ECE 375 LAB 8

Remotely Operated Vehicle (USART)

Lab Time: Thursday 12-2

Mason Obery

Benjamin Leonard

INTRODUCTION

The purpose of this lab is to learn the basics of USART transmission by using the IR light/sensor. The lab uses two TekBot boards, one which transmits a signal and another which receives the signal. When a button is pressed, the transmitter board selects an opcode to send to the board, and transmits the code along with the receiving bots individual address. The receiver functions as a normal BumpBot, but will set its behavior to that of the most recently issued command from the transmitter. This lab will demonstrate and understanding of how to efficiently make use of all the I/O capabilities of the board, including buttons, interrupts, IR USART, timer/counters, and LEDs.

PROGRAM OVERVIEW

There are two programs that were created for this lab: the Receiver code (often abbreviated as RX) and the Transmitter code (often referred to as TX). The transmitter determines if one of the buttons is pressed, and if they have been pressed, to send the bot ID via USART. The program waits until the full signal has been transmitted, and then loads the opcode for the movement indicated. The receiver waits until an receive interrupt has been triggered, which then sets the output LEDs to the correct settings based on the message received. If the robot receives the Send Freeze signal, the TekBot switches the settings of the USART port to Transmit mode, and sends the signal 01010101. If the 01010101 signal is received, the TekBot stops and does not respond to external commands for 5 seconds. The TekBot can survive 3 Freezes until it ceases to function entirely.

For the TX remote, there are no interrupts. The body of the program consists of an initialization routine, followed by a main program which polls PORTD to check if any buttons have been pressed. If buttons have been pressed, it jumps to a function that transmits the data, and uses polling to determine when the transmitter has sent the first bit and is prepared to send another message.

For the RX remote, interrupts are used extensively. The program retains the BumpBot functionality from Lab 2, which means that when External Interupt1 and External Interrupt0 are triggered, the robot will back up and turn away from the object it bumped into. When the robot receives a signal from the transmitter with the corresponding Bot ID, it will begin executing the command and await another instruction.

TX INITIALIZATION ROUTINE

First, the stack pointer is set up so that function calls can be made. PORTD is configured to input on all pins except PIND4, and the pull-up resistor is enabled for all input bits. PORTB is set up to allow for LED output. These LEDs are not part of the assignment, but help to display the opcode that was most recently transmitted to the receiver. For setting up the USART communication, the value 832 is set into the UBBR1 Registers, the double data rate is enabled, the transmitter is enabled, asynchronous mode is enabled, parity is disabled, n-bit data transfer is set to n=8, and 2 stop bits is selected.

RX INITIALIZATION ROUTINE

For the initialization of the RX TekBot, first the stack pointer is set up. PIND3 is set as an output, and the pullup resistor is enabled. PORTB (LEDs) is set up as an output, and the initial output of the PORTB is set to $00. For setting up the USART communication, the value 832 is set into the UBBR1 Registers, the double data rate is enabled, the receiver mode is enabled, asynchronous mode is enabled, parity is disabled, n-bit data transfer is set to n=8, and 2 stop bits is selected. The number of health points (HP) is initialized to 3. The external interrupts are set to falling edge, the robot is set to MoveForward, and the interrupt is enabled.

TX Main Routine

The TX remote program is relatively straightforward. The program polls for button inputs every 100ms. If any of the buttons have been pressed, their respective buttons presses are converted into the opcode pattern. The USART_Transmit function is called twice, first to transmit the specified BotID, then to transmit the opcode. The USART_Transmit function is described in detail below. The program then jumps back to the top of the main function.

TX Subroutines

1. USART_Transmit Routine

This function loops until the USART data register Is empty, and then loads the next piece of data into the transmit data buffer. This is a polling implementation of USART communication. While an interrupt-based implementation would be more resource-efficient, the polling system is sufficient for the demonstration of concept.

2. Wait Routine

This subroutine is directly taken from the BumpBot code. This function cycles repeatedly to wait roughly waitcnt * 10ms. This is used to prevent button debouncing, which could result in multiple freeze signals sent simultaneously.

RX Main Routine

The TX remote program is relatively straightforward. The program polls for button inputs every 100ms. If any of the buttons have been pressed, their respective buttons presses are converted into the opcode pattern. The USART_Transmit function is called twice, first to transmit the specified BotID, then to transmit the opcode. The USART_Transmit function is described in detail below. Additionally, external interrupts 1 and 0 are enabled so that the TekBot will turn if it bumps into an object.

