Moving Car Project

-Team no.2
Peter Essam
Mario Saad
Nadeen Adel

Table Of Content:

1. Introduction:

This application aims to create a specific driving pattern for the car, involving forward movement, turns, and stops, with the ability to stop the car instantly with a certain button.

The application description is as follows:

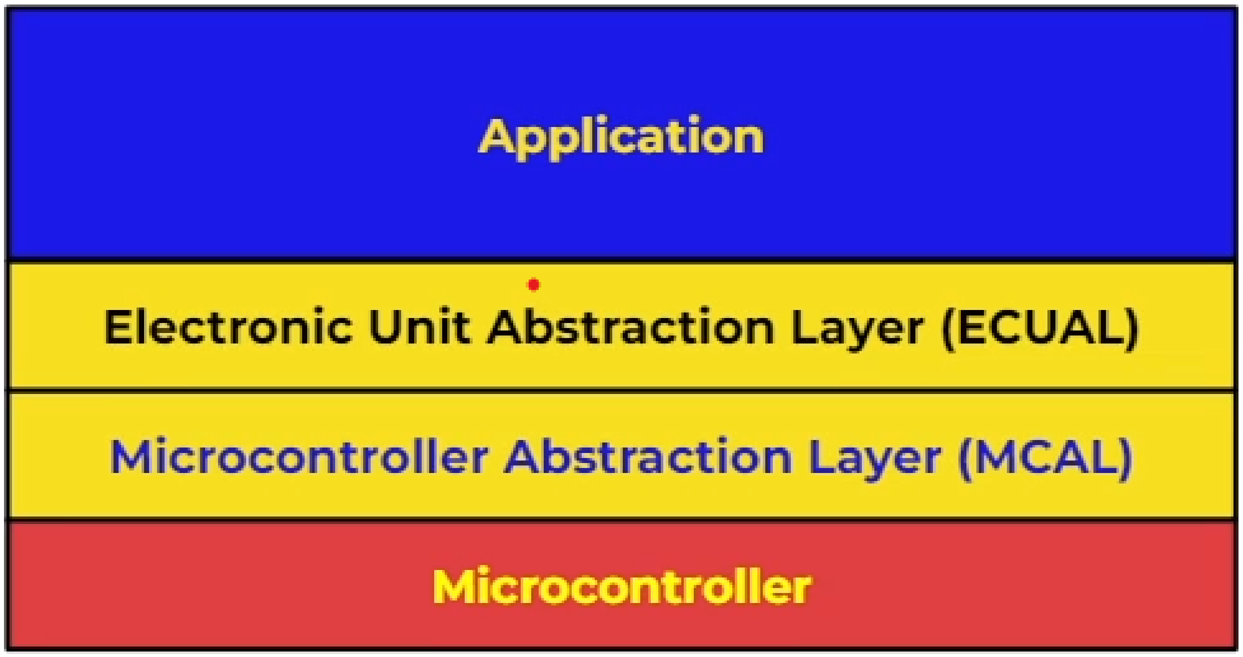
When PB1 is pressed the car will wait for 1 second then move forward for 3 seconds with 50% of its speed to create the longest side of the rectangle. Then it will stop for 0.5 seconds ,rotate 90 degrees to the right then stop again for 0.5 seconds.

After this the car will move forward for 2 seconds with 30% of its speed to create the shortest side,stop for 0.5 seconds,make another 90-degree right turn and stop for another 0.5 seconds.

And then it will repeat the same sequence to create the rectangle shape of movement and continue infinitely until PB2 is pressed and the car stops as this is a sudden break and has the highest priority.

2. High Level Design:

2.1 Layered Architecture:


Application
Electronic Unit Abstraction Layer (ECUAL)
Microcontroller Abstraction Layer (MCAL)
Microcontroller

2.2 Modules Description:

2.2.1 MCAL Layer:

- **DIO:**

  The DIO (digital input output) driver is used to set up pins configuration of the microcontroller whether input or output, pull up or pull down.

- **External Interrupt**
- **Timer**
- **PWM**
- **PWM Normal Mode**

2.2.2 HAL Layer:

- **LED:**

  LED driver is used to set up and control the LEDs of the microcontroller.

- **Button:**

  Button driver is used to get the value on the button pin.

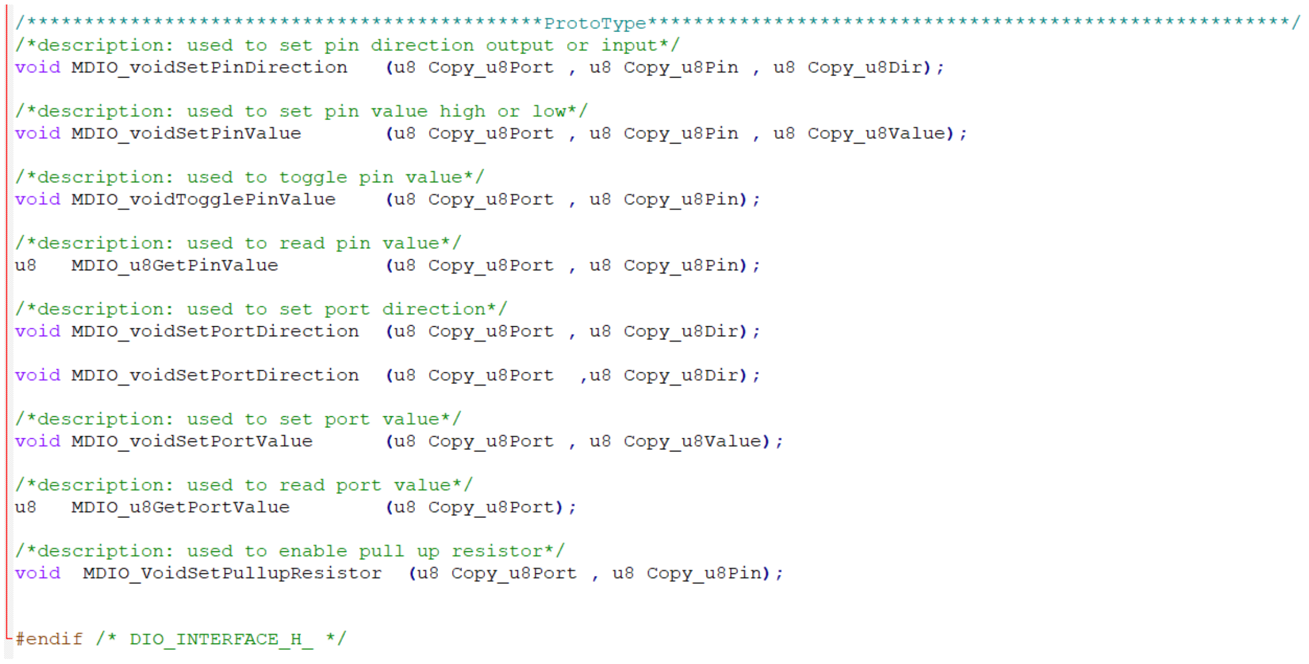
- **Motor**

2.2.3 Application Layer:

Implementation of the code to do the project demands.

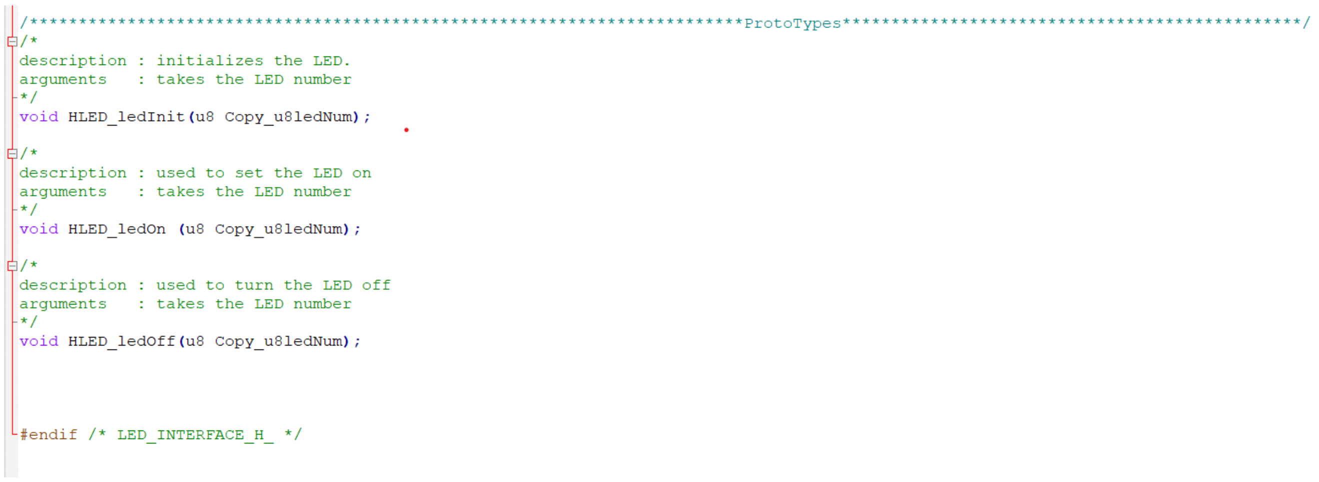
2.3 Drivers Documentation:

2.3.1 DIO:



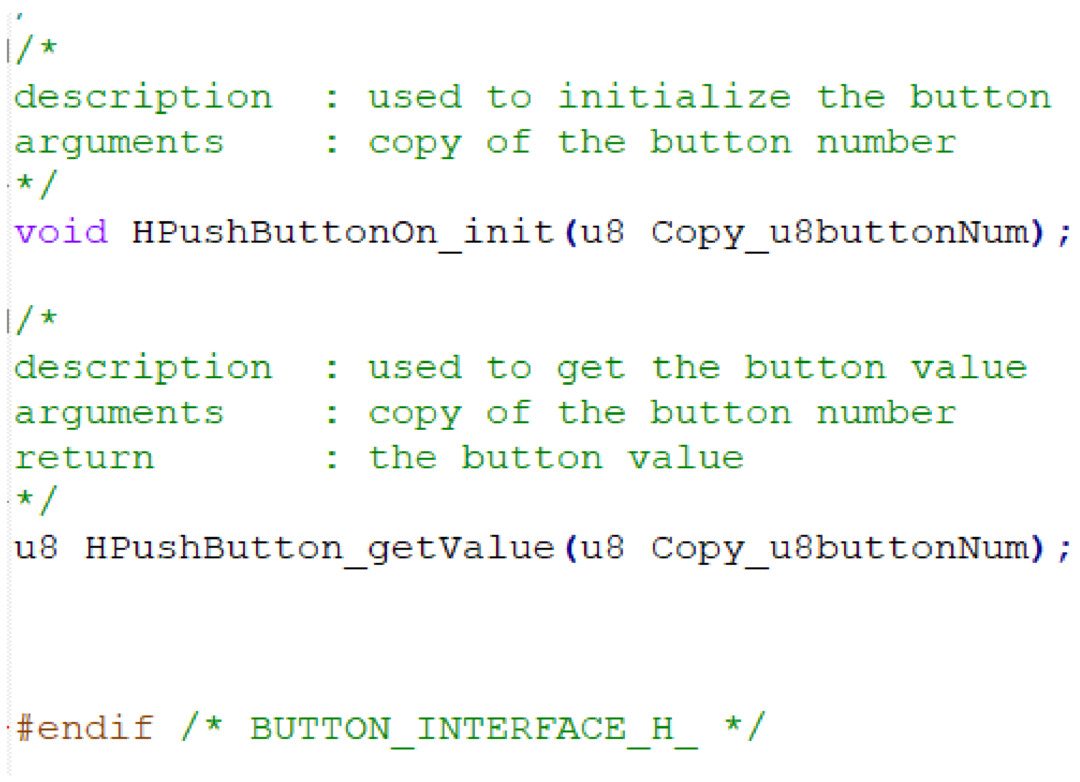
```
/**********************************************ProtoType********************************************************/
/*description: used to set pin direction output or input*/
void MDIO_voidSetPinDirection   (u8 Copy_u8Port , u8 Copy_u8Pin , u8 Copy_u8Dir);

/*description: used to set pin value high or low*/
void MDIO_voidSetPinValue       (u8 Copy_u8Port , u8 Copy_u8Pin , u8 Copy_u8Value);

/*description: used to toggle pin value*/
void MDIO_voidTogglePinValue    (u8 Copy_u8Port , u8 Copy_u8Pin);

/*description: used to read pin value*/
u8   MDIO_u8GetPinValue         (u8 Copy_u8Port , u8 Copy_u8Pin);

/*description: used to set port direction*/
void MDIO_voidSetPortDirection  (u8 Copy_u8Port , u8 Copy_u8Dir);

void MDIO_voidSetPortDirection  (u8 Copy_u8Port  ,u8 Copy_u8Dir);

/*description: used to set port value*/
void MDIO_voidSetPortValue      (u8 Copy_u8Port , u8 Copy_u8Value);

/*description: used to read port value*/
u8   MDIO_u8GetPortValue        (u8 Copy_u8Port);

/*description: used to enable pull up resistor*/
void  MDIO_VoidSetPullupResistor  (u8 Copy_u8Port , u8 Copy_u8Pin);


#endif /* DIO_INTERFACE_H_ */
```

2.3.2 LED:



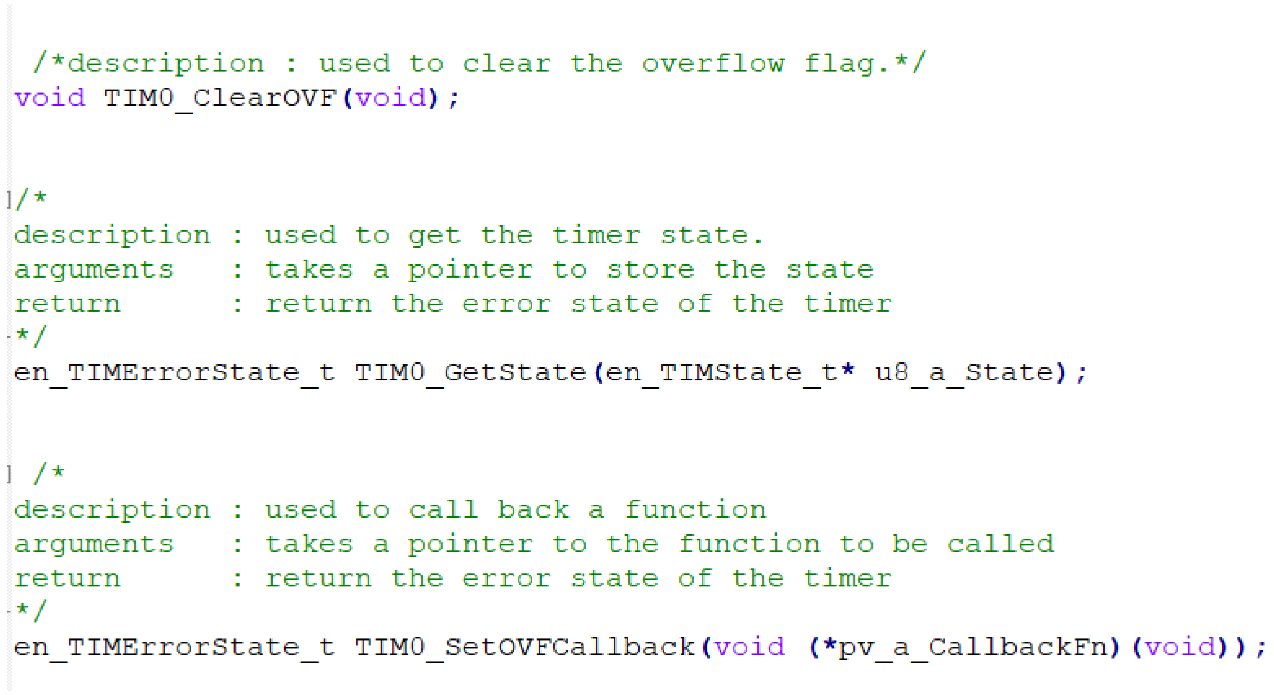
```
/***************************************************************************ProtoTypes************************************************/
/*
description : initializes the LED.
arguments   : takes the LED number
*/
void HLED_ledInit(u8 Copy_u8ledNum);

/*
description : used to set the LED on
arguments   : takes the LED number
*/
void HLED_ledOn (u8 Copy_u8ledNum);

/*
description : used to turn the LED off
arguments   : takes the LED number
*/
void HLED_ledOff(u8 Copy_u8ledNum);



#endif /* LED_INTERFACE_H_ */
```

