ECE 375 Lab 8

Remotely operated Vehicles

**Lab Time: Friday 4-6**

Hyunjae Kim

# Introduction

In this lab section, the Tekbot will be upgraded by using USART feature of AVR board. The Tekbot can send the command to other Tekbot to do the received command, send freeze command to other Tekbot when the Tekbot receives the command, or responds to external interrupts. Furthermore, the Tekbot operates the commands when it received the correct Bot Address.

# Program Overview

In the program, there are transmitter and receiver program.

First, the transmitter program sends the bot address and command. The transmitter sends corresponding Bot address, and then sends the commands: Move Forward, Move Backward, Turn Right, Turn Left, Halt, Send Freeze command.

Second, the receiver program first receives the frame, and then check it is right Bot address. If the address is correct, then the receiver gets and executes the command. Furthermore, the receiver program reacts to external interrupts, and when the external interrupt occurs, the receiver program does not receive the command from the transmitter.

## Initialization Routine

1) Transmitter

The stack is initialized, and Port D is initialized as input except pd3 and pd4. Then, in USART1, transmitter function is enabled.

2) Receiver

The stack is initialized, and Port B is initialized as output. Then in USART1, transmitter function, receiver function, and receive interrupt are enabled. Furthermore, the external interrupts, INT 0 and INT 1 are initialized for responding external interrupts.

## Main Routine

1)Transmitter

In main routine, the program reads input from PIND, and then jumps to corresponding subroutines.

2)Receiver

In main routine, the program does nothing with maintaining previous state, and jumps to subroutine whenever interrupts are occurred. (When the program returns from the interrupt, it maintains the previous state before the interrupt.)

## Subroutines

1)Transmitter

* GoFoward : Sends Bot address and 0b10110000 command code to receiver bot.
* GoBack: Sends Bot address and 0b10000000 command code to receiver bot
* TurnRight: Sends Bot address and 0b10100000 command code to receiver bot
* TurnLeft: Sends Bot address and 0b10010000 command code to receiver bot
* Halt: Sends Bot address and 0b11001000 command code to receiver bot
* Freeze: Sends Bot address and 0b11111000 command code to receiver bot

2)Receiver

* GoForward : Checks the received Bot address, and execute the command code – moving forward.
* GoBack: Checks the received Bot address, and execute the command code – moving backward.
* TurnRight: Checks the received Bot address, and execute the command code – turning right.
* TurnLeft: Checks the received Bot address, and execute the command code – turning left.
* Halt: Checks the received Bot address, and execute the command code – halts.
* Freeze: Checks the received Bot address, and execute the command code – send command $55 to other Bot.
* External Interrupt1 – Right whisker Hit

Moves Backward for a second, Turn left for a second, then Move forward.

* External Interrupt2 – Left whisker Hit

Moves Backward for a second, Turn right for a second, then Move forward.

(Note: If the address does not match, the receiver does nothing. The command cannot be received during the external ISR.)

# Difficulties

Since the concept of USART was novel concept, it had some time to understand and exploit those concepts to implement the Tekbot. However, after sufficiently understanding and learning about the concept, the challenges of this lab were clear and able to solve effectively. Thus, by reviewing the concepts and features of USART, the challenges were resolved.

# Conclusion

Through this lab activity, the prospect and perspective of the computation machine have been expanded and enable the connection between the real-world. The lessons from this activity will make the problem solving more effective and various by adapting to the problems. Thus, through this lab, the bridge from fundamentals to practical/complex is completed.

# Source Code

1)Transmitter

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

;\*

;\* This is the TRANSMIT skeleton file for Lab 8 of ECE 375

;\*

;\* Author: Hyunjae Kim

;\* Date: 02/25/2022

;\*

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

.include "m128def.inc" ; Include definition file

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

;\* Internal Register Definitions and Constants

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

.def mpr = r16 ; Multi-Purpose Register

.equ EngEnR = 4 ; Right Engine Enable Bit

.equ EngEnL = 7 ; Left Engine Enable Bit

.equ EngDirR = 5 ; Right Engine Direction Bit

.equ EngDirL = 6 ; Left Engine Direction Bit

; Use these action codes between the remote and robot

; MSB = 1 thus:

; control signals are shifted right by one and ORed with 0b10000000 = $80

.equ MovFwd = ($80|1<<(EngDirR-1)|1<<(EngDirL-1)) ;0b10110000 Move Forward Action Code

.equ MovBck = ($80|$00) ;0b10000000 Move Backward Action Code

.equ TurnR = ($80|1<<(EngDirL-1)) ;0b10100000 Turn Right Action Code

.equ TurnL = ($80|1<<(EngDirR-1)) ;0b10010000 Turn Left Action Code

.equ Halt = ($80|1<<(EngEnR-1)|1<<(EngEnL-1)) ;0b11001000 Halt Action Code

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

;\* 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

ldi mpr, $04;PD3 output 0000 1000

out DDRD, mpr

ldi mpr, $ff

out DDRB, mpr

out PORTB, mpr

ldi mpr, $f7

out PORTD, mpr

;USART1

;Set baudrate at 2400bps

ldi mpr, high(416)

sts UBRR1H, mpr

ldi mpr, low(416)

