ECE 375 LAB 8

Remotely Operated Vehicle (USART)

Lab Time: Wednesday 5-7

Zhenggan Zheng

Abhishek Raol

INTRODUCTION

The purpose of this lab is to introduce the USART feature of the AVR Atmega128. The end result of this lab is a robot-transmitter pair that can move via USART commands and freeze other robots that are nearby. After doing this lab, students will learn how to use and control the USART feature to transmit information to other devices.

PROGRAM OVERVIEW

This program is split into two parts: a transmitter and a receiver. The transmitter polls from PORTD and checks to see which button the user presses. Then it transmit the signal that corresponds to the button the user is pressing. The receiver receives the signal from the transmitter and outputs to PORTB the movement corresponding to that signal. The receiver can also transmit a freeze signal that freezes the other robots. When the receiver is frozen 3 times, it will stay frozen until it is reset.

INITIALIZATION ROUTINE

The initialization routine for both the receiver and transmitter are very similar. Both are initialized to 2400bps baud rate by loading 832 into UBRR1 and setting it to double baud rate by setting U2X1 on UCSR1A. They are both initialized to 8 data bits and 2 stop bits by setting the bits on UCSR1C. Then finally the transmitter has TXEN1 set in UCSR1B to enable it to transmit, while the receiver has TXEN1, RXEN1, and RXCIE1 set to enable it to both transmit and receive, as well as enabling the interrupt for when it receives a command. The Receiver needs TXEN1 set to transmit the freeze command. The receiver also has external interrupts set up for falling edge detection by setting up bits in EICRA and masking set by setting up EIMSK. This is to implement the bumpbot behavior.

MAIN ROUTINE

The Main routine for the receiver simply loops infinitely and waits to an interrupt from either the receiver or from the push buttons. The transmitter main routine polls for inputs from PORTD.

RECEIVE ROUTINE

This subroutine is part of the receiver. It loads in input from UDR1 and determines if it is a robot ID, a command or a freeze. It does this by ANDing the input with 0b1000000 and checking to see if it is set. This way only the MSB matters and if it is set then it will go to Testflag subroutine. Otherwise it will go to test Bot ID, if it is the correct Bot ID then it will go Setflag routine, otherwise it will go to Frozen routine

TESTFLAG ROUTINE

This routine is part of the receiver. This subroutine tests to see if register 21 is set, if it is then go to Run subroutine.

RUN ROUTINE

This routine is part of the receiver. The run routine checks the command stored in register 17 and compares it to different binary numbers. When it reaches a number that matches register 17 it will break into the action that corresponds to that command.

Setflag Routine

The Setflag routine is part of the receiver and simply loads 1 into register 21 which is used as a flag.

Moveforward / Movebackward / Turnright / Turnleft / Stop Routines

The above routines are all part of the receiver and are basically identical and is what tells the robot what to do. It loads its respective command into mpr and then outputs to PORTB. It also loads the command into register 25 which is saved for later use. Afterwards it clears r21 to reset the robot for the next command.

ReceiveFreeze Routine

This routine is part of the receiver and is ran when the robot receives the freeze command from the transmitter. It will then output 0b01010101 and freeze other robot.

Frozen Routine

This subroutine is part of the receiver. It will freeze the robot if the flag in register 21 is not set.

Freeze Routine

This routine is ran when the robot is frozen. It output 0b00000001 to PORTB to display that it is frozen. Then it will back up EIMSK and UCSR1B. Then it will write 0s to both of the previous mentioned registers which will prevent the robot from receiving any outside interrupts. It will then call the wait subroutine 5 times to wait for 5 seconds. Then it will restore EIMSK and UCSR1B. Then it increments register 24 and compares it to 3. If register 24 is equal to 3 then run Dead routine.

Dead Routine

The Dead subroutine is part of the receiver and is called when the robot has been frozen 3 times. It will write 0s to EIMSK and UCSR1B and not restore it until the robot is reset.

HitLeft / HitRight Routines

These subroutines are part of the receiver and implements bumpbot behavior. They are called when the interrupts are triggered that calls them. They will turn to robot left or right.

Wait Routine

The wait routine waits for 1 second. This time depends on WTime

Loop / OLoop/ ILoop

These routines are part of the receiver and uses waitcnt to allow wait to last for 1 seconds. This time depends on WTime

CheckBack / Checkleft/ Checkright/ Checkfreeze/ Checkstop Routines

These Routines are part of the transmitter and polls to see which one of them will be activated. When they are activated, rjmp to their respective commands.

