

**Moving Car Project**

**Team Members:**

|  |
| --- |
| Alaa Ibrahim |
| Bassel Yasser |
| Sharpel Malek |
| Sherif Khadr |

Contents

[INTRODUCTION 1](#_Toc132230076)

[High Level Design 2](#_Toc132230077)

[**01)** **Layered Architecture** 2](#_Toc132230078)

[**02)** **Modules Description** 3](#_Toc132230079)

[**03)** **Drivers’ Documentation** 4](#_Toc132230080)

[Low Level Design 14](#_Toc132230081)

[Figure 1: Project Layered Architecture 2](file:///D:\02_workspace\MovingCarProject\Moving_Car.docx#_Toc132213547)

# INTRODUCTION

In our project we have a four-driving wheel robot and moves in a rectangular shape, on this document we will illustrate the module design and how they integrate with each other, we’ll also discuss the used APIs in more detail and providing the flowchart for each function in each module, and making layered architecture.

In this project we used PWM for controlling motor speed, TIMER on normal mode for controlling the duration of motor, DIO for GPIO Pins and External Interrupt Module for Start / Stop Motor

# High Level Design

## **Layered Architecture**

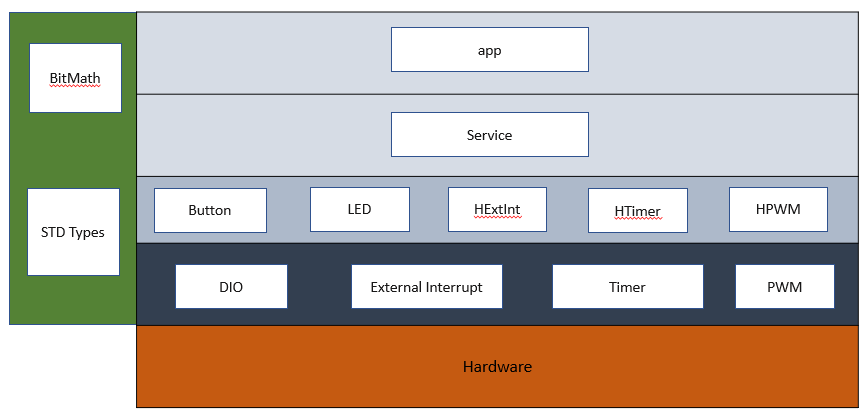


Figure : Project Layered Architecture

## **Modules Description**

**MCAL Layer:**

* **DIO:**
* **External Interrupt:** this module takes place in MCAL layer. it controlsthree external hardware interrupts on pins PD2, PD3, and PB2 which are referred to as INT0, INT1, and INT2 respectively. External interrupts can be level-triggered or edge-triggered.  
  We program this triggering. INT0 and INT1 can be level-triggered and edge-triggered whereas INT2 can be only edge-triggered.
* **Timer:**
* **PWM:** a modulation technique that generates variable-width pulses to represent the amplitude of an analog input signal, and directly communicates to hardware.
* **PWM NORMAL:** it generates the functionality of PWM using the normal mode of timer.

**HAL Layer:**

* **Button:**
* **LED**
* **HPWM:** Is in Middle layer which Application can communicate to MCAL layer**.**
* **HTimer:**
* **HExtInt:** In the HAL layer, which the Application can communicate to the External interrupt module.
* **HPWM\_NORMAL:** In the HAL layer, which the Application can communicate to the PWM NORMAL module.

