Moving Car System Design

By : Team 4

# 

# 

# 

# 

# 

# 

# 

# **Table Of Content:**

1. Detailed Requirements
2. Layered architecture
3. System module
4. Module architecture
5. MCAL APIs  
   3.2.1 : DIO API  
    3.2.1.1 : Flowchart  
    3.2.1.2 : Type definitions  
    3.2.1.3 : Services  
   3.2.2 : Timer API  
    3.2.2.1 : Flowchart  
    3.2.2.2 : Type definitions  
    3.2.2.3 : Services  
   3.2.3 : Timer API  
    3.2.3.1 : Flowchart  
    3.2.3.2 : Type definitions  
    3.2.3.3 : Services  
   3.2.4 : External Interrupt API  
    3.2.4.1 : Flowchart  
    3.2.4.2 : Type definitions  
    3.2.4.3 : Services
6. HAL APIs  
   3.3.1 : LED API  
    3.3.1.1 : Flowchart  
    3.3.1.2 : Type definitions  
    3.3.1.3 : Services  
   3.3.2 : Motor API  
    3.3.2.1 : Flowchart  
    3.3.2.2 : Type definitions  
    3.3.2.3 : Services  
   3.3.3 : Car Control API  
    3.3.3.1 : Flowchart  
    3.3.3.2 : Type definitions  
    3.3.3.3 : Services  
   3.3.4 : Timer Manager API  
    3.3.4.1 : Flowchart  
    3.3.4.2 : Type definitions  
    3.3.4.3 : Services  
    3.3.5 : Interrupt Manager API  
    3.3.5.1 : Flowchart  
    3.3.5.2 : Type definitions  
    3.3.5.3 : Services  
    3.3.6 : Button API  
    3.3.6.1 : Flowchart  
    3.3.6.2 : Type definitions  
    3.3.6.3 : Services
7. APP APIs  
   3.4.1 : APP API  
    3.4.1.1 : Flowchart  
    3.4.1.2 : Type definitions  
    3.4.1.3 : Services

# 

# 

# 

# 

# 

# **1 : Detailed Requirements**

System Requirements:

1. The car starts initially from 0 speed

2. When PB1 is pressed, the car will move forward after 1 second

3. The car will move forward to create the longest side of the rectangle for 3 seconds with 50% of its maximum speed

4. After finishing the first longest side, the car will stop for 0.5 seconds, rotate 90 degrees to the right, and stop for 0.5 second

5. The car will move to create the short side of the rectangle at 30% of its speed for 2 seconds

6. After finishing the shortest side, the car will stop for 0.5 seconds, rotate 90 degrees to the right, and stop for 0.5 second

