Sprints

OBSTACLE AVOIDING ROBOT

Table of Contents

Sprints

Obstacle Avoidance Robot V1.0 Design

Assigned By: Mohamed Adel Abdel-Salam

Project Description:

Hardware Requirements

1.1. ATmega32 microcontroller
1.2. Four motors (M1, M2, M3, M4)
1.3. One button to change default direction of rotation (PBUTTON0)
1.4. Keypad button 1 to start
1.5. Keypad button 2 to stop
1.6. One Ultrasonic sensor connected as follows
  1.6.1. Vcc to 5V in the Board
  1.6.2. GND to the ground In the Board
  1.6.3. Trig to PB3 (Port B, Pin 3)
  1.6.4. Echo to PB2 (Port B, Pin 2)
1.7. LCD

Software Requirements

1. The car starts initially from 0 speed
2. The default rotation direction is to the right
3. Press (Keypad Btn 1), (Keypad Btn 2) to start or stop the robot respectively
4. After Pressing Start:
    1. The LCD will display a centered message in line 1 “Set Def. Rot.”
    2.The LCD will display the selected option in line 2 “Right”
    3. The robot will wait for 5 seconds to choose between Right and Left
        1. When PBUTTON0 is pressed once, the default rotation will be Left and the LCD line 2 will be updated
        2. When PBUTTON0 is pressed again, the default rotation will be Right and the LCD line 2 will be updated

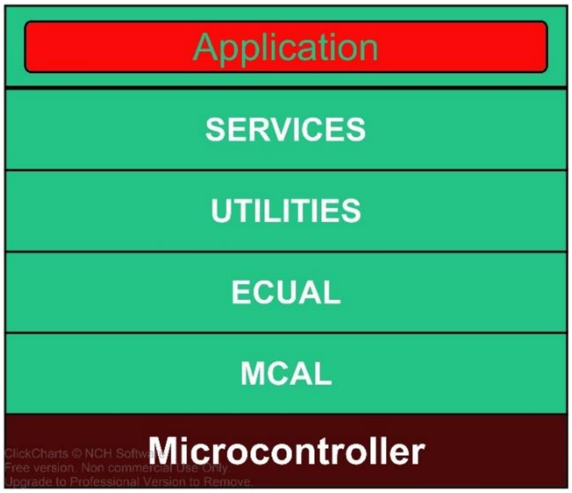
   3. For each press the default rotation will changed and the LCD line 2 is updated
   4. After the 5 seconds the default value of rotation is set

4. The robot will move after 2 seconds from setting the default direction of rotation.
5. For No obstacles or object is far than 70 centimeters:
   1. The robot will move forward with 30% speed for 5 seconds
   2. After 5 seconds it will move with 50% speed as long as there was no object or objects are located at more than 70 centimeters distance
   3. The LCD will display the speed and moving direction in line 1: “Speed:00% Dir: F/B/R/S”, F: forward, B: Backwards, R: Rotating, and S: Stopped
   4. The LCD will display Object distance in line 2 “Dist.: 000 Cm”
6. For Obstacles located between 30 and 70 centimeters
   1. The robot will decrease its speed to 30%
   2. LCD data is updated
7. For Obstacles located between 20 and 30 centimeters
   1. The robot will stop and rotates 90 degrees to right/left according to the chosen configuration
   2. The LCD data is updated
8. For Obstacles located less than 20 centimeters
   1. The robot will stop, move backwards with 30% speed until distance is greater than 20 and less than 30
   2. The LCD data is updated
   3. Then preform point 7

High level design

Layered architecture:

1- Microcontroller
2- MCAL
3- ECUAL
4- UTILITIES
5- SERVICES


Application
SERVICES
UTILITIES
ECUAL
MCAL
Microcontroller
ClickCharts © NCH Software
Free version. Non commercial Use Only
Upgrade to Professional Version to Remove

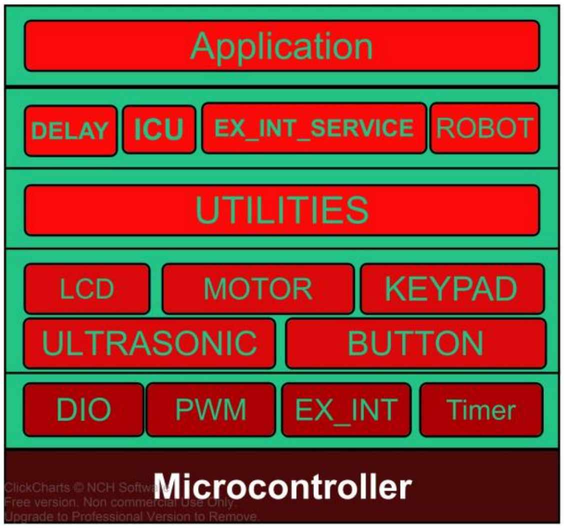
Modules descriptions:

1- Specify system modules/drivers:

- DIO, TIMER, EX_INT,PWM,
- LCD, KEYPAD, BUTTON, ULTRASONIC, MOTOR
- DELAY, ICU, EX_INT_SERVICE, ROBOT

2- Assign each module to its related layer:

- By drawing



Drivers' documentation APIs:

DIO APIs:

```
void DIO_init (uint8_t portNumber,uint8_t
pinNumber,uint8_t direction);
void DIO_write (uint8_t portNumber,uint8_t
pinNumber,uint8_t value);
void DIO_read (uint8_t portNumber,uint8_t
```

TIMER APIs:

```
void TIMER_init (uint8_t Mode,uint8_t intial_value);
void TIMER_start (uint8_t prescaler_value);
void TIMER_set(uint8_t intial_value);
void TIMER_getStatus(uint8_t *value);
void TIMER_Stop (void);
```

EX_INT APIs:

```
void INT_VECT(void) __attribute__ ((signal,used));
void SIE(void);
void CLI(void);
void INT_SENSE(uint8_t inerrupt_number,uint8_t sense);
void EX_INT_Enable(uint8_t inerrupt_number);
void EX_INT_Disable(uint8_t inerrupt_number);
void EX_INT0_SET_CALLBACK (void (*copyFuncptr) (void));
void EX_INT0_SET_CALLBACK (void (*copyFuncptr) (void));
void EX_INT_init(uint8_t interrupt , uint8_t sense);
```

PWM APIs:

```
void PWM_Stop (void);
void PWM_init (void);
void PWM_start (uint8_t duty_percent);
void PWM_set (uint8_t duty_percent , uint8_t blinks);
```

LCD APIs:

```
void LCD_init (void);
void LCD _sendcommand (uint8_t cmnd);
void LCD_sendchar (uint8_t char_data);
void LCD _sendstring(uint8_t *str);
void LCD_setcursor (uint8_t row, uint8_t column);
void LCD_clear (void);
void LCD_customchar(uint8_t *pattern, uint8_t location);
LCD_floattostring (f32_t float_value);
```

KEYPAD APIs:

```
void KEYPAD_init (uint8_t port);
uint8_t KEYPAD_getkey (void);
```

BUTTON APIs

```
void BUTTON_init (uint8_t buttonport, uint8_t buttonpin);
void BUTTON_read (uint8_t buttonport, uint8_t buttonpin, uint8_t *value);
```

ULTRASONIC APIs:

```
void ULTRASONIC_init (void);
u16 ULTRASONIC _read (void);
```

MOTOR APIs:

```
void MOTOR_init (u8 motorA,u8 motorB);
void MOTOR_writeCw (u8 motorA,u8 motorB);
void MOTOR_writeCcw (u8 motorA,u8 motorB);
void MOTOR_stop (u8 motorA,u8 motorB);
```

DELAY APIs:

```
void delay_start_ms (uint32_t ms);
void delay_start_us (uint32_t us);
void Delay_ms (uint8_t milliseconds);
```

ICU APIs:

```
void ICU_rising_ISR (void);
void ICU_falling_ISR (void);
u16 ICU_timeNow (void);
void ICU_init (u8 interrupt);
```

ROBOT APIs:

```
void ROBOT_init (void);                void ROBOT_left (void);
void ROBOT_forward (void);
void ROBOT_right (void);
void ROBOT_backward (void);            void ROBOT_stop (void);
```