**Service Layer:**

**Application Layer:**

## **Drivers’ Documentation**

**MCAL Layer:**

* **DIO:**
* **External Interrupt:**

**// EXT\_INT TYPEDEFS**

**typedef enum {EXTINT\_OK=0,EXTINT\_NOT\_O }EN\_EXTINT\_ERROR;**

**typedef enum {LOW\_LEVEL=0,FALLING\_EDGE,RISING\_EDGE,ANY\_LOGICAL\_CHANGE}EN\_Sence\_Control;**

**typedef enum {EXTINT0=0,EXTINT1,EXTINT2}EN\_EXINT\_NUMBER;**

**typedef enum {DISABLE=0,ENABLE}EN\_GLOBAL\_INT;**

**// EXT\_INT prototypes**

**/\*Description : This function initializes the GLOBAL\_INTERRUPT**

**ARGS : takes the state ( ENABLE OR DISABLE )**

**return : return EXTINT\_OK if the PIN initializes correctly, EXTINT\_NOT\_OK otherwise\*/**

**EN\_EXTINT\_ERROR SET\_GLOBAL\_INTERRUPT(EN\_GLOBAL\_INT state);**

**/\*Description: This function initializes the external interrupt number and its detecting type**

**ARGS : takes the EXINT\_NUMBER( INT0,INT1 OR INT2) and sence control.**

**return : EXTINT\_OK if the EXINT\_NUMBER initializes correctly, EXTINT\_NOT\_OK otherwise\*/**

**EN\_EXTINT\_ERROR EXTINT\_init(EN\_EXINT\_NUMBER INTx ,EN\_Sence\_Control INTxSense);**

**/\*Description: This function takes the external interrupt number and initializes the call-back function.**

**ARGS : takes the EXINT\_NUMBER( INT0,INT1 OR INT2) and pointer to the function we want to execute.**

**return : return EXTINT\_OK if the EXINT\_NUMBER initializes correctly, EXTINT\_NOT\_OK otherwise\*/**

**EN\_EXTINT\_ERROR EXTINT\_CallBack(EN\_EXINT\_NUMBER INTx,void(\*ptrfunc)(void));**

* **Timer:**
* **PWM:** 
  + **enu\_timer1Status\_t Timer1\_enuInit (enu\_timer1Mode\_t)**

