ECE 375 Lab 8

Remotely Operated Vehicle

**Lab Time: Wednesday 8-10**

Jiongcheng Luo

# Introduction

For this lab, we are supposed to write an assembly program that allows “remotely controlled” by connection two boards (one as a transmitter and the other one as a receiver), specifically use the USART module on the robot board and on the remote board. The proof-of-concept robot is able to o implement a few different actions: move forward, move backward, turn right, turn left, halt and send freeze signal as well as received freeze command. In addition, the robot has some minimal intelligence on the robot which can perform some BumpBot behavior by hitting the left and right whiskers.

# Program Overview

--- Transmitter ---

INITIALIZE includes:

* Set stack Pointer
* Set up I/O ports, transmitter will only use PORTD for inputs (PORTB may be used for testing purpose)
* Set up USART1
  + We used double data rate and baudrate at 2400 bps.
  + Enable the device as a transmitter (TXEN1)
  + Set frame format: 8 data bits, 2 stop bits

MAIN (Functions & Subroutines) includes:

* Read input from PIND (as command)
* Check which command is inputted (relatively check every possible command)
* Send out message (2 bytes) to the transmit buffer (UDR1)
  + The first byte is a unique robot address
  + The second byte is the actual action command

--- Receiver ---

INITIALIZE includes:

* Set interrupt for left, right whiskers and USART receive respectively addresses with $0002, $0004 and $003C
* Set stack Pointer
* Set up I/O ports, Receiver (robot) uses PORTD as the inputs from hitting left and right whisker of the robot, PORTB as output for corresponding actions.
* Set up USART1
  + We used double data rate and baudrate at 2400 bps.
  + Enable the device as both transmitter (TXEN1), receiver (RXEN1), as well as enable receiver interrupt (RXCIE1)
  + Set frame format: 8 data bits, 2 stop bits
* Enable external interrupt requests for PIND 0 and 1 (left & right whiskers)
* Enable global interrupt

MAIN (Functions & Subroutines) includes:

* The MAIN function always outputs to PORTB for the latest action, and it acts as an infinite loop and waits for interrupt to be triggered.
* Function “ChkFreeze” is the service routine for USART interrupt, which checks the MSB (most significant bit from the received message). The message would be either a freeze signal or a robot ID if MSB is 0, and if the MSB is 1 then it is an action command
* The “ChkFreeze” routine (MSB as 0) checks if the message a Robot ID ($66) or a freeze signal (0b01010101), if it’s a Robot ID, it branches to the “ChkRobotID” routine, otherwise it’s a freeze signal, then it checks if the signal is sent by itself (by checking a flag is set), if the flag is not set (not sent by itself) then it checks if the freeze counter has been up to 3 times, if so then frozen forever, otherwise (normal situation) halts for 5 seconds.
* The “ChkRobotID” simply compare the received robot address with the correct value, if they matched it stores the ID to a temp register and jump back to the “ChkFreeze” routine to get the next byte (action command), if not matched then return and wait for the next command.
* The “ChkAction” routine would be called if the last byte (robot ID) is verified to be the same, if not then return back and do nothing; if matched, it checks if the action command is normal action or a freeze command. If it’s a normal action it simply left shifts the current message (to the correct output command) and outputs to PORTB, and clear the temp register that stores the ID. Otherwise (a freeze command), it jumps to “FreeAct” routine.
* The “FreeAct” routine simply transmits out the freeze signal to UDR1 in order to not interrupt by itself, the receiver (RXEN1) was disable temporarily before sending the signal, and it will be enable again after sending out the signal, so that during this period the receiver won’t be interrupted by itself.
* In addition, there are two interrupt respectively corresponding to the left/right bumper is hit on the robot, and HitRight and HitLeft service routines implement for the interrupt.
* A Wait routine implements a delay for waiting for certain period (second).

# Conclusion

This lab requires to display LEDs (PORTB on receiver board) correctly for corresponding command/signal, as well as allows to perform correctly when the bumpers are hit on the receiver board (indicate hitleft/hitright). The problem I encountered in designing this program was to how to let the robot correctly perform what it was doing after resume from any interrupt routines, the way I solved it was every time I output an action command, I let another register to store the command value and I let it always to perform the latest/updated action command in the MAIN function (when it’s not interrupted by other signal). In overall, the program runs successfully but somehow with unexpected error output that was not caused by logical errors.

# Source Code

--- Transmitter

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* Enter remote.asm

;\*

;\* ECE375 - Lab 8 Transmitter

;\*

;\* This is the TRANSMIT remote.asm file for Lab 8 of ECE 375

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* Author: Jiongcheng Luo

;\* Date: 2/24/16

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.include "m128def.inc" ; Include definition file

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Internal Register Definitions and Constants

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.def mpr = r16 ; Multi-Purpose Register

.def CMD = r17

.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 RobotID = $66 ; set robot address/ID

; 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 Command

.equ Forward = 0 ; PORTD PIN 0 as forward input

.equ Reverse = 1 ; PORTD PIN 1 as reverse input

.equ TRight = 6 ; PORTD PIN 6 as turn right input

.equ TLeft = 7 ; PORTD PIN 7 as turn left input

.equ THalt = 5 ; PORTD PIN 5 as hult input

.equ TFreeze = 4 ; PORTD PIN 4 as freeze input

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Start of Code Segment

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.cseg ; Beginning of code segment

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Interrupt Vectors

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.org $0000 ; Beginning of IVs

rjmp INIT ; Reset interrupt

.org $0046 ; End of Interrupt Vectors

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Program Initialization

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

INIT:

;Stack Pointer (VERY IMPORTANT!!!!)