EX_INT_SERVICE

```
#  define ISR(vector,...)            \
void vector (void)    __attribute__
((signal,used))__VA_ARGS__ ; \
void vector (void)
```

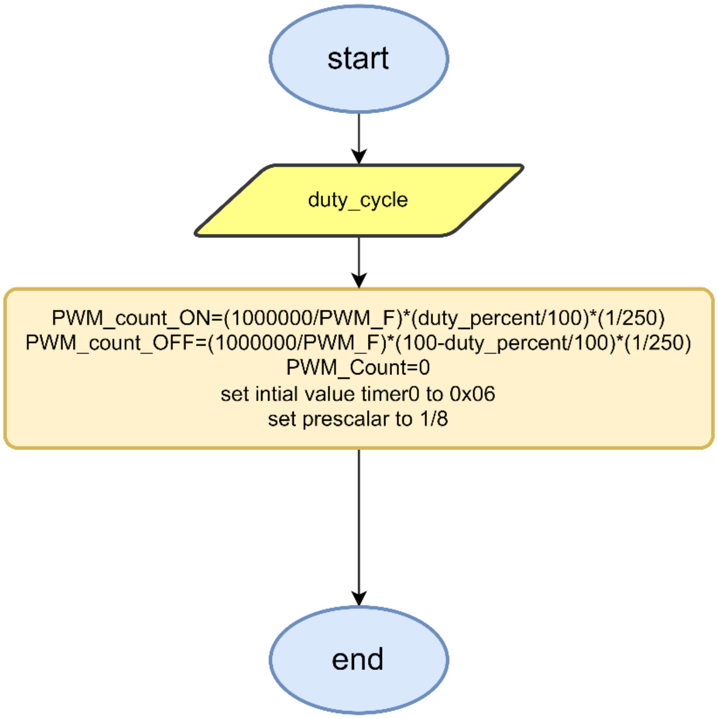
APPLICATION APIs:

```
void APP_init(void);
void APP_start(void);
void APP_stop(void);
```

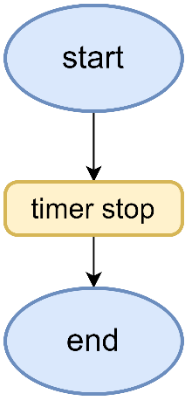
Low level design

Flowcharts APIs:

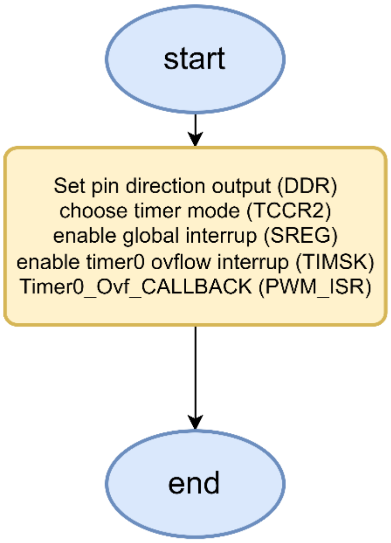
void PWM_start (u8 duty_cycle)


start
duty_cycle
PWM_count_ON=(1000000/PWM_F)*(duty_percent/100)*(1/250)
PWM_count_OFF=(1000000/PWM_F)*(100-duty_percent/100)*(1/250)
PWM_Count=0
set intial value timer0 to 0x06
set prescalar to 1/8
end

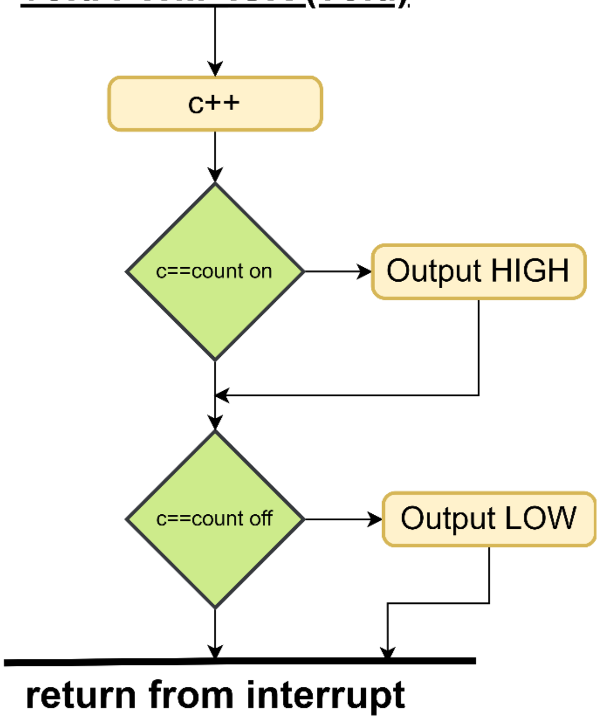
void PWM_stop (void)



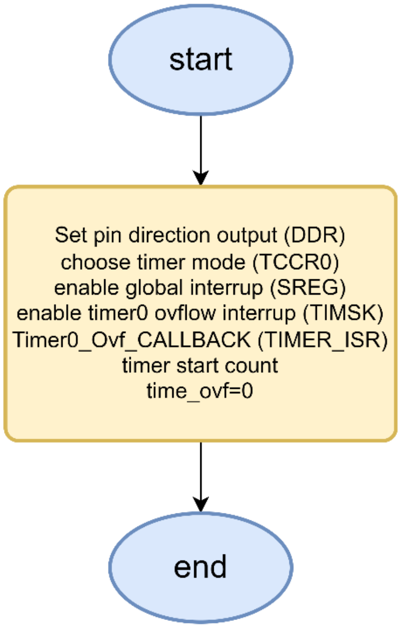
void PWM_init (void)


start
Set pin direction output (DDR)
choose timer mode (TCCR2)
enable global interrupt (SREG)
enable timer0 ovflow interrupt (TIMSK)
Timer0_Ovf_CALLBACK (PWM_ISR)
end

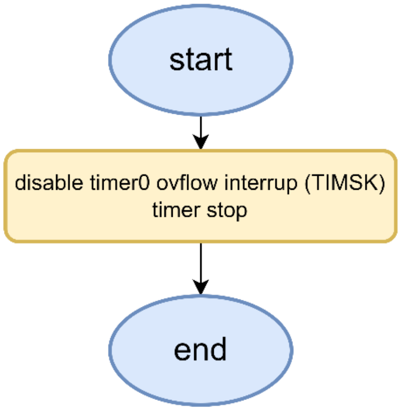
void PWM_ISR (void)



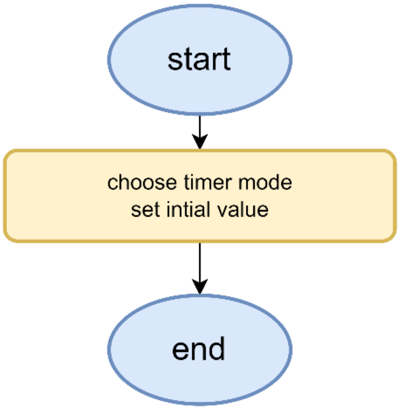
void TIMER_start (u8 ms)


start
Set pin direction output (DDR)
choose timer mode (TCCR0)
enable global interrupp (SREG)
enable timer0 ovflow interrup (TIMSK)
Timer0_Ovf_CALLBACK (TIMER_ISR)
timer start count
time_ovf=0
end

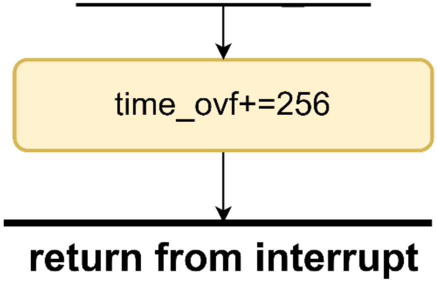
void TIMER_stop (void)



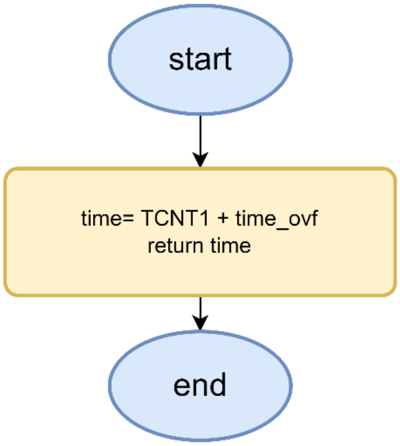
void TIMER_init (u8 mode,u8 intial_value)