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

;Other

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

;\* Main Program

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

MAIN:

in mpr, PIND

sbrs mpr, 0

rcall Send\_Forward

sbrs mpr, 1

rcall Send\_Backward

sbrs mpr, 4

rcall Send\_TurnRight

sbrs mpr, 5

rcall Send\_TurnLeft

sbrs mpr, 6

rcall Send\_Halt

sbrs mpr, 7

rcall Send\_Freeze

rjmp MAIN

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

;\* Functions and Subroutines

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

Send\_Address:

;lds mpr, UCSR1A

;sbrs mpr, UDRE1

;rjmp Send\_Address

ldi mpr, $2A ; Robot Address

sts UDR1, mpr

ret

Send\_Forward:

rcall Send\_Address

; lds mpr, UCSR1A

; sbrs mpr, UDRE1

; rjmp Send\_Forward

ldi mpr, $b0

sts UDR1, mpr

ret

Send\_Backward:

rcall Send\_Address

;lds mpr, UCSR1A

;sbrs mpr, UDRE1

;rjmp Send\_Backward

ldi mpr, $80

sts UDR1, mpr

ret

Send\_TurnRight:

rcall Send\_Address

;lds mpr, UCSR1A

;sbrs mpr, UDRE1

;rjmp Send\_TurnRight

ldi mpr, $a0

sts UDR1, mpr

ret

Send\_TurnLeft:

rcall Send\_Address

;lds mpr, UCSR1A

;sbrs mpr, UDRE1

;rjmp Send\_TurnLeft

ldi mpr, $90

sts UDR1, mpr

ret

Send\_Halt:

lds mpr, UCSR1A

sbrs mpr, UDRE1

rjmp Send\_Halt

ldi mpr, $c8

sts UDR1, mpr

ret

Send\_Freeze:

rcall Send\_Address

;lds mpr, UCSR1A

;sbrs mpr, 5

;rjmp Send\_Freeze

ldi mpr, $f8

sts UDR1, mpr

ret

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

;\* Stored Program Data

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

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

;\* Additional Program Includes

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

2)Receiver

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

;\*

;\* 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 mmpr = r17 ; Multi-Purpose Register 2

.def Fcnt = r18 ; Freeze count register

.def waitcnt = r19 ; Wait Loop Counter

.def ilcnt = r20 ; Inner Loop Counter

.def olcnt = r21 ; Outer Loop Counter

.def mmmpr = r22

.def Flag = r23

.def remember = r24

.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 = $2A;(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 $0002

rcall HitRight

reti

.org $0004

rcall HitLeft

reti

.org $003C

rcall Receive\_Data

reti

;Should have Interrupt vectors for:

;- Left whisker

;- Right whisker

;- USART receive

.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

out DDRB, mpr

ldi mpr, $04 ; 0000 1000

out DDRD, mpr

ldi mpr, $03 ; 0000 0011

out PORTD, mpr

;USART1

;Set baudrate at 2400bps

ldi mpr, high(416)

sts UBRR1H, mpr

ldi mpr, low(416)

sts UBRR1L, mpr

;Enable receiver and enable receive interrupts

ldi mpr, (1 << RXEN1 | 1 << RXCIE1 | 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

;External Interrupts

;Set the External Interrupt Mask

;Set the Interrupt Sense Control to falling edge detection

ldi mpr, $0A

sts EICRA,mpr

ldi mpr, $03

out EIMSK, mpr

sei

;Other

ldi Fcnt, 3

clr Flag

clr remember

ldi mpr, (1 << 5| 1 << 6)

out PORTB, mpr

ldi remember, 1

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

;\* Main Program

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

MAIN:

;TODO: Maintain the previous state of the external interruption

rcall Memory

rjmp MAIN

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

;\* Functions and Subroutines

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

;External Interruptions

HitRight:

push mpr ; Save mpr register

push waitcnt ; Save wait register

in mpr, SREG ; Save program state

push mpr ;