MoveForward / MoveBackward / Turnleft / Turnright / Stop / Freeze Routines

These subroutines are part of the transmitter and they transmit whatever command they are assigned to the receiver.

CONCLUSION

The purpose of this lab was to learn how to use USART communication on embedded systems such as the AVR Atmega128. The end result was a robot that can play freeze tag with other similarly programmed robots. This lab is also an accumulative test of skills that students have learned in ECE 375. It requires students to use all the skills they have learned in their previous labs and implement them alongside USART. Upon completing this lab, students will learn not only USART communication but also how to implement a complete system with many features such as USART, bumpbot and freeze.

SOURCE CODE

```
;***********************************************************
;*
;*      AssemblerApplication8Transmitbackup.asm
;*
;*
;*
;*      This is the TRANSMIT file for Lab 8 of ECE 375
;*
;***********************************************************
;*
;*       Author: Zhenggan Zheng and Abhishek Raol
;*         Date: 3/11/2016
;*
;***********************************************************

.include "m128def.inc"              ; Include definition file

;***********************************************************
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                           ; Multi-Purpose Register

.equ    EngEnR = 4                          ; Right Engine Enable Bit
.equ    EngEnL = 7                          ; Left Engine Enable Bit
.equ    EngDirR = 5                         ; Right Engine Direction Bit
.equ    EngDirL = 6                         ; Left Engine Direction Bit
; Use these action codes between the remote and robot
; MSB = 1 thus:
; control signals are shifted right by one and ORed with 0b10000000 = $80
.equ    MovFwd =  ($80|1<<(EngDirR-1)|1<<(EngDirL-1))       ;0b10110000 Move Forward Action Code
.equ    MovBck =  ($80|$00)                                                   ;0b10000000
Move Backward Action Code
.equ    TurnR =   ($80|1<<(EngDirL-1))                              ;0b10100000 Turn Right
Action Code
.equ    TurnL =   ($80|1<<(EngDirR-1))                              ;0b10010000 Turn Left
Action Code
.equ    Halt =    ($80|1<<(EngEnR-1)|1<<(EngEnL-1))         ;0b11001000 Halt Action Code
.equ    Frze = 0b11111000
.equ    BotAddress = $2A
;***********************************************************
;*      Start of Code Segment
;***********************************************************
```

```
.cseg                                                   ; Beginning of code segment

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

.org    $0046                                   ; End of Interrupt Vectors

;***************************************************************
;*      Program Initialization
;***************************************************************
INIT:
        ;Stack Pointer (VERY IMPORTANT!!!!)
        ldi mpr, high(RAMEND)
        out SPH, mpr
        ldi mpr, low(RAMEND)
        out SPL, mpr
        ;I/O Ports
        ldi mpr, 0b00000000
        out DDRD, mpr
        ldi mpr, 0b11111111
        out PORTD, mpr

        ldi             mpr, $FF                ; Set Port B Data Direction Register
        out             DDRB, mpr               ; for output
        ldi             mpr, $00                ; Initialize Port B Data Register
        out             PORTB, mpr              ; so all Port B outputs are low

        ;USART1
                ;Set baudrate at 2400bps

        ldi mpr, (1<<U2X1)
        sts UCSR1A, mpr
        ldi mpr, high(832)
        sts UBRR1H, mpr
        ldi mpr, low(832)
        sts UBRR1L, mpr
                ;Enable transmitter
        ldi mpr, (1<<TXEN1)
    sts UCSR1B, mpr
                ;Set frame format: 8 data bits, 2 stop bits
        ldi mpr, (0<<UMSEL1|1<<USBS1|1<<UCSZ11|1<<UCSZ10)
        sts UCSR1C, mpr

;***************************************************************
;*      Main Program
;***************************************************************
MAIN:   ;Main subroutine which polls PORTD for input
                in mpr, PIND    ;Reads input from PORTD and compares to each possible command and
calls the subroutine accordingly
                cpi mpr, 0b11111110
                brne Checkback
                rcall MoveForward
                jmp MAIN
CheckBack:
                cpi mpr, 0b11111101     ;All these subroutines do the same thing which is basically
call their respective routine when
                brne Checkleft          ;Their command is read
                rcall MoveBackward