RX Subroutines

1. rCommand Routine

This subroutine is triggered when a message is received by the USART module. When the USART module receives data, it stores the data and clears the buffer. The program checks if the data was equal to the freeze command. If the TekBot has been frozen, it jumps to a portion of the routine that will decrement the total number of health points, wait for 5 seconds, and stops the robot permanently if the robot is out of health points.

If the robot did not receive a freeze command, it checks if it received the BotID. If is equal to the bot ID, it sets register ereg(r17) to $FF and returns to main. If not, it checks first if ereg=$FF, then checks if a valid command was transmitted. For all the standard functions (Forward, Left, Right, Stop, Back), the LEDs are updated to the correct configuration. If the 'Issue Freeze' opcode was received, the Tekbot changes the configuration of the USART module to a transmitter, identical to that of the TX program. The freeze command uses the USART_Transmit1 function (polling) to set the freeze command '01010101.' After the transmission has been sent, the USART module is restored to its initial configuration.

2. HitLeft Routine

The function is taken directly from the standard BumpBot code. If the External interrupt 1 is triggered (S2), the robot backs up for 1 second, drives forward for one second with the left motor enabled, and then continues forward.

3. HitRightRoutine

The function is taken directly from the standard BumpBot code. If the External interrupt 0 is trigge red (S1), the robot backs up for 1 second, drives forward for one second with the left motor enabled, and then continues forward.

3. Wait Routine

This subroutine is directly taken from the BumpBot code. This function cycles repeatedly to wait roughly waitcnt * 10ms. This is used to prevent button debouncing, which could result in multiple freeze signals sent simultaneously.

4. USART_Transmit1 Routine

This function loops until the USART data register Is empty (UDRE1), loads the next piece of data into the transmit data buffer, and then loops until the data buffer is empty again. This is a polling implementation of USART communication. While an interrupt-based implementation would be more resource-efficient, the polling system is sufficient.

DIFFICULTIES

One of the main difficulties we had was correctly implementing the freeze and frozen command because we did not have possession of a third TekBot to test on. Coordination of schedules with three or four p eople to test the bots is very challenging, especially if not everyone is on the same page. Correctly configuring the timer/counters to work consistently for the challenge code was also rather difficult and took considerably more time than implementing the PWM mode. Most of the trouble encountered in the challenge section was due to unfamiliar hardware and software, as there is a steep learning curve when experimenting with new AVR material. Our understanding of timers/counters and their implementation on the ATMEGA128 has greatly improved after hours of poring over the ATMEGA128 Datasheet in an attempt to correctly configure the control registers.

CONCLUSION

In this lab, we were asked to utilize USART communication to a create a 'remote control' for a TekBot, giving it a unique BotID to ensure communication came from the associated remote, as well as the TekBot itself. We implemented forward, backward, turn left, turn right, halt, speed control, and freeze commands. The lab was an opportunity to learn how to use the USART module and how to utilize the datasheet to correctly configure I/O ports on the microcontroller. Over the course of the multi-week lab, we gained a more in depth understanding of the material covered in class, and this lab helped to demonstrate the importance of good programming techniques and etiquette, especially for larger, collaborative programming projects.

TX SOURCE/CHALLENGE CODE

```
;***********************************************************
;*
;*      This is the TRANSMIT skeleton file for Lab 8 of ECE 375
;*
;*       Author: Mason Obery and Benjamin Leonard
;*         Date: 2/24/2022 12:23:27 PM
;*
;***********************************************************

.include "m128def.inc"              ; Include definition file

;***********************************************************
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                               ; Multi-Purpose Register
.def    ereg = r17                              ;extra register'
.def    waitcnt = r23                           ; Wait Loop Counter
.def    ilcnt = r24                             ; Inner Loop Counter
.def    olcnt = r25                   ; Outer Loop Counter

.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    BotAddress = 0x2B;(Enter your robot's address here (8 bits))


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

;***********************************************************
;*      Interrupt Vectors
;***********************************************************
.org    $0000                                   ; Beginning of IVs
rjmp    INIT                  ; Reset interrupt
;.org UDRE1addr
;               rcall finishsend
;               ret
.org    $0046                                   ; End of Interrupt Vectors

;***********************************************************
;*      Program Initialization
;***********************************************************
INIT:
;Stack Pointer (VERY IMPORTANT!!!!)
ldi mpr, low(RAMEND)
out SPL, mpr
ldi mpr, high(RAMEND)
out SPH, mpr

;I/O Ports
ldi mpr, 0b0000_1000
out DDRD, mpr
ldi mpr, 0b11110111
out PORTD, mpr
```

