ECE 375 LAB 8

Remotely Operated Vehicle

Lab Time: Tuesday 6-7:50

Aditya Kothari

Mac

INTRODUCTION

The Goal of this lab was to familiarize ourselves with USART (Universal Synchronous/Asynchronous Receiver/Transmitter) facility on the ATmega128. We used USART and its ability to send and receive bytes of data, to build a remote-controlled vehicle that could play freeze tag with other remotely operated vehicles.

PROGRAM OVERVIEW

The program is divided in two parts: The Transmitter part for the remote control and the receiver part for the remote-controlled vehicle. The Transmitter (remote control) implements polling to detect button presses and send an appropriate action code to the Receiver (Remote Controlled Car). The Receiver than performs an appropriate action depending on the incoming signal.

TRANSMITTER:

INITIALIZATION ROUTINE

Initialize the stack pointer. Set Port D for input and enable pullup on first tow and last four buttons (0,1,4,5,6,7). In addition, we set the 4th button (3) to output since it is the transmitter. Configure USART, set Baud rate to 2400 bps by writing $01A0 to UBRR, enable transmitter and set frame format to 8 data bits and 2 stop bits. Lastly we set the global interrupt flag in SREG

MAIN ROUTINE

W use Polling to detect the button presses. Since we have initialized first tow and last four to pull up, by default input from PIND is 1 i.e 0b11110011. If a button is pressed, corresponding bit becomes 0 and the appropriate subroutine calls is made that sends the action code. For instance, if first button pressed 0th bit becomes 0, SBRS doesn't skip the next instruction rcall sendfwd (sends MovFwd action code).

SENDFWD

This subroutine sends robot address as the first 8 bits and The MovFwd action code to the receiver. This is done by making a call to send robot subroutine that sends first byte as the robot address. Then we use a loop to check for UDRE1 flag in UCSR1A is set. UDRE1 flag set signifies the buffer is empty and is ready to be written. Once the UDRE1 flag is set write action code to UDR1 that sends the action code to the receiver. Finally, we make a call wait function (~30ms) to handle debouncing and make busy wait loop until the data is sent.

SENDBCK

This subroutine sends robot address as the first 8 bits and The MovBck action code to the receiver. This is done by making a call to send robot subroutine that sends first byte as the robot address. Then we use a loop to check for UDRE1 flag in UCSR1A is set. UDRE1 flag set signifies the buffer is empty and is ready to be written. Once the UDRE1 flag is set, we write action code to UDR1 that sends the action code to the receiver. Finally, we make a call wait function (~30ms) to handle debouncing and make busy wait loop until the data is sent.

SendRight

This subroutine sends robot address as the first 8 bits and The TurnR action code to the receiver. This is done by making a call to send robot subroutine that sends first byte as the robot address. Then we use a loop to check for UDRE1 flag in UCSR1A is set. UDRE1 flag set signifies the buffer is empty and is ready to be written. Once the UDRE1 flag is set, we write action code to UDR1 that sends the action code to the receiver. Finally, we make a call wait function (~30ms) to handle debouncing and make busy wait loop until the data is sent.

SendLeft

This subroutine sends robot address as the first 8 bits and The TurnL action code to the receiver. This is done by making a call to send robot subroutine that sends first byte as the robot address. Then we use a loop to check for UDRE1 flag in UCSR1A is set. UDRE1 flag set signifies the buffer is empty and is ready to be written. Once the UDRE1 flag is set, we write action code to UDR1 that sends the action code to the receiver. Finally, we make a call wait function (~30ms) to handle debouncing and make busy wait loop until the data is sent.

SendHalt

This subroutine sends robot address as the first 8 bits and The Halt action code to the receiver. This is done by making a call to send robot subroutine that sends first byte as the robot address. Then we use a loop to check for UDRE1 flag in UCSR1A is set. UDRE1 flag set signifies the buffer is empty and is ready to be written. Once the UDRE1 flag is set, we write action code to UDR1 that sends the action code to the receiver. Finally, we make a call wait function (~30ms) to handle debouncing and make busy wait loop until the data is sent.