**Author :** Bassel Yasser Mahmoud

**Function Name:** Timer1\_enuInit

**Function Description**: Initialize Timer1 to Fast PWM Mode

**Arguments:** copy\_enTmerMode {TIMER1\_OVF\_MODE, TIMER1\_FAST\_PWM\_MODE,}

**Return:** enu\_timer1Status\_t {TIMER1\_OK or TIMER1\_NOK}

* + **enu\_timer1Status\_t Timer1\_enuSetPrescallar(enu\_timer1Prescalar\_t)**

**Author**  : Bassel Yasser Mahmoud

**Function Nam**e : Timer1\_enuSetPrescallar

**Function Description** : Set Prescaller

**Arguments:** Timer1\_enuSetPrescallar {TIMER1\_PRE\_1, TIMER1\_PRE\_64, TIMER1\_PRE\_256}

**Return** : enu\_timer1Status\_t {TIMER1\_OK or TIMER1\_NOK}

* + **enu\_timer1Status\_t Timer1\_enuFastPWMInit(enu\_pwm1Mode\_t)**

**Author**  : Bassel Yasser Mahmoud

**Function Name** : Timer1\_enuFastPWMInit

**Function Description :** Set PWM Mode

**Arguments: c**opy\_enPWMMode {TIMER1\_PWM\_NORMAL, TIMER1\_PWM\_CLR\_ON\_CMP, TIMER1\_PWM\_SET\_ON\_CMP}

**Return**  : enu\_timer1Status\_t {TIMER1\_OK or TIMER1\_NOK}

* + **enu\_timer1Status\_t Timer1\_enuPWMGenerate (Uchar8\_t)**

**Author**  : Bassel Yasser Mahmoud

**Function Name** : Timer1\_enuPWMGenerate

**Function Description** : Generate PWM

**Arguments** : copy\_u8DutyCycle {1 ~ 100}

**Return**  : enu\_timer1Status\_t {TIMER1\_OK or TIMER1\_NOK}

* + **enu\_timer2Status\_t Timer2\_enuInit (enu\_timer2Mode\_t)**

**Author :** Bassel Yasser Mahmoud

**Function Name:** Timer1\_enuInit

**Function Description**: Initialize Timer2 to Fast PWM Mode

**Arguments:** copy\_enTmerMode {TIMER2\_OVF\_MODE, TIMER2\_FAST\_PWM\_MODE,}

**Return:** enu\_timer1Status\_t {TIMER2\_OK or TIMER2\_NOK}

* + **enu\_timer2Status\_t Timer2\_enuSetPrescallar(enu\_timer2Prescalar\_t)**

**Author**  : Bassel Yasser Mahmoud

**Function Nam**e : Timer2\_enuSetPrescallar

**Function Description** : Set Prescaller

**Arguments:** Timer2\_enuSetPrescallar {TIMER2\_PRE\_1, TIMER2\_PRE\_64, TIMER2\_PRE\_256}

**Return** : enu\_timer2Status\_t {TIMER2\_OK or TIMER2\_NOK}

* + **enu\_timer2Status\_t Timer2\_enuFastPWMInit(enu\_pwm2Mode\_t)**

**Author**  : Bassel Yasser Mahmoud

**Function Name** : Timer2\_enuFastPWMInit

**Function Description :** Set PWM Mode

**Arguments: c**opy\_enPWMMode {TIMER2\_PWM\_NORMAL, TIMER2\_PWM\_CLR\_ON\_CMP, TIMER2\_PWM\_SET\_ON\_CMP}

**Return**  : enu\_timer2Status\_t {TIMER2\_OK or TIMER2\_NOK}

* + **enu\_timer2Status\_t Timer2\_enuPWMGenerate (Uchar8\_t)**

**Author**  : Bassel Yasser Mahmoud

**Function Name** : Timer2\_enuPWMGenerate

**Function Description** : Generate PWM

**Arguments** : copy\_u8DutyCycle {1 ~ 100}

**Return**  : enu\_timer2Status\_t {TIMER2\_OK or TIMER2\_NOK}

* **PWM NORMAL:**

**/\*Description : This function selects the normal mode and enables the GLOBAL\_INTERRUPT and overflow interrupt for timer2**

**ARGS : void**

**return : void\*/**

**void timer2\_init(void);**

**/\*Description : This function selects the prescaler (clk/1024). the timer start counting once we call this function.**

**ARGS : void**

**return : void\*/**

**void timer2\_start(void);**

**/\*Description : This function selects the no clock source option. the timer stop counting once we call this function.**

**ARGS : void**

**return : void\*/**

**void timer2\_stop(void);**

**/\*Description : This function calculate the on time based on duty cycle we need .**

**ARGS : takes the duty cycle**

**return : void\*/**

**void timer2\_set\_pwm\_normal(uint8\_t a\_dutycycle);**

**HAL Layer:**

* **Button:**
* **LED:**
* **HPWM:** 
  + **enu\_pwmStatus\_t pwm\_enInit(void)**

**Author** : Bassel Yasser Mahmoud

**Function Name** : pwm\_enInit

**Function Description :** Initialize PWM to be fast PWM, set prescaller, Set PWM Mode

**Arguments** : void

**Return**  : enu\_pwmStatus\_t {PWM\_OK or PWM\_NOK}

* + **enu\_pwmStatus\_t pwm\_enGenerate(Uchar8\_t)**

**Author** : Bassel Yasser Mahmoud

**Function Name** : pwm\_enGenerate

**Function Description :** Generate PWM signal

**Arguments** : Uchar8\_t

**Return**  : enu\_pwmStatus\_t {PWM\_OK or PWM\_NOK}

* **HTimer:**
* **HExtInt:**

/\*

Description : This function initializes the external interrupt number and it's detecting type and initialize call back function.