LDI mpr, low(RAMEND)

OUT SPL, mpr

LDI mpr, high(RAMEND)

OUT SPH, mpr

;I/O Ports (For Remote, input as PORTD)

LDI mpr, $00

OUT DDRD, mpr ;set PORTD as input

LDI mpr, $FF ;set PORTB as output for test only

OUT DDRB, mpr

;USART1

;Set baudrate at 2400bps

LDI mpr, (1<<U2X1) ; set double data rate

STS UCSR1A, mpr

LDI mpr, high(832)

STS UBRR1H, mpr

LDI mpr, low(832)

STS UBRR1L, mpr

;Enable transmitter

LDI mpr, (1<<TXEN1)

STS UCSR1B, mpr

;Set frame format: 8 data bits, 2 stop bits

LDI mpr, (0<<UMSEL1 | 1<<USBS1 | 1<<UCSZ11 | 1<<UCSZ10)

STS UCSR1C, mpr

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Main Program (Check input from pressing buttons)

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MAIN:

IN MPR, PIND ; Get inputs from PORTD

ANDI MPR, (1<<Forward)|(1<<Reverse)|(1<<TRight)|(1<<TLeft)|(1<<THalt)|(1<<TFreeze) ;$E3 ; MASK out Inputs

CPI MPR, $F2 ; Compare ob11110010 (Forward)

BRNE ChkReverse

LDI CMD, MovFwd ; save action command

RCALL SendID

ChkReverse:

CPI MPR, $F1 ; Compare ob11110001 (Reverse)

BRNE ChkTRight

LDI CMD, MovBck

RCALL SendID

ChkTRight:

CPI MPR, $B3 ; Compare ob10110011 (Turn right)

BRNE ChkTLeft

LDI CMD, TurnR

RCALL SendID

ChkTLeft:

CPI MPR, $73 ; Compare ob01110011 (Turn left)

BRNE ChkHalt

LDI CMD, TurnL

RCALL SendID

ChkHalt:

CPI MPR, $D3 ; Compare ob11010011 (Halt)

BRNE ChkFreeze

LDI CMD, Halt

RCALL SendID

ChkFreeze:

CPI MPR, $E3 ; Compare ob11100011 (Halt)

BRNE SKIP

LDI CMD, Freeze

RCALL SendID

SKIP:

RJMP MAIN

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Functions and Subroutines

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SendID:

LDS mpr, UCSR1A

SBRS mpr, UDRE1 ; check if transmit buffer is empty

RJMP SendID ; loop over if buffer is empty

LDI mpr, RobotID ; Send Robot Address

STS UDR1, mpr

RCALL SendCMD

SendCMD:

LDS mpr, UCSR1A

SBRS mpr, UDRE1 ; check if transmit buffer is empty

RJMP SendCMD ; loop over if buffer is empty

STS UDR1, CMD ; send action command

OUT PORTB, CMD ; for test

RJMP MAIN

--- Receiver

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* robot.asm

;\*

;\* ECE375 - Lab 8 Robot

;\*

;\* This is the RECEIVE robot.asm file for Lab 8 of ECE 375

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* Author: Jiongcheng Luo

;\* Date: 3/11/16

;\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.include "m128def.inc" ; Include definition file

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Internal Register Definitions and Constants

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.def mpr = r16 ; Multi-Purpose Register

.def waitcnt = r17 ; Wait Loop Counter

.def ilcnt = r18 ; Inner Loop Counter

.def olcnt = r19 ; Outer Loop Counter

.def MSG = r20

.def OldMSG = r21

.def IDtemp = r22

.def FreezeFlag = r23

.def FreezeCNT = r24

.equ Halt\_WTime = 140 ; Time to wait in wait loop (2.5s)

.equ Whisker\_WTime = 100 ; Time to wait in wait loop (1.0s)

.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 = $66 ;(Enter your robot's address here (8 bits))

.equ FreezeSIGN = 0b01010101 ;Freeze signal from other robot

;/////////////////////////////////////////////////////////////

;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

.equ FreezeCMD = $F8 ;ob11111000 Freeze command (from remote)

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Start of Code Segment

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.cseg ; Beginning of code segment

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Interrupt Vectors

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.org $0000 ; Beginning of IVs

rjmp INIT ; Reset interrupt

;Should have Interrupt vectors for:

;- Left whisker

.org $0002

RCALL HitLeft ; Reset interrupt

RETI

;- Right whisker

.org $0004

RCALL HitRight ; Reset interrupt

RETI

;- USART receive

.org $003C

RCALL ChkReceived

RETI

.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, $FF ; set PORTB as output