2.3.3 BUTTON:

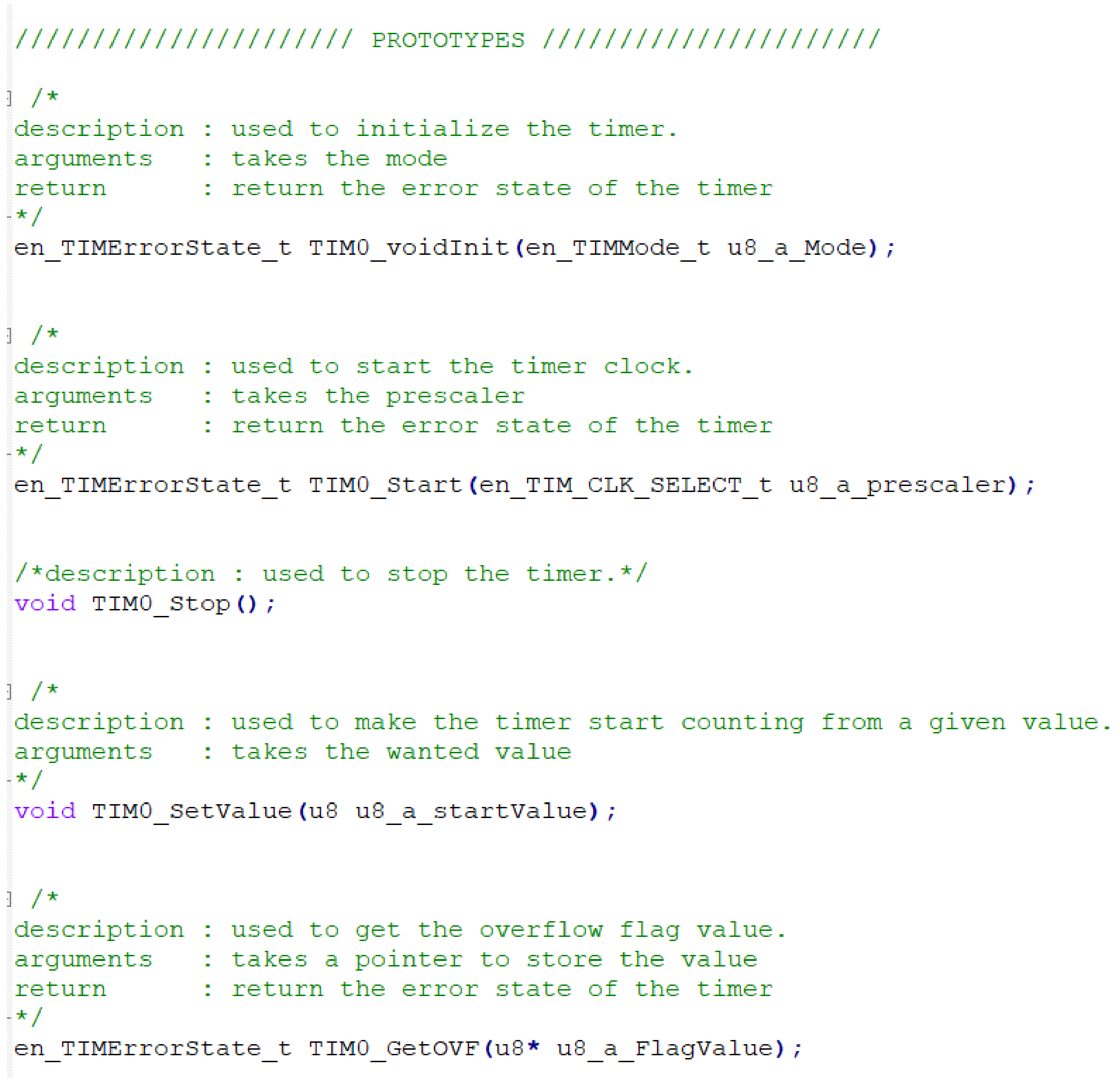


```c
/*
description   : used to initialize the button
arguments     : copy of the button number
*/
void HPushButtonOn_init(u8 Copy_u8buttonNum);

/*
description   : used to get the button value
arguments     : copy of the button number
return        : the button value
*/
u8 HPushButton_getValue(u8 Copy_u8buttonNum);



#endif /* BUTTON_INTERFACE_H_ */
```

2.3.4 TIMER:

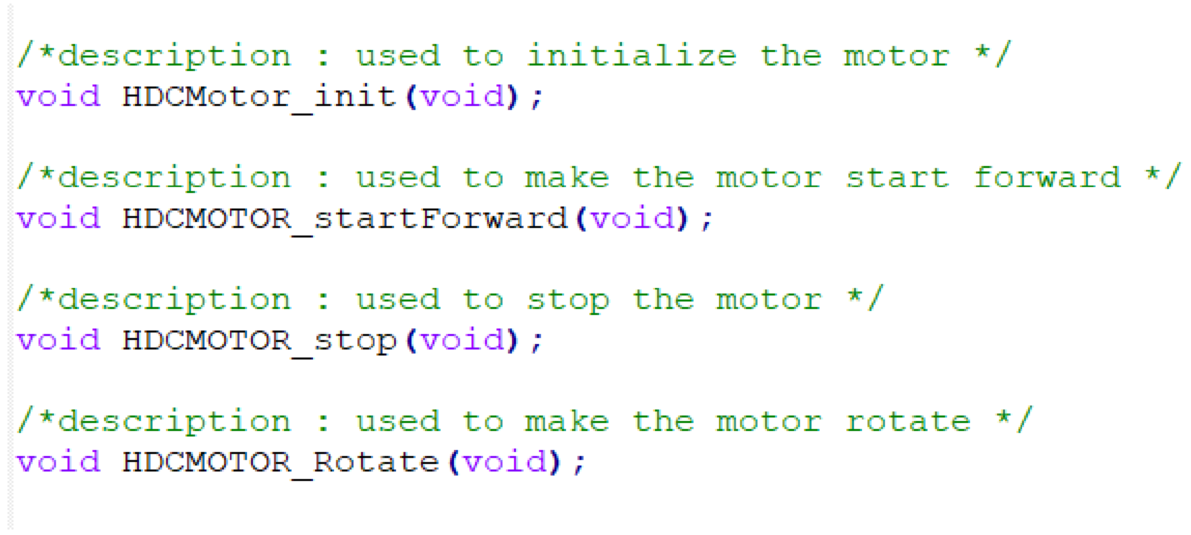


```c
 /*description : used to clear the overflow flag.*/
void TIM0_ClearOVF(void);


/*
description : used to get the timer state.
arguments    : takes a pointer to store the state
return       : return the error state of the timer
*/
en_TIMErrorState_t TIM0_GetState(en_TIMState_t* u8_a_State);


 /*
description : used to call back a function
arguments    : takes a pointer to the function to be called
return       : return the error state of the timer
*/
en_TIMErrorState_t TIM0_SetOVFCallback(void (*pv_a_CallbackFn)(void));
```



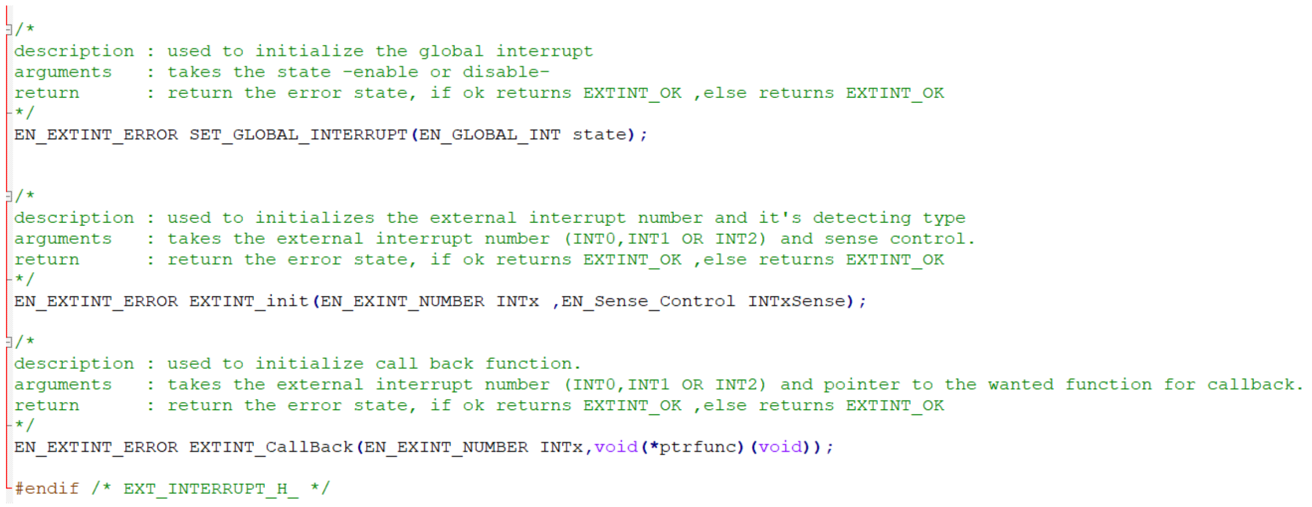
```
///////////////////// PROTOTYPES /////////////////////

 /*
description : used to initialize the timer.
arguments   : takes the mode
return      : return the error state of the timer
*/
en_TIMErrorState_t TIM0_voidInit(en_TIMMode_t u8_a_Mode);


 /*
description : used to start the timer clock.
arguments   : takes the prescaler
return      : return the error state of the timer
*/
en_TIMErrorState_t TIM0_Start(en_TIM_CLK_SELECT_t u8_a_prescaler);


/*description : used to stop the timer.*/
void TIM0_Stop();


 /*
description : used to make the timer start counting from a given value.
arguments   : takes the wanted value
*/
void TIM0_SetValue(u8 u8_a_startValue);


 /*
description : used to get the overflow flag value.
arguments   : takes a pointer to store the value
return      : return the error state of the timer
*/
en_TIMErrorState_t TIM0_GetOVF(u8* u8_a_FlagValue);
```

2.3.5 MOTOR:



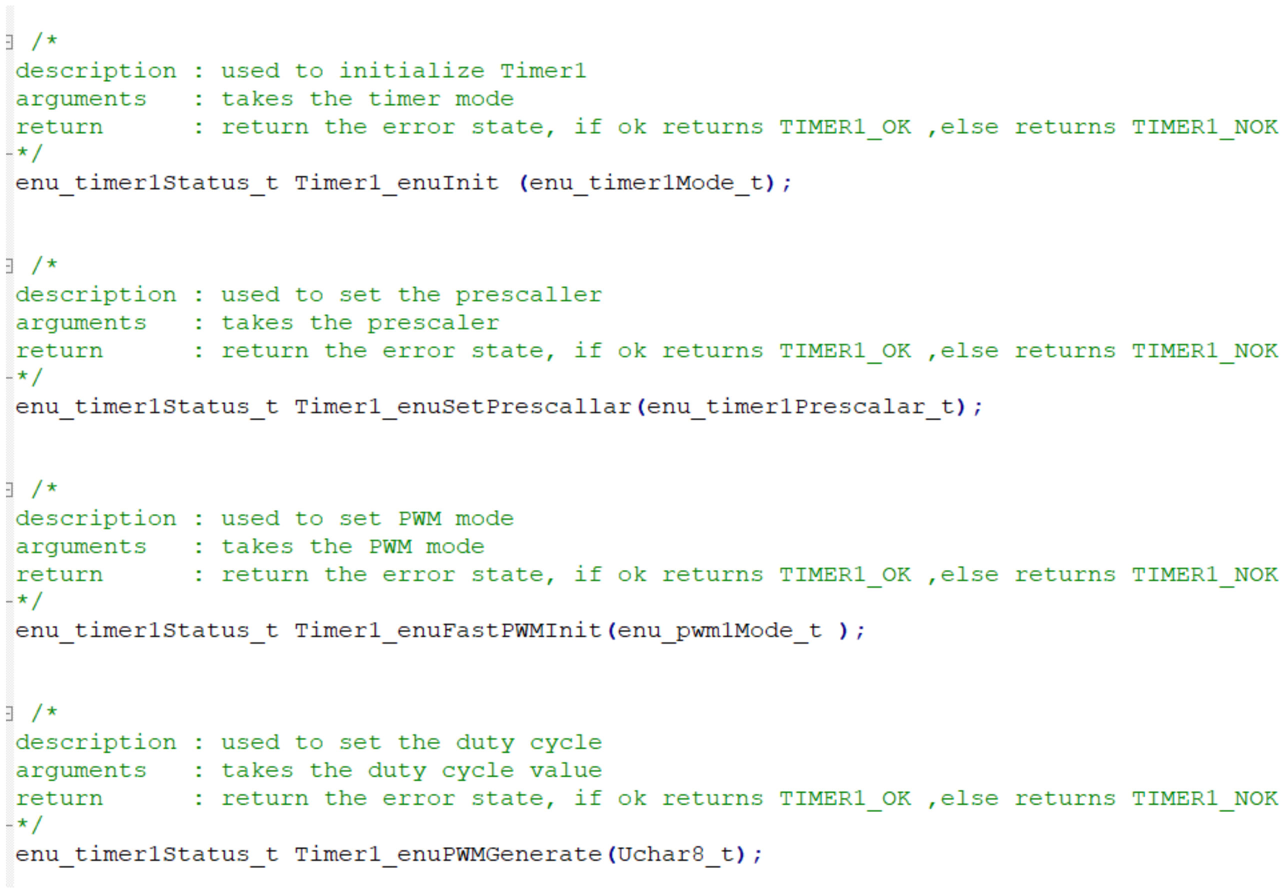
```
/*description : used to initialize the motor */
void HDCMotor_init(void);

/*description : used to make the motor start forward */
void HDCMOTOR_startForward(void);

/*description : used to stop the motor */
void HDCMOTOR_stop(void);

/*description : used to make the motor rotate */
void HDCMOTOR_Rotate(void);
```

2.3.6 External Interrupt:



```
/*
description : used to initialize the global interrupt
arguments   : takes the state -enable or disable-
return      : return the error state, if ok returns EXTINT_OK ,else returns EXTINT_OK
*/
EN_EXTINT_ERROR SET_GLOBAL_INTERRUPT(EN_GLOBAL_INT state);


/*
description : used to initializes the external interrupt number and it's detecting type
arguments   : takes the external interrupt number (INT0,INT1 OR INT2) and sense control.
return      : return the error state, if ok returns EXTINT_OK ,else returns EXTINT_OK
*/
EN_EXTINT_ERROR EXTINT_init(EN_EXINT_NUMBER INTx ,EN_Sense_Control INTxSense);

/*
description : used to initialize call back function.
arguments   : takes the external interrupt number (INT0,INT1 OR INT2) and pointer to the wanted function for callback.
return      : return the error state, if ok returns EXTINT_OK ,else returns EXTINT_OK
*/
EN_EXTINT_ERROR EXTINT_CallBack(EN_EXINT_NUMBER INTx,void(*ptrfunc)(void));

#endif /* EXT_INTERRUPT_H_ */
```

2.3.7 PWM:

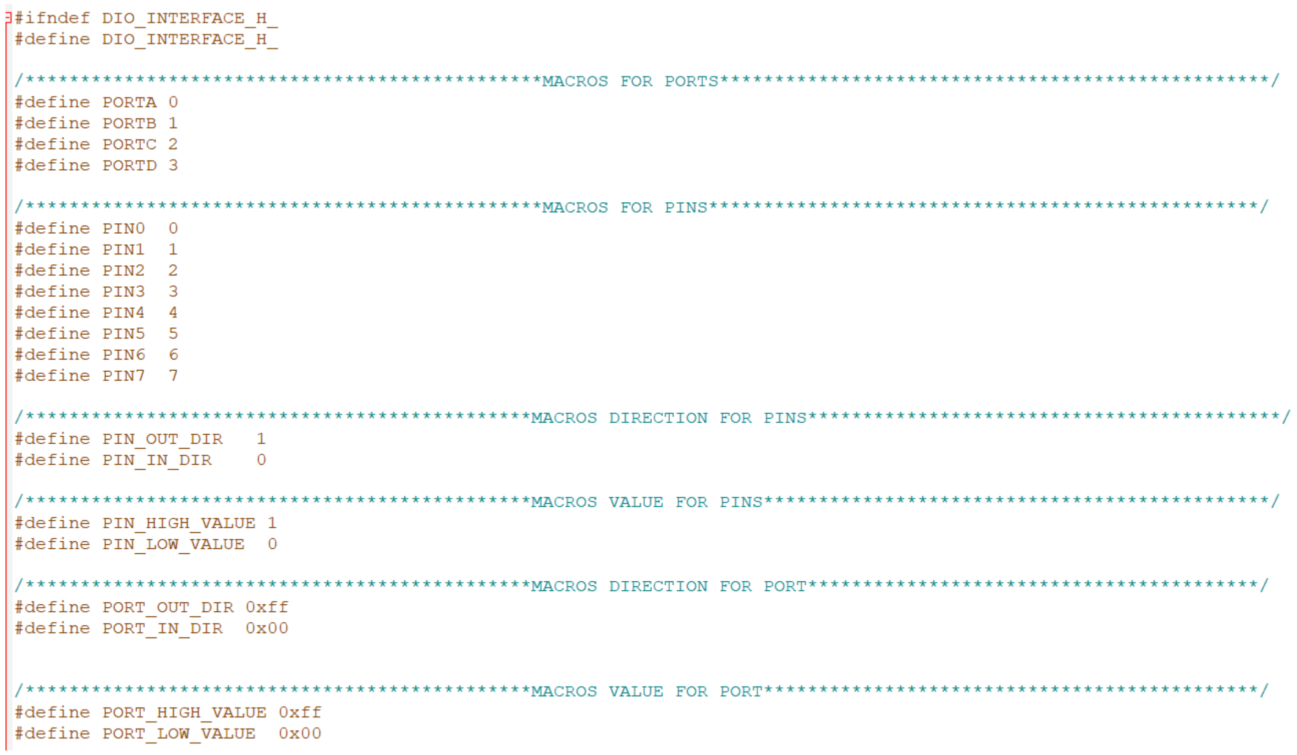


```
/*
description : used to initialize Timer1
arguments   : takes the timer mode
return      : return the error state, if ok returns TIMER1_OK ,else returns TIMER1_NOK
*/
enu_timer1Status_t Timer1_enuInit (enu_timer1Mode_t);


/*
description : used to set the prescaller
arguments   : takes the prescaler
return      : return the error state, if ok returns TIMER1_OK ,else returns TIMER1_NOK
*/
enu_timer1Status_t Timer1_enuSetPrescallar(enu_timer1Prescalar_t);


/*
description : used to set PWM mode
arguments   : takes the PWM mode
return      : return the error state, if ok returns TIMER1_OK ,else returns TIMER1_NOK
*/
enu_timer1Status_t Timer1_enuFastPWMInit(enu_pwm1Mode_t );


/*
description : used to set the duty cycle
arguments   : takes the duty cycle value
return      : return the error state, if ok returns TIMER1_OK ,else returns TIMER1_NOK
*/
enu_timer1Status_t Timer1_enuPWMGenerate(Uchar8_t);
```