ARGS : takes the EXINT\_NUMBER( INT0,INT1 OR INT2) and sense control and and pointer to the function we want to execute when interrupt occurs.

return : return EXTINT\_OK if the EXINT\_NUMBER initializes correctly, EXTINT\_NOT\_OK otherwise

\*/

EN\_EXTINT\_ERROR H\_EXTINT\_create(EN\_EXINT\_NUMBER INTx ,EN\_Sence\_Control INTxSense,void(\*ptrfunc)(void));

* **HPWM\_NORMAL:**

/\*

Description : This function selects the normal mode and enables the GLOBAL\_INTERRUPT and overflow interrupt for timer2

ARGS : void

return : void

\*/

void H\_PWM\_NORMAL\_init(void);

/\*

Description : This function selects the no clock source option. the timer stop counting once we call this function.

ARGS : void

return : void

\*/

void H\_PWM\_NORMAL\_stop(void);

/\*

Description : This function calculate the on time based on duty cycle we need . then start the timer

ARGS : takes the duty cycle

return : void

\*/

void H\_PWM\_NORMAL\_setDutyCycle(Uchar8\_t dutycycle);

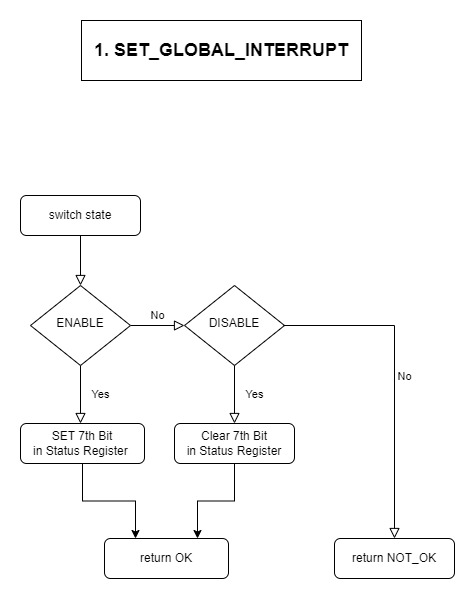
**Service Layer:**

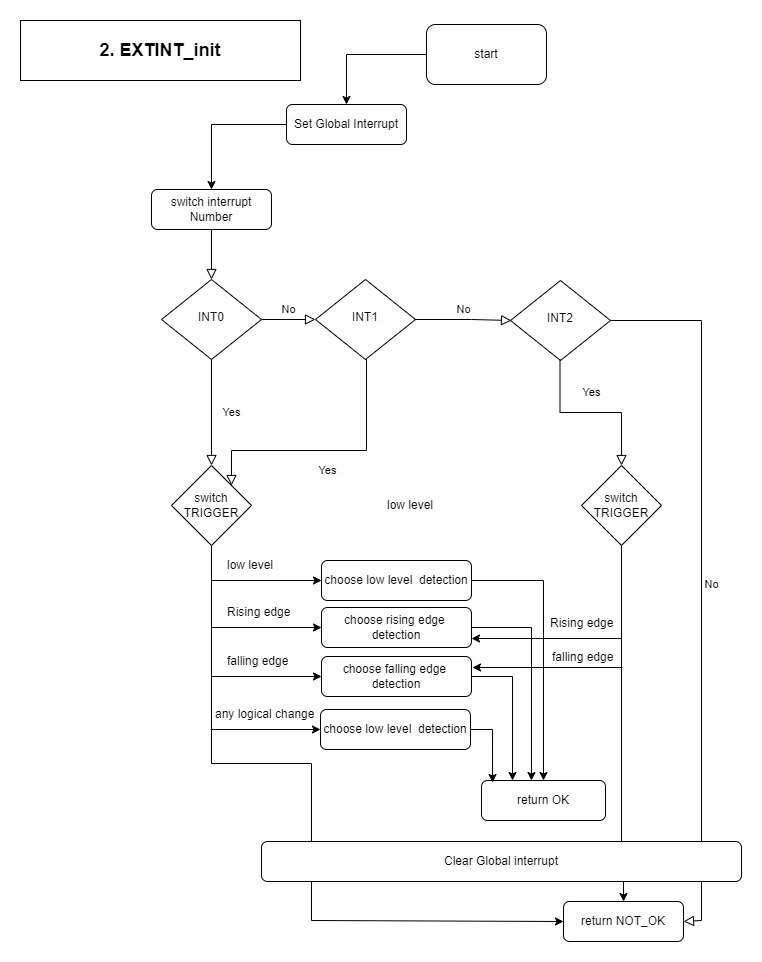
**Application Layer:**

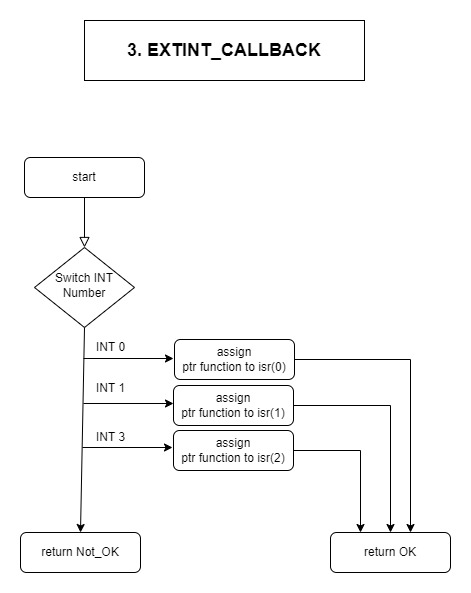
# Low Level Design

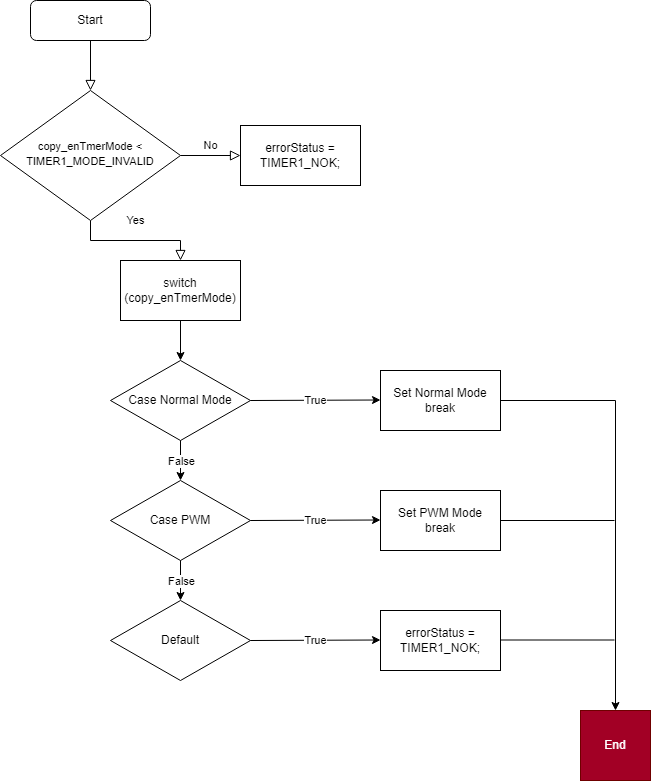
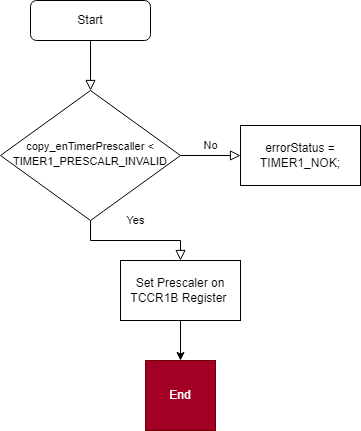
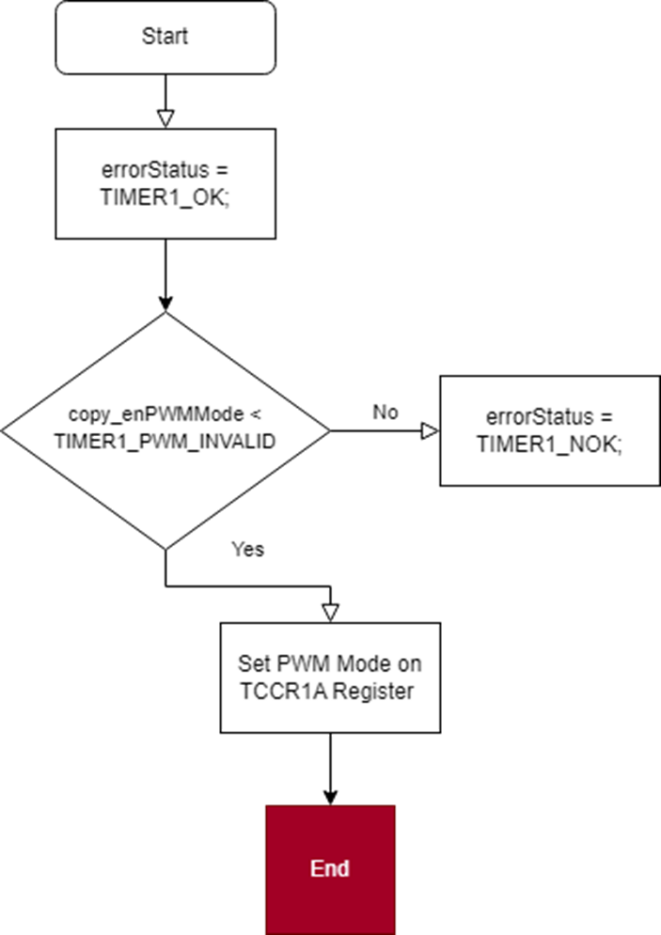
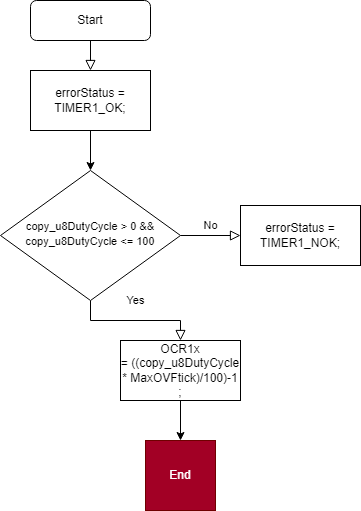
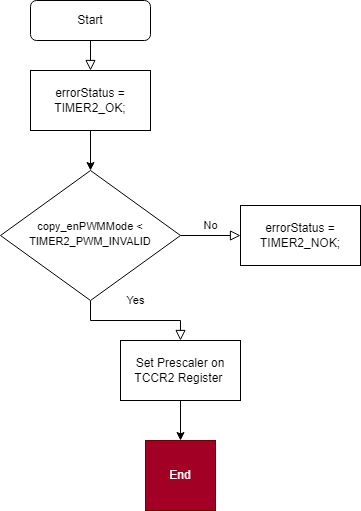
**MCAL Layer:**