7. Steps 3 to 6 will be repeated infinitely until you press the stop button (PB2)

8. PB2 acts as a sudden break, and it has the highest priority

9. LEDs Operations

1. LED1: On means moving forward on the long side

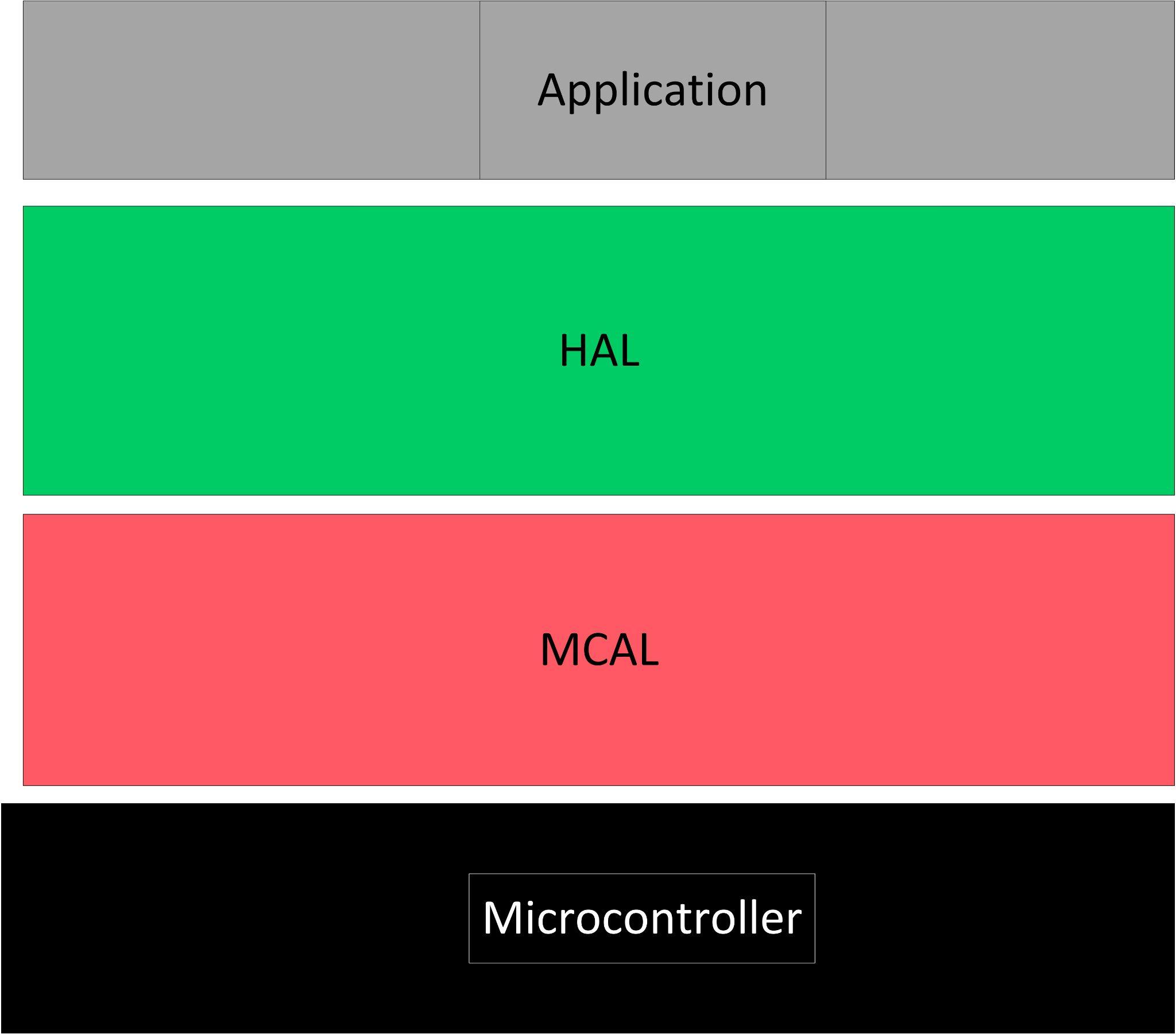
2. LED2: On means moving forward on the short side