; Move Backwards for a second

ldi mpr, MovBck ; Load Move Backward command

out PORTB, mpr ; Send command to port

ldi waitcnt, WTime ; Wait for 1 second

rcall Wait ; Call wait function

; Turn left for a second

ldi mpr, TurnL ; Load Turn Left Command

out PORTB, mpr ; Send command to port

ldi waitcnt, WTime ; Wait for 1 second

rcall Wait ; Call wait function

; Move Forward again

ldi mpr, MovFwd ; Load Move Forward command

out PORTB, mpr ; Send command to port

ldi waitcnt, WTime ; Wait for 1 second

rcall Wait ; Call wait function

;Clear INT0, INT1 interruption

ldi mpr, $03

out EIFR, mpr

;Clear RXC flag to clear interrupt queue

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

sts UCSR1B, mpr

pop mpr ; Restore program state

out SREG, mpr ;

pop waitcnt ; Restore wait register

pop mpr ; Restore mpr

out PORTB, 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, 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 waitcnt, WTime ; Wait for 1 second

rcall Wait ; Call wait function

;Clear INT0, INT1 interruption

ldi mpr, $03

out EIFR, mpr

;Clear RXC flag to clear interrupt queue

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

sts UCSR1B, mpr

pop mpr ; Restore program state

out SREG, mpr ;

pop waitcnt ; Restore wait register

pop mpr ; Restore mpr

ret ; Return from subroutine

Wait:

push waitcnt ; Save wait register

push ilcnt ; Save ilcnt register

push olcnt ; Save olcnt register

Loop: ldi olcnt, 224 ; load olcnt register

OLoop: ldi ilcnt, 237 ; load ilcnt register

ILoop: dec ilcnt ; decrement ilcnt

brne ILoop ; Continue Inner Loop

dec olcnt ; decrement olcnt

brne OLoop ; Continue Outer Loop

dec waitcnt ; Decrement wait

brne Loop ; Continue Wait loop

pop olcnt ; Restore olcnt register

pop ilcnt ; Restore ilcnt register

pop waitcnt ; Restore wait register

ret ; Return from subroutine

;USART Receive Interruption

Receive\_Data:

lds mpr, UDR1

;Check the robot Address

cpi mpr, $2A

brne SetFlag

Commands:

;Check the command

lds mpr, UDR1

;Move Forward

cpi mpr, $b0

breq MoveForward

;Move Backward

cpi mpr, $80

breq MoveBackward

;Turn Right

cpi mpr, $a0

breq TurnRight

; Turn Left

cpi mpr, $90

breq TurnLeft

; Halt

cpi mpr, $c8

breq Halt\_2

;Freeze

cpi mpr, $f8

breq Send\_Freeze

cpi mpr, $55

breq Freeze

Exit:

clr Flag

ret

SetFlag:

ldi Flag, $ff

jmp Commands

MoveForward:

cpi Flag, $ff

breq Exit

ldi mpr, (1 << 5 | 1 << 6)

out PORTB, mpr

clr remember

ldi remember, 1

rjmp Exit

MoveBackward:

cpi Flag, $ff

breq Exit

ldi mpr, $00

out PORTB, mpr

clr remember

ldi remember, 2

rjmp Exit

TurnRight:

cpi Flag, $ff

breq Exit

ldi mpr, (1 << 6)

out PORTB, mpr

clr remember

ldi remember, 4

rjmp Exit

TurnLeft:

cpi Flag, $ff

breq Exit

ldi mpr, (1 << 5)

out PORTB, mpr

clr remember

ldi remember, 8

rjmp Exit

Halt\_2:

cpi Flag, $ff

breq Exit

ldi mpr, (1 << 4 | 1 << 7)

out PORTB, mpr

clr remember

ldi remember, 16 ; 0000 0000

rjmp Exit

Send\_Freeze:

cpi Flag, $ff

breq Exit

;ldi mpr, $2B ; Robot Address

;sts UDR1, mpr

ldi mpr, $55

sts UDR1, mpr

rjmp Exit

Freeze:

;Check it is not the command for itself

;If the address is same, then Flag will be still cleared.

cpi Flag, $00

breq Exit

dec Fcnt

breq Forever

ldi mpr, (1 << 4 | 1<< 5 | 1 << 6| 1 << 7)

out PORTB, mpr

;Clear INT0, INT1 interruption

ldi mpr, $03

out EIFR, mpr

ldi mmpr, 3

ThreeSeconds:

ldi waitcnt, WTime ; Wait for 1 second

rcall Wait ; Call wait function

dec mmpr

;Clear INT0, INT1 interruption

ldi mpr, $03

out EIFR, mpr

brne ThreeSeconds

rjmp Exit

Forever:

ldi mpr, (1 << 4 | 1<< 5 | 1 << 6| 1 << 7)

out PORTB, mpr

;Clear INT0, INT1 interruption

ldi mpr, $03

out EIFR, mpr

;Clear RXC flag to clear interrupt queue

ldi mpr, $80

sts UCSR1B, mpr

rjmp Forever

Memory:

sbrc remember, 0

rcall MoveForward

sbrc remember, 1

rcall MoveBackward

sbrc remember, 2

rcall TurnRight

sbrc remember, 3

rcall TurnLeft

sbrc remember, 4

rcall Halt\_2

ret

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

;\* Stored Program Data

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

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

;\* Additional Program Includes

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