3. Low Level Design:

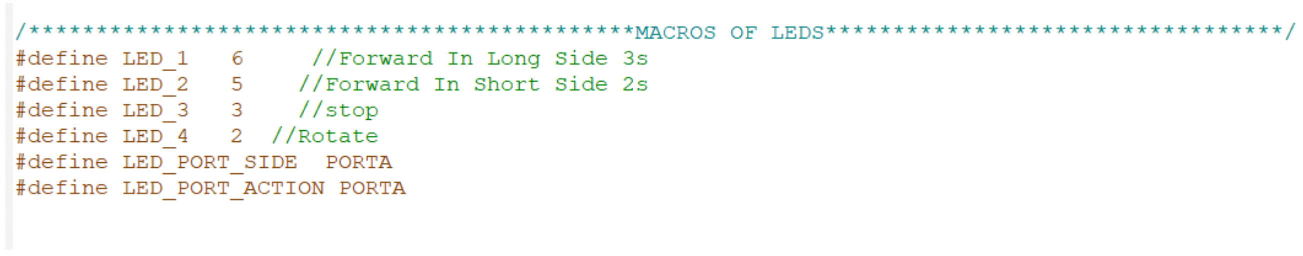
3.1 Configurations:

3.1.1 DIO:



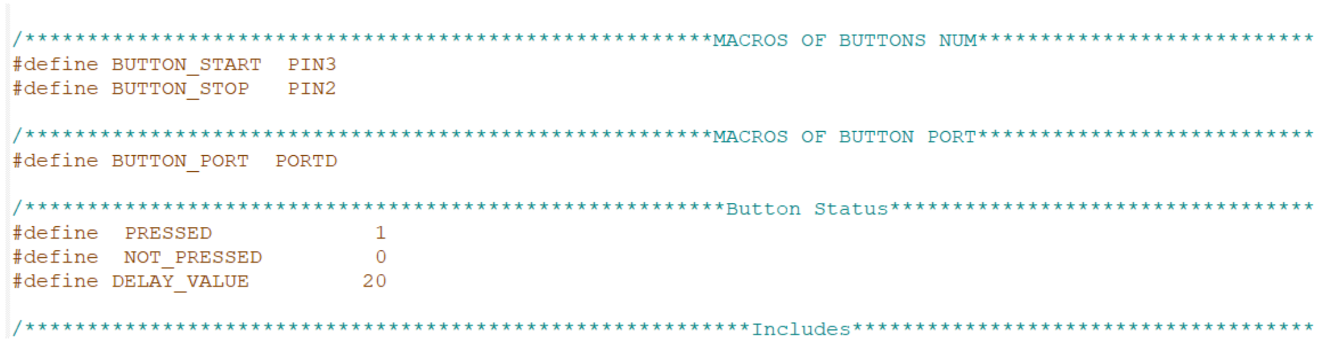
```
#ifndef DIO_INTERFACE_H_
#define DIO_INTERFACE_H_

/***********************************************MACROS FOR PORTS**************************************************/
#define PORTA 0
#define PORTB 1
#define PORTC 2
#define PORTD 3

/***********************************************MACROS FOR PINS**************************************************/
#define PIN0  0
#define PIN1  1
#define PIN2  2
#define PIN3  3
#define PIN4  4
#define PIN5  5
#define PIN6  6
#define PIN7  7

/**********************************************MACROS DIRECTION FOR PINS********************************************/
#define PIN_OUT_DIR   1
#define PIN_IN_DIR    0

/**********************************************MACROS VALUE FOR PINS************************************************/
#define PIN_HIGH_VALUE 1
#define PIN_LOW_VALUE  0

/**********************************************MACROS DIRECTION FOR PORT*******************************************/
#define PORT_OUT_DIR 0xff
#define PORT_IN_DIR  0x00


/**********************************************MACROS VALUE FOR PORT***********************************************/
#define PORT_HIGH_VALUE 0xff
#define PORT_LOW_VALUE  0x00
```

3.2.2 LED:



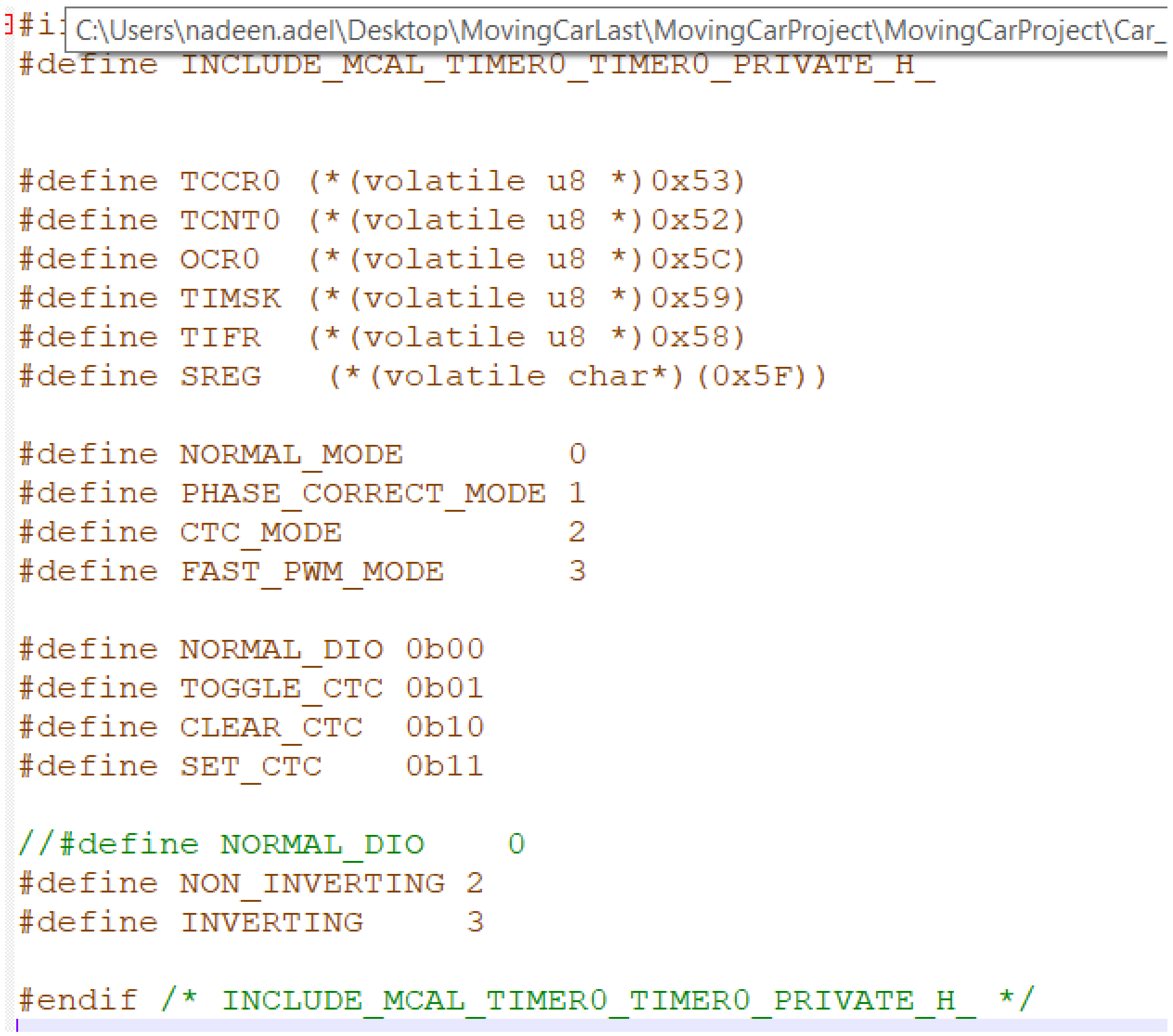
```
/**********************************************MACROS OF LEDS************************************/
#define LED_1   6     //Forward In Long Side 3s
#define LED_2   5    //Forward In Short Side 2s
#define LED_3   3    //stop
#define LED_4   2  //Rotate
#define LED_PORT_SIDE   PORTA
#define LED_PORT_ACTION PORTA
```

3.1.3 Button:



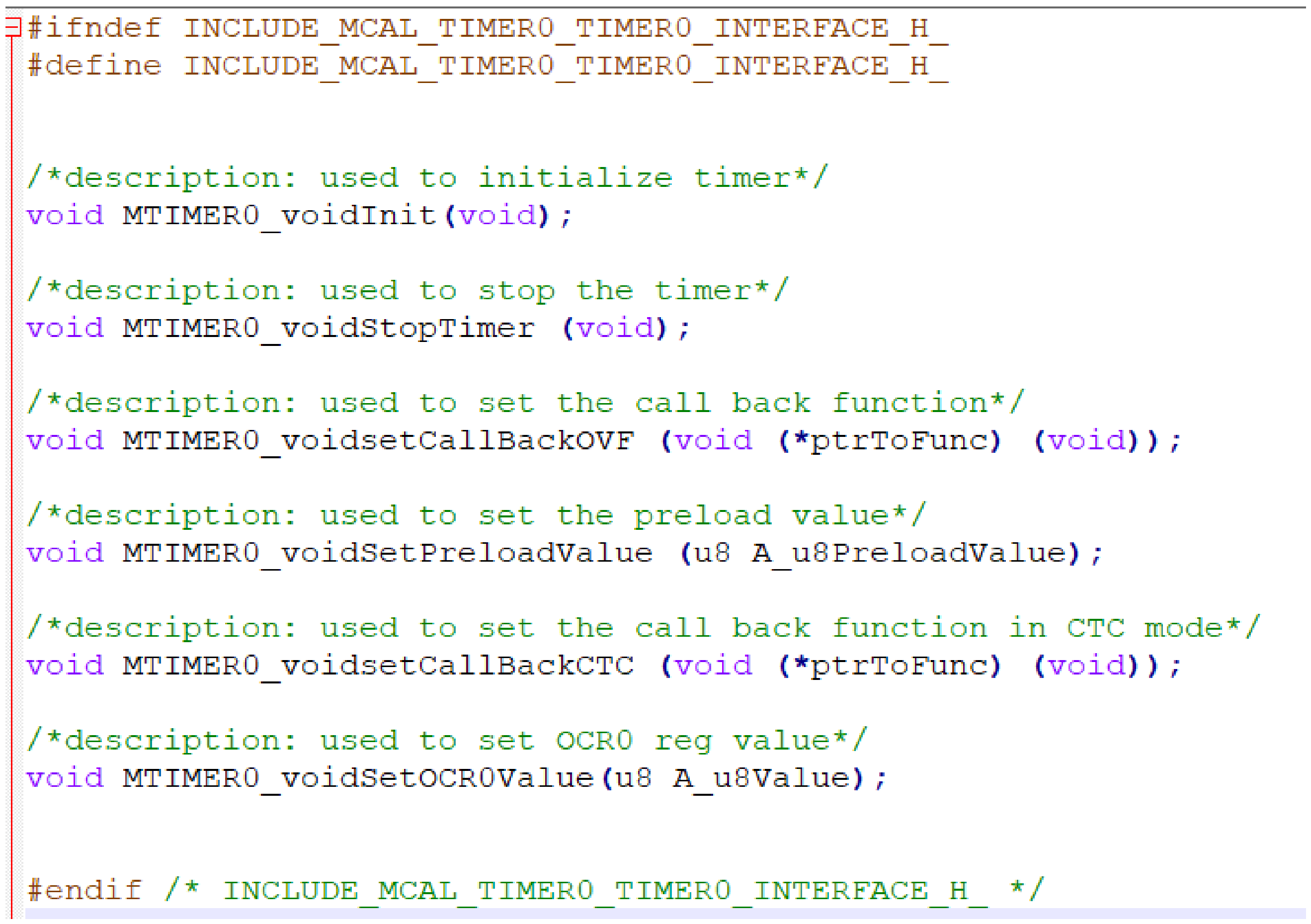
```
/*******************************************************MACROS OF BUTTONS NUM***************************
#define BUTTON_START  PIN3
#define BUTTON_STOP   PIN2

/*******************************************************MACROS OF BUTTON PORT***************************
#define BUTTON_PORT  PORTD

/********************************************************Button Status*********************************
#define  PRESSED            1
#define  NOT_PRESSED        0
#define DELAY_VALUE        20

/***********************************************************Includes************************************
```

3.1.4 Timer:



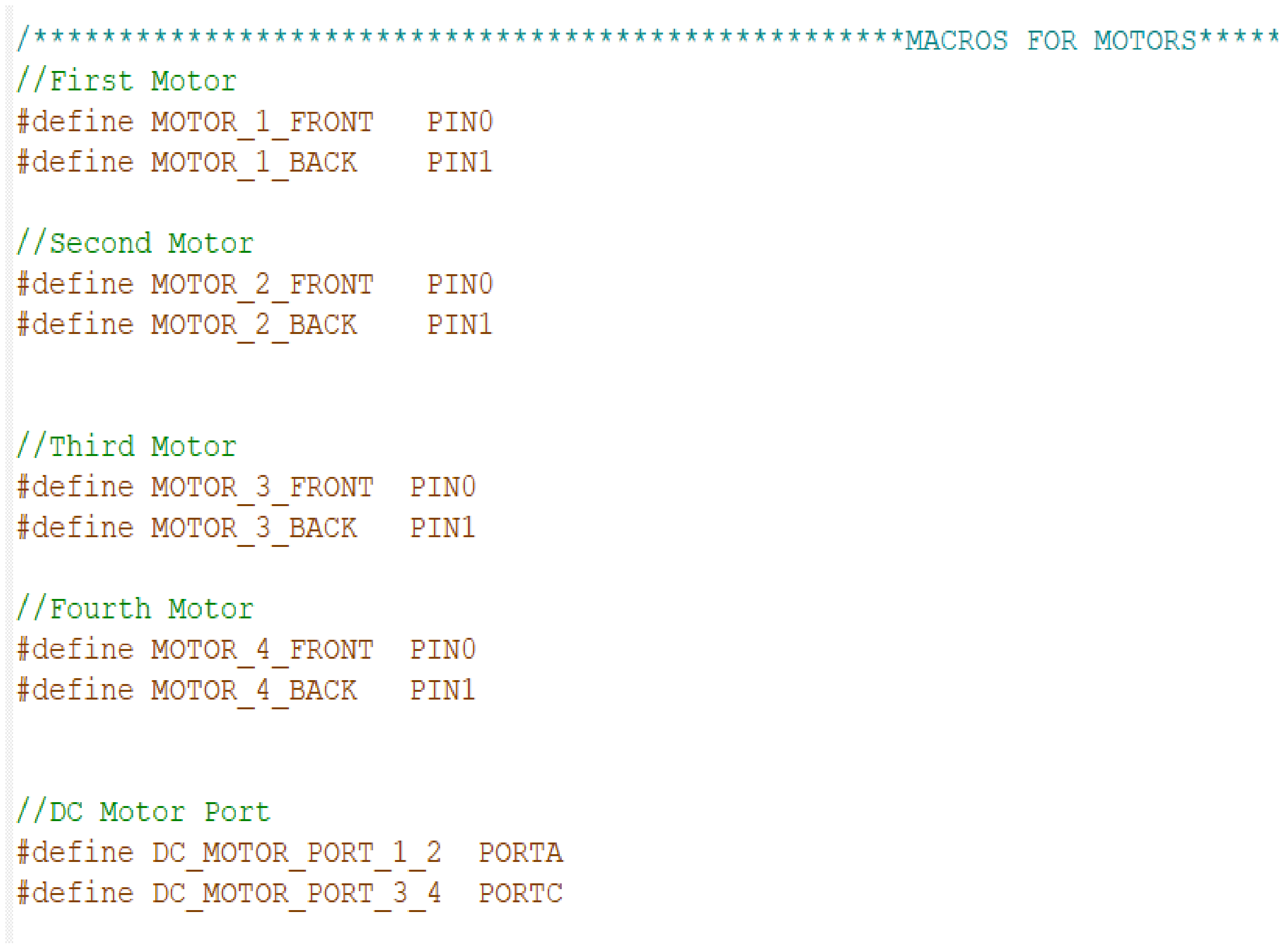
```
#i C:\Users\nadeen.adel\Desktop\MovingCarLast\MovingCarProject\MovingCarProject\Car_
#define INCLUDE_MCAL_TIMER0_TIMER0_PRIVATE_H_


#define TCCR0  (*(volatile u8 *)0x53)
#define TCNT0  (*(volatile u8 *)0x52)
#define OCR0   (*(volatile u8 *)0x5C)
#define TIMSK  (*(volatile u8 *)0x59)
#define TIFR   (*(volatile u8 *)0x58)
#define SREG    (*(volatile char*)(0x5F))

#define NORMAL_MODE        0
#define PHASE_CORRECT_MODE 1
#define CTC_MODE           2
#define FAST_PWM_MODE      3

#define NORMAL_DIO 0b00
#define TOGGLE_CTC 0b01
#define CLEAR_CTC  0b10
#define SET_CTC    0b11

//#define NORMAL_DIO    0
#define NON_INVERTING 2
#define INVERTING     3

#endif /* INCLUDE_MCAL_TIMER0_TIMER0_PRIVATE_H_ */
```

```
#ifndef INCLUDE_MCAL_TIMER0_TIMER0_CONFIGURATION_H_
#define INCLUDE_MCAL_TIMER0_TIMER0_CONFIGURATION_H_



/*
 * NORMAL_MODE
 * PHASE_CORRECT_MODE
 * CTC_MODE
 * FAST_PWM_MODE
 * */
#define TIMER0_MODE NORMAL_MODE


//NORMAL_DIO
//TOGGLE_CTC
//CLEAR_CTC
//SET_CTC

#define CTC_OC0_PIN_ACTION NORMAL_DIO


// NORMAL_DIO
// NON_INVERTING
// INVERTING
#define FAST_PWM_OC0_PIn_ACTION INVERTING

#define OCR0_VALUE  99

/*
 * 0b000  No clock source (Timer/Counter stopped).
 * 0b001  clkI/O /(No prescaling)
 * 0b010  clkI/O /8 (From prescaler)
 * 0b011  clkI/O /64 (From prescaler)
 * 0b100  clkI/O /256 (From prescaler)
 * 0b101  clkI/O /1024 (From prescaler)
 * 0b110  External clock source on T0 pin. Clock on falling edge.
 * 0b111  External clock source on T0 pin. Clock on rising edge.
 * */
#define CLK_CONFIGURATION  0b010
```