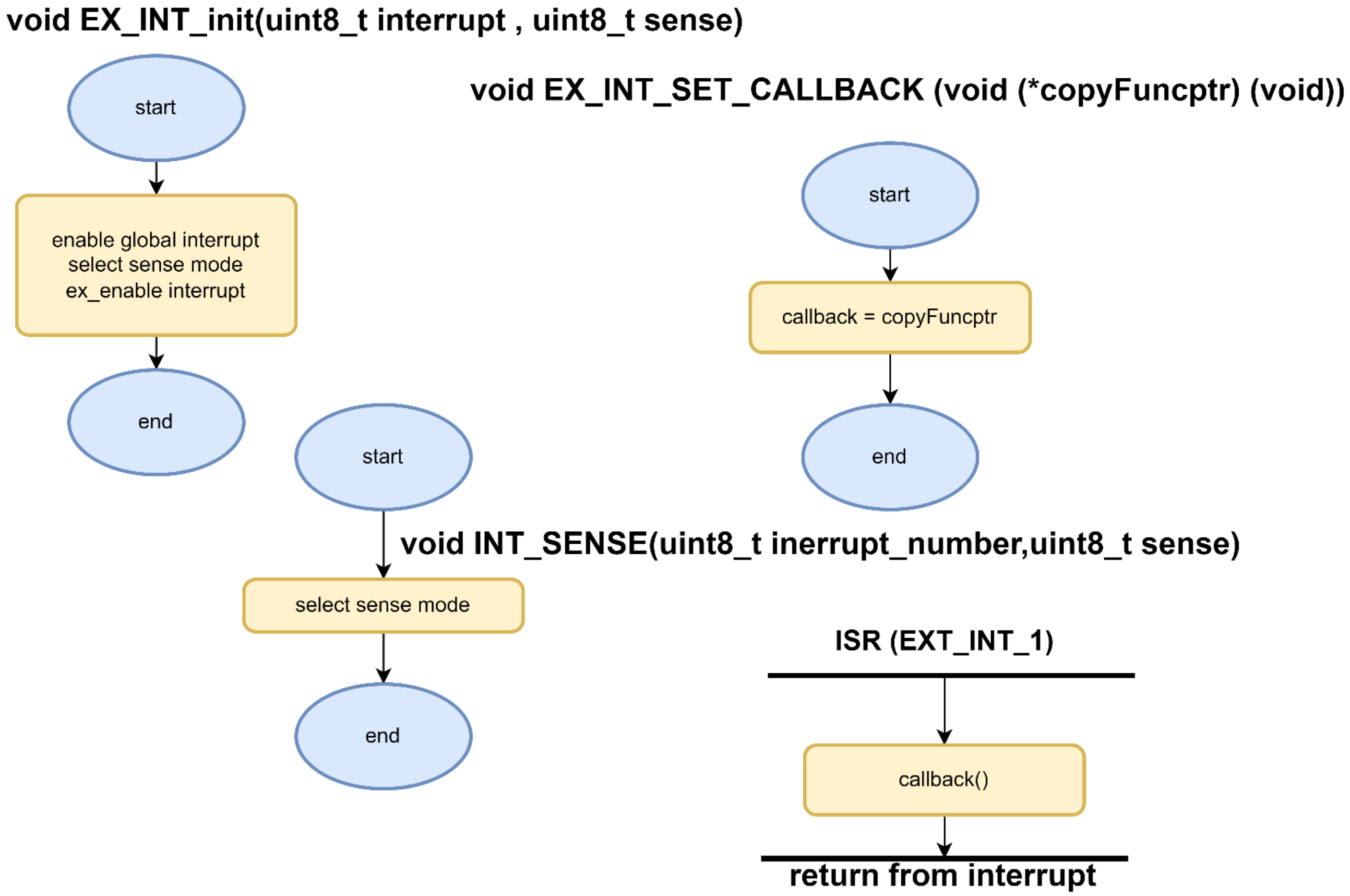


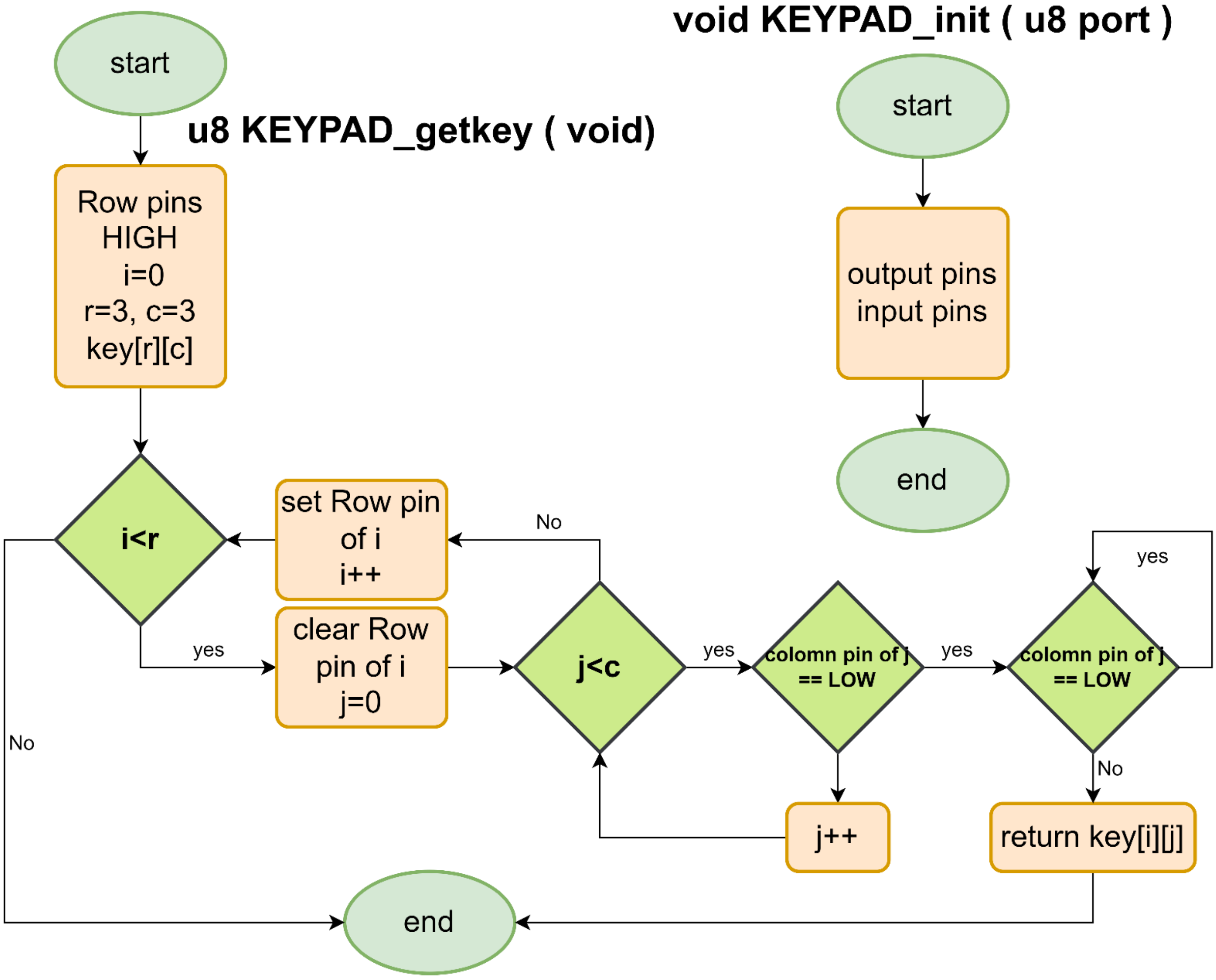
TIMEROVF_ISR



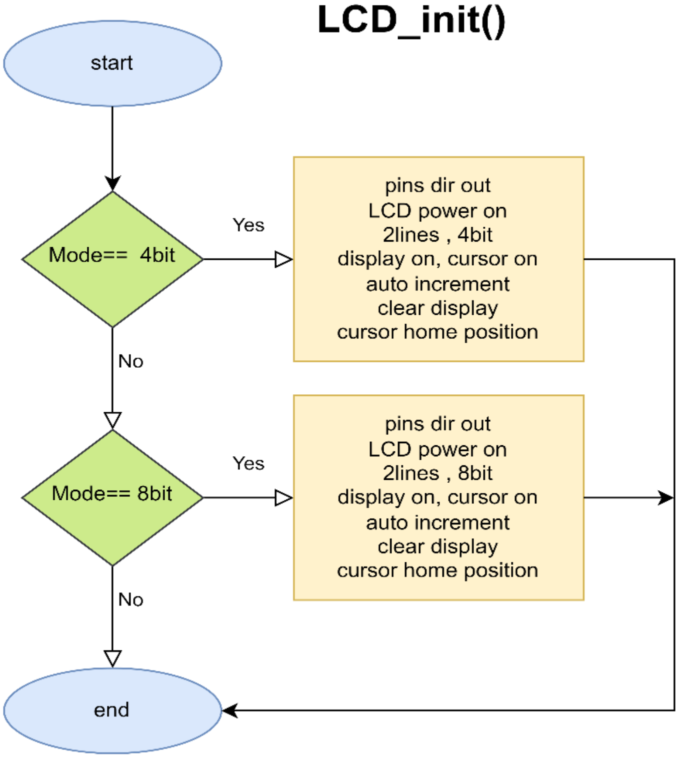
u16 TIMER_get (void)



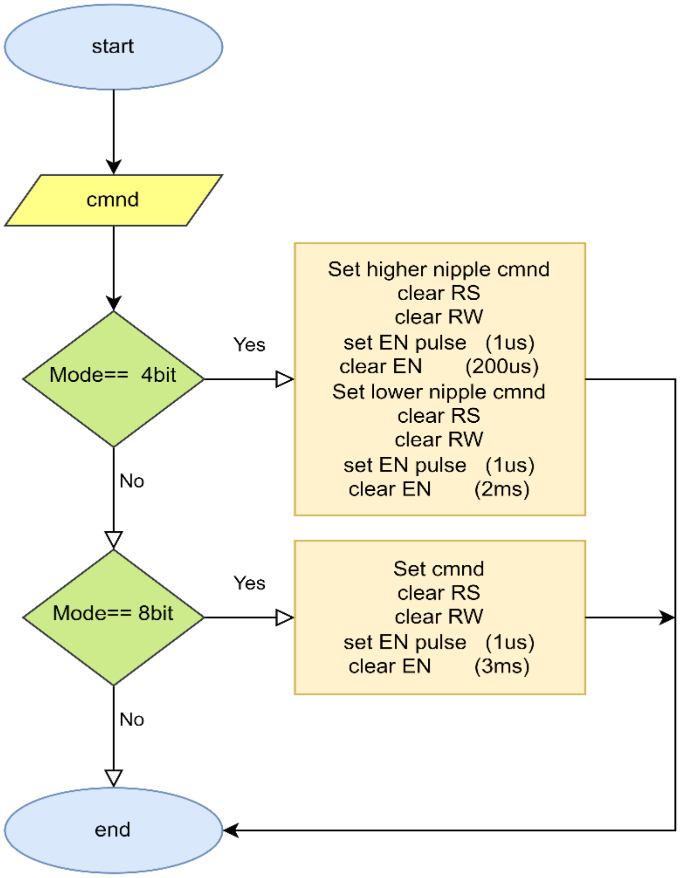

void EX_INT_init(uint8_t interrupt , uint8_t sense)
start
enable global interrupt
select sense mode
ex_enable interrupt
end
void EX_INT_SET_CALLBACK (void (*copyFuncptr) (void))
start
callback = copyFuncptr
end
start