3. LED3: On means stop

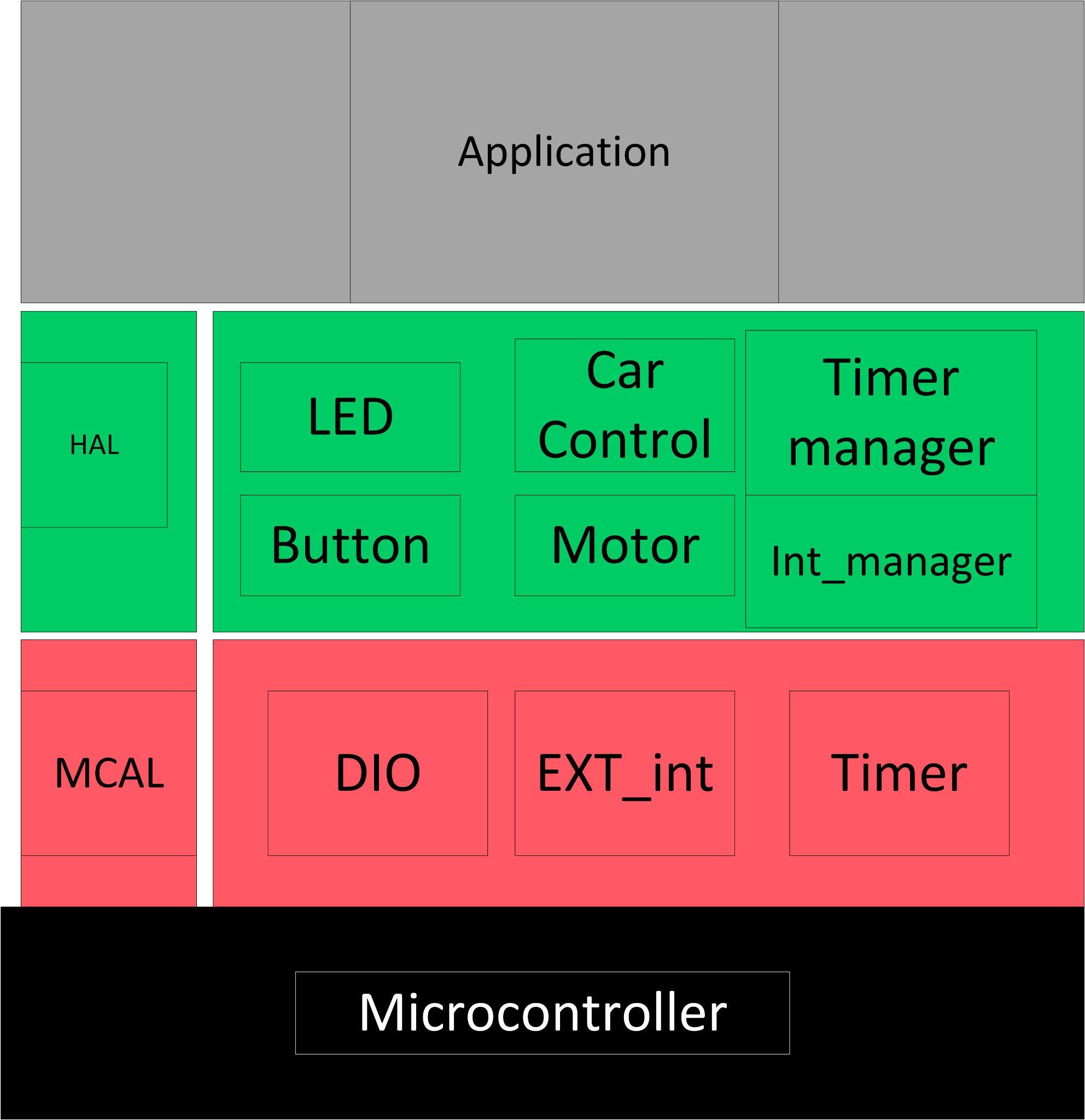
4. LED4: On means Rotating

# 

# **2 : Layered architecture**



# **3 : System modules**



## **3.2 : MCAL APIs**

### 3.2.1 : DIO API :

#### 3.2.1.1 :Flowcharts:

#### 

#### 

#### 

#### 3.2.1.2 : Type definitions:

* en\_dioPinsType

|  |  |
| --- | --- |
| Name | en\_dioPinsType |
| Type | Enumeration |
| Range | Shall contain all pins ID |
| Description | en\_dioPinsType |
| Available via | dio.h |

* en\_dioPortsType

|  |  |
| --- | --- |
| Name | en\_dioPortsType |
| Type | Enumeration |
| Range | Shall contain all ports ID |
| Description | en\_dioPortsType |
| Available via | dio.h |

* u8\_en\_dioErrors

|  |  |
| --- | --- |
| Name | u8\_en\_dioErrors |
| Type | Enumeration |
| Range | |  |  |  | | --- | --- | --- | | DIO\_E\_OK | 0x00 | DIO error OK | | DIO\_InvalidPin | 0x01 | DIO error, invalid pin number. | | DIO\_InvalidPort | 0x02 | DIO error, invalid port number. | |
| Description | u8\_en\_dioErrors |
| Available via | dio.h |

* u8\_en\_dioLevelType

|  |  |
| --- | --- |
| Name | u8\_en\_dioLevelType |
| Type | Enumeration |
| Range | |  |  |  | | --- | --- | --- | | STD\_LOW | 0x00 | Physical state 0V | | STD\_HIGH | 0x01 | Physical state 5V or 3.3V. | |
| Description | u8\_en\_dioLevelType |
| Available via | dio.h |

* u8\_en\_dioDirType

|  |  |
| --- | --- |
| Name | u8\_en\_dioDirType |
| Type | Enumeration |
| Range | |  |  |  | | --- | --- | --- | | STD\_INPUT | 0x00 | Set pin as input pin | | STD\_OUTPUT | 0x01 | Set pin as output pin | |
| Description | u8\_en\_dioDirType |
| Available via | dio.h |

#### 

#### 3.2.1.3 : Services affecting the hardware unit:

* DIO\_readPIN

|  |  |  |  |
| --- | --- | --- | --- |
| Service name | DIO\_readPIN | | |
| Syntax | u8\_en\_dioErrors DIO\_readPIN (  en\_dioPortsType port,  en\_dioPinsType pin,  uint8\_t\* value  ); | | |
| Parameters (in) | Port, pin | Channel ID | |
| value | Pointer to store the level | STD\_HIGH |
| STD\_LOW |
| Return | |  |  | | --- | --- | | u8\_en\_dioErrors | DIO\_E\_OK | | DIO\_InvalidPin | | DIO\_InvalidPort | | | |
| Description | This Function gets the level of the pin | | |