SendFreeze

This subroutine sends robot address as the first 8 bits and The Freeze action code to the receiver. This is done by making a call to send robot subroutine that sends first byte as the robot address. Then we use a loop to check for UDRE1 flag in UCSR1A is set. UDRE1 flag set signifies the buffer is empty and is ready to be written. Once the UDRE1 flag is set, we write action code to UDR1 that sends the action code to the receiver. Finally, we make a call wait function (~30ms) to handle debouncing and make a busy wait loop until the data is sent.

SendRobot

This subroutine is called in all other subroutines that send the action code. This subroutine sends the robot address. This is done by checking for UDRE1 flag in UCSR1A is set. UDRE1 flag set signifies the buffer is empty and is ready to be written. Once the UDRE1 flag is set, we write robot address to UDR1 that sends the action code to the receiver.

Wait

A wait loop that initializes wait for the specific amount of time in 30ms intervals.

RECIEVER:

INTERRUPT VECTORS

Defined interrupts for left and right whiskers, which make call to HitLeft and HitRight Subroutine if interrupt detected. In addition, also defined USART receive interrupt that makes calls to Checkflag subroutine.

INITIALIZATION ROUTINE

Initialize the stack pointer. Set Port D for input and enable pullup on first tow buttons (0,1). Set Port B for output and turn on 5th and 6th LED by writing to PORTB. Next we Configure USART: set Baud rate to 2400 bps by writing $01A0 to UBRR, enable transmitter, receiver, receiver interrupt and set frame format to 8 data bits and 2 stop bits. Then enable external Interrupts, set sense control to falling edge and enable the two interrupts by writing to EIMSK. Lastly we set the global interrupt flag in SREG.

MAIN

Jump back to main, infinite loop.

CHECKFLAG

This subroutine is called by the USART Tx complete interrupt vector. It Makes sense out of the received byte from the transmitter. Incoming byte is read from UDR1 and is compared to FreezeSignal action code. If the received byte is FreezeSignal it branches to freezeRobot Subroutine. Then it checks for flag (set by setRobot subroutine). If flag is not set (flag = 0), branches to setRobot subroutine and if flag set it branches to pickAction Subroutine.

PICKACTION

This subroutine compares the received byte with action code. If the received byte and action code match, it branches to appropriate subroutine.

FREEZEACTION

This subroutine is called from pickAction subroutine, if the received byte matches with action code for Freeze. This subroutine disables the receiver and receiver interrupt in UCSR1B and sends out freeze signal to all other receivers. Once the freeze signal is sent out receiver and receiver interrupt are enabled once again.

FREEZEROBOT

This subroutine disables the global interrupt flag and halts for 5 seconds. After 5 seconds the robot resumes functioning. R20 is used to count number of times FreezeRobot subroutine is called. After three freeze signal the robot freezes permanently.

MOVFWDCOM

This subroutine is called from pickAction subroutine, if the received byte matches with action code – MovFwd. This makes the bot move forward by writing out MovFwd1 command to PORTB.

MovBckCom

This subroutine is called from pickAction subroutine, if the received byte matches with action code – MovBck. This makes the bot move backwards by writing out MovBck1 command to PORTB.

TurnLCom

This subroutine is called from pickAction subroutine, if the received byte matches with action code – TurnL. This makes the bot turn left by writing out TurnL1 command to PORTB.

TurnRCom

This subroutine is called from pickAction subroutine, if the received byte matches with action code – TurnR. This makes the bot turn right by writing out TurnR1 command to PORTB.

HaltCom

This subroutine is called from pickAction subroutine, if the received byte matches with action code – Halt. This makes the bot halt by writing out Halt1 command to PORTB

HitRight

The HitRight routine causes the TekBot to move backwards for a 1 second by writing Move Backwards command to PORTB, followed by call to the Wait routine. The Turn Left command is written to PORTB followed by another call to the Wait routine. Finally, EIFR is cleared so it doesn't lock the interrupt signal and therefore doesn't create a que of interrupts

HitLeft

The HitLeft routine causes the TekBot to move backwards for a 1 second by writing Move Backwards command to PORTB, followed by call to the Wait routine. The Turn right command is written to PORTB followed by another call to the Wait routine. Finally, EIFR is cleared so it doesn't lock the interrupt signal and therefore doesn't create a que of interrupts.