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

ldi mpr, (1<<U2X1)
sts UCSR1A, mpr

ldi mpr, high(832)
sts UBRR1H, mpr
ldi mpr, low(832)
sts UBRR1L, mpr

ldi mpr, ((1<<TXEN1))
sts UCSR1B, mpr
ldi mpr, (1<<UCSZ11 | 1<<UCSZ10 | 1<<USBS1 | 0<<UMSEL1)
sts UCSR1C, mpr

;Set baudrate at 2400bps
;SEI



rjmp MAIN
;Other

;***********************************************************
;*      Main Program
;***********************************************************
MAIN:
;TODO: ???
rcall transmit

rjmp    MAIN

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

transmit:
in mpr, PIND
andi    mpr, 0b1111_0111

cpi mpr, 0b1111_0110
breq forward
cpi mpr, 0b1111_0101
breq backward
cpi mpr, 0b1110_0111
breq right
cpi mpr, 0b1101_0111
breq left
cpi mpr, 0b1011_0111
breq stop
cpi mpr, 0b0111_0111
breq freeze

ret

forward:
ldi mpr, 0b10110000
rjmp check
backward:
ldi mpr, 0b1000_0000
rjmp check
left:
ldi mpr, 0b1010_0000
rjmp check
right:
ldi mpr, 0b1001_0000
rjmp check
stop:
ldi mpr, 0b1100_1000
```

```
rjmp check
freeze:
ldi mpr, 0b1111_1000 ;set to freeze
;ldi mpr, 0b0101_0101
rjmp check

check:
push mpr
ldi mpr, BotAddress ;transmit bot ID
RCALL USART_Transmit
pop mpr
out PORTB, mpr
RCALL USART_Transmit
ret

USART_Transmit:
lds ereg, UCSR1A
sbrs ereg, UDRE1 ; Loop until UDR0 is empty
rjmp USART_Transmit
sts UDR1, mpr ; Move data to transmit data buffer
ret
;-----------------------------------------------------------------
; 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:
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
;***********************************************************
.org $0200
.dseg
data: .byte 1
```

RX SOURCE CODE

```
;***********************************************************
;*
;*      This is the RECEIVE file for Lab 8 of ECE 375
;*
;*       Author: Mason Obery and Benjamin Leonard
;*         Date: 2/24/2022
;*
;***********************************************************

.include "m128def.inc"              ; Include definition file
```

```
;***********************************************************
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                               ; Multi-Purpose Register
.def    ereg = r17
.def    waitcnt = r23                           ; Wait Loop Counter
.def    ilcnt = r18                             ; Inner Loop Counter
.def    olcnt = r19                             ; Outer Loop Counter

.equ    WTime = 100                             ; Time to wait in wait loop

.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 = 0x2B;(Enter your robot's address here (8 bits))

;/////////////////////////////////////////////////////////////
;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    INT0addr
rcall HitRight
reti

.org    INT1addr
rcall HitLeft
reti

.org    $003C
rcall rCommand
reti
;Should have Interrupt vectors for:
;- Left whisker
;- Right whisker
;- USART receive

.org    $0046                                   ; End of Interrupt Vectors

;***********************************************************
;*      Program Initialization
;***********************************************************
INIT:
ldi ereg, $00

;Stack Pointer (VERY IMPORTANT!!!!)
ldi             mpr, low(RAMEND)
out             SPL, mpr
ldi             mpr, high(RAMEND)
out             SPH, mpr
;I/O Ports
ldi             mpr, 0b0000_1000 ; 1 is for IR output for Freeze
```

```
out             DDRD, mpr
ldi             mpr, $FF
out             PORTD, mpr
; Initialize Port B for output
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

;USART0 to recieve
;Set baudrate at 2400bps
ldi mpr, high(832)
sts UBRR1H, mpr
ldi mpr, low(832)
sts UBRR1L, mpr

;Enable receiver and enable receive interrupts
;Set frame format: 8 data bits, 2 stop bits
ldi mpr, (1<<U2X1)
sts UCSR1A, mpr
ldi mpr, ((1<<RXCIE1) | (1<<RXEN1));
sts UCSR1B, mpr
ldi mpr, ((1<<UCSZ11)|(1<<UCSZ10)|(1<<USBS1)) ;modified the paratheses
sts UCSR1C, mpr