                rjmp MAIN
Checkleft:
                cpi mpr, 0b11101111
                brne Checkright
                rcall Turnleft
                rjmp MAIN
Checkright:
                cpi mpr, 0b11011111
                brne Checkfreeze
                rcall Turnright
                rjmp MAIN
Checkfreeze:
                cpi mpr, 0b01111111
                brne Checkstop
                rcall Freeze
                rjmp MAIN
Checkstop:
                cpi mpr, 0b10111111
                brne MAIN
                rcall Stop
                rjmp MAIN
                ;cpi r19, 0b00000010
                ;breq MoveBackward
                ;cpi r19, 0b00000100
                ;breq Turnleft
                ;cpi r19, 0b00001000
                ;breq Turnright
                ;cpi r19, 0b00010000
                ;breq Stop
        ;TODO: ???
                ;rjmp   MAIN

;***********************************************************
;*      Functions and Subroutines
;***********************************************************
MoveForward:    ;Move Forward subroutine
        out PORTB, mpr       ;Outputs to LED to display button press
        ldi mpr, BotAddress           ;Loads Address to transmit
        sts UDR1, mpr                 ;Transmits
        ldi mpr, MovFwd                      ;Loads MovFwd
        sts UDR1, mpr                 ;Transmits
        ret
MoveBackward:
        out PORTB, mpr                ;The rest of the subroutines are basically the same as
MoveForward
        ldi mpr, BotAddress           ;except they load their own commands into UDR1
        sts UDR1, mpr
        ldi mpr, MovBck
        sts UDR1, mpr
        ret
Turnleft:
        out PORTB, mpr
        ldi mpr, BotAddress
        sts UDR1, mpr
        ldi mpr, TurnL
        sts UDR1, mpr
        ret
Turnright:
        out PORTB, mpr
        ldi mpr, BotAddress
        sts UDR1, mpr
        ldi mpr, TurnR
        sts UDR1, mpr
        ret
Stop:
        out PORTB, mpr
        ldi mpr, BotAddress
        sts UDR1, mpr
        ldi mpr, Halt
        sts UDR1, mpr
        ret
Freeze:
        out PORTB, mpr
        ldi mpr, BotAddress
        sts UDR1, mpr
        ldi mpr, Frze
        sts UDR1, mpr
        ret

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

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

;**********************************************************
;*     Additional Program Includes
;**********************************************************

;**********************************************************
;*
;*     AssemblerApplication8.asm
;*
;*     Receives input via USART1 and runs a robot accordingly. Can also issue freeze commands to
;*     to other robots nearby.
;*
;*     This is the RECEIVE skeleton file for Lab 8 of ECE 375
;*
;**********************************************************
;*
;*      Author: Zhenggan Zheng and Abhishek Raol
;*        Date: 3/11/2016
;*
;**********************************************************

.include "m128def.inc"               ; Include definition file

;**********************************************************
;*     Internal Register Definitions and Constants
;**********************************************************
.def   mpr = r16                            ; Multi-Purpose Register
.def   waitcnt = r20                ; Register to store count
.def   ilcnt = r18                          ; Register to store inner loop
.def   olcnt = r19                          ; Register to store outer loop

.equ   WTime = 100                          ; Wait time
.equ   WskrR = 0                            ; Right Whisker Input Bit
.equ   WskrL = 1                            ; Left Whisker Input Bit
.equ   EngEnR = 4                           ; Right Engine Enable Bit
.equ   EngEnL = 7                           ; Left Engine Enable Bit
.equ   EngDirR = 5                          ; Right Engine Direction Bit
.equ   EngDirL = 6                          ; Left Engine Direction Bit
.equ   BotAddress = $2A;(Enter your robot's address here (8 bits));7F

;////////////////////////////////////////////////////////////
;These macros are the values to make the TekBot Move.
;////////////////////////////////////////////////////////////
.equ   MovFwd =  (1<<EngDirR|1<<EngDirL)    ;0b01100000 Move Forward Action Code
.equ   MovBck =  $00                                       ;0b00000000 Move Backward Action Code
.equ   TurnR =   (1<<EngDirL)                      ;0b01000000 Turn Right Action Code
.equ   TurnL =   (1<<EngDirR)                      ;0b00100000 Turn Left Action Code
.equ   Halt =    (1<<EngEnR|1<<EngEnL)             ;0b10010000 Halt Action Code