Wait

A wait loop that initializes wait for the specific amount of time in 30ms intervals.

Conclusion

In this lab, I learned how to configure USART and handle interrupts from multiple sources to build a remote-controlled car. This lab was by far the most challenging and fun lab and has given me a firm grip over the AVR assembly.

TRANSMITTER SOURCE CODE:

```
;***********************************************************
;*
;*      Enter Name of file here
;*
;*      Enter the description of the program here
;*
;*      This is the TRANSMIT skeleton file for Lab 8 of ECE 375
;*
;***********************************************************
;*
;*       Author: Enter your name
;*         Date: Enter Date
;*
;***********************************************************

.include "m128def.inc"               ; Include definition file

;***********************************************************
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                           ; Multi-Purpose Register
.def    mpr2 = r17
.def    mpr3 = r18
.def    mpr1 = r19


.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     Freeze = $F8   ;0b11111000 Freeze Action Code
.equ    Robot = $2A           ;Robot address
.equ    fwd = 0
.equ    bck = 1
.equ    right = 4
.equ    left = 5
.equ    haltnum = 6
.equ    freezenum = 7
;***********************************************************
;*      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:
        ;Initialize Stack Pointer
        LDI mpr, High(RAMEND)
        OUT SPH, mpr
        LDI mpr, Low(RAMEND)
        OUT SPL, mpr

        ;Intilize Port D
        LDI mpr, $08 ; DDRD to input and
        OUT DDRD, mpr; 1 set into the 4th bit

        ldi mpr, $F3 ; pull up on first two buttons
        out PORTD, mpr
        ;USART1
        LDI mpr, $01   ;Set baudrate at 2400bps
        STS UBRR1H, mpr
        LDI mpr, $A0
        STS UBRR1L, mpr

        LDI mpr, (1<<TXEN1) ;Enable transmitter
        STS UCSR1B, mpr

        LDI mpr, ((1<<USBS1)|(1<<UCSZ11)|(1<<UCSZ10)) ;Set frame format: 8 data bits, 2 stop bits
        STS UCSR1C, mpr

        sei ;Set Global Interrupt
;************************************************************
;*      Main Program
;************************************************************
MAIN:
                in mpr, PIND ;Input from PortD

                SBRS mpr, 0     ;if button pressed (0th bit = 0)
                rcall sendFwd   ;send the following action code

                SBRS mpr, 1
                rcall sendBck


                SBRS mpr, 4
                rcall sendRight


                SBRS mpr, 5
                rcall sendLeft

                SBRS mpr, 6
                rcall sendHalt

                SBRS mpr, 7
                rcall sendFreeze


                rjmp    MAIN
;************************************************************
;*      Functions and Subroutines
;************************************************************

;------------------------------------------------------------
; Func: sendFwd
; Desc: Sends Robot address (first 8 bits) and MovFwd action code
;              to the reciever.
;------------------------------------------------------------
sendFwd:
        rcall sendRobot ;Calls sendRobot function, that sends the robot address first 8 bits
        ldi mpr1, MovFwd

        fwdLoop2:             ;check if UDRE flag is set
        LDS mpr2, UCSR1A
        SBRS mpr2, 5   ;if UDRE1 is set skip next insturction. (UDRE set, buffer empty and ready
to be written)
```

```
        rjmp fwdLoop2  ;if not set check again

        sts UDR1, mpr1 ;send the action code to the reciever
        rcall wait             ;handle debouncing

        ret
;-----------------------------------------------------------
; Func: sendBck
; Desc: Sends Robot address (first 8 bits) and MovBck action code
;              to the reciever.