void INT_SENSE(uint8_t inerrupt_number,uint8_t sense)
select sense mode
end
ISR (EXT_INT_1)
callback()
return from interrupt


void KEYPAD_init (u8 port)
start
output pins
input pins
end
start
u8 KEYPAD_getkey (void)
Row pins
HIGH
i=0
r=3, c=3
key[r][c]
i<r
set Row pin
of i
i++
No
clear Row
pin of i
j=0
yes
j<c
yes
colomn pin of j
== LOW
yes
colomn pin of j
== LOW
yes
No
j++
No
return key[i][j]
end

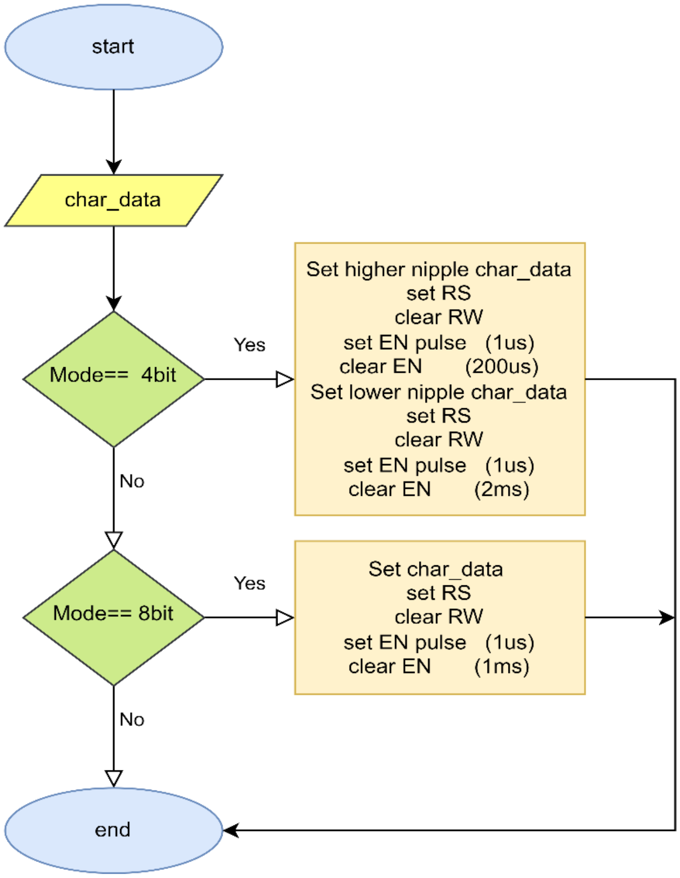
LCD_init()