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

;**********************************************************
;*     Interrupt Vectors
;**********************************************************
.org   $0000                                ; Beginning of IVs
               rjmp   INIT                  ; Reset interrupt
.org   $0002
               rcall HitRight               ; Interrupt to trigger HitRight routine
               reti
.org   $0004
               rcall HitLeft                ; Interrupt to trigger HitLeft Routine
               reti
.org   $003C
               rcall Receive                ;Interrupt that triggers when receives a command
from USART
               reti
```

```
.org    $0046                               ; End of Interrupt Vectors

;***********************************************************
;*      Program Initialization
;***********************************************************
INIT:
        ;Stack Pointer (VERY IMPORTANT!!!!)
        ldi mpr, high(RAMEND)
        out sph, mpr
        ldi mpr, low(RAMEND)
        out spl, mpr
        ;I/O Ports
        ldi mpr, $ff
        out DDRB, mpr
        ldi mpr, $00
        out PORTB, mpr

        ldi mpr, $00
        out DDRD, mpr

        ldi mpr, $00
        out DDRE, mpr
        ldi r24, 0
        ldi r26, 0
        ;USART1
        ;Set baudrate at 2400bps
        ldi mpr, (1<<U2X1)
        sts UCSR1A, mpr

        ldi mpr, high(832)
        sts UBRR1H, mpr
        ldi mpr, low(832)
        sts UBRR1L, mpr
        ;Enable receiver and enable receive interrupts
        ldi mpr, (1<<TXEN1|1<<RXEN1|1<<RXCIE1)
        sts UCSR1B, mpr
        ;Set frame format: 8 data bits, 2 stop bits
        ldi mpr, (0<<UMSEL1|1<<USBS1|1<<UCSZ11|1<<UCSZ10)
        sts UCSR1C, mpr
        ;External Interrupts
        ;Set the External Interrupt Mask
        ldi mpr, (1<<INT0)|(1<<INT1)
        out EIMSK, mpr
        ;Set the Interrupt Sense Control to falling edge detection
        ldi mpr, (1<<ISC01)|(0<<ISC00)|(1<<ISC11)|(0<<ISC10)
        sts EICRA, mpr

        ;Other
        sei
;***********************************************************
;*      Main Program
;***********************************************************
MAIN:
        ;TODO: ???

                rjmp    MAIN

;***********************************************************
;*      Functions and Subroutines
;***********************************************************
Receive:                              ;Subroutine to recieve USART
                lds r17, UDR1  ; Loads USART into register 17
                mov mpr, r17   ; Move it to MPR
                andi mpr, 0b10000000   ;AND it to mask out the other bits
                tst mpr                             ; If mpr is 0
                brne Testflag                ; Branch to Testflag if it is not
                cpi r17, BotAddress          ; Compare r17 to BotAddress
                breq Setflag                 ; If it is then set flag
                cpi r17, 0b01010101          ; Compare r17 to freeze command, if it is then go to
freeze
                breq Frozen
```

```
Testflag:              ;Subroutine to test if r27 is set
               tst r21 ;If r17 is set go to Run
               brne Run
               ret
Run:                   ;Subroutine that actually runs the robot
        cpi r17, 0b10110000   ;Runs through every posibility of commands received from the remote
        BREQ Moveforward      ;Then breaks to the subroutine that corresponds to that command
        cpi r17, 0b10000000
        BREQ Movebackward
        cpi r17, 0b10100000
        BREQ Turnright
        cpi    r17, 0b10010000
        BREQ Turnleft
        cpi r17, 0b11001000
        BREQ Stop
        cpi r17, 0b11111000
        BREQ ReceiveFreeze
        reti
Setflag:       ;Subroutine to set flag
        ldi r21, 1            ;Loads 1 into r21
        ret
Moveforward:           ;Subroutine to move forward
        ldi mpr, MovFwd       ; Loads MovFwd into r25 and outputs to PORTB
        ldi r25, MovFwd       ;Backs up MovFwd to r25 for later
        out PORTB, mpr
        clr r21               ;Clear the flag
        ret
Movebackward:          ;The other subroutines are identical to Moveforward except they have their
own
        ldi mpr, MovBck       ;corresponding commands
        ldi r25, MovBck
        out PORTB, mpr
        clr r21
        ret
Turnright:
        ldi mpr, TurnR
        out PORTB, mpr
        ldi r25, TurnR
        clr r21
        ret
Turnleft:
        ldi mpr, TurnL
        out PORTB, mpr
        ldi r25, TurnL
        clr r21
        ret
ReceiveFreeze: ;Receives freeze command and outputs the command to freeze other robots