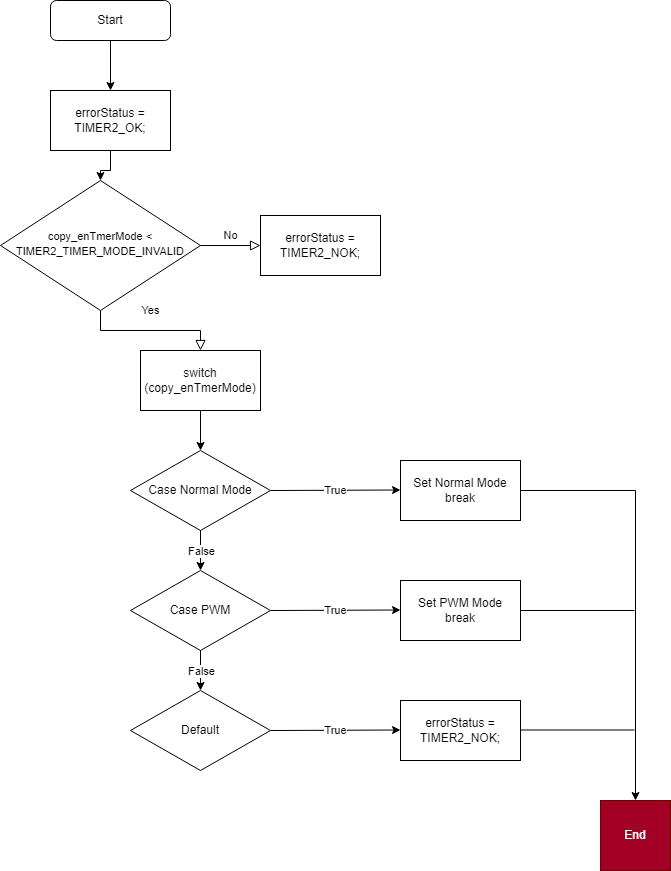
* **DIO:**
* **External Interrupt:**



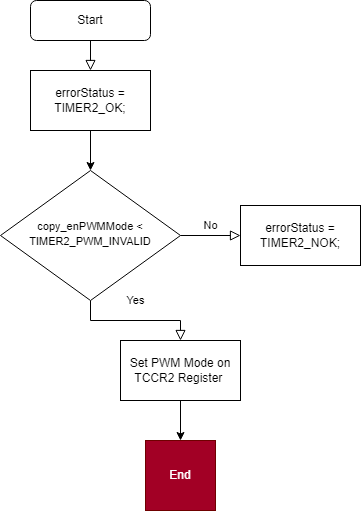


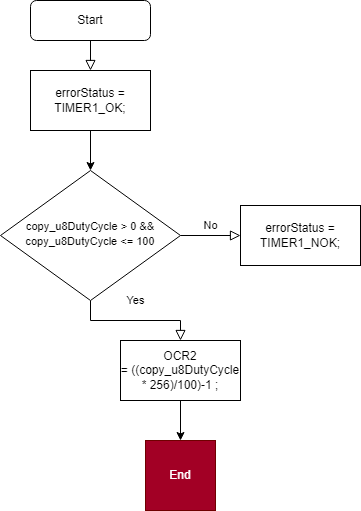


* **Timer:**
* **PWM:** 
  + **enu\_timer1Status\_t Timer1\_enuInit (enu\_timer1Mode\_t)**
  + **enu\_timer1Status\_t Timer1\_enuSetPrescallar(enu\_timer1Prescalar\_t)**
  + **enu\_timer1Status\_t Timer1\_enuFastPWMInit(enu\_pwm1Mode\_t)**
  + **enu\_timer1Status\_t Timer1\_enuPWMGenerate (Uchar8\_t)**
  + **enu\_timer2Status\_t Timer2\_enuSetPrescallar(enu\_timer2Prescalar\_t)**
  + **enu\_timer2Status\_t Timer2\_enuInit (enu\_timer2Mode\_t)**