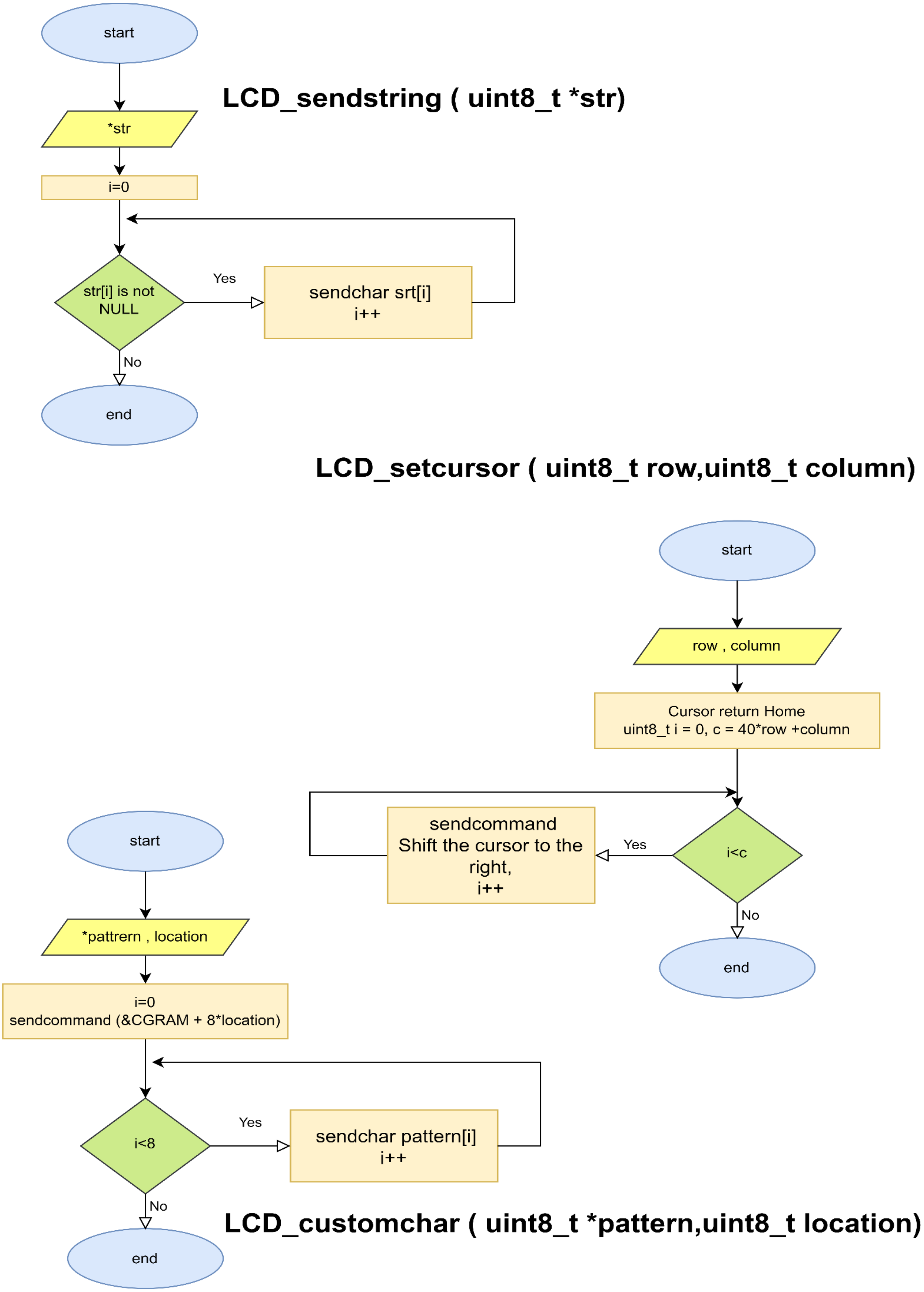


LCD_sendcommand(uint8_t cmnd)



LCD_sendchar (uint8_t char_data)





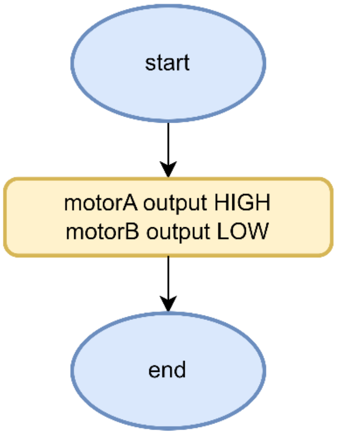
LCD_sendstring (uint8_t *str)

LCD_setcursor (uint8_t row,uint8_t column)

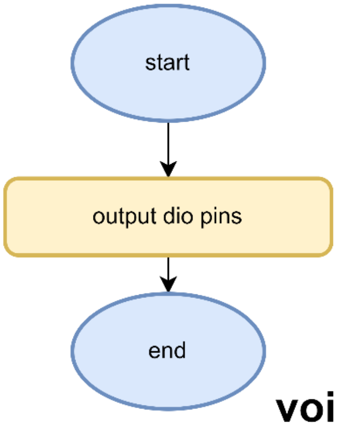
start
row , column
Cursor return Home
uint8_t i = 0, c = 40*row +column
i<c
Yes
sendcommand
Shift the cursor to the right,
i++
No
end

LCD_customchar (uint8_t *pattern,uint8_t location)

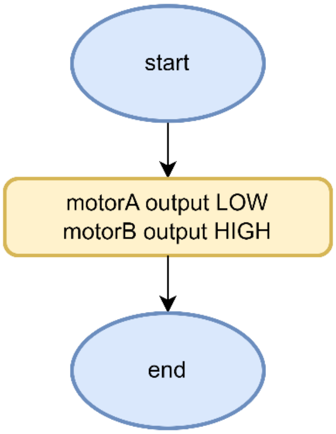
void MOTOR_writeCw (u8 motorA,u8 motorB)



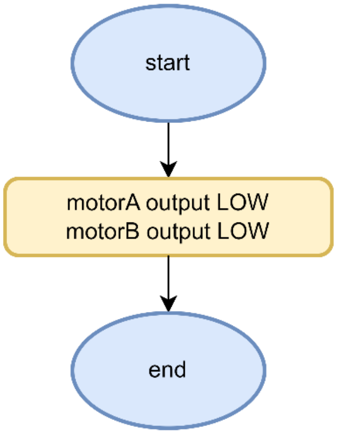
void MOTOR_init (u8 motorA,u8 motorB)



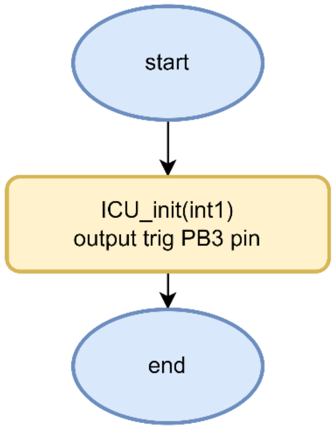
void MOTOR_writeCcw (u8 motorA,u8 motorB)



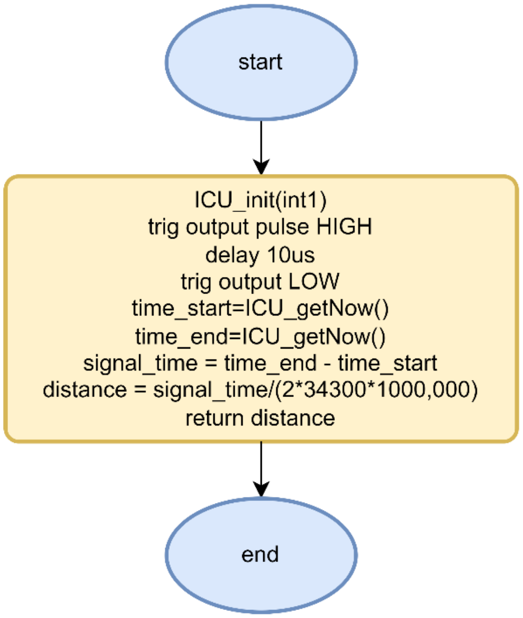
void MOTOR_stop (u8 motorA,u8 motorB)



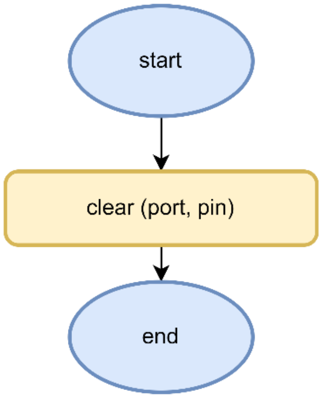
void ULTRASONIC_init (void)



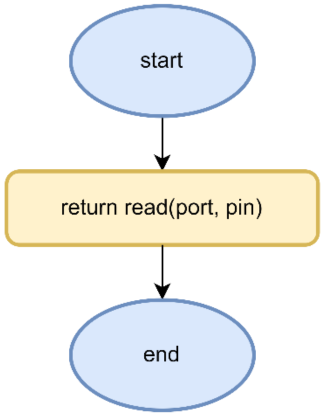
u16 ULTRASONIC_read (void)



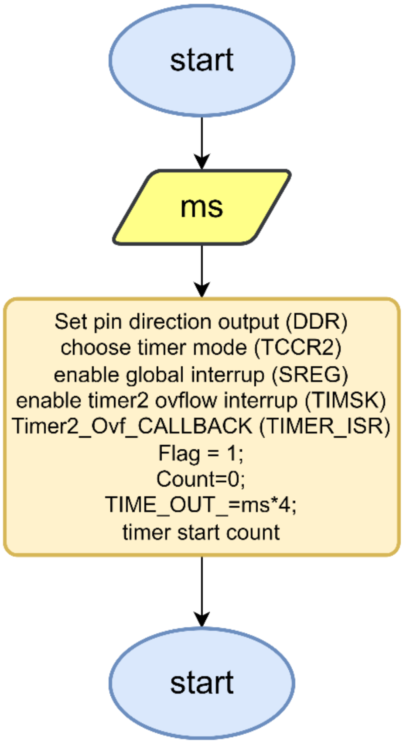
void BUTTON_init (u8 port, u8 pin)