* This function shall return DIO\_InvalidPin if pin number is invalid.
* This function shall return DIO\_InvalidPort if port number is invalid.
* DIO\_writePIN

|  |  |  |  |
| --- | --- | --- | --- |
| Service name | DIO\_writePIN | | |
| Syntax | u8\_en\_dioErrors DIO\_writePIN (  en\_dioPortsType port, en\_dioPinsType pin, u8\_en\_dioLevelType state  ); | | |
| Parameters (in) | Port, pin | Channel ID | |
| state | Value to be set | STD\_HIGH |
| STD\_LOW |
| Return | |  |  | | --- | --- | | u8\_en\_dioErrors | DIO\_E\_OK | | DIO\_InvalidPin | | DIO\_InvalidPort | | | |
| Description | This Function sets the level of the pin | | |

* This function shall return DIO\_InvalidPin if pin number is invalid.
* This function shall return DIO\_InvalidPort if port number is invalid.
* DIO\_init

|  |  |  |  |
| --- | --- | --- | --- |
| Service name | DIO\_init | | |
| Syntax | u8\_en\_dioErrors DIO\_init (  en\_dioPortsType port,  en\_dioPinsType pin,  u8\_en\_dioDirType direction  ); | | |
| Parameters (in) | Port, pin | Channel ID | |
| direction | Value to be set | STD\_INPUT |
| STD\_OUTPUT |
| Return | |  |  | | --- | --- | | DIO\_Errors | DIO\_E\_OK | | DIO\_InvalidPin | | DIO\_InvalidPort | | | |
| Description | This Function sets the Direction of the pin | | |

* This function shall return DIO\_InvalidPin if pin number is invalid
* This function shall return DIO\_InvalidPort if port number is invalid.

### 3.2.2 : Timer API :

#### 3.2.2.1 :Flowcharts:

#### 3.2.2.2 : Type definitions:

typedef enum{

E\_NOT\_OK = 0,

E\_OK

}Std\_ReturnType;

typedef enum

{

Timer0,Timer1,Timer2,INVALID\_TIMER\_TYPE

}TimerType\_t;

typedef enum

{

NO\_CLOCK,F\_CPU\_CLOCK,F\_CPU\_8,F\_CPU\_32,F\_CPU\_64,F\_CPU\_128,F\_CPU\_256,F\_CPU\_1024,

TIMER\_EXTERNAL\_CLK\_FALLING\_EDGE,TIMER\_EXTERNAL\_CLK\_RISING\_EDGE,

INVALID\_TIMER\_CLK

}TimerClock\_t;

typedef enum

{

TIMER\_NORMAL\_MODE=0,

TIMER\_PHASE\_CORRECT\_PWM\_MODE,

TIMER\_CTC\_MODE,

TIMER\_FAST\_PWM\_MODE,

INVALID\_TIMER\_MODE

}TimerMode\_t;

typedef enum

{

CTC\_Output\_Compare\_Mode\_DISCONNECTED=0, /\*Normal port operation, OCx disconnected.\*/

CTC\_Output\_Compare\_Mode\_TOGGLE, /\*Toggle OCx on compare match\*/

CTC\_Output\_Compare\_Mode\_CLEAR, /\*Clear OCx on compare match\*/

CTC\_Output\_Compare\_SET, /\*Set OCx on compare match\*/

CTC\_INVALID\_TIMER\_OUTPUT\_COMPARE\_MODE

}Output\_Compare\_Mode\_t;

typedef enum

{

FAST\_Output\_Compare\_Mode\_DISCONNECTED=0, /\*Normal port operation, OCx disconnected.\*/

FAST\_Output\_Compare\_Mode\_TOGGLE, /\*Toggle OCx on compare match\*/

FAST\_Output\_Compare\_Mode\_NON\_INVERTED, /\*Clear OCx on compare match, set OCx at BOTTOM\*/

FAST\_Output\_Compare\_Mode\_INVERTED, /\* Set OCx on compare match, clear OCx at BOTTOM\*/

FAST\_INVALID\_TIMER\_FAST\_PWM\_MODE

}FAST\_PWM\_MODE\_t;

typedef enum

{

PC\_Output\_Compare\_Mode\_DISCONNECTED=0, /\*Normal port operation, OCx disconnected.\*/

PC\_Output\_Compare\_Mode\_TOGGLE, /\*Toggle OCx on compare match\*/

PC\_Output\_Compare\_Mode\_NON\_INVERTED, /\*Clear OCx on compare match when up-counting Set OCx on compare match when down counting\*/