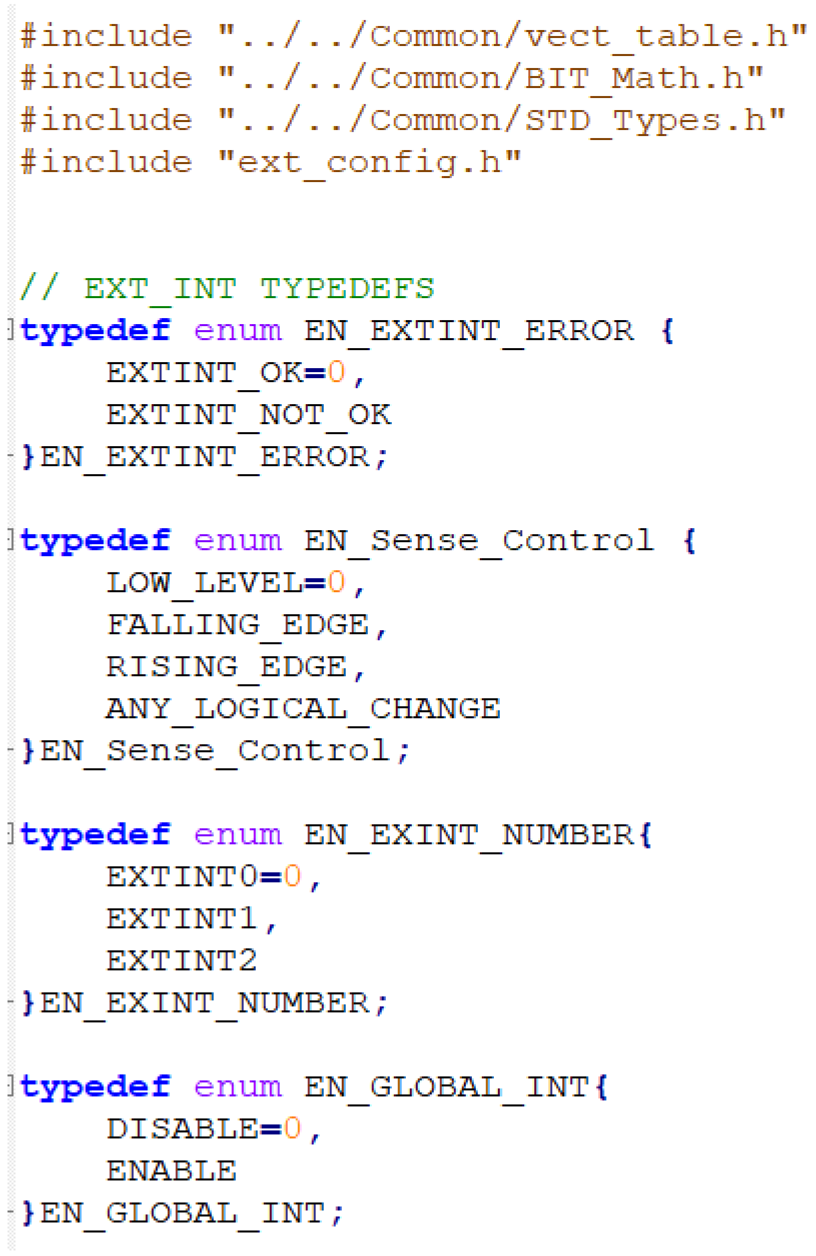
```c
#ifndef INCLUDE_MCAL_TIMER0_TIMER0_INTERFACE_H_
#define INCLUDE_MCAL_TIMER0_TIMER0_INTERFACE_H_


/*description: used to initialize timer*/
void MTIMER0_voidInit(void);

/*description: used to stop the timer*/
void MTIMER0_voidStopTimer (void);

/*description: used to set the call back function*/
void MTIMER0_voidsetCallBackOVF (void (*ptrToFunc) (void));

/*description: used to set the preload value*/
void MTIMER0_voidSetPreloadValue (u8 A_u8PreloadValue);

/*description: used to set the call back function in CTC mode*/
void MTIMER0_voidsetCallBackCTC (void (*ptrToFunc) (void));

/*description: used to set OCR0 reg value*/
void MTIMER0_voidSetOCR0Value(u8 A_u8Value);


#endif /* INCLUDE_MCAL_TIMER0_TIMER0_INTERFACE_H_ */
```

3.1.5 Motor:



```
/***************************************************MACROS FOR MOTORS*****
//First Motor
#define MOTOR_1_FRONT   PIN0
#define MOTOR_1_BACK    PIN1

//Second Motor
#define MOTOR_2_FRONT   PIN0
#define MOTOR_2_BACK    PIN1


//Third Motor
#define MOTOR_3_FRONT  PIN0
#define MOTOR_3_BACK   PIN1

//Fourth Motor
#define MOTOR_4_FRONT  PIN0
#define MOTOR_4_BACK   PIN1


//DC Motor Port
#define DC_MOTOR_PORT_1_2  PORTA
#define DC_MOTOR_PORT_3_4  PORTC
```

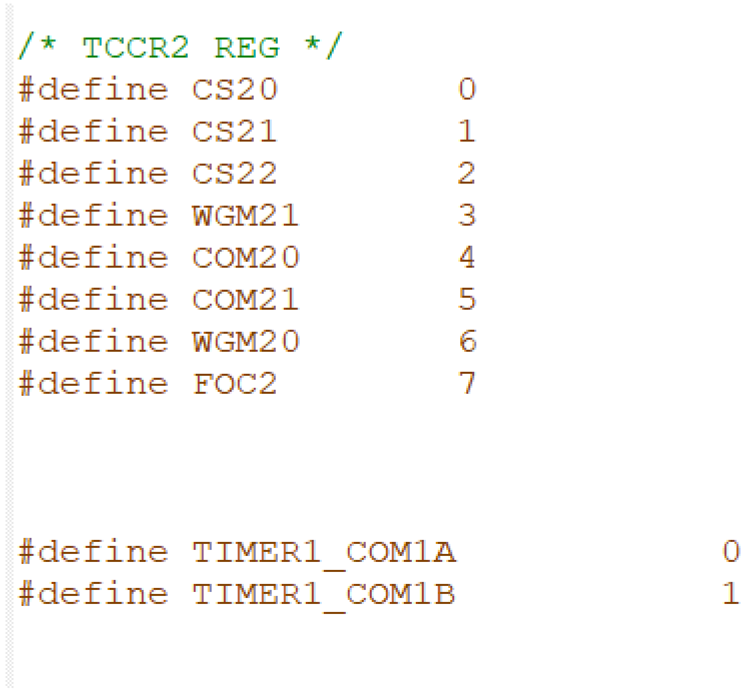
3.1.6 External Interrupt:



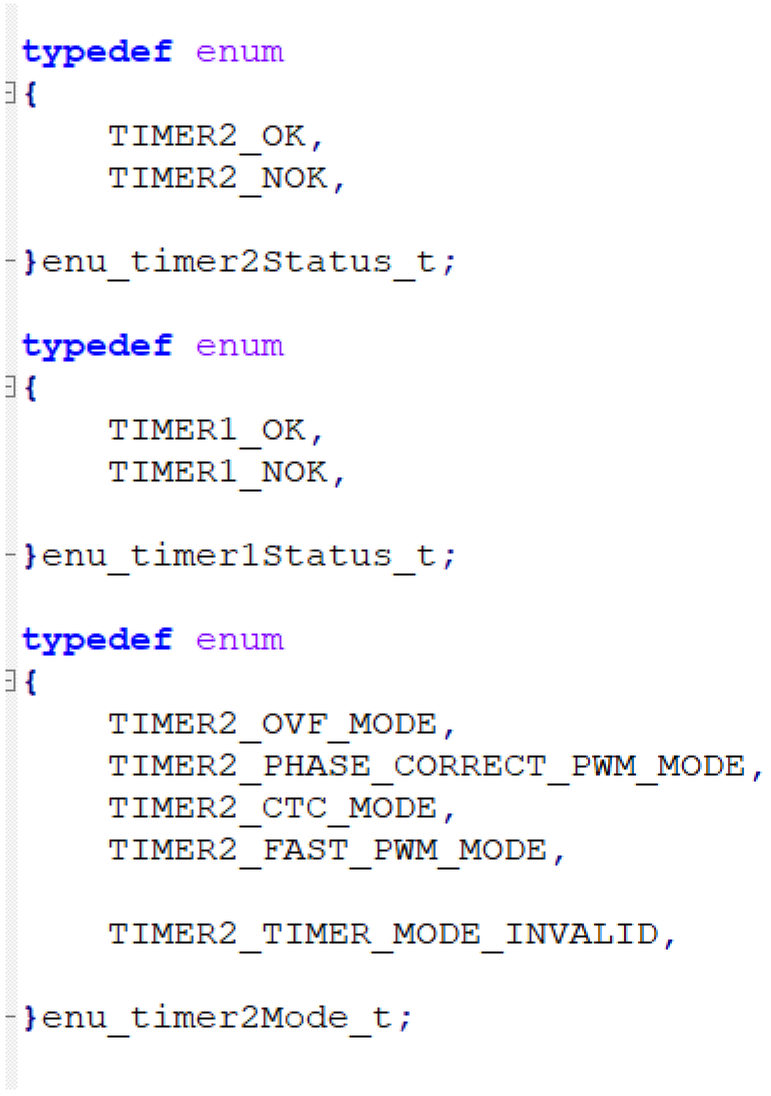
```
#include "../../Common/vect_table.h"
#include "../../Common/BIT_Math.h"
#include "../../Common/STD_Types.h"
#include "ext_config.h"


// EXT_INT TYPEDEFS
typedef enum EN_EXTINT_ERROR {
    EXTINT_OK=0,
    EXTINT_NOT_OK
}EN_EXTINT_ERROR;

typedef enum EN_Sense_Control {
    LOW_LEVEL=0,
    FALLING_EDGE,
    RISING_EDGE,
    ANY_LOGICAL_CHANGE
}EN_Sense_Control;

typedef enum EN_EXINT_NUMBER{
    EXTINT0=0,
    EXTINT1,
    EXTINT2
}EN_EXINT_NUMBER;

typedef enum EN_GLOBAL_INT{
    DISABLE=0,
    ENABLE
}EN_GLOBAL_INT;
```

3.1.7 PWM:

```
/* TIMSK REG */
#define TOIE0       0
#define OCIE0       1
#define TOIE1       2
#define OCIE1A      3
#define OCIE1B      4
#define TICIE1      5
#define TOIE2       6
#define OCIE2       7

/* TIFR REG */
#define TOV0        0
#define OCF0        1
#define TOV1        2
#define OCF1A       3
#define OCF1B       4
#define ICF1        5
#define TOV2        6
#define OCF2        7


/* TCCR1A REG */
#define WGM10       0
#define WGM11       1
#define FOC1B       2
#define FOC1A       3
#define COM1B0      4
#define COM1B1      5
#define COM1A0      6
#define COM1A1      7

/* TCCR1B REG */
#define CS10        0
#define CS11        1
#define CS12        2
#define WGM12       3
#define WGM13       4
#define ICES1       6
#define ICNC1       7
```

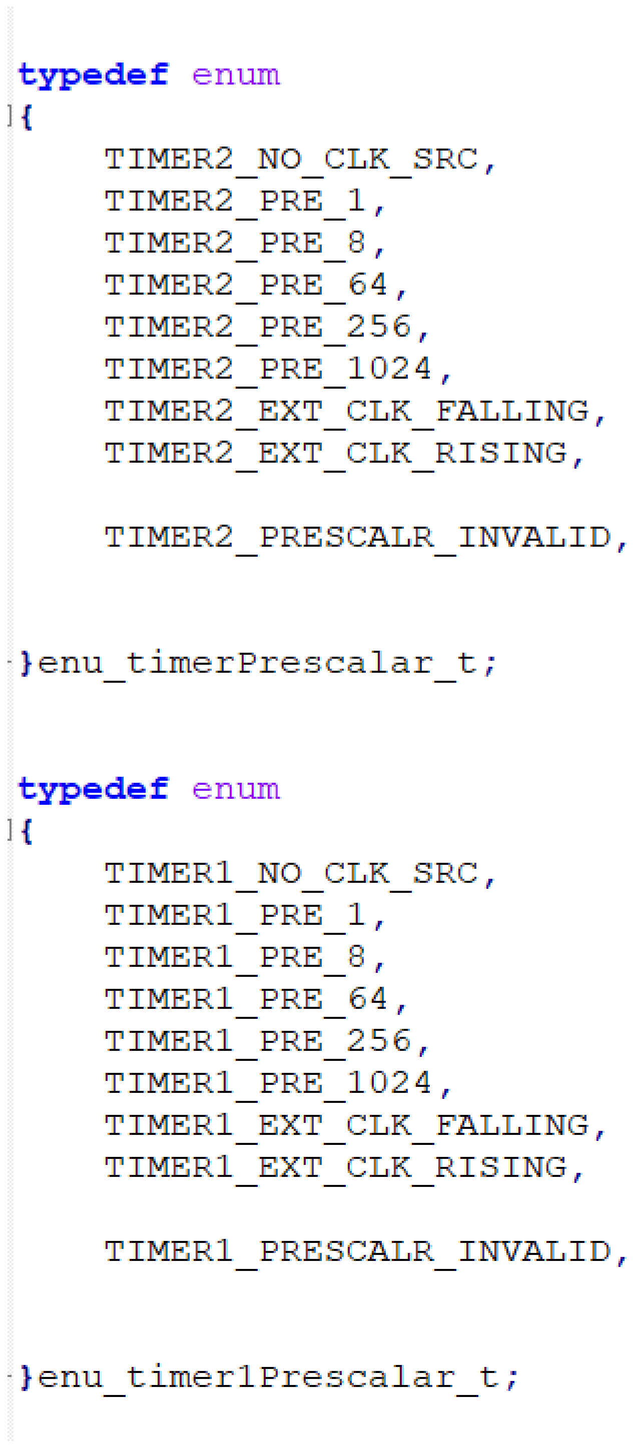


```c
/* TCCR2 REG */
#define CS20        0
#define CS21        1
#define CS22        2
#define WGM21       3
#define COM20       4
#define COM21       5
#define WGM20       6
#define FOC2        7



#define TIMER1_COM1A            0
#define TIMER1_COM1B            1
```