;-----------------------------------------------------------
sendBck:
        rcall sendRobot
        ldi mpr1, MovBck

        bckLoop2:
        LDS mpr2, UCSR1A
        SBRS mpr2, 5
        rjmp bckLoop2

        sts UDR1, mpr1
        rcall wait

        ret
;-----------------------------------------------------------
; Func: sendRight
; Desc: Sends Robot address (first 8 bits) and TurnR action code
;              to the reciever.
;-----------------------------------------------------------
sendRight:
        rcall sendRobot
        ldi mpr1, TurnR

        rLoop2:
        LDS mpr2, UCSR1A
        SBRS mpr2, 5
        rjmp rLoop2

        sts UDR1, mpr1
        rcall wait

        ret
;-----------------------------------------------------------
; Func: sendLeft
; Desc: Sends Robot address (first 8 bits) and TurnL action code
;              to the reciever.
;-----------------------------------------------------------
sendLeft:
        rcall sendRobot
        ldi mpr1, TurnL

        lLoop2:
        LDS mpr2, UCSR1A
        SBRS mpr2, 5
        rjmp lLoop2

        sts UDR1, mpr1
        rcall wait

        ret
;-----------------------------------------------------------
; Func: sendHalt
; Desc: Sends Robot address (first 8 bits) and Halt action code
;              to the reciever.
;-----------------------------------------------------------
sendHalt:
        rcall sendRobot
        ldi mpr1, Halt

        hLoop2:
        LDS mpr2, UCSR1A
        SBRS mpr2, 5
        rjmp hLoop2

        sts UDR1, mpr1
        rcall wait

        ret
;-----------------------------------------------------------
; Func: sendFreeze
; Desc: Sends Robot address (first 8 bits) and MovBck action code
;               to the reciever.
;-----------------------------------------------------------
sendFreeze:
        rcall sendRobot
        ldi mpr1, Freeze

        fLoop2:
        LDS mpr2, UCSR1A
        SBRS mpr2, 5
        rjmp fLoop2

        sts UDR1, mpr1
        rcall wait

        ret
;-----------------------------------------------------------
; Func: sendRobot
; Desc: Sends Robot address. Called in other subroutines that
;               are required to send robot address before the action code.
;-----------------------------------------------------------
sendRobot:
robLoop:
        LDS mpr2, UCSR1A
        SBRS mpr2, 5
        rjmp robLoop

        ldi mpr1, Robot
        sts UDR1,mpr1

        ret

;----------------------------------------------------------------
; Func: Wait
; Desc: A wait loop that is 16 + 159975*waitcnt cycles or roughly
;               waitcnt*10ms.  Just initialize wait for the specific amount
;               of time in 10ms intervals. Here is the general eqaution
;               for the number of clock cycles in the wait loop:
;                       ((3 * ilcnt + 3) * olcnt + 3) * waitcnt + 13 + call
;----------------------------------------------------------------

Wait:

                ldi             r24, 30
Loop:   ldi             r22, 224                ; load olcnt register
OLoop:  ldi             r23, 237                ; load ilcnt register
ILoop:  dec             r23                             ; decrement ilcnt
                brne    ILoop                   ; Continue Inner Loop
                dec             r22                             ; decrement olcnt
                brne    OLoop                   ; Continue Outer Loop
                dec             r24                             ; Decrement wait
                brne    Loop                    ; Continue Wait loop


                ret                             ; Return from subroutine
;***********************************************************
;*      Stored Program Data
;***********************************************************

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

RECEIVER SOURCE CODE:

```
;***********************************************************
;*
;*      Enter Name of file here
;*
;*      Enter the description of the program here
;*
;*      This is the RECEIVE skeleton file for Lab 8 of ECE 375
;*
;***********************************************************
;*
;*       Author: Enter your name
;*         Date: Enter Date
;*
;***********************************************************

.include "m128def.inc"               ; Include definition file

;***********************************************************
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                            ; Multi-Purpose Register
.def    mpr2 = r17
.def    flag = r18
.def    mpr1 = r19
.def    waitcnt = r20                ; Wait Loop Counter
.def    ilcnt = r21                          ; Inner Loop Counter
.def    olcnt = r22                          ; Outer Loop Counter
.def    freezeCount = r15
.equ    WTime = 100                          ; Time to wait in wait loop


.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    MovFwd1 = (1<<EngDirR|1<<EngDirL)     ;Move Forward Command
.equ    MovBck1 = $00                                ;Move Backward Command
.equ    TurnR1 = (1<<EngDirL)                       ;Turn Right Command
.equ    TurnL1 = (1<<EngDirR)                       ;Turn Left Command
.equ    Halt1 = (1<<EngEnR|1<<EngEnL)         ;Halt Command

.equ     Freeze = $F8 ;Freeze Action code
.equ    Robot = $2A    ;Robot address
.equ    fwd = 0
.equ    bck = 1
.equ    right = 4
.equ    left = 5
.equ    haltnum = 6
.equ    freezenum = 7
.equ    freezeSignal = $55     ;Frezee signal from the reciever to other recievers
;***********************************************************
;*      Start of Code Segment
;***********************************************************
```

```
.cseg                                                   ; Beginning of code segment

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

.org    $0002                                   ; External Interrupt: INT0
                rcall Hitright                  ; Calls HitRight Subroutine
                reti

.org    $0004                                   ; External Interrupt: INT1
                rcall Hitleft                   ; Calls Hitleft Subroutine
                reti

.org    $003C                                   ; USART recieve Complete Interrupt
                rcall Checkflag                         ; Calls Checkflag subroutine
                reti


.org    $0046                                   ; End of Interrupt Vectors

;****************************************************************
;*      Program Initialization
;****************************************************************
INIT:
        ;Initialize Stack Pointer
        LDI mpr, High(RAMEND)
        OUT SPH, mpr
        LDI mpr, Low(RAMEND)
        OUT SPL, mpr

        ;Intilize Ports
        ;Intilize Port D
        LDI mpr, $00 ; DDRD to input
        OUT DDRD, mpr; Port D - 0,1 buttons and 2 reciever for input

        ldi mpr, $03 ; pull up on first two buttons - for BumpBot
        out PORTD, mpr

        ;Intilize Port B
        ldi mpr, $FF; DDRB to Output
        OUT DDRB, mpr
        LDI MPR, (1<<EngDirR)|(1<<EngDirL);Turn on 5th and 6th LED
        OUT PORTB, MPR

        ;USART1
        LDI mpr, $01    ; set baud rate 2400 bps
        STS UBRR1H, mpr
        LDI mpr, $A0
        STS UBRR1L, mpr

        LDI mpr, (1<<TXEN1|1<<RXEN1 |1<<RXCIE1); Enable Transmitter, reciever and reciever
interrupt
        STS UCSR1B, mpr

        LDI mpr, ((1<<USBS1)|(1<<UCSZ11)|(1<<UCSZ10)) ;Set frame format: 8 data bits, 2 stop
bits, Async
        STS UCSR1C, mpr

        ;Enable ExternalInterrupts
        LDI mpr, (1<<ISC01)|(0<<ISC00)|(1<<ISC11)|(0<<ISC10); Set the Interrupt Sense Control to
falling edge (0th and 1st buttons)
        STS EICRA, mpr
        LDI mpr, (1<<INT0)|(1<<INT1); Enable two interrupts
        OUT EIMSK, mpr

        ldi R20, 0;Load 0 in register 20 (Used as count for freeze Robot subroutine)

        sei ;Set Global Interrupt flag in SREG
```

```
;***********************************************************
;*      Main Program
;***********************************************************
MAIN:
               rjmp    MAIN
;***********************************************************
;*      Functions and Subroutines
;***********************************************************

;-----------------------------------------------------------------
; Sub: Checkflag
; Desc: Makes sense out of the received byte from the transmitter.
;              if the received signal is FreezeSignal it branches to freezeRobot Subroutine.
;              if flag not set (flag = 0), branches to setRobot subroutine and if flag set
;              it branches to pickAction Subroutine.
;-----------------------------------------------------------------
Checkflag:
               LDS MPR, UDR1 ;Loads recived byte from UDR1 to MPR
               CPI MPR, FreezeSignal  ;Checks if the received byte is a Freeze Signal
               BREQ freezeRobot ;branch to freezeRobot subroutine

               CPI flag, 0            ;check if flag set to 0
               BREQ setRobot  ;branch to setRobot SubRoutine
               CPI flag, 1            ;check if flag set to 1
               BREQ pickAction ;branch to pickAction