OUT DDRB, mpr

LDI mpr, $00 ; set PORTD PIN 0&1 as input (Right&Left whisker)

OUT DDRD, mpr

LDI mpr, $FF ; Initialize Port D Data Register

OUT PORTD, mpr ; so all Port D inputs are Tri-State

;USART1

;Set baudrate at 2400bps

LDI mpr, (1<<U2X1) ; set double data rate

STS UCSR1A, mpr

LDI mpr, high(832)

STS UBRR1H, mpr

LDI mpr, low(832)

STS UBRR1L, mpr

;Enable receiver and enable receive interrupts

LDI mpr, (1<<RXCIE1 | 1<<RXEN1 | 1<<TXEN1)

STS UCSR1B, mpr

;Set frame format: 8 data bits, 2 stop bits

LDI mpr, (0<<UMSEL1 | 1<<USBS1 | 1<<UCSZ11 | 1<<UCSZ10)

STS UCSR1C, mpr

;Other

LDI mpr, (1<<ISC01)|(0<<ISC00)|(1<<ISC11)|(0<<ISC10) ; using falling edge

STS EICRA, mpr

; Enable External interrupt requests 0 and 1

LDI mpr, (1<<INT0)|(1<<INT1)

OUT EIMSK, mpr

CLR IDtemp

CLR FreezeFlag

CLR FreezeCNT

LDI MSG, MovFwd ; initialize first action

OUT PORTB, MSG

MOV OldMSG, MSG

SEI ; enable global interrupt

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Main Program

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MAIN:

OUT PORTB, OldMSG

RJMP MAIN ; otherwise loop over until interrupt triggered

ChkReceived:

LDS MSG, UDR1 ; receive new signal/message (FreezeSIGN/ID)

SBRS MSG, 7 ; check MSB, 0 (Freeze or ID) or 1 (Action)

RCALL ChkFreeze ; MSB is 0

RCALL ChkAction ; check action if MSB is 1

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Functions and Subroutines

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ChkFreeze:

LDI mpr, FreezeSIGN

CP mpr, MSG ; check if the MSG is freeze signal

BRNE ChkRobotID ; branch check id if not equal

CPI FreezeCNT, 6 ; check if freeze up to 3 times

BREQ FrozenAct

LDI mpr, 1

ADD FreezeCNT, mpr ; increment counter

LDI mpr, Halt ; otherwise, halt robot

OUT PORTB, mpr

LDI waitcnt, Halt\_WTime ; Wait for 2.5 second

RCALL Wait ; Call wait function

RCALL Wait ; Call wait function 2nd time (5s)

RET

SKIPFreeze:

OUT PORTB, OldMSG

CLR FreezeFlag

ldi waitcnt, 5 ; Wait for a moment

rcall Wait ; Call wait function

RJMP ChkReceived

FrozenAct:

LDI mpr, Halt ; otherwise, halt robot

OUT PORTB, mpr

RJMP FrozenAct ; inifitely loop

ChkRobotID:

LDI mpr, BotAddress ;

CP mpr, MSG ; check robot ID

BRNE SKIP ; if not matched, return

MOV IDtemp, MSG ; if matched, save ID

RCALL ChkReceived ; RJMP and get the next byte of message

ChkAction:

LDI mpr, BotAddress ;

CP mpr, IDtemp ; check robot ID for previous byte (ID)

BRNE SKIP ; skip if not equal

; otherwise, output action command

CLR IDtemp ; clear ID after output

LDI mpr, FreezeCMD

CP mpr, MSG ; check current msg with freeze command

BREQ SendFreeze ; if equal then let robot send freeze signal

LSL MSG ; otherwise, left shift action command

MOV OldMSG, MSG

RET

SKIP: RET

SendFreeze:

OUT PORTB, OldMSG

LDI mpr, (0<<RXEN1) ; disable receiver for temp

STS UCSR1B, mpr

ldi waitcnt, 10 ; Wait for a moment

rcall Wait ; Call wait function

LDI mpr, FreezeSIGN

STS UDR1, mpr

ldi waitcnt, 10 ; Wait for a moment

rcall Wait

LDI mpr, (1<<RXEN1) ;enable receiver

STS UCSR1B, mpr

OUT PORTB, OldMSG

RET

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\* Additional Program Includes

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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, Whisker\_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, Whisker\_WTime ; Wait for 1 second

rcall Wait ; Call wait function

; Resume previous action again

;out PORTB, OldMSG ; Send command to port

pop mpr ; Restore program state

out SREG, mpr ;

pop waitcnt ; Restore wait register

pop mpr ; Restore mpr

ret ; Return from subroutine

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, Whisker\_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, Whisker\_WTime ; Wait for 1 second

rcall Wait ; Call wait function

; Resume previous action again

;out PORTB, OldMSG ; Send command to port

pop mpr ; Restore program state

out SREG, mpr ;

pop waitcnt ; Restore wait register

pop mpr ; Restore mpr

ret ; Return from subroutine

;----------------------------------------------------------------

; 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