;USART1 (to send freexe)

ldi             ZL, low(HP)
ldi             ZH, high(HP)
ldi             mpr, 3
st              Z, mpr

;External Interrupts
;Set the External Interrupt Mask
;Set the Interrupt Sense Control to falling edge detection
ldi             mpr, 0b00001010 ; falling edge
sts             EICRA, mpr
ldi ZL, low(preFreeze)
ldi ZH, high(preFreeze)
in mpr, PORTB
st Z, mpr
ldi             mpr, 0b00000011 ; int0-3 activated
out             EIMSK, mpr
;Other
; Initialize TekBot Forward Movement
ldi             mpr, MovFwd             ; Load Move Forward Command
out             PORTB, mpr              ; Send command to motors
SEI
rjmp Main

;***********************************************************
;*      Main Program
;***********************************************************
MAIN:
;TODO: ???
rjmp    MAIN

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

; ;-----------------------------------------------------------
;       HitRight
; Desc: Handles functionality of the TekBot when the right whisker
;               is triggered.
;-----------------------------------------------------------
HitRight:
push    mpr                     ; Save mpr register
push    waitcnt                 ; Save wait register
in              mpr, SREG       ; Save program state
push    mpr                     ;
```

```
in              mpr, PORTB
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

ldi mpr, $0F
out EIFR, mpr

pop     mpr
out             PORTB, mpr

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

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

ldi mpr, $0F
out EIFR, mpr
pop     mpr
out             PORTB, mpr

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

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

rCommand:
lds mpr, UDR1 ; reads transmitted info

push mpr
ldi mpr, $00
sts UCSR1B, mpr
ldi mpr, ((1<<RXCIE1) | (1<<RXEN1))
sts UCSR1B, mpr
pop mpr

cpi mpr, 0b01010101;check if freeze command recieved
breq frozen

cpi mpr, BotAddress   ;botID recieved
brne notEqual
ldi ereg, $FF
ret

NotEqual:
cpi ereg, $FF  ;check that the botID preceded it
breq ValidCommand
ret

ValidCommand:  ;check for valid opcodes
cpi mpr, 0b1011_0000
breq forward
cpi mpr, 0b1000_0000
breq backward
cpi mpr, 0b1010_0000
breq right
cpi mpr, 0b1001_0000
breq left
cpi mpr, 0b1100_1000
breq stop
cpi mpr, 0b1111_1000
breq freeze
rjmp toEnd

forward:
ldi mpr, MovFwd
out PORTB, mpr
rjmp toEnd
backward:
ldi mpr, MovBck
out PORTB, mpr
rjmp toEnd
right:
ldi mpr, TurnR
out PORTB, mpr
rjmp toEnd
left:
ldi mpr, TurnL
out PORTB, mpr
rjmp toEnd
stop:
ldi mpr, Halt
out PORTB, mpr
rjmp toEnd

freeze:;set up transmitter, disable recever
ldi mpr, ((1<<TXEN1))
sts UCSR1B, mpr
ldi mpr, 0b01010101
rcall USART_Transmit1
ldi mpr, ((1<<RXCIE1) | (1<<RXEN1))   ;reenable reciever to work properly
sts UCSR1B, mpr
rjmp toEnd
```

```
toEnd:
ldi ereg, $00
ret

USART_Transmit1:
lds ereg, UCSR1A
sbrs ereg, UDRE1 ; Loop until UDR0 is empty
rjmp USART_Transmit1
sts UDR1, mpr ; Move data to transmit data buffer
USART_Transmit2:
lds ereg, UCSR1A
sbrs ereg, UDRE1 ; Loop until UDR0 is empty
rjmp USART_Transmit2
ret

frozen:
ldi ZL, low(preFreeze)
ldi ZH, high(preFreeze)
in mpr, PORTB
st Z, mpr

ldi    mpr, halt
out PORTB, mpr

ldi ZL, low(HP)
ldi ZH, high(HP)
ld mpr, Z
dec mpr
st Z, mpr

ld mpr, Z
cpi mpr, 0
breq killed

ldi waitcnt, 250
rcall Wait
ldi waitcnt, 240
rcall Wait

ldi ZL, low(preFreeze)
ldi ZH, high(preFreeze)
ld     mpr, Z
out    PORTB, mpr

rjmp toEnd

killed:
ldi mpr, 0
out EIMSK, mpr
sts UCSR1B, mpr

rjmp toEnd

;-----------------------------------------------------------
; 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:
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
;***********************************************************
.dseg