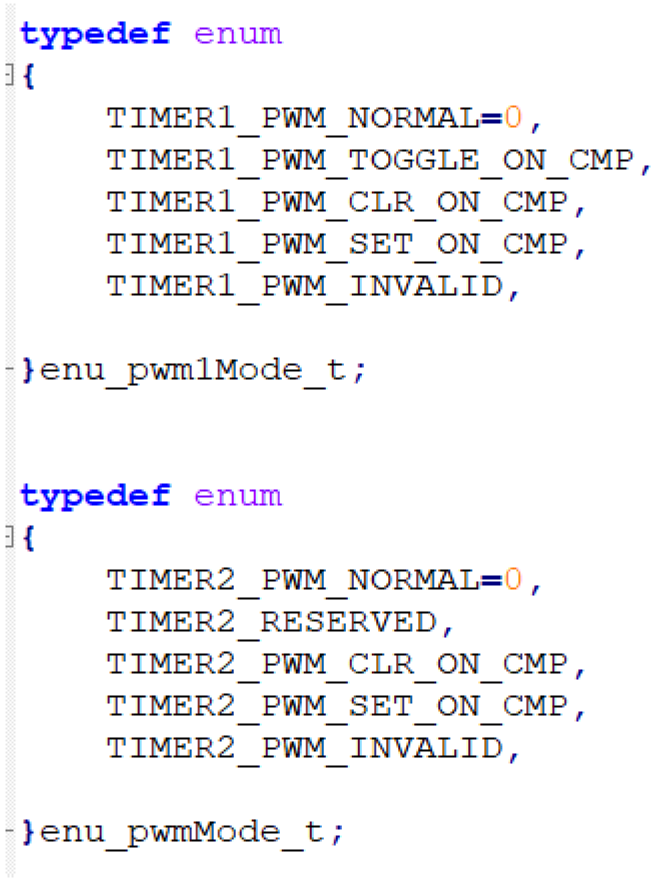


```c
typedef enum
{
    TIMER2_OK,
    TIMER2_NOK,

}enu_timer2Status_t;

typedef enum
{
    TIMER1_OK,
    TIMER1_NOK,

}enu_timer1Status_t;

typedef enum
{
    TIMER2_OVF_MODE,
    TIMER2_PHASE_CORRECT_PWM_MODE,
    TIMER2_CTC_MODE,
    TIMER2_FAST_PWM_MODE,

    TIMER2_TIMER_MODE_INVALID,

}enu_timer2Mode_t;
```

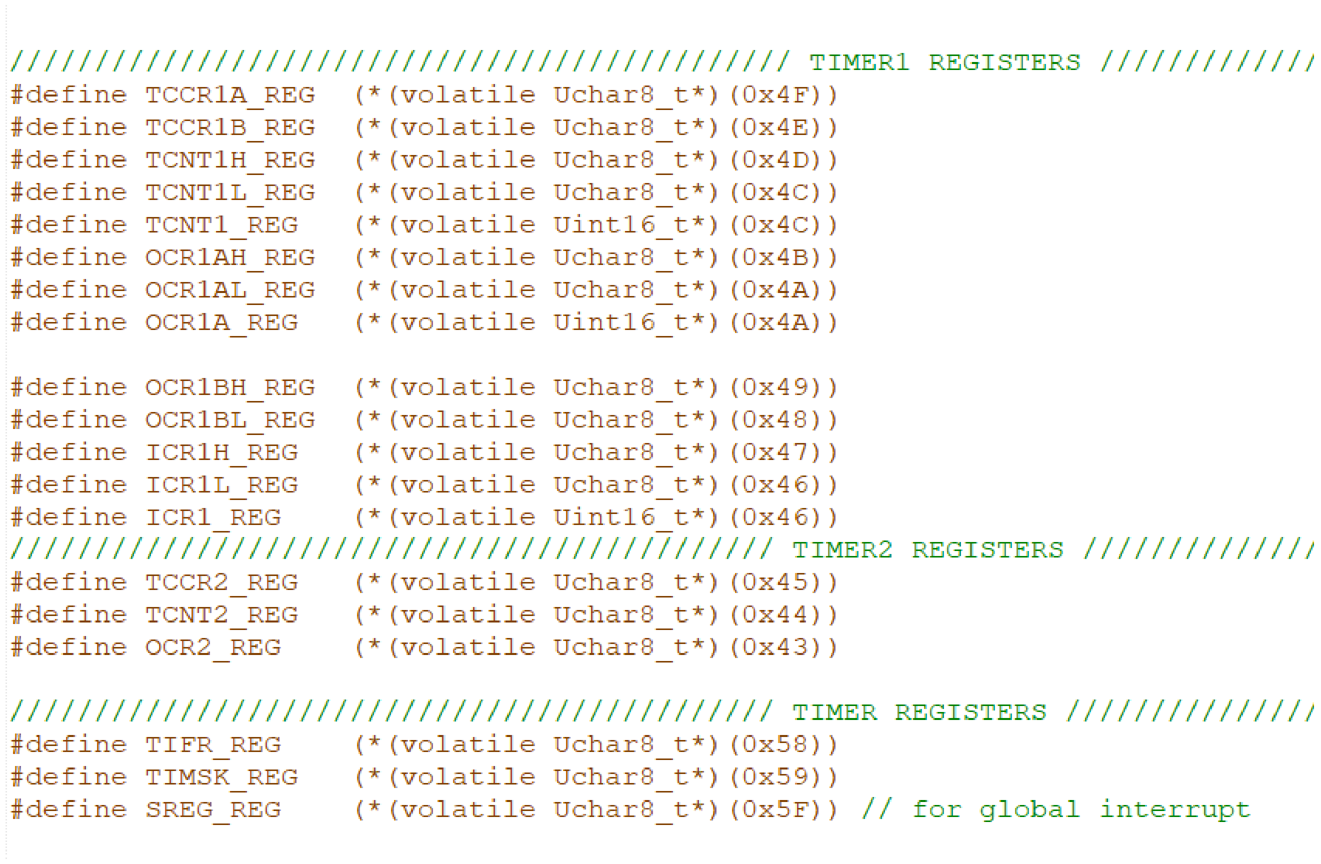
```
typedef enum
{
    TIMER1_NORMAL_0xFFFF,

    TIMER1_PWM_PHASE_CORRECT_8_0x00FF,
    TIMER1_PWM_PHASE_CORRECT_9_0x01FF,
    TIMER1_PWM_PHASE_CORRECT_10_0x03FF,

    TIMER1_CTC_OCR1A,

    TIMER1_FAST_PWM_8_0x00FF,
    TIMER1_FAST_PWM_9_0x01FF,
    TIMER1_FAST_PWM_10_0x03FF,

    TIMER1_PWM_PHASE_FREQ_CORRECT_ICR1,
    TIMER1_PWM_PHASE_FREQ_CORRECT_OCR1A,

    TIMER1_PWM_PHASE_CORRECT_ICR1,
    TIMER1_PWM_PHASE_CORRECT_OCR1A,

    TIMER1_CTC_ICR1,

    TIMER1_FAST_PWM_ICR1,
    TIMER1_FAST_PWM_OCR1A,

    TIMER1_MODE_INVALID,


}enu_timer1Mode_t;
```



```
typedef enum
{
    TIMER2_NO_CLK_SRC,
    TIMER2_PRE_1,
    TIMER2_PRE_8,
    TIMER2_PRE_64,
    TIMER2_PRE_256,
    TIMER2_PRE_1024,
    TIMER2_EXT_CLK_FALLING,
    TIMER2_EXT_CLK_RISING,

    TIMER2_PRESCALR_INVALID,


}enu_timerPrescalar_t;


typedef enum
{
    TIMER1_NO_CLK_SRC,
    TIMER1_PRE_1,
    TIMER1_PRE_8,
    TIMER1_PRE_64,
    TIMER1_PRE_256,
    TIMER1_PRE_1024,
    TIMER1_EXT_CLK_FALLING,
    TIMER1_EXT_CLK_RISING,

    TIMER1_PRESCALR_INVALID,


}enu_timer1Prescalar_t;
```



```c
typedef enum
{
    TIMER1_PWM_NORMAL=0,
    TIMER1_PWM_TOGGLE_ON_CMP,
    TIMER1_PWM_CLR_ON_CMP,
    TIMER1_PWM_SET_ON_CMP,
    TIMER1_PWM_INVALID,

}enu_pwm1Mode_t;


typedef enum
{
    TIMER2_PWM_NORMAL=0,
    TIMER2_RESERVED,
    TIMER2_PWM_CLR_ON_CMP,
    TIMER2_PWM_SET_ON_CMP,
    TIMER2_PWM_INVALID,

}enu_pwmMode_t;
```

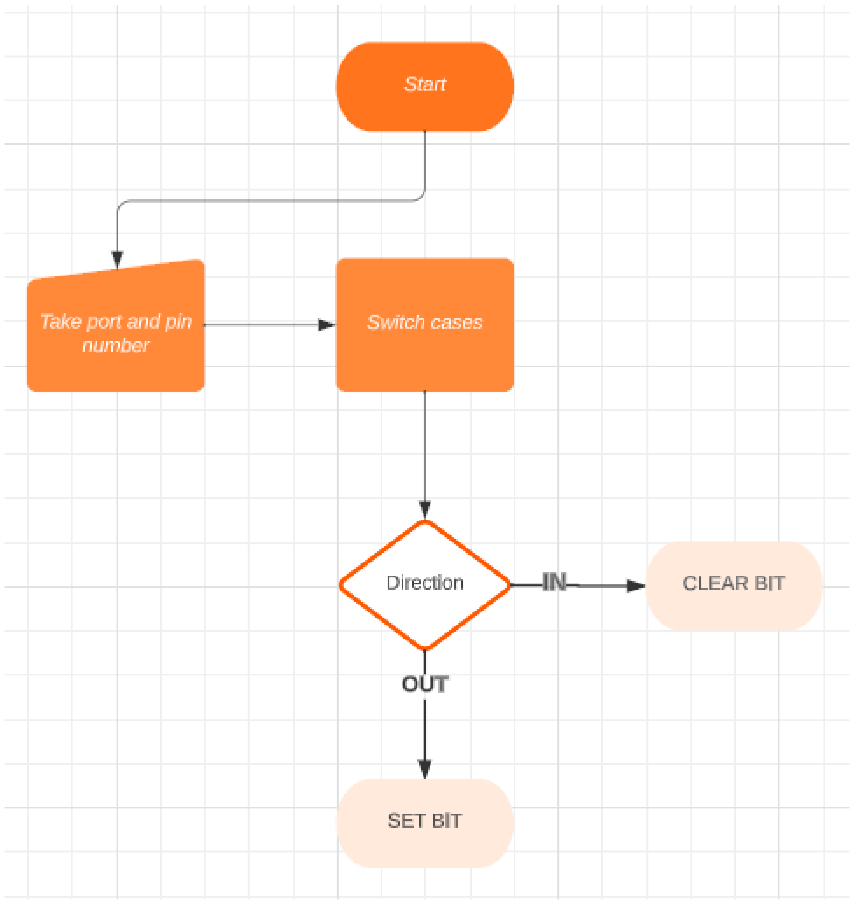


```c
////////////////////////////////////////////// TIMER1 REGISTERS /////////////
#define TCCR1A_REG  (*(volatile Uchar8_t*)(0x4F))
#define TCCR1B_REG  (*(volatile Uchar8_t*)(0x4E))
#define TCNT1H_REG  (*(volatile Uchar8_t*)(0x4D))
#define TCNT1L_REG  (*(volatile Uchar8_t*)(0x4C))
#define TCNT1_REG   (*(volatile Uint16_t*)(0x4C))
#define OCR1AH_REG  (*(volatile Uchar8_t*)(0x4B))
#define OCR1AL_REG  (*(volatile Uchar8_t*)(0x4A))
#define OCR1A_REG   (*(volatile Uint16_t*)(0x4A))

#define OCR1BH_REG  (*(volatile Uchar8_t*)(0x49))
#define OCR1BL_REG  (*(volatile Uchar8_t*)(0x48))
#define ICR1H_REG   (*(volatile Uchar8_t*)(0x47))
#define ICR1L_REG   (*(volatile Uchar8_t*)(0x46))
#define ICR1_REG    (*(volatile Uint16_t*)(0x46))
///////////////////////////////////////////// TIMER2 REGISTERS //////////////
#define TCCR2_REG   (*(volatile Uchar8_t*)(0x45))
#define TCNT2_REG   (*(volatile Uchar8_t*)(0x44))
#define OCR2_REG    (*(volatile Uchar8_t*)(0x43))

///////////////////////////////////////////// TIMER REGISTERS ///////////////
#define TIFR_REG    (*(volatile Uchar8_t*)(0x58))
#define TIMSK_REG   (*(volatile Uchar8_t*)(0x59))
#define SREG_REG    (*(volatile Uchar8_t*)(0x5F)) // for global interrupt
```

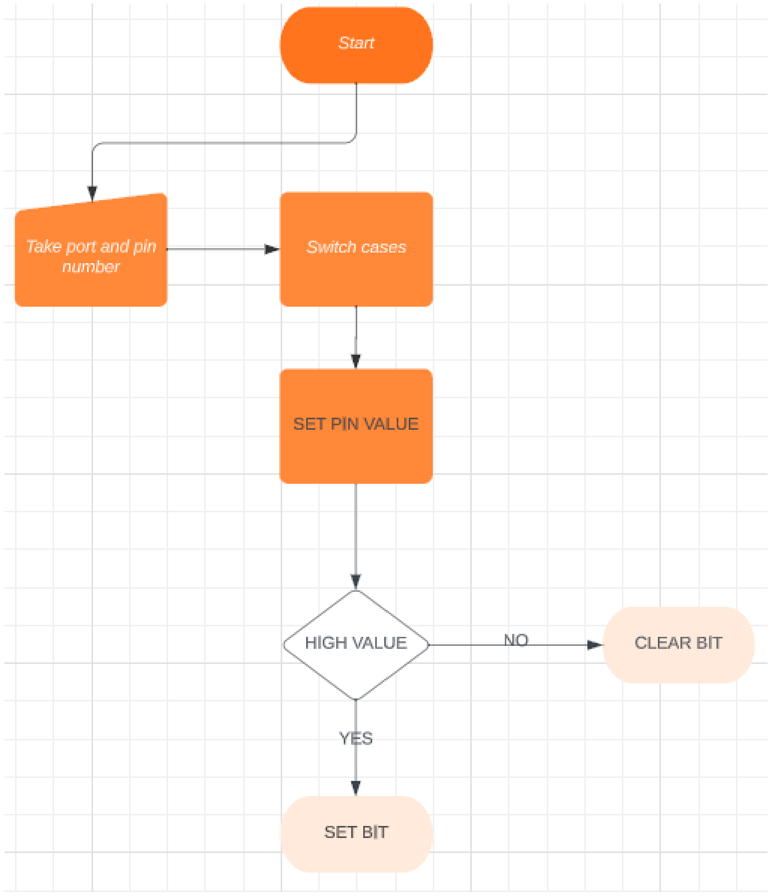
3.2 Functions Flowcharts:

3.2.1 DIO:

void MDIO_voidSetPinDirection (u8 Copy_u8Port , u8 Copy_u8Pin , u8 Copy_u8Dir)

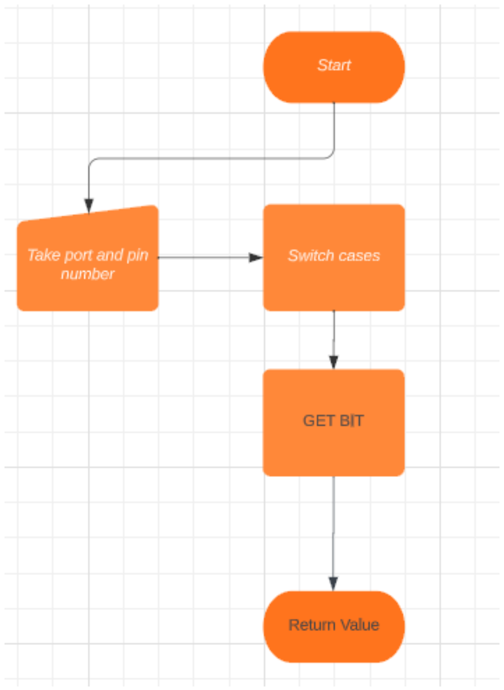


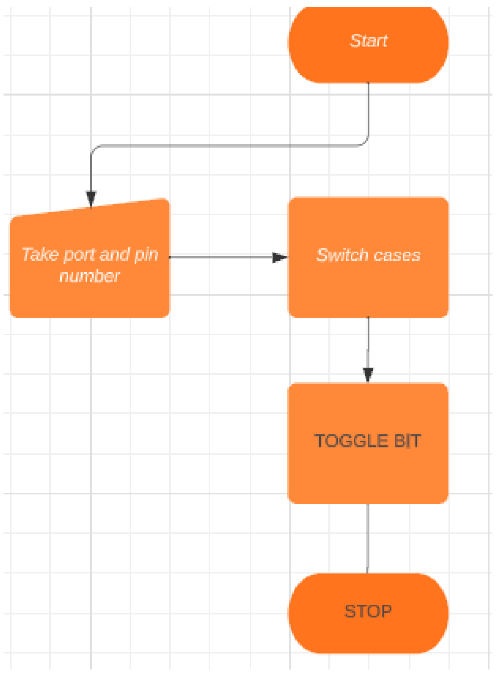
void MDIO_voidSetPinValue(u8 Copy_u8Port,u8 Copy_u8Pin,u8 Copy_u8Value)



u8   MDIO_u8GetPinValue (u8 Copy_u8Port , u8 Copy_u8Pin)