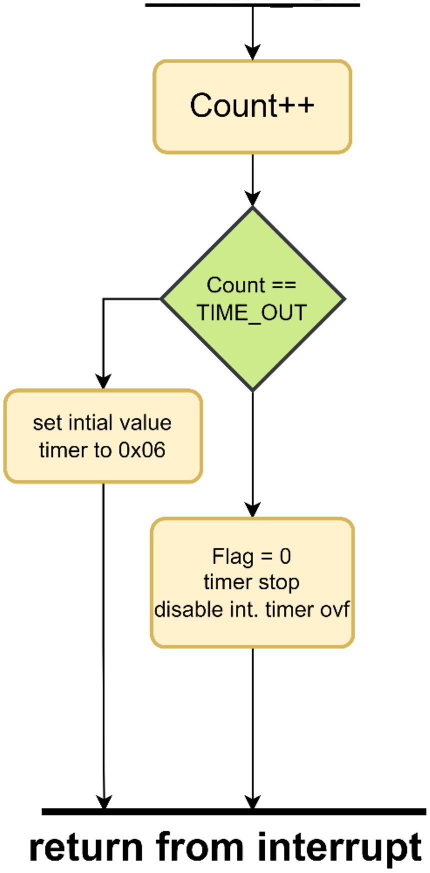
u8 BUTTON_read (u8 port, u8 pin)



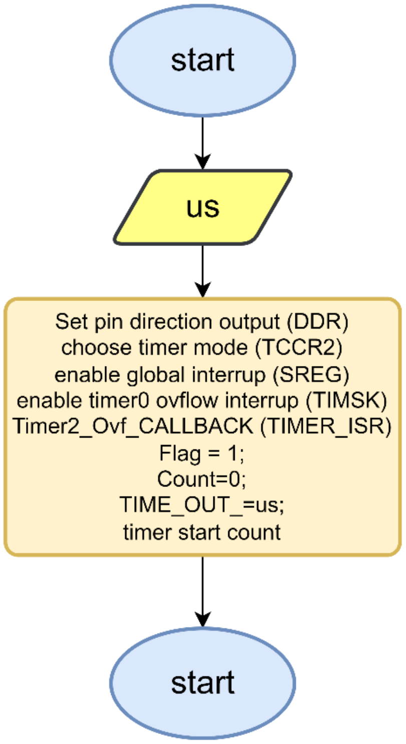
void delay_start_ms (u8 ms)


start
ms
Set pin direction output (DDR)
choose timer mode (TCCR2)
enable global interrupp (SREG)
enable timer2 ovflow interrup (TIMSK)
Timer2_Ovf_CALLBACK (TIMER_ISR)
Flag = 1;
Count=0;
TIME_OUT_=ms*4;
timer start count
start

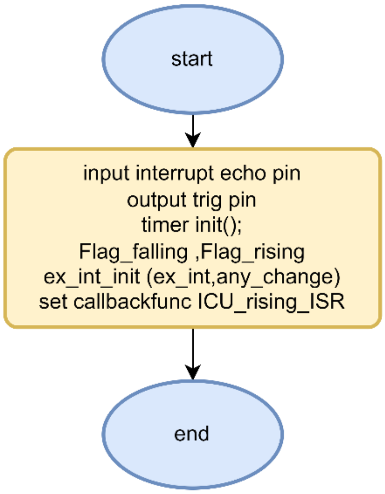
TIMER_ISR



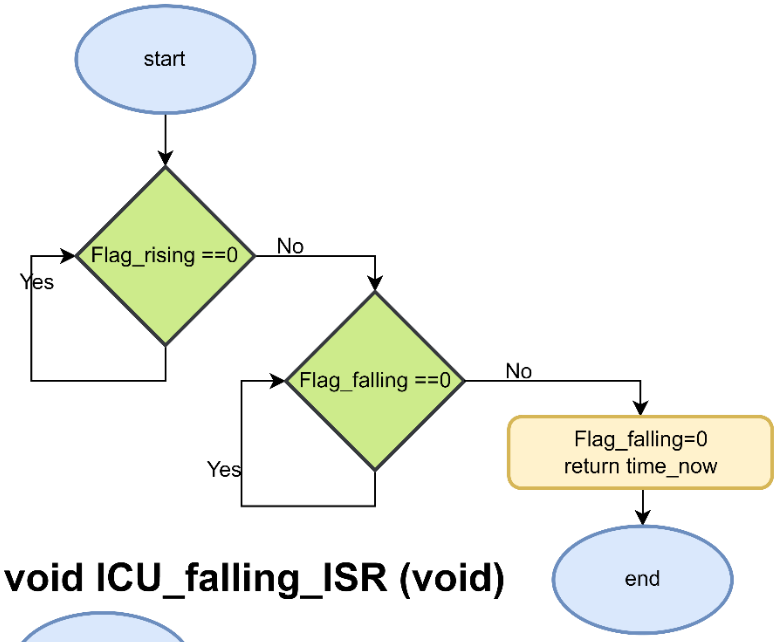
void delay_start_us (u8 us)


start
us
Set pin direction output (DDR)
choose timer mode (TCCR2)
enable global interrupp (SREG)
enable timer0 ovflow interrup (TIMSK)
Timer2_Ovf_CALLBACK (TIMER_ISR)
Flag = 1;
Count=0;
TIME_OUT_=us;
timer start count
start

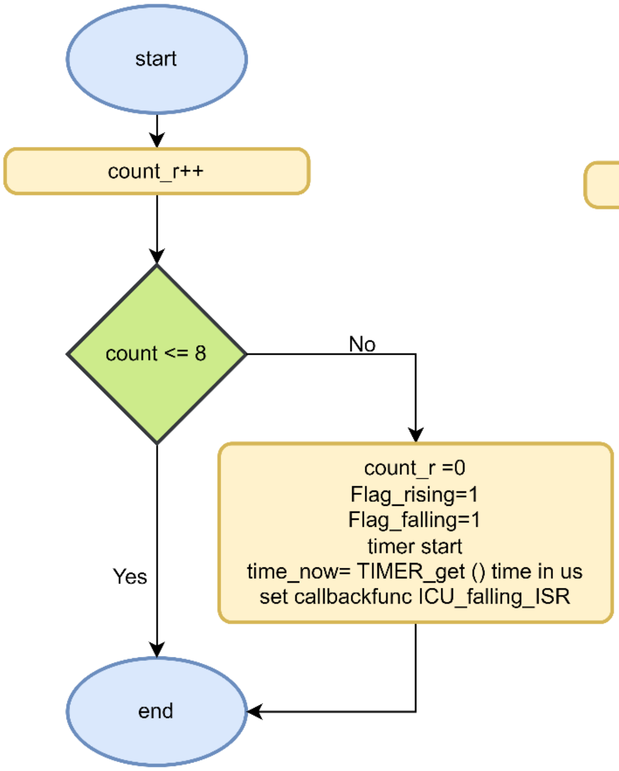
void ICU_init (u8 interrupt,u8 trig_pin)


start
input interrupt echo pin
output trig pin
timer init();
Flag_falling ,Flag_rising
ex_int_init (ex_int,any_change)
set callbackfunc ICU_rising_ISR
end

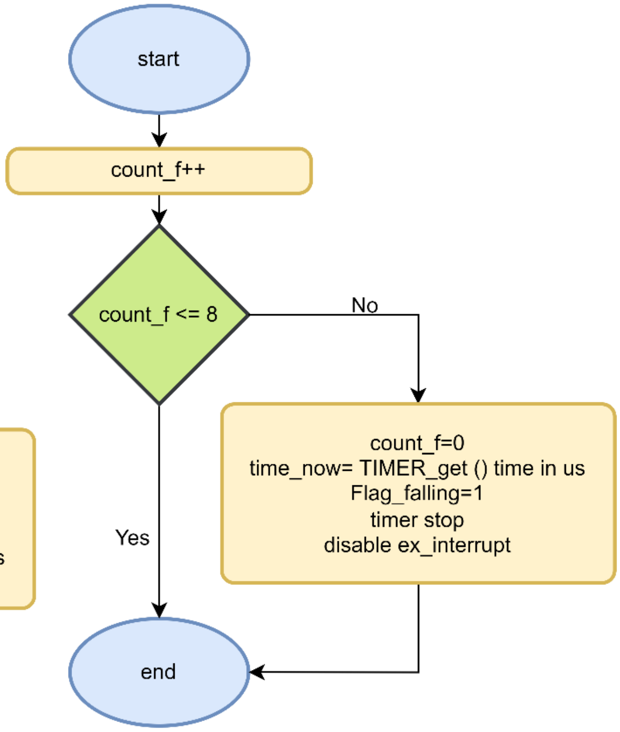
u16 ICU_timeNow (void)