.org $0200
HP: .byte 1
PreFreeze: .byte 1
```

RX CHALLENGE CODE

```
;*;***********************************************************
;*
;*      This    is      the     RECEIVE     skeleton     file     for     Lab     8     of     ECE     375
;*
;*       Author:            Mason            Obery            and            Benjamin            Leonard
;*         Date:                                                                                  2/24/2022
;*
;***********************************************************

.include "m128def.inc"                ;             Include             definition             file

;***********************************************************
;*      Internal            Register            Definitions             and             Constants
;***********************************************************
.def    mpr = r16                               ;               Multi-Purpose                Register
.def    ereg                                            =                                            r17
.def    waitcnt = r23                           ;            Wait            Loop            Counter
.def    ilcnt = r18                             ;            Inner           Loop            Counter
.def    olcnt = r19                             ;            Outer           Loop            Counter

.equ    WTime = 100                             ;     Time     to     wait     in     wait     loop

.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      =       0x2B;(Enter     your     robot's     address     here     (8     bits))

;///////////////////////////////////////////////////////////
;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    INT0addr
        rcall                                                                     HitRight
        reti

.org    INT1addr
        rcall                                                                      HitLeft
        reti

.org    $0034
        rcall                                                                   nowgoleft
        reti

.org    $0036
        rcall                                                                  nowgoright
        reti

.org    $003A
        rcall                                                                     finish
        reti

.org    $003C
        rcall                                                                   rCommand
        reti
;Should          have            Interrupt           vectors             for:
;-                                Left                                  whisker
;-                                Right                                 whisker
;-                                USART                                 receive

.org    $0046                           ;       End       of       Interrupt       Vectors

;***********************************************************
;*      Program                                                        Initialization
;***********************************************************
INIT:
        ;new                                                                    plan

        /*ldi                                   mpr,                              $00
        sts                                   TCCR3A,                           mpr*/

        ldi                                     ereg,                             $00

        ;Stack                  Pointer                  (VERY                IMPORTANT!!!!)
        ldi                                   mpr,                          low(RAMEND)
        out                                       SPL,                            mpr
        ldi                                   mpr,                         high(RAMEND)
        out                                       SPH,                            mpr
        ;I/O                                                                   Ports
        ldi   mpr,   0b0000_1000   ;button   inputs,   1   is   for   IR   output   for   Freeze
        out                                     DDRD,                             mpr
        ldi                                     mpr,                              $FF
        out                                    PORTD,                             mpr
        ;            Initialize            Port            B            for           output
        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

        ldi       mpr,       $FF       ;       output       for       COM       of       timer3
        out                                     DDRE,                             mpr
```

```
        ;USART0                                 to                                      recieve
               ;Set                    baudrate                    at                  2400bps
               ldi                             mpr,                                    high(832)
               sts                              UBRR1H,                                    mpr
               ldi                             mpr,                                    low(832)
               sts                              UBRR1L,                                    mpr

               ;Enable        receiver        and        enable        receive        interrupts
               ;Set frame format: 8 data bits, 2 stop bits
               ldi                             mpr,                                    (1<<U2X1)
               sts                              UCSR1A,                                    mpr
               ldi            mpr,            ((1<<RXCIE1)            |            (1<<RXEN1));
               sts                              UCSR1B,                                    mpr
               ldi    mpr,    ((1<<UCSZ11)|(1<<UCSZ10)|(1<<USBS1))    ;modified    the    paratheses
               sts                             UCSR1C,                                    mpr

        ;USART1                        (to                        send                        freexe)

        ldi                                    ZL,                                     low(HP)
        ldi                                    ZH,                                    high(HP)
        ldi                                       mpr,                                    3
        st                                     Z,                                         mpr



        ;                     Configure                     8-bit                     Timer/Counters
               ldi                             mpr,                                    0b01_11_1_001
               out                                 TCCR2,                                 mpr

               ldi                             mpr,                                    0b01_11_1_001
               out                                 TCCR0,                                 mpr
        ldi mpr, 0b0000_0000 ; outputcompare interrupt match enable for timer0 and timer2. and com1
and               com2              and               ovf               for               timer1
               out                             TIMSK,                                     mpr
        ldi                                    ZL,                                     low(speed)
               ldi                                ZH,                                    high(speed)
               ldi                                    mpr,                                   $0F
               st                                  Z,                                     mpr