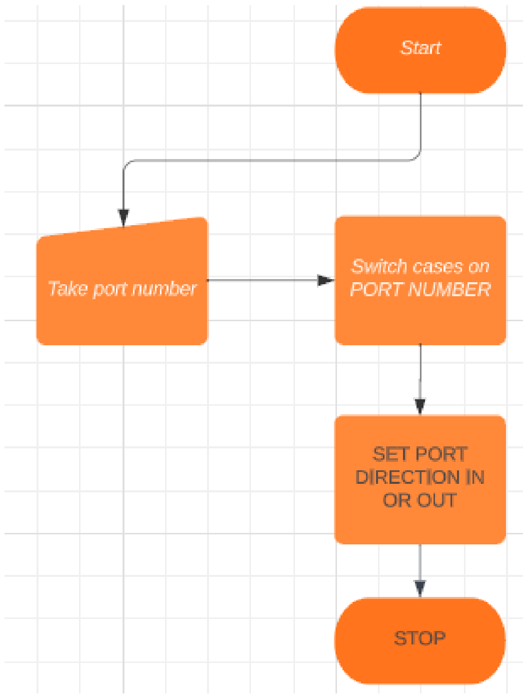
void MDIO_voidTogglePinValue(u8 Copy_u8Port , u8 Copy_u8Pin)





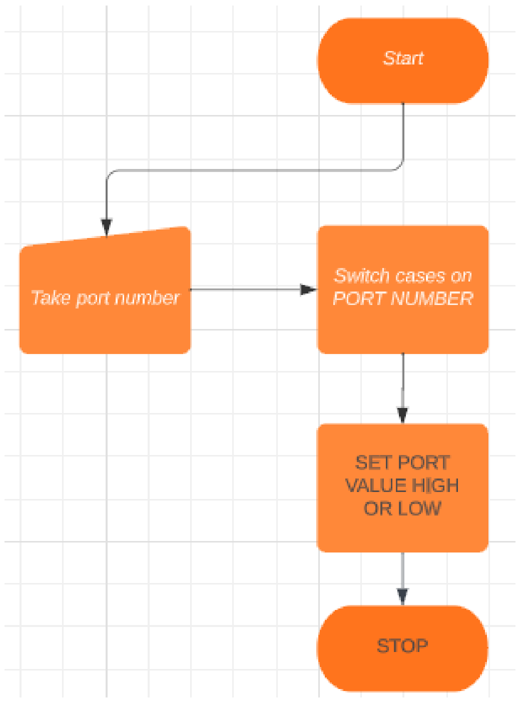
void MDIO_voidSetPortDirection

(u8 Copy_u8Port , u8 Copy_u8Dir)

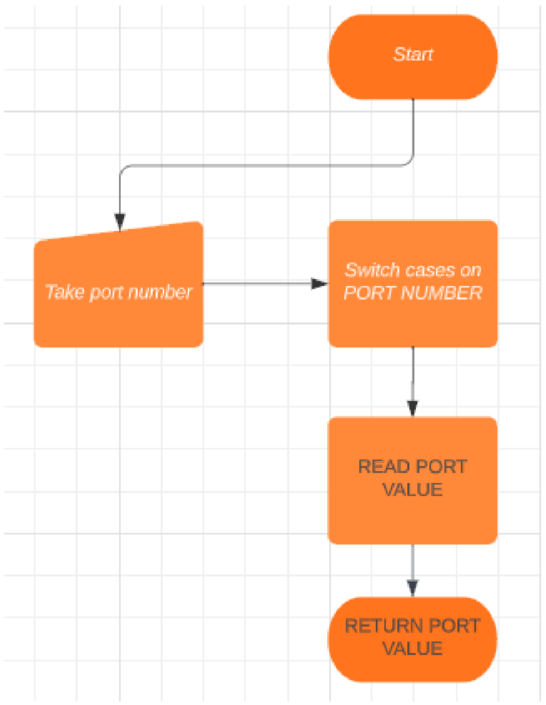


void MDIO_voidSetPortValue

(u8 Copy_u8Port , u8 Copy_u8Value)

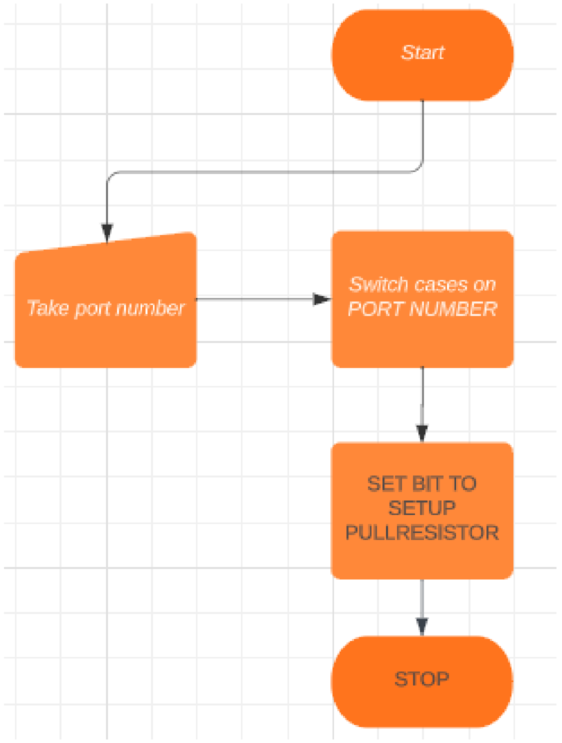


u8 MDIO_u8GetPortValue (u8 Copy_u8Port)



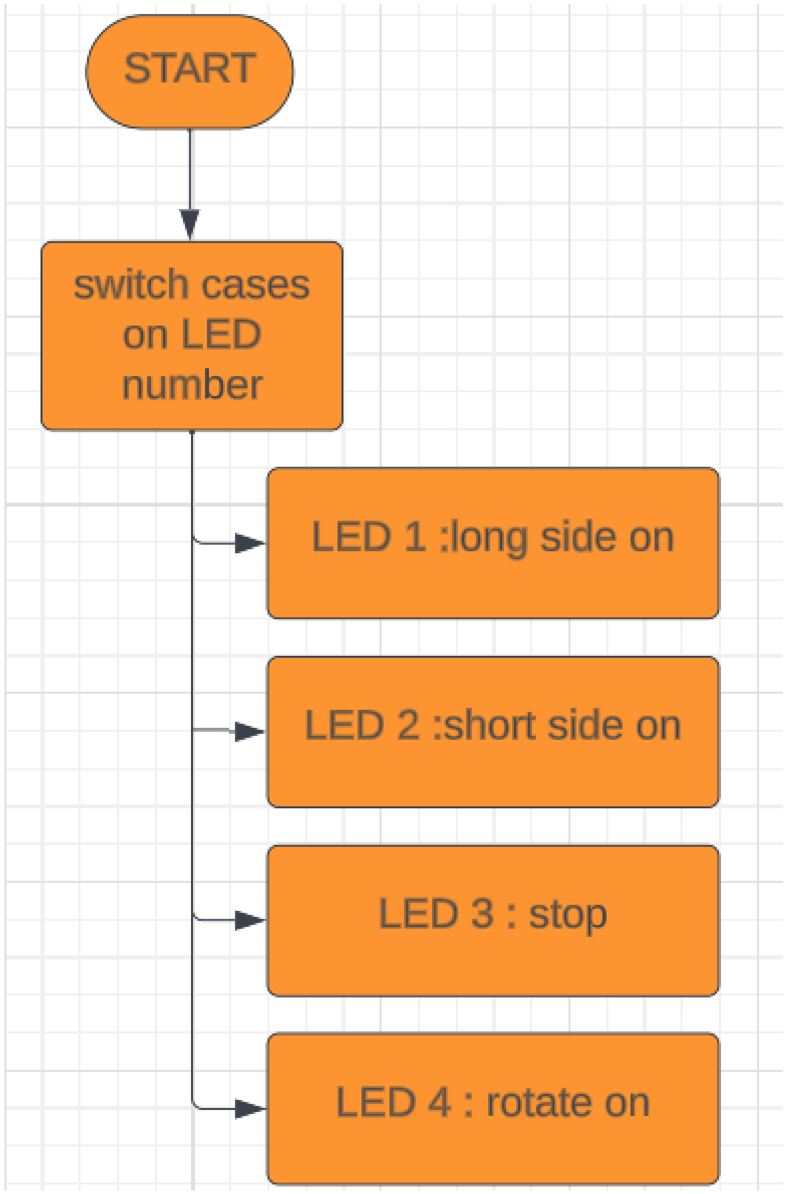
void MDIO_VoidSetPullupResistor

(u8 Copy_u8Port , u8 Copy_u8Pin)



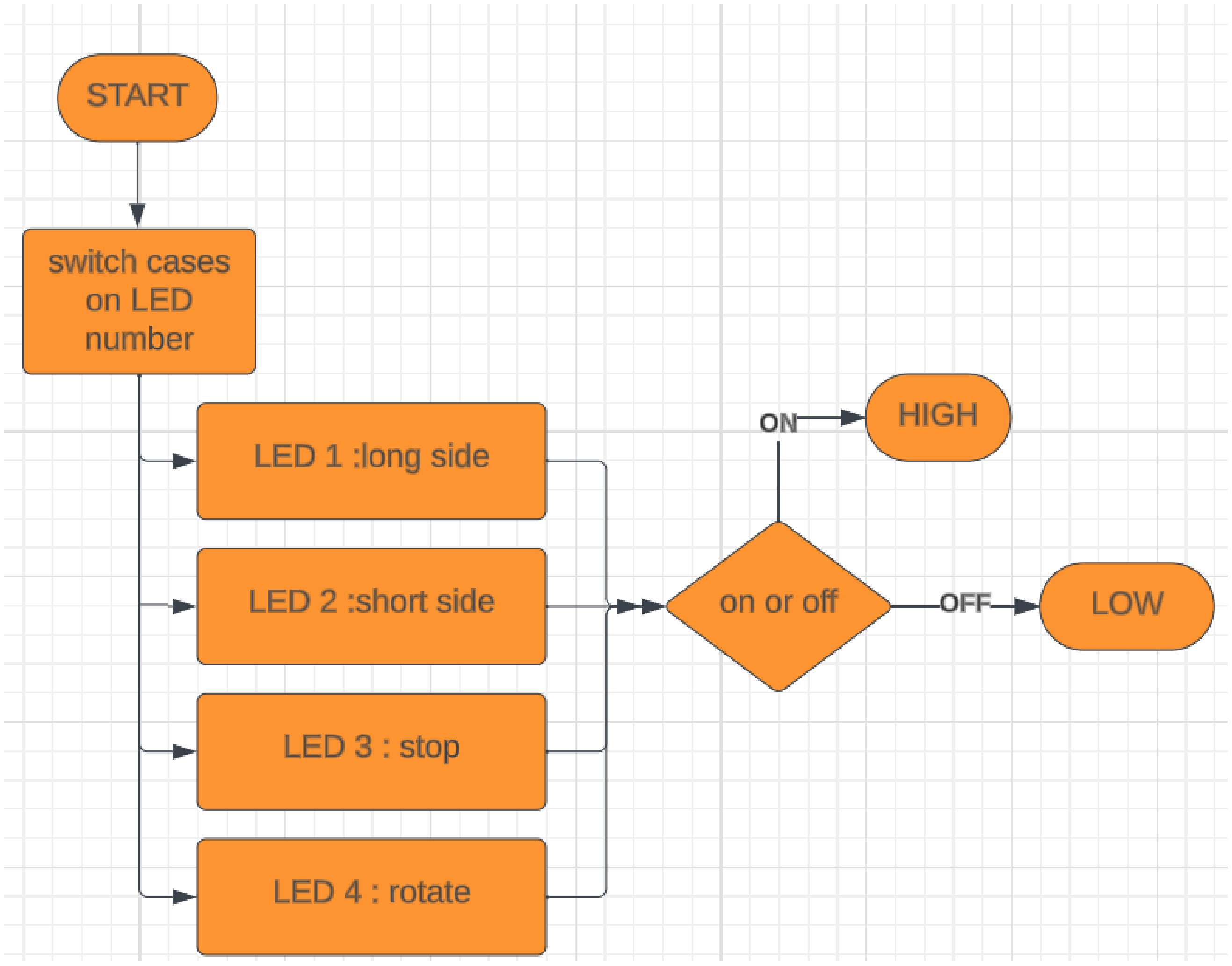
3.2.2 LED:

void HLED_ledInit(u8 Copy_u8ledNum)



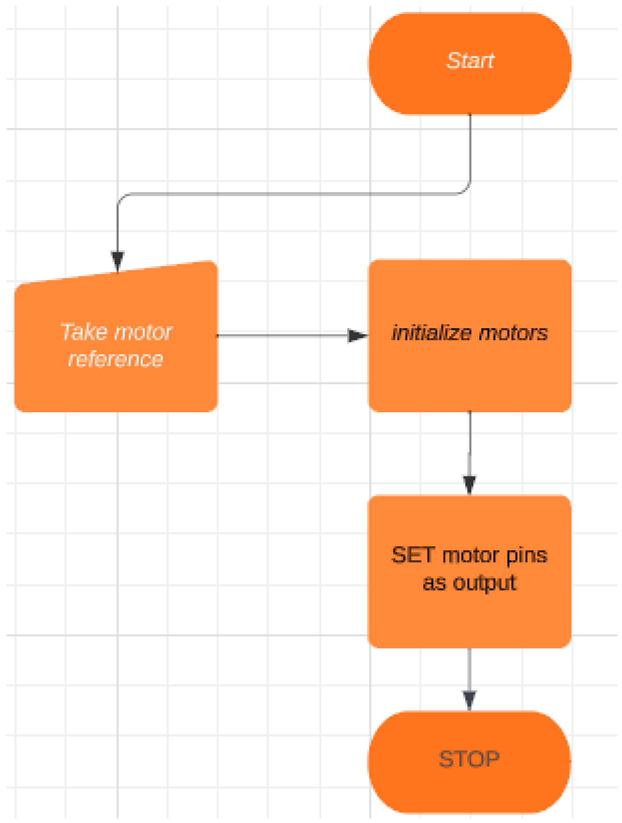
-void HLED_ledOn (u8 Copy_u8ledNum)

-void HLED_ledOff(u8 Copy_u8ledNum)