;-----------------------------------------------------------------
; Sub:  setRobot
; Desc: Checks if the received byte of robot address, matches ours. if they match
;              set flag to 1, allowing the transmitter's next byte to pickAction.
;-----------------------------------------------------------------
setRobot:
               CPI MPR, Robot ; check for robot address
               BREQ setFlag   ;Set register bit for correct robot
               ret ;return to Rcall Checkflag

SetFlag:
               LDI flag, 1; Load flag with 1
               ret
;-----------------------------------------------------------------
; Sub:  pickAction
; Desc: Compares recieved byte with action code. If they match perform the appropriate action
;              by branching to appropriate subroutine
;-----------------------------------------------------------------
pickAction:
               CPI MPR, MovFwd ; check if the recieved action code is to move forward?
               breq MovFwdCom

               CPI MPR, MovBck ; check if the recieved action code is to move Back?
               breq MovBckCom

               CPI MPR, TurnR ; check if the recieved action code is to turn right?
               breq TurnRCom

               CPI MPR, TurnL ; check if the recieved action code is to turn left?
               breq TurnLCom

               CPI MPR, Halt ; check if the recieved action code is to halt?
               breq HaltCom

               CPI MPR, Freeze ;  check if the recieved action code is to Freeze?
               breq freezeAction

               ret
;-----------------------------------------------------------------
; Sub:  freezeAction
; Desc: if frezee action code is recieved the receiver sends out frezee signal to all nerarby
;              recievers.
;-----------------------------------------------------------------
freezeAction:
               ldi mpr2, freezeSignal;Load Freeze signal to mpr2
               LDI mpr, (1<<TXEN1|0<<RXEN1 |0<<RXCIE1); disable  reciever and reciever interrupt
               STS UCSR1B, mpr

freezeLoop2:   ;check if UDRE flag is set
               LDS mpr1, UCSR1A ;load UCSR1A to mpr1
               SBRS mpr1, 5    ;if UDRE1 is set skip next insturction. (UDRE set, buffer empty
and ready to be written)
               rjmp freezeLoop2 ;if not set check again

               sts UDR1, mpr2 ;send the freeze signal to other nearby recievers
               rcall wait

               LDI mpr, (1<<TXEN1|1<<RXEN1 |1<<RXCIE1); Enable reciever and reciever interrupt
               STS UCSR1B, mpr

               ret
;-----------------------------------------------------------------
; Sub:  freezeRobot
; Desc: Disables the interrupt flag in SREG so it cannot recieve any more interrupts and halts
the bot for 5 seconds.
;              After 3 freezes the robot freezes forever.
;-----------------------------------------------------------------
freezeRobot:
               CLI            ;Clear global Interrupt flag
               IN r25, PORTB  ;saves the current behavior bot before freezing

               LDI mpr1, Halt1 ;load mpr1 with halt command
               LDI MPR, 10 ;load mpr with 10, used in waitloop
               OUT PORTB, mpr1 ;write halt command to port B

               inc r20 ;counts for number of freeze robot signal
               cpi r20, 3 ;checks if robot has been frozen for three times
               breq permFreeze ;if robot has been frozen for three times, branch to permFreeze
subroutine

waitLoop: ;frezees robot for 5 sconds
               rcall wait     ;calls wait function
               dec MPR ;decreases wait count (50ms*10 = 5 sec)
               cpi MPR, 0 ;check for MPR until it is 0
               brne waitLoop ;branch to  waitloop until count is not equal to 0

               OUT PORTB, r25 ;returns to the same behavior bot was performing before freezing

               LDI mpr1, $FF
               OUT EIFR, mpr1;Clears EIFR

               SEI     ;Set global Interrupt Flag

               ret
permFreeze: ;Traps robot in infinite loop that freeze's it forever
               rjmp permFreeze
;-----------------------------------------------------------------
; Sub: MovFwdCom
; Desc: Makes the bot move forward.
;-----------------------------------------------------------------
MovFwdCom:
                       LDI mpr2, MovFwd1 ;loads MoveFwd command to mpr2
                       out PORTB, mpr2 ;writes it out to PORTB

                       LDI flag, 0


                       ret

;-----------------------------------------------------------------
; Sub:  MovBckcom
; Desc: Makes the bot move backwards.
;-----------------------------------------------------------------