        ;External                                                                      Interrupts
               ;Set             the             External             Interrupt             Mask
               ;Set    the    Interrupt    Sense    Control    to    falling    edge    detection
               ldi           mpr,           0b00001010           ;           falling           edge
               sts                              EICRA,                                     mpr
               ldi          mpr,          0b00000011          ;          int0-3          activated
               out                             EIMSK,                                     mpr
        ;Other
        ;               Initialize              TekBot              Forward              Movement
               ldi             mpr, MovFwd             ;      Load      Move      Forward      Command
               out             PORTB, mpr              ;       Send      command      to       motors

                      ldi                              mpr,                                HIGH(49911)
               sts                             OCR3AH,                                    mpr
               ldi                            mpr,                                    LOW(49911)
               sts                              OCR3AL,                                    mpr

                      ;right                                                                  side
               ldi                               mpr,                                    HIGH(49911)
               sts                                 OCR3BH,                                    mpr
               ldi                               mpr,                                    LOW(49911)
               sts                                 OCR3BL,                                    mpr


               /*      ldi                              mpr,                                0b1111_0000*/
                      ldi                               mpr,                                 0b0000_0000
```

```
        sts                                     TCCR3A,                                         mpr
        ldi                                    mpr,                                     0b0000_0101
        sts                                     TCCR3B,                                         mpr


        SEI
        rjmp                                                                                    Main
;*********************************************************
;*      Main                                                                             Program
;*********************************************************
MAIN:
        ;TODO:                                                                                  ???
        nop
        nop
        nop
        nop

busywait:
        rjmp                                                                             busywait



        rjmp    MAIN

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

;*********************************************************
;*      Stored                                    Program                                   Data
;*********************************************************
;*************************************************************
;*                      Subroutines                        and                        Functions
;*************************************************************

;---------------------------------------------------------
; Func: INCspeed
; Desc: Increments             speed             of             less             than             15
;---------------------------------------------------------

INCspeed:
        PUSH                                                                                    ZL
        PUSH                                                                                    ZH
        PUSH                                                                                   mpr

        ldi                                     ZL,                                     low(speed)
        ldi                                    ZH,                                     high(speed)

        ld                                         mpr,                                         Z
        CPI                                        mpr,                                        15
        BREQ exit       ;check    if    speed    level    is    at    15,    exit    if    it    is
        inc mpr         ;if          it's          not          inc          speed          value
        st                                          Z,                                         mpr

        rjmp                                                                                   exit


;---------------------------------------------------------
; Func: DECspeed
; Desc: Decrements             speed             of             less             than             0
;---------------------------------------------------------

DECspeed:
        PUSH                                                                                    ZL
        PUSH                                                                                    ZH
```

```
        PUSH                                                                    mpr

        ldi                                     ZL,                         low(speed)
        ldi                                     ZH,                        high(speed)

        ld                                      mpr,                            Z
        CPI mpr, 0      ;check          if          speed           level           is          at          0
        BREQ                                                                    exit
        dec mpr         ;if             not,            decrement           speed           by          1
        st                                      Z,                              mpr


exit:           ;all functions share the same exit seqeunce, so to save space it is abbrieviated to
        ldi                                     waitcnt,                        40
        rcall                                                                   Wait

        rcall                                                                   LeftOn
        rcall                                                                   RightOn


        ldi         mpr,        $0F             ;clear      external        interrupt       register
        out                                     EIFR,                           mpr

        POP                                                                     MPR
        POP                                                                     ZH
        POP                                                                     ZL
        ret


;-----------------------------------------------------------
; Func: LeftOn
; Desc: Sets                        PWM                         signal                      on
;-----------------------------------------------------------

LeftOn:
        push                                                                    mpr
        push                                                                    ZL
        push                                                                    ZH
        push                                                                  waitcnt

        ldi                                     ZL,                         low(speed)
        ldi                                     ZH,                        high(speed)
        ld                                          mpr,                        Z
        ldi                                       waitcnt,                      17
        mul mpr, waitcnt        ;convert    4-bit    nibble    to    8-bit    even    distirbution
        mov                                         mpr,                        r0
        out                                       OCR2,                         mpr

        /*ldi                                       mpr,                        $FF
        out TIFR, mpr   ;clear              timer               interrupt               register*/

        pop                                                                   waitcnt
        pop                                                                     ZH
        pop                                                                     ZL
        pop                                                                     mpr
        ret

;-----------------------------------------------------------
; Func: RightOn
; Desc: Sets                PWM                 signal              to                approriate
;-----------------------------------------------------------

RightOn:
        push                                                                    mpr
        push                                                                    ZL
```