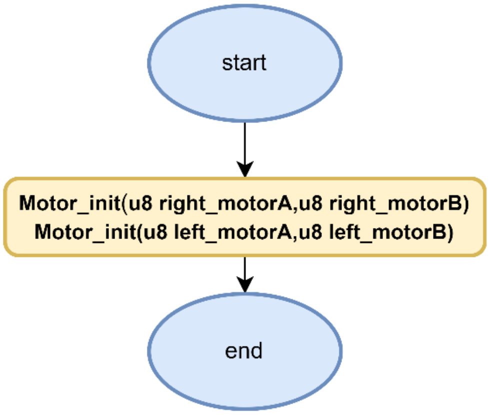
void ICU_rising_ISR (void)



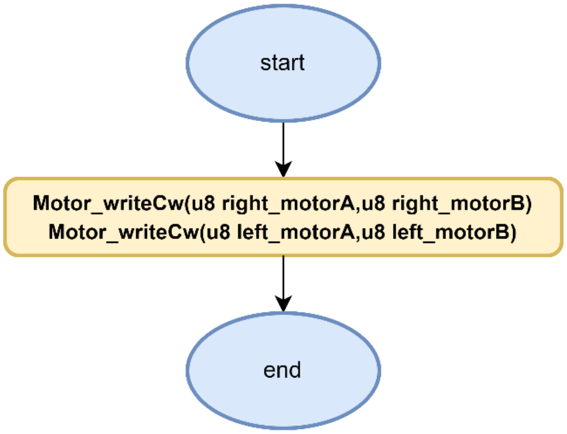
void ICU_falling_ISR (void)



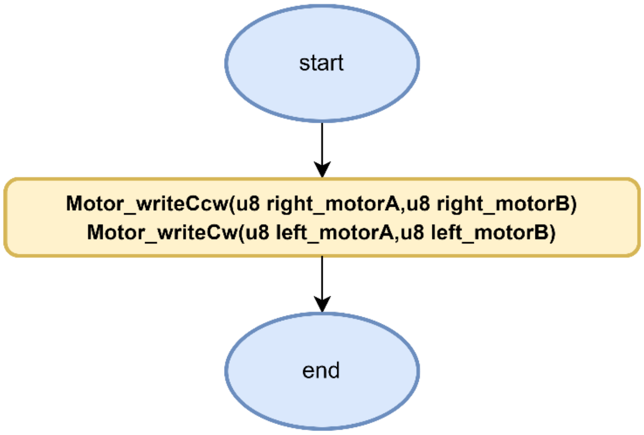
void ROBOT_init (void)


start
Motor_init(u8 right_motorA,u8 right_motorB)
Motor_init(u8 left_motorA,u8 left_motorB)
end

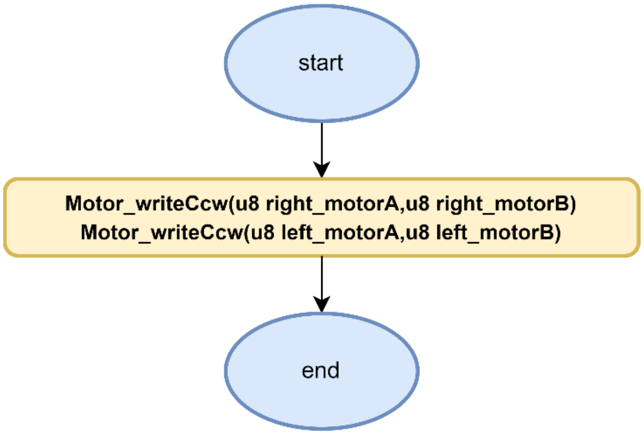
void ROBOT_forward (void)


start
Motor_writeCw(u8 right_motorA,u8 right_motorB)
Motor_writeCw(u8 left_motorA,u8 left_motorB)
end

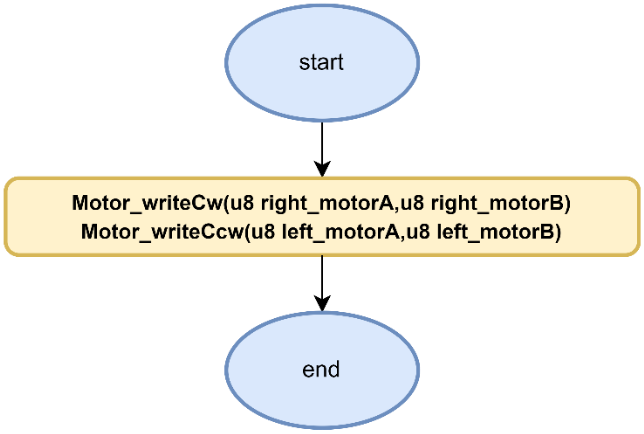
void ROBOT_right (void)


start
Motor_writeCcw(u8 right_motorA,u8 right_motorB)
Motor_writeCw(u8 left_motorA,u8 left_motorB)
end

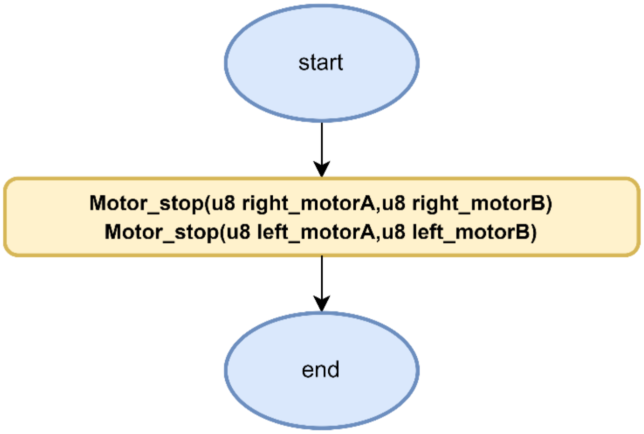
void ROBOT_backward (void)