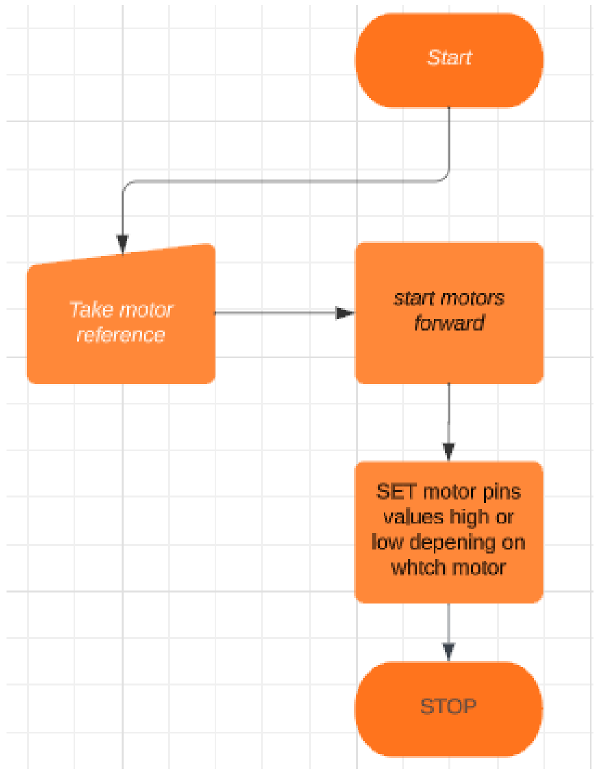


3.2.3 MOTOR:

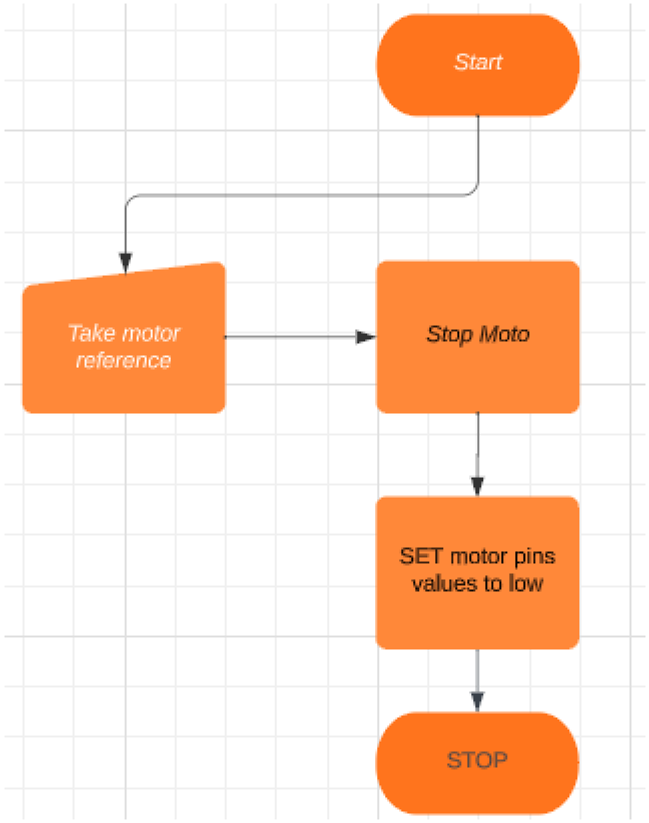
Init function



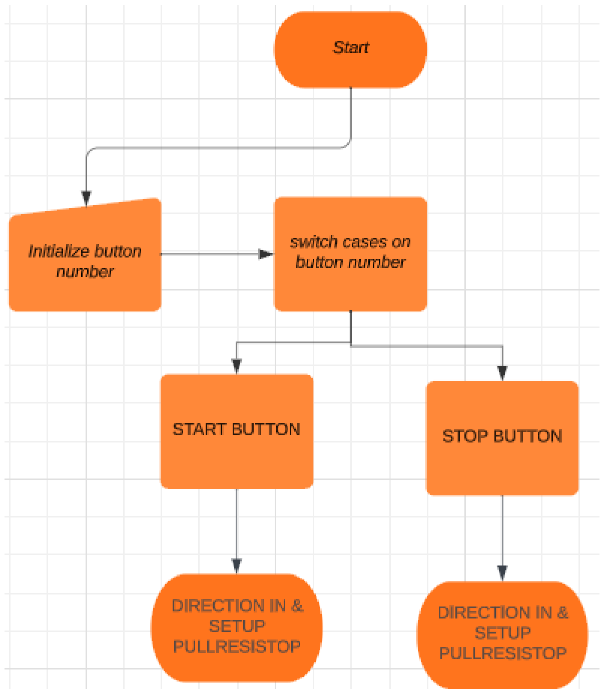
start forward function



Stop function

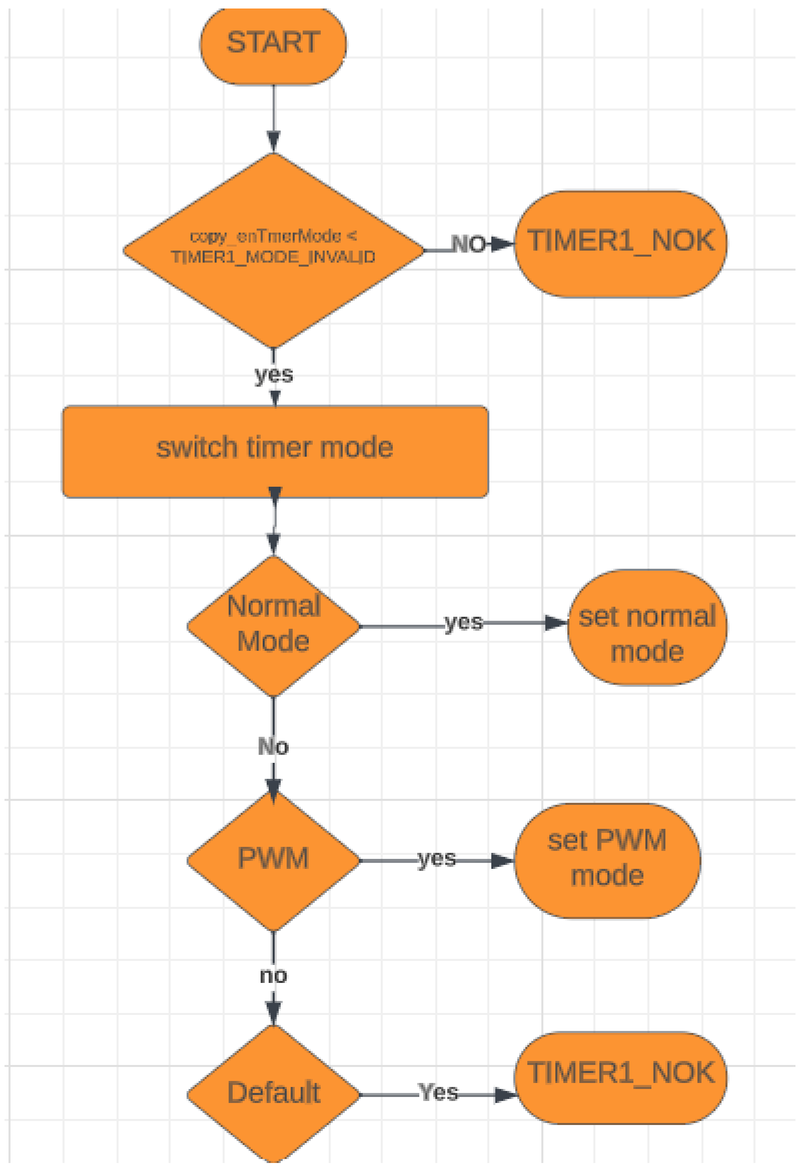


3.2.4 BUTTON:

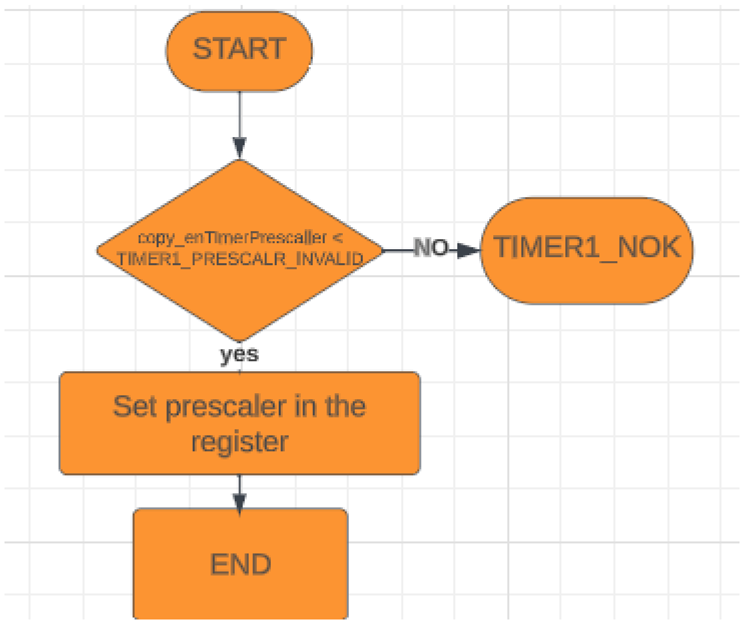

Start
Initialize button number
switch cases on button number
START BUTTON
STOP BUTTON
DIRECTION IN & SETUP PULLRESISTOP
DIRECTION IN & SETUP PULLRESISTOP

3.2.5 PWM:

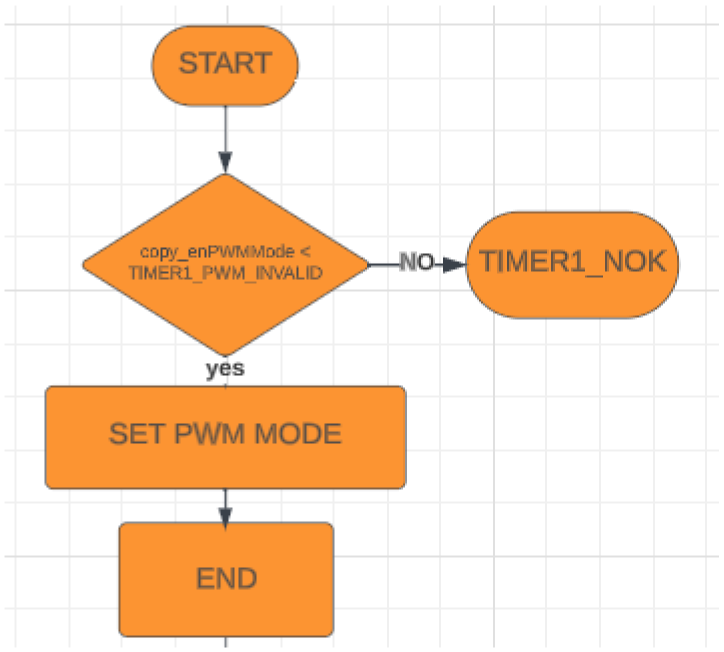
enu_timer1Status_t Timer1_enuInit (enu_timer1Mode_t copy_enTmerMode)



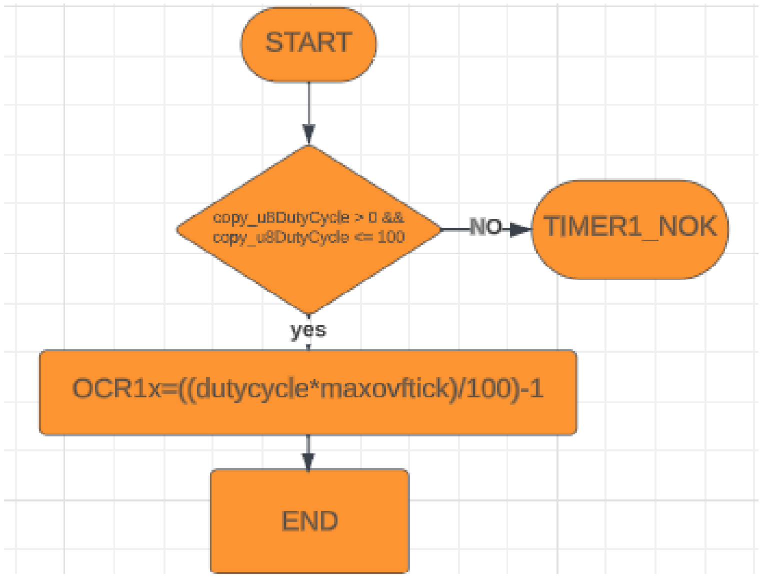
enu_timer1Status_t Timer1_enuSetPrescallar(enu_timer1Prescalar_t copy_enTimerPrescaller)


START
copy_enTimerPrescaller < TIMER1_PRESCALR_INVALID
NO
TIMER1_NOK
yes
Set prescaler in the register
END

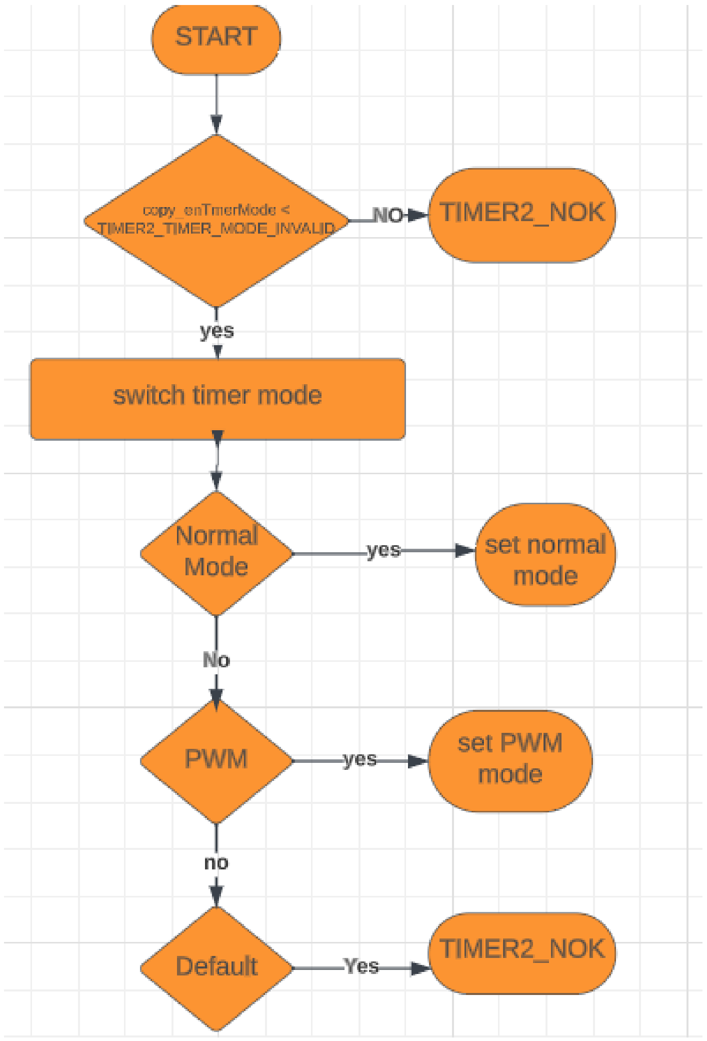
enu_timer1Status_t Timer1_enuFastPWMInit(enu_pwm1Mode_t copy_enPWMMode)



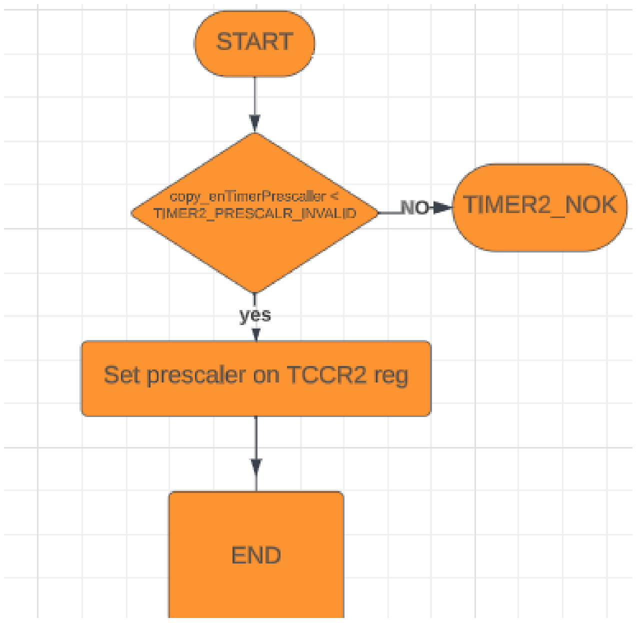
enu_timer1Status_t Timer1_enuPWMGenerate(Uchar8_t copy_u8DutyCycle)


START
copy_u8DutyCycle > 0 && copy_u8DutyCycle <= 100
NO
TIMER1_NOK
yes
OCR1x=((dutycycle*maxovftick)/100)-1
END

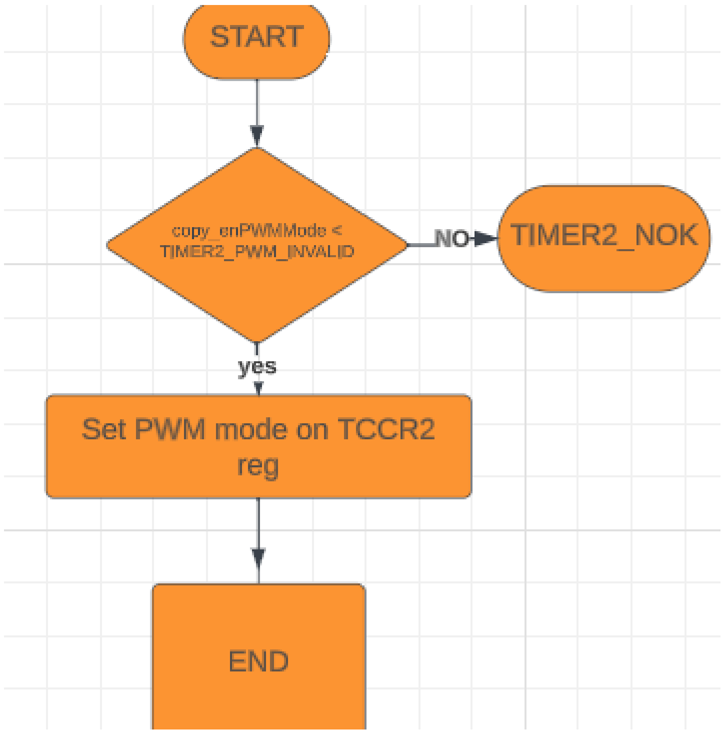
enu_timer2Status_t Timer2_enuInit (enu_timer2Mode_t copy_enTmerMode)



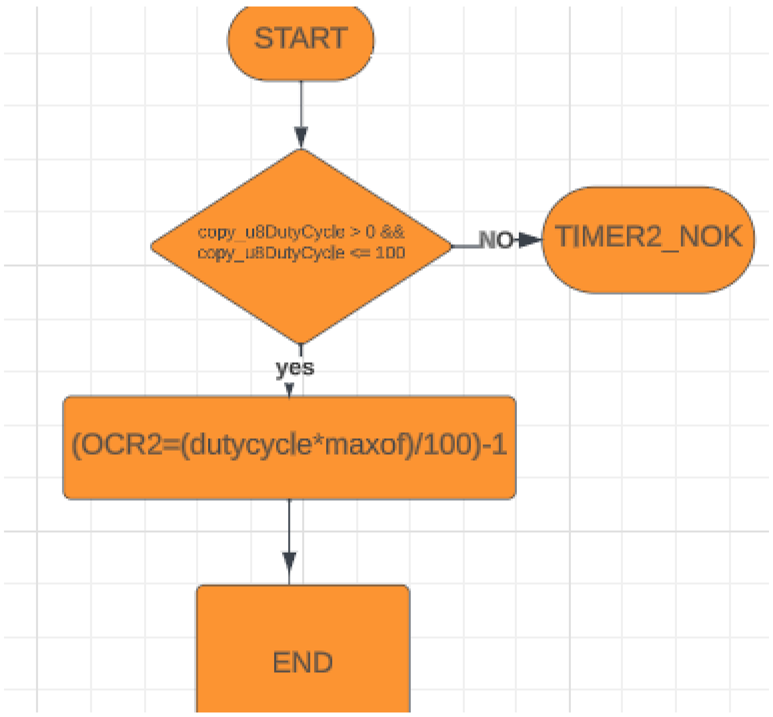
enu_timer2Status_t Timer2_enuSetPrescallar(enu_timerPrescalar_t copy_enTimerPrescaller)