```
        push                                                                    ZH
        push                                                                 waitcnt

        ldi                                     ZL,                       low(speed)
        ldi                                     ZH,                      high(speed)
        ld                                     mpr,                             Z
        ldi                                  waitcnt,                           17
        mul mpr, waitcnt      ;convert    4-bit    nibble    to    8-bit    even    distirbution
        mov                                    mpr,                            r0
        out                                   OCR0,                           mpr

        /*ldi                                    mpr,                          $FF
        out TIFR, mpr  ;clear              timer              interrupt              register*/

        pop                                                                  waitcnt
        pop                                                                       ZH
        pop                                                                       ZL
        pop                                                                      mpr
        ret
```

```
;-----------------------------------------------------------
; Sub:  HitRight
; Desc: Handles     functionality     of     the     TekBot     when     the     right     whisker
;             is                                                                  triggered.
;-----------------------------------------------------------
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
              rcall              RWait              ;right              side              wait

              ;            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
              rcall                                                              NewWait*/

              ldi            mpr, MovBck  ;       Load       Move       Backward       command
              out            PORTB, mpr   ;         Send        command        to         port




              rjmp                                                          rdisableandenable

/*hitrmid:           pop           mpr          ;        Restore        program        state
              out            SREG, mpr    ;
              pop            waitcnt      ;          Restore          wait          register
              pop            mpr          ;                  Restore                    mpr
              ret                         ;         Return         from         subroutine*/
```

```
;-----------------------------------------------------------
; Sub:  HitLeft
; Desc: Handles functionality of the TekBot when the left whisker
;       is triggered.
;-----------------------------------------------------------
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
        rcall   RWait           ;right side wait

        ;       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
        rcall   NewWait*/



    ldi     mpr, MovBck     ; Load Move Backward command
        out     PORTB, mpr      ; Send command to port

        rjmp    ldisableandenable;
/*
hitlmid: pop     mpr             ; Restore program state
        out     SREG, mpr       ;
        pop     waitcnt         ; Restore wait register
        pop     mpr             ; Restore mpr
        ret                     ; Return from subroutine*/
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


rdisableandenable:
    ldi     mpr, 0b0000_0000    ; int0-3 de activated
    out     EIMSK, mpr

    ldi     mpr, MovBck     ; Load Move Backward command
    out     PORTB, mpr      ; Send command to port

    ldi     mpr, $00
    sts     UCSR1B, mpr

    ;use timer 1 with two different output compares and the overflow
/*  ldi     mpr, 0b0011_0000
    sts     TCCR3A, mpr*/
/*  ldi     mpr, 0b0000_0101
    sts     TCCR3B, mpr*/
    ;nothing for TCCR1C, used to force compare match


    ;now load values
    ;starting new timer attempt HERE:
        ldi     mpr, HIGH(34286)    ;34286
        sts     TCNT3H, mpr
        ldi     mpr, LOW(34286)
        sts     TCNT3L, mpr
```

```
        ;configure                                                      TIMSK
        ;ldi   mpr,   0b0001_1100   both,   bit   4   is   A,   3   is   B,   and   2   is   OVF
        ldi                             mpr,                                    0b0001_0100
        sts                               ETIMSK,                                       mpr


        ldi           mpr, MovBck    ;        Load        Move        Backward        command
        out           PORTB, mpr     ;          Send          command          to          port

        ldi                             mpr,                                    0b0001_1100
        sts ETIFR, mpr ;clear qued timer compare matches that set too early because they queued


        pop                                                                             mpr
              out                                 SREG,                             mpr
              pop                                                                 waitcnt
              pop                                                                   mpr

        ret

ldisableandenable:
        ldi         mpr,         0b0000_0000         ;         int0-3         de         activated
        out                               EIMSK,                                        mpr

        ldi           mpr, MovBck    ;        Load        Move        Backward        command
        out           PORTB, mpr     ;          Send          command          to          port


        ldi                               mpr,                                        $00
        sts                              UCSR1B,                                      mpr


        ;use   timer   1   with   two   different   output   compares   and   the   overflow
/*      ldi                             mpr,                                    0b1100_0000
        sts                               TCCR3A,                                       mpr*/
/*      ldi                             mpr,                                    0b0000_0101
        sts                               TCCR3B,                                       mpr*/
        ;nothing       for       TCCR1C,       used       to       force       compare       match


        ;now                              load                                    values
        ;starting             new             timer             attempt             HERE:
        ;right                                                                     side


              ldi                          mpr,                                HIGH(34286)
              sts                            TCNT3H,                                 mpr
              ldi                          mpr,                                 LOW(34286)
              sts                            TCNT3L,                                 mpr