        ldi mpr, 0b01010101
        sts UDR1, mpr ;Outputs freeze robot command to UDR1
        ret
Frozen:
        sbrs r21, 0    ;Freeze the robot when register 21 is not set and it receives a freeze
signal
        rcall Freeze
        ret
        ;rcall Freeze
        ;ret
        ;rcall Freeze
Freeze: ;The subrountine happens when the robot is frozen
        ldi mpr, 0b00000001   ;Loads LED display into mpr
        out PORTB, mpr        ;Outputs that display to PORTB
        ldi mpr, EIMSK        ;Back up EIMSK
        ldi r22, UCSR1B               ;Back up UCSR1B
        out EIMSK, r26        ;Outputs 0 to EIMSK to prevent interrupts
        sts UCSR1B, r26               ;Outputs 0 to USCR1B to prevent USART signals
        ldi waitcnt, WTime    ;Loads WTime into Waitcnt
        rcall Wait                    ;Wait for 5 seconds
        rcall Wait
        rcall Wait
        rcall Wait
        rcall Wait
        out EIMSK, mpr ;Restore EIMSK
        sts UCSR1B, r22         ;Restore UCSR1B
        inc r24                 ;increment counter to see how many times robot has been frozen
        cpi r24, 3              ;If counter reaches 3, go to Dead
        breq Dead
        out PORTB, r25 ;Output original content of r25 before it was frozen

        ret
Dead:
        out EIMSK, r26 ;Write 0s to EIMSK
        sts UCSR1B, r26         ;Write 0s to UCSR1B
        ret
Stop:   ;Subroutine for stop
        ldi mpr, Halt  ;Loads halt into mpr
        out PORTB, mpr ;Output to PORTB
        clr r21                 ;Clears the flag

HitLeft:
                push    mpr                     ; Save mpr register
                push    waitcnt         ; Save wait register
                in              mpr, SREG       ; Save program state
                push    mpr                     ;

                ; Move Backwards for a second
                ldi             mpr, MovBck     ; Load Move Backward command
                out             PORTB, mpr      ; Send command to port
                ldi             waitcnt, WTime  ; Wait for 1 second
                rcall   Wait                    ; Call wait function

                ; Turn right for a second
                ldi             mpr, TurnR      ; Load Turn Left Command
                out             PORTB, mpr      ; Send command to port
                ldi             waitcnt, WTime  ; Wait for 1 second
                rcall   Wait                    ; Call wait function

                ; Move Forward again
                ldi             mpr, MovFwd     ; Load Move Forward command
                out             PORTB, mpr      ; Send command to port

                pop             mpr             ; Restore program state
                out             SREG, mpr       ;
                pop             waitcnt         ; Restore wait register
                pop             mpr             ; Restore mpr
                ret                             ; Return from subroutine

HitRight:
                push    mpr                     ; Save mpr register
                push    waitcnt                 ; Save wait register
                in              mpr, SREG       ; Save program state
                push    mpr                     ;

                ; Move Backwards for a second
                ldi             mpr, MovBck     ; Load Move Backward command
                out             PORTB, mpr      ; Send command to port
                ldi             waitcnt, WTime  ; Wait for 1 second
                rcall   Wait                    ; Call wait function

                ; Turn left for a second
                ldi             mpr, TurnL      ; Load Turn Left Command
                out             PORTB, mpr      ; Send command to port
                ldi             waitcnt, WTime  ; Wait for 1 second
                rcall   Wait                    ; Call wait function

                ; Move Forward again
                ldi             mpr, MovFwd     ; Load Move Forward command
                out             PORTB, mpr      ; Send command to port

                pop             mpr             ; Restore program state
                out             SREG, mpr       ;
```

```
               pop            waitcnt        ; Restore wait register
               pop            mpr            ; Restore mpr
               ret                           ; Return from subroutine

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

;************************************************************
;*     Additional Program Includes
;************************************************************
```