MovBckCom:
```

```
                      LDI mpr2, MovBck1 ;loads MovBck command to mpr2
                      out PORTB, mpr2 ;writes it out to PORTB

                      LDI flag, 0

                      ret

;----------------------------------------------------------------
; Sub:  TurnLCom
; Desc: Makes the bot Turn Left.
;----------------------------------------------------------------

TurnLCom:
                      LDI mpr2, TurnL1 ;loads TurnL command to mpr2
                      out PORTB, mpr2 ;writes it out to PORTB

                      LDI flag, 0

                      ret

;----------------------------------------------------------------
; Sub:  TurnRCom
; Desc:  Makes the bot Turn Right.
;----------------------------------------------------------------

TurnRCom:
                      LDI mpr2, TurnR1 ;loads TurnR command to mpr2
                      out PORTB, mpr2 ;writes it out to PORTB

                      LDI flag, 0

                      ret


;----------------------------------------------------------------
; Sub:  HaltCom
; Desc: Makes the bot halt.
;----------------------------------------------------------------

HaltCom:
                      LDI mpr2, Halt1 ;loads Halt command to mpr2
                      out PORTB, mpr2 ;writes it out to PORTB

                      LDI flag, 0

                      ret


;----------------------------------------------------------------
; Sub:  HitRight
; Desc: Handles functionality of the TekBot when the right whisker
;              is triggered.
;----------------------------------------------------------------
HitRight:

               push   mpr                    ; Save mpr register

               in             mpr, SREG      ; Save program state
               push   mpr                    ;

               ; Move Backwards for a second
               ldi            mpr, MovBck1   ; 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, TurnL1    ; 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, MovFwd1   ; Load Move Forward command
               out            PORTB, mpr     ; Send command to port

               pop            mpr                ; Restore program state
               out            SREG, mpr      ;

               pop            mpr                ; Restore mpr
               LDI            mpr, 0x03
               OUT            EIFR, mpr   ; clearing EIFR

               ret                           ; Return from subroutine

;----------------------------------------------------------------
; Sub:  HitLeft
; Desc: Handles functionality of the TekBot when the left whisker
;              is triggered.
;----------------------------------------------------------------
HitLeft:
               push    mpr                   ; Save mpr register
               in             mpr, SREG      ; Save program state
               push    mpr                   ;

               ; Move Backwards for a second
               ldi            mpr, MovBck1   ; Load Move Backward command
               out            PORTB, mpr     ; Send command to port
               rcall   Wait                  ; Call wait function

               ; Turn right for a second
               ldi            mpr, TurnR1    ; Load Turn Left Command
               out            PORTB, mpr     ; Send command to port
               rcall   Wait                  ; Call wait function

               ; Move Forward again
               ldi            mpr, MovFwd1   ; Load Move Forward command
               out            PORTB, mpr     ; Send command to port

               pop            mpr                ; Restore program state
               out            SREG, mpr      ;

               pop            mpr                ; Restore mpr
               LDI            mpr, 0x03
               OUT            EIFR, mpr   ; clearing EIFR
               ret                               ; Return from subroutine

;----------------------------------------------------------------
; Sub:  Wait
; Desc: A wait loop that is 16 + 159975*waitcnt cycles or roughly
;              waitcnt*10ms.  Just initialize wait for the specific amount
;              of time in 10ms intervals. Here is the general eqaution
;              for the number of clock cycles in the wait loop:
;                      ((3 * ilcnt + 3) * olcnt + 3) * waitcnt + 13 + call
;----------------------------------------------------------------

Wait:

               ldi            r24, 50
Loop:   ldi            r22, 224              ; load olcnt register
OLoop:  ldi            r23, 237              ; load ilcnt register
ILoop:  dec            r23                          ; decrement ilcnt
               brne    ILoop                 ; Continue Inner Loop
               dec            r22                           ; decrement olcnt
               brne    OLoop                 ; Continue Outer Loop
               dec            r24                           ; Decrement wait
               brne    Loop                  ; Continue Wait loop


               ret                           ; Return from subroutine
;***********************************************************
;*      Stored Program Data
```

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

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