        ;configure                                                      TIMSK
        ;ldi   mpr,   0b0001_1100   both,   bit   4   is   A,   3   is   B,   and   2   is   OVF
        ldi                             mpr,                                    0b0000_1100
        sts                               ETIMSK,                                       mpr
```

```
        ldi             mpr, MovBck     ;         Load         Move         Backward         command
        out             PORTB, mpr      ;          Send           command          to           port

        ldi                                     mpr,                                        0b0001_1100
        sts ETIFR, mpr ;clear qued timer compare matches that set too early because they queued

        pop                                                                                     mpr
                out                                     SREG,                                   mpr
                pop                                                                           waitcnt
                pop                                                                             mpr

        ret



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

rCommand:
        ;in                             did                             not                         work
        lds          mpr,          UDR1          ;          reads          transmitted          info

        cpi                                     mpr,                                        BotAddress
        brne                                                                                  notEqual
        ldi                                        ereg,                                        $FF
        ret

NotEqual:
        cpi                                        ereg,                                        $FF
        breq                                                                                ValidCommand
        ret

ValidCommand:
        cpi                                     mpr,                                        0b1011_0000
        breq                                                                                    forward
        cpi                                     mpr,                                        0b1000_0000
        breq                                                                                   backward
        cpi                                     mpr,                                        0b1010_0000
        breq                                                                                     right
        cpi                                     mpr,                                        0b1001_0000
        breq                                                                                      left
        cpi                                     mpr,                                        0b1100_1000
        breq                                                                                    speedUp
        cpi                                     mpr,                                        0b1111_1000
        breq                                                                                  speedDown
        rjmp                                                                                    toEnd

forward:
        ldi                                       mpr,                                          MovFwd
        out                                        PORTB,                                         mpr
        rjmp                                                                                    toEnd
backward:
        ldi                                       mpr,                                          MovBck
        out                                        PORTB,                                         mpr
        rjmp                                                                                    toEnd
right:
        ldi                                        mpr,                                          TurnR
        out                                        PORTB,                                         mpr
        rjmp                                                                                    toEnd
left:
        ldi                                        mpr,                                          TurnL
        out                                        PORTB,                                         mpr
        rjmp                                                                                    toEnd
stop:
        ldi                                         mpr,                                           Halt
        out                                        PORTB,                                         mpr
        rjmp                                                                                    toEnd
```

```
speedUp:
        rcall                                                                   INCSPEED
        rjmp                                                                      toEnd
speedDown:
        rcall                                                                   DECSPEED
        rjmp                                                                      toEnd
toEnd:
        ldi                                     ereg,                               $00
        ret



        ;stop                com                and                turn                signal
nowgoleft:
        /*ldi                                  mpr,                                0b0000_0000
        sts                                    TCCR3A,                                mpr*/

        ldi             mpr, TurnL      ;          Load          Turn          Left          Command
        out             PORTB, mpr      ;          Send          command          to          port
/*      ldi             mpr, MovBck     ;         Load         Move         Backward         command
                out             PORTB, mpr      ;         Send         command         to         port*/

        ret

nowgoright:
/*      ldi                                   mpr,                                0b0000_0000
        sts                                     TCCR3A,                                mpr*/

        ldi             mpr, TurnR      ;          Load          Turn          Left          Command
        out             PORTB, mpr      ;          Send          command          to          port
/*      ldi             mpr, MovBck     ;         Load         Move         Backward         command
        out             PORTB, mpr      ;          Send          command          to          port*/

        ret

finish:
        ldi             mpr, MovFwd             ;        Load        Move        Forward        Command
        out             PORTB, mpr              ;         Send         command         to         motors

        ldi                         mpr,                          0b0000_0000                          ;
        sts                                     ETIMSK,                                 mpr

        ldi                                       mpr,                                  $FF
        out                                       EIFR,                                 mpr

        ldi             mpr,            0b00000011              ;            int0-1            activated
        out                                     EIMSK,                                  mpr

                ldi             mpr,            ((1<<RXCIE1)            |            (1<<RXEN1));
                sts                                  UCSR1B,                                mpr

        ret




------------------------------------------------------------
; 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:
```

```
              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
;*******************************************************
;*     Additional                                Program                                Includes
;*******************************************************
.dseg

.org                                                                                      $0200
       HP:                                          .byte                                   1
       speed: .byte 1
```