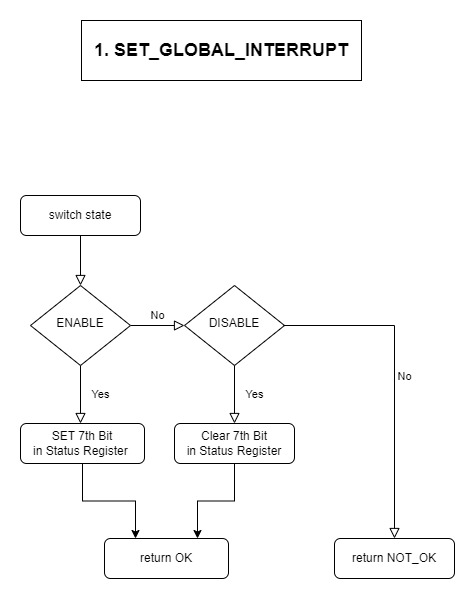
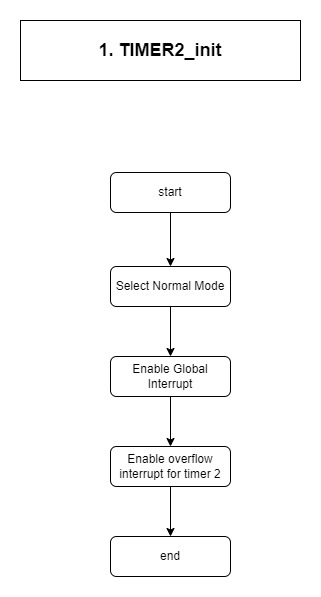
****

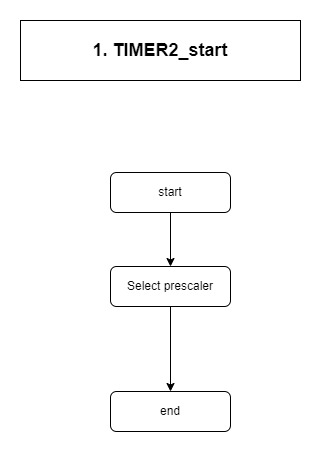
* + **enu\_timer2Status\_t Timer2\_enuFastPWMInit(enu\_pwm2Mode\_t)**

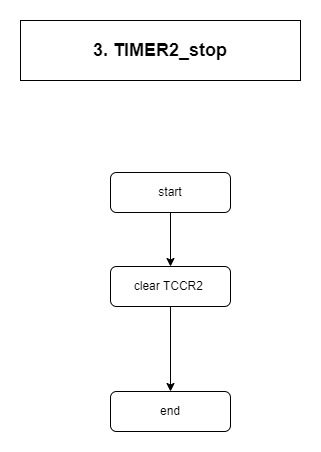
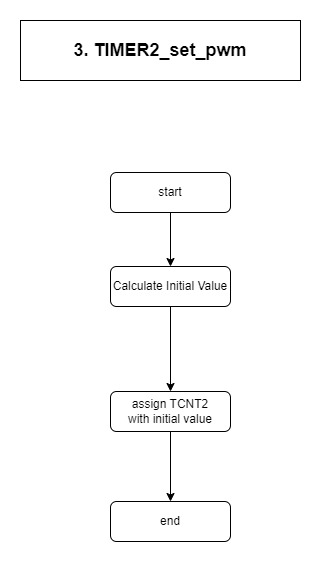
****

****

* + **enu\_timer2Status\_t Timer2\_enuPWMGenerate (Uchar8\_t)**
* **PWM Normal**

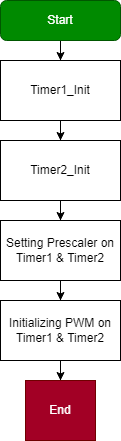




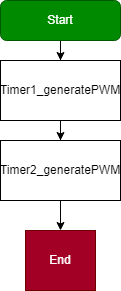


**HAL Layer:**

* **Button:**
* **LED:**
* **HPWM:** 
  + **enu\_pwmStatus\_t pwm\_enInit(void)**



* + **enu\_pwmStatus\_t pwm\_enGenerate(Uchar8\_t)**

****

* **HTimer:**
* **HExtInt:**

**Service Layer:**

**Application Layer:**