START
copy_enTimerPrescaller < TIMER2_PRESCALR_INVALID
NO
TIMER2_NOK
yes
Set prescaler on TCCR2 reg
END

enu_timer2Status_t Timer2_enuFastPWMInit(enu_pwmMode_t copy_enPWMMode)

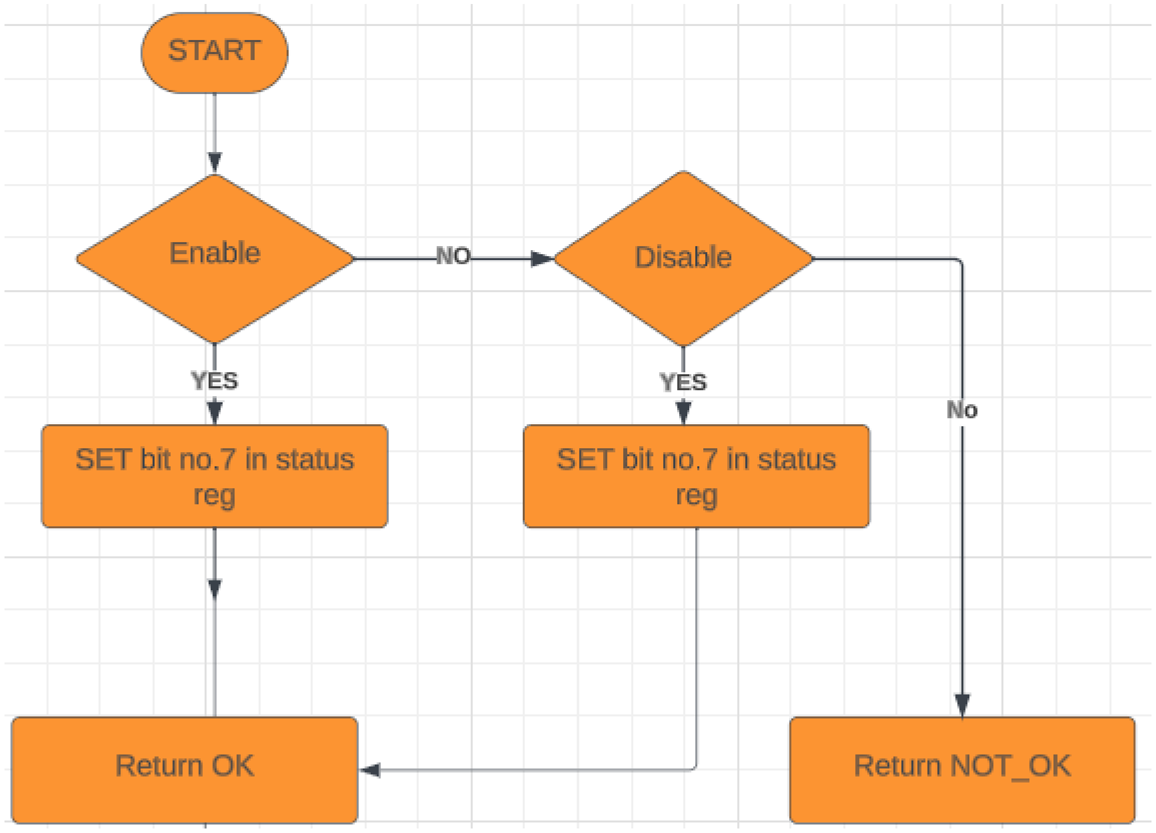

START
copy_enPWMMode < TIMER2_PWM_INVALID
NO
TIMER2_NOK
yes
Set PWM mode on TCCR2 reg
END

enu_timer2Status_t Timer2_enuPWMGenerate(Uchar8_t copy_u8DutyCycle)

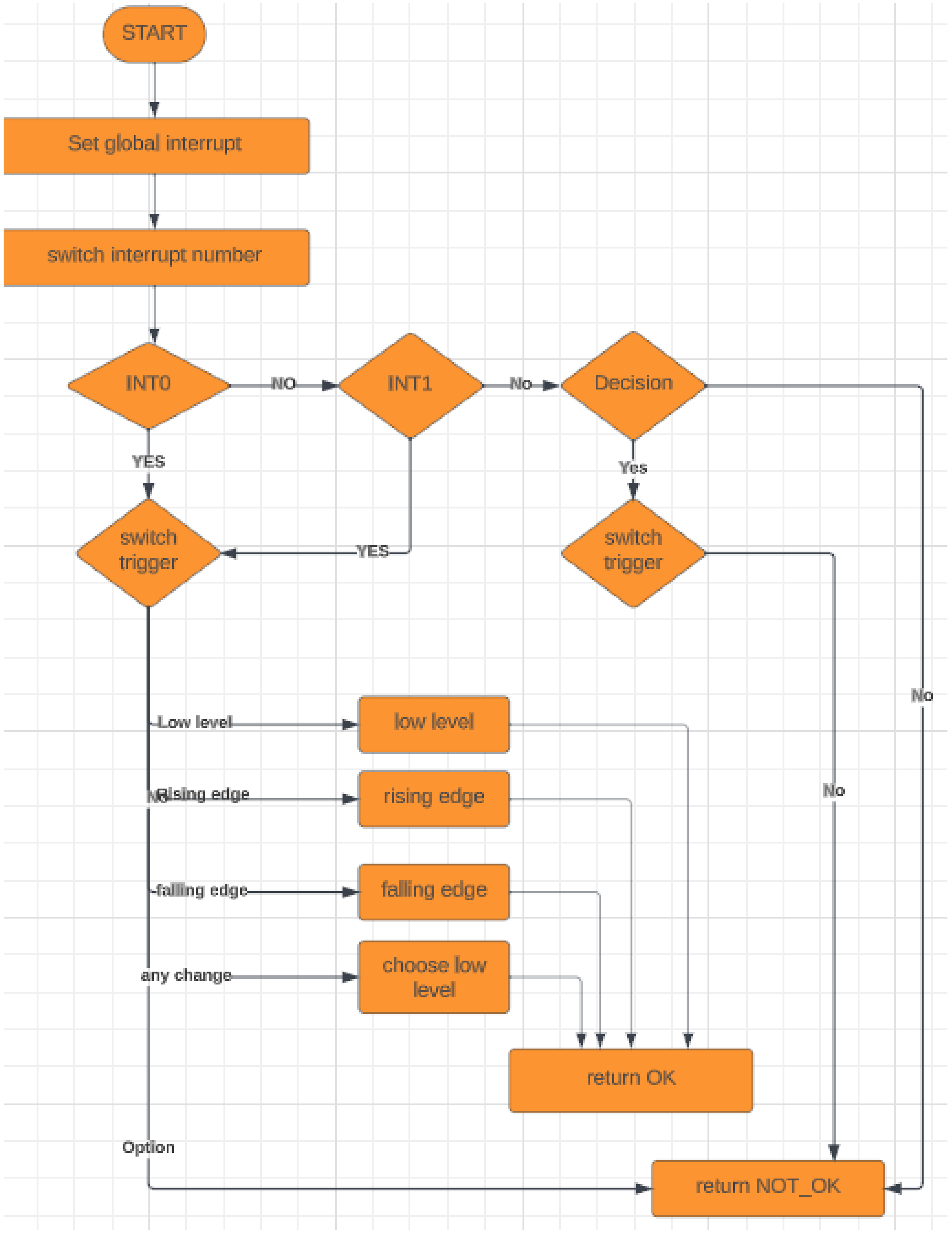

START
copy_u8DutyCycle > 0 && copy_u8DutyCycle <= 100
NO
TIMER2_NOK
yes
(OCR2=(dutycycle*maxof)/100)-1
END

3.2.6 External Interrupt:

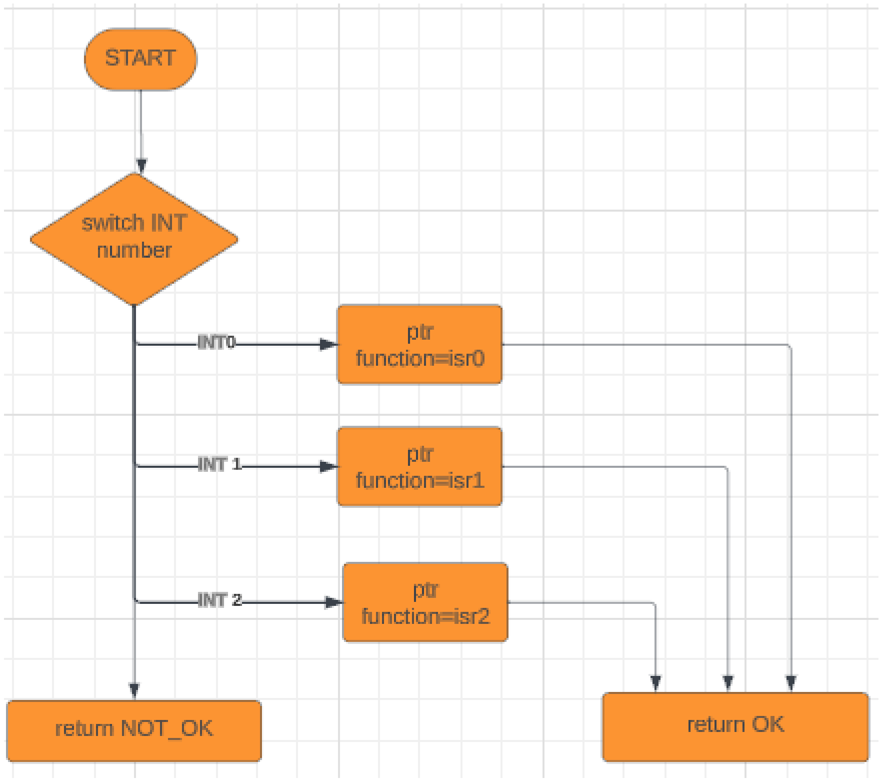
EN_EXTINT_ERROR SET_GLOBAL_INTERRUPT(EN_GLOBAL_INT state)



EN_EXTINT_ERROR EXTINT_init(EN_EXINT_NUMBER INTx ,EN_Sense_Control INTxSense)

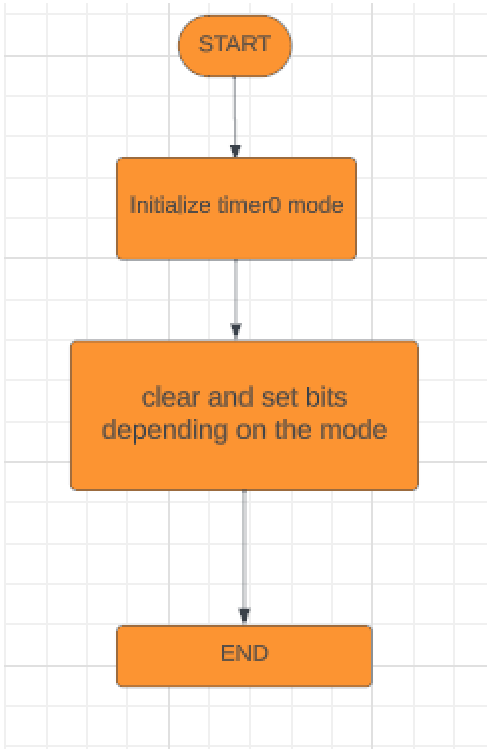


EN_EXTINT_ERROR EXTINT_CallBack(EN_EXINT_NUMBER INTx,void(*ptrfunc)(void))

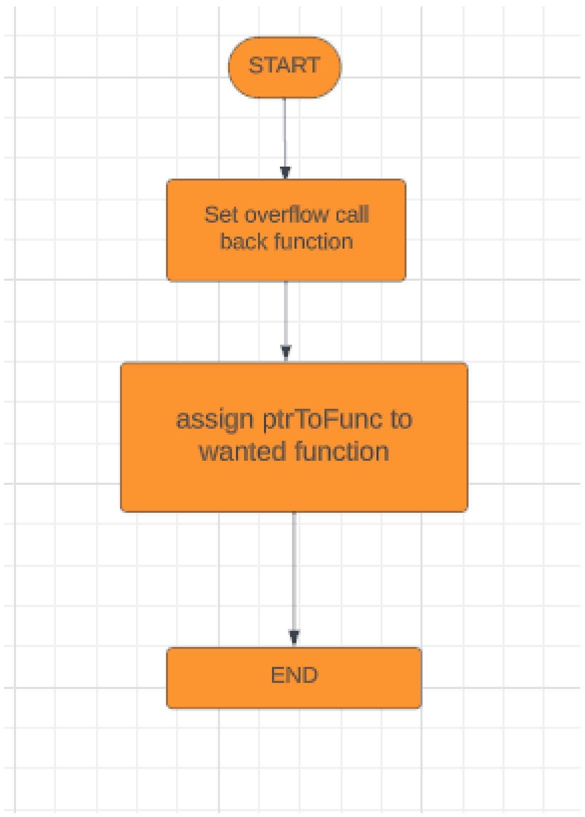


3.2.7 TIMER:

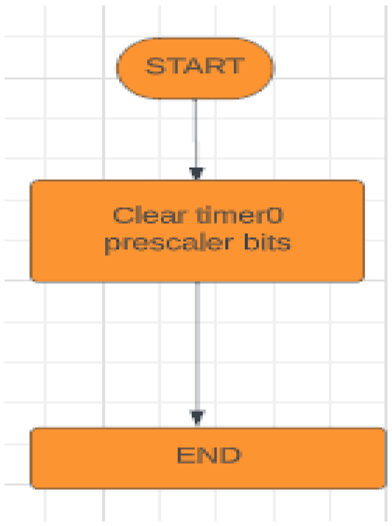
Timer initialize function



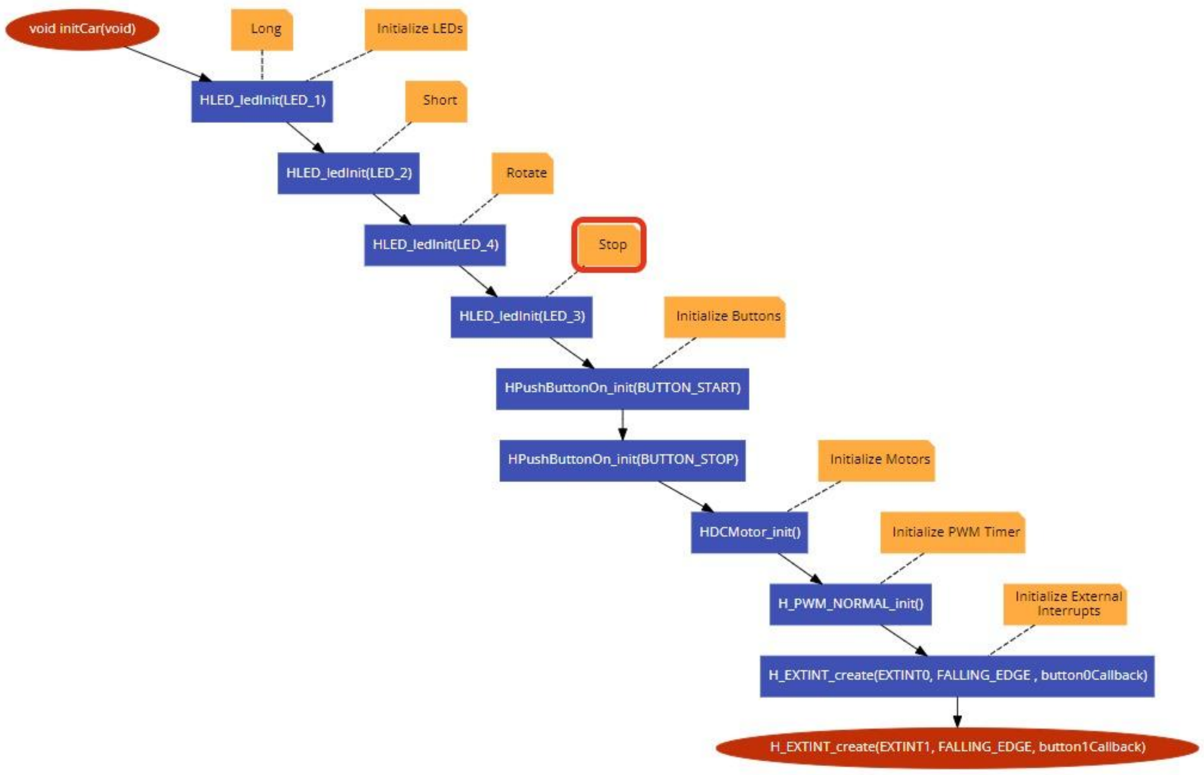
setCallBack function


START
Set overflow call back function
assign ptrToFunc to wanted function
END

Timer stop function



InitCar function:


void initCar(void)
Long
Initialize LEDs
HLED_ledInit(LED_1)
Short
HLED_ledInit(LED_2)
Rotate
HLED_ledInit(LED_4)
Stop
HLED_ledInit(LED_3)
Initialize Buttons
HPushButtonOn_init(BUTTON_START)
HPushButtonOn_init(BUTTON_STOP)
Initialize Motors
HDCMotor_init()
Initialize PWM Timer
H_PWM_NORMAL_init()
Initialize External Interrupts
H_EXTINT_create(EXTINT0, FALLING_EDGE , button0Callback)
H_EXTINT_create(EXTINT1, FALLING_EDGE, button1Callback)