        ldi mpr, $30
        add opplay, mpr
        st Y+, opplay

        rcall LCDWrLn2
        ret

;*********************************************************

WritePlay1Ln1:
        cpi play, $00
        brne WritePlay2Ln1
        ldi ZL, low(Rock_START<<1)
        ldi ZH, high(Rock_START<<1)
        ldi YL, $00
        ldi YH, $01
        ldi counter, 4

        rcall WriteRockLn1
        ret
WritePlay2Ln1:
        cpi play, $01
        brne WritePlay3Ln1
        ldi ZL, low(Paper_START<<1)
        ldi ZH, high(Paper_START<<1)
        ldi YL, $00
        ldi YH, $01
        ldi counter, 5

        rcall WritePaperLn1
        ret
WritePlay3Ln1:
        ldi ZL, low(Scissor_START<<1)
        ldi ZH, high(Scissor_START<<1)
        ldi YL, $00
        ldi YH, $01
        ldi counter, 8
        rcall WriteScissorsLn1
        ret

WritePlay4Ln1:
        ret

;**************************************************************

WriteRock:
        lpm mpr, Z+
        st Y+, mpr
        dec counter
        brne WriteRock

        rcall LCDWrLn2
        ret

WritePaper:
        lpm mpr, Z+
        st Y+, mpr
        dec counter
        brne WritePaper

        rcall LCDWrLn2
        ret

WriteScissors:
        lpm mpr, Z+
        st Y+, mpr
        dec counter
        brne WriteScissors

        rcall LCDWrLn2
        ret

;*************************************************************

WriteRockLn1:
        lpm mpr, Z+
        st Y+, mpr
        dec counter
        brne WriteRockLn1

        rcall LCDWrLn1
        ret

WritePaperLn1:
        lpm mpr, Z+
        st Y+, mpr
        dec counter
        brne WritePaperLn1

        rcall LCDWrLn1
        ret

WriteScissorsLn1:
        lpm mpr, Z+
        st Y+, mpr
        dec counter
        brne WriteScissorsLn1

        rcall LCDWrLn1
        ret

;***********************************************************
;*    Writes Play (RPS)
;***********************************************************

InitWriteL1:
        lpm mpr, Z+
        st Y+, mpr
        dec counter
        brne InitWriteL1

        rcall LCDWrLn1
        ret

InitWriteL2:
        lpm mpr, Z+
        st Y+, mpr
        dec counter
        brne InitWriteL2

        rcall LCDWrLn2
        ret

;***********************************************************
;*    Write Initial Message (Welcome!)
;***********************************************************

ReadyWrLn1:
        lpm mpr, Z+
        st Y+, mpr
        dec counter
        brne ReadyWrLn1

        rcall LCDWrLn1
        ret

ReadyWrLn2:
        lpm mpr, Z+
        st Y+, mpr
        dec counter
        brne ReadyWrLn2

        rcall LCDWrLn2
        ret

;***********************************************************
;*  Write Functions to write Ready, waiting for other player
;***********************************************************

WriteResultWin:
        ldi ZL, low(WIN_START<<1)
        ldi ZH, high(WIN_START<<1)
        ldi YL, $00
        ldi YH, $01
        ldi counter, 4
```

```
WriteWin2:
        lpm mpr, Z+
        st Y+, mpr
        dec counter
        brne WriteWin2

        rcall LCDWrLn1
        ret

WriteResultLoss:
        ldi ZL, low(LOSS_START<<1)
        ldi ZH, high(LOSS_START<<1)
        ldi YL, $00
        ldi YH, $01
        ldi counter, 5

WriteLoss2:
        lpm mpr, Z+
        st Y+, mpr
        dec counter
        brne WriteLoss2

        rcall LCDWrLn1
        ret

WriteResultDraw:
        ldi ZL, low(DRAW_START<<1)
        ldi ZH, high(DRAW_START<<1)
        ldi YL, $00
        ldi YH, $01
        ldi counter, 5

WriteDraw2:
        lpm mpr, Z+
        st Y+, mpr
        dec counter
        brne WriteDraw2

        rcall LCDWrLn1
        ret
;***********************************************************
;*  Write Functions to write Result
;***********************************************************


Wait:
        push    waitcnt              ; Save wait register
        push    ilcnt                ; Save ilcnt register
        push    olcnt                ; Save olcnt register

Loop:    ldi           olcnt, 224    ; load olcnt register
OLoop:    ldi          ilcnt, 237    ; load ilcnt register
ILoop:    dec          ilcnt                 ; decrement ilcnt
        brne    ILoop                ; Continue Inner Loop
        dec    olcnt          ; decrement olcnt
        brne    OLoop                ; Continue Outer Loop
        dec    waitcnt        ; Decrement wait
        brne    Loop                 ; Continue Wait loop

        pop    olcnt          ; Restore olcnt register
        pop    ilcnt          ; Restore ilcnt register
        pop    waitcnt        ; Restore wait register
        ret                   ; Return from subroutine

;***********************************************************
;*    Stored Program Data
;***********************************************************

;-----------------------------------------------------------
; An example of storing a string. Note the labels before and
; after the .DB directive; these can help to access the data
;-----------------------------------------------------------

Init_START:
    .DB        "Welcome!"
Init_END:

InitL2_START:
    .DB        "Please Press PD7"
InitL2_END:

PressedL1_START:
    .DB        "Ready. Waiting"
PressedL1_END:

PressedL2_START:
    .DB        "for the opponent"
PressedL2_END:

Start_START:
    .DB        "Game Start"
Start_END:

Rock_START:
       .DB     "Rock"         ; Declaring data in ProgMem
Rock_END:

Paper_START:
       .DB     "Paper"        ; Declaring data in ProgMem
Paper_END:

Scissor_START:
       .DB     "Scissors"     ; Declaring data in ProgMem
Scissor_END:

WIN_START:
       .DB     "WIN!"         ; Declaring data in ProgMem
WIN_END:

LOSS_START:
       .DB     "LOSS!"        ; Declaring data in ProgMem
LOSS_END:

DRAW_START:
       .DB     "DRAW!"        ; Declaring data in ProgMem
DRAW_END:

;***********************************************************
;*    Additional Program Includes
;***********************************************************
.include "LCDDriver.asm"      ; Include the LCD Driver
```