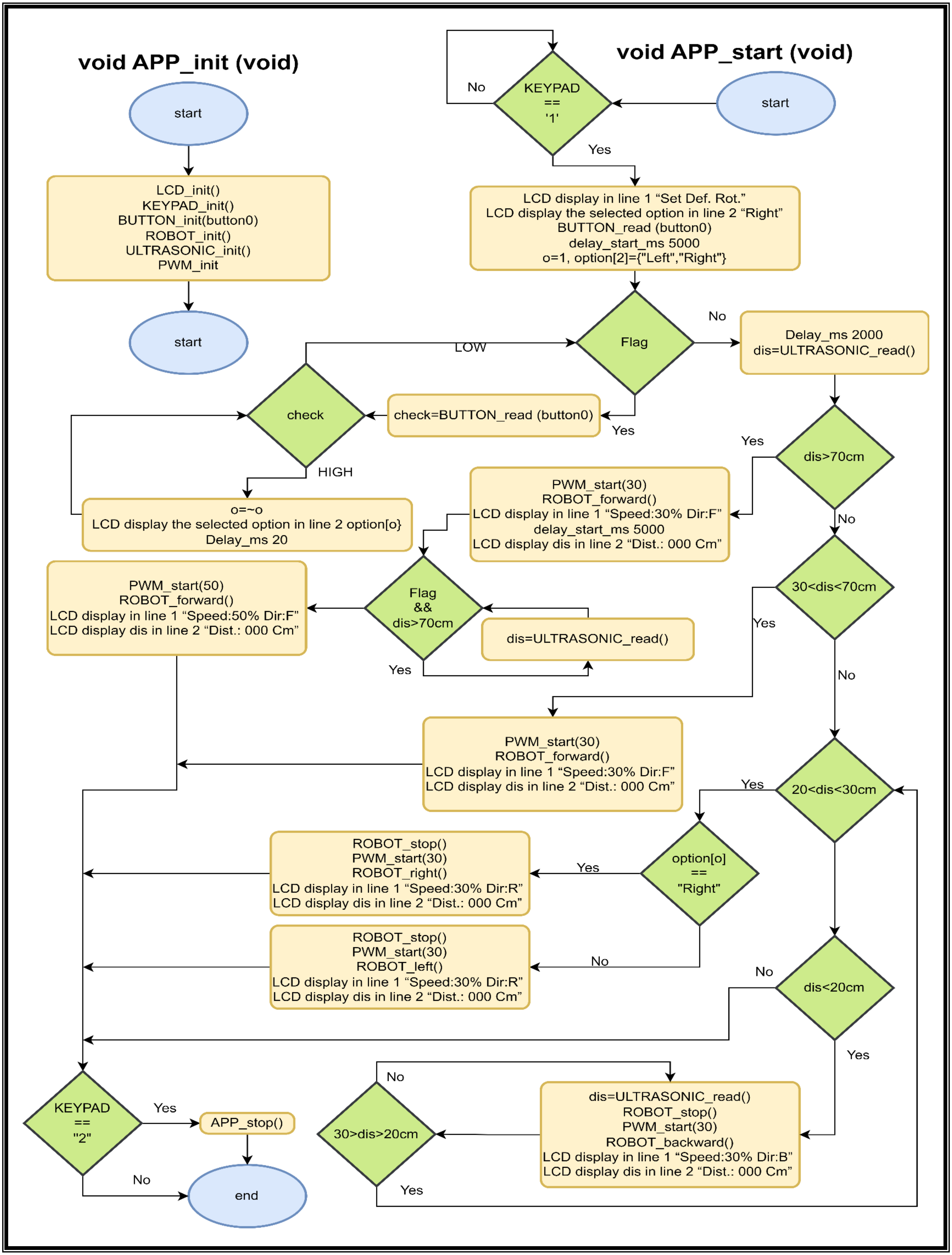

start
Motor_writeCcw(u8 right_motorA,u8 right_motorB)
Motor_writeCcw(u8 left_motorA,u8 left_motorB)
end

void ROBOT_left (void)


start
Motor_writeCw(u8 right_motorA,u8 right_motorB)
Motor_writeCcw(u8 left_motorA,u8 left_motorB)
end

void ROBOT_stop (void)


start
Motor_stop(u8 right_motorA,u8 right_motorB)
Motor_stop(u8 left_motorA,u8 left_motorB)
end


void APP_init (void)
start
LCD_init()
KEYPAD_init()
BUTTON_init(button0)
ROBOT_init()
ULTRASONIC_init()
PWM_init
start
void APP_start (void)
start
KEYPAD
==
'1'
No
Yes
LCD display in line 1 “Set Def. Rot.”
LCD display the selected option in line 2 “Right”
BUTTON_read (button0)
delay_start_ms 5000
o=1, option[2]={"Left","Right"}
Flag
No
Delay_ms 2000
dis=ULTRASONIC_read()
LOW
check
check=BUTTON_read (button0)
Yes
HIGH
o=~o
LCD display the selected option in line 2 option[o]
Delay_ms 20
dis>70cm
Yes
PWM_start(30)
ROBOT_forward()
LCD display in line 1 “Speed:30% Dir:F”
delay_start_ms 5000
LCD display dis in line 2 “Dist.: 000 Cm”
No
PWM_start(50)
ROBOT_forward()
LCD display in line 1 “Speed:50% Dir:F”
LCD display dis in line 2 “Dist.: 000 Cm”
Flag
&&
dis>70cm
dis=ULTRASONIC_read()
Yes
30<dis<70cm
Yes
No
PWM_start(30)
ROBOT_forward()
LCD display in line 1 “Speed:30% Dir:F”
LCD display dis in line 2 “Dist.: 000 Cm”
20<dis<30cm
Yes
option[o]
==
"Right"
Yes
ROBOT_stop()
PWM_start(30)
ROBOT_right()
LCD display in line 1 “Speed:30% Dir:R”
LCD display dis in line 2 “Dist.: 000 Cm”
No
ROBOT_stop()
PWM_start(30)
ROBOT_left()
LCD display in line 1 “Speed:30% Dir:R”
LCD display dis in line 2 “Dist.: 000 Cm”
No
dis<20cm
Yes
No
dis=ULTRASONIC_read()
ROBOT_stop()
PWM_start(30)
ROBOT_backward()
LCD display in line 1 “Speed:30% Dir:B”
LCD display dis in line 2 “Dist.: 000 Cm”
30>dis>20cm
Yes
KEYPAD
==
"2"
Yes
APP_stop()
No
end

Precompiling and linking configurations:

LCD

```
/************************************************************************/
/* LCD Modes                                                                     */
/************************************************************************/
#define bit_4 4
#define bit_8 8
/************************************************************************/
/************************************************************************/
/*  Mode for data transfer Choose ( bit_4 or bit_8 )
*/
/************************************************************************/
#define Mode bit_4
/************************************************************************/
/************************************************************************/
/*                                               Definitions
                     */
/************************************************************************/
/*
                                   4 bits mode definitions
*/
#if Mode == bit_4                                       //if LCD mode chosen in 4bit mode
#define D7 PINA7                                        //Data7 Pin and Port
#define D6 PINA6                                        //Data6 Pin and Port
#define D5 PINA5                                        //Data5 Pin and Port
#define D4 PINA4                                        //Data4 Pin and Port
#define EN PINA3                                        //EN Pin and Port
#define RW PINA2                                        //RW Pin and Port
#define RS PINA1                                        //RS Pin and Port
#define LCD_Data_Port PORTA                      //Data Port
/*
                                   8 bits mode definations
*/
#elif Mode == bit_8                                     //if LCD mode chosen in 8bit mode
#define D7 PINA7                                        //Data7 Pin and Port
#define D6 PINA6                                        //Data6 Pin and Port
#define D5 PINA5                                        //Data5 Pin and Port
#define D4 PINA4                                        //Data4 Pin and Port
#define D3 PINA3                                        //Data3 Pin and Port
#define D2 PINA2                                        //Data2 Pin and Port
#define D1 PINA1                                        //Data1 Pin and Port
#define D0 PINA0                                        //Data0 Pin and Port
#define RS PINB0                                        //RS Pin and Port
#define RW PINB1                                        //RW Pin and Port
#define EN PINB2                                        //EN Pin and Port
#define LCD_Data_Port PA                         //Data Port
#endif
/************************************************************************/
/************************************************************************/
```

KEYPAD

```
#define COL   3
#define ROW   3

#define  NO_KEY  'N'

static u8 KeysArray[ROW][COL]={
       {'1','2','3'},
       {'4','5','6'},
       {'7','8','9'}
       };

/********************************************************/
typedef enum{

       Done,
       Error

}Keypad_Status_en;

/********************************************************/
```

PWM

```
//pwm output pin
#define percent_30 30
#define percent_50 50

//pwm output pin
#define OC0_port 'B'
#define OC0_pin 3
//PWM Modes
#define NonInv_correct_phase 0x60
#define Normal 0x00
//intial values
#define no_clk 0x00
#define zero_intial 0x00
#define Intial_value_PWM 0x06
//prescaler values
#define pres_1 0x01
#define pres_8 0x02
#define pres_64 0x03
#define pres_256 0x04
#define pres_1024 0x05
#define clear_PWM 0x01
```

DIO

```
typedef enum {
        PINA0,
        PINA1,
        PINA2,
        PINA3,
        PINA4,
        PINA5,
        PINA6,
        PINA7,
        PINB0,
        PINB1,
        PINB2,
        PINB3,
        PINB4,
        PINB5,
        PINB6,
        PINB7,
        PINC0,
        PINC1,
        PINC2,
        PINC3,
        PINC4,
        PINC5,
        PINC6,
        PINC7,
        PIND0,
        PIND1,
        PIND2,
        PIND3,
        PIND4,
        PIND5,
        PIND6,
        PIND7,
}PIn_name;

typedef enum{
        OUTPUT,
        INFREE,
        INPUT
}PIN_Status;

typedef enum {
        PA,
        PB,
        PC,
        PD
}PORT_Type;

typedef enum {
        LOW,
        HIGH
}Voltage_type;
```