PC\_Output\_Compare\_Mode\_INVERTED, /\*Set OCx on compare match when up-counting Clear OCx on compare match when down counting\*/

PC\_INVALID\_TIMER\_Phase\_Correct\_PWM\_MODE

}Phase\_Correct\_PWM\_Mode;

typedef struct

{

TimerType\_t timer\_type; /\* @ref TimerType\_t\*/

TimerClock\_t timer\_clock; /\* @ref TimerClock\_t\*/

TimerMode\_t timer\_mode; /\* @ref TimerMode\_t\*/

Output\_Compare\_Mode\_t output\_compare\_mode; /\* @ref Output\_Compare\_Mode\_t\*/

FAST\_PWM\_MODE\_t fast\_pwm\_mode; /\* @ref FAST\_PWM\_MODE\_t 8-bit resolution only\*/

Phase\_Correct\_PWM\_Mode phase\_correct\_pwm\_mode; /\* @ref Phase\_Correct\_PWM\_Mode 8-bit resolution only\*/

*uint16\_t* timer\_InitialValue; /\* the pre-loaded value on Timer/Counter Register\*/

*uint16\_t* timer\_compare\_MatchValue; /\* the top value on Output Compare Register\*/

}Timer\_Config\_t;

#### 3.2.2.3 : Services affecting the hardware unit

/\*

\* Description: Function to Initialize Timer Driver

\* - Working in Interrupt Mode

\* - Choose Timer initial value

\* - Choose Timer\_ID (Timer0, Timer1, Timer2)

\* - Choose Timer\_Mode (OverFlow, Compare,PWM)

\* - if using CTC mode choose Timer compare match value And choose q output\_compare\_mode

\*

\*@param A Reference of the Timer configuration

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType TIMERx\_init(const Timer\_Config\_t \*stPtr\_a\_Config);

/\*

\* Description : TIMER START COUNTING BY CONFIGURE THE TIMER CLOCK

\* @param en\_a\_timer\_clk :timer clock configuration with pres-scaler

en\_a\_timer\_type :timer channel : timer0,timer1,timer2

\* @return Std\_ReturnType: status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType TIMERx\_start(const TimerClock\_t en\_a\_timer\_clk,const TimerType\_t en\_a\_timer\_type);

/\*

\* Description : Call the Call Back function in the application after timer did its job

\* @param A pointer to function & the timer type

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType TIMERx\_setCallBack(void(\*a\_fptr)(void),const TimerType\_t en\_a\_timer\_type );

/\*

\* Description :set a certain value on the timer counting register

\* @param the timer type and the initial value to be set

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType TIMERx\_setValue(const TimerType\_t en\_a\_timer\_type ,const *uint16\_t* u16\_a\_timer\_value);

/\*

\* Description :this function sets the offset of the compare unit

\* @param timer type and the top value to be compared with the TCNCx

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType TIMERx\_CTC\_SetCompare(const TimerType\_t en\_a\_timer\_type ,const *uint16\_t* u16\_a\_compareValue);

/\*

\* Description :Function to make the timer to start again from beginning(reset)

\* @param the timer type and the initial value to be set

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType TIMERx\_reset(const TimerType\_t en\_a\_timer\_type);

/\*

\* Description :Function to Halt the timer (stop)

\* @param the timer type and the initial value to be set

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType TIMERx\_stop(const TimerType\_t en\_a\_timer\_type);

### 3.2.3 : External Interrupt API :

#### 3.2.3.1 : Flowcharts:

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

#### 3.2.3.2 : Type definitions:

/\*\*datatype to hold the state of function and it has two options INT\_E\_OK || INT\_E\_NOK\*\*/

typedef uint8\_t u8\_en\_interruptErrorType;

#define INT\_E\_OK ((u8\_en\_interruptErrorType)0x00)// function done

#define INT\_E\_NOK ((u8\_en\_interruptErrorType)0x01)// function didn't do its behavior correctly

/\* datatype of enum has three choices of interrupts

external interrupt 0, external interrupt 1, external interrupt 2

\*/

typedef enum{

EXT\_0=0,

EXT\_1,

EXT\_2

}ext\_interrupt\_no\_t;

/\* datatype of enum has four options of interrupt detection

low level, locical change, falling edge, rising edge

\*/

typedef enum{

LOW\_LEVEL,

LOGICAL\_CHANGE,

FALLING\_EDGE,

RISING\_EDGE

}EDGE\_detection\_t;

#### 3.2.3.3 : Services affecting the hardware unit

/\*

\* -Description-

\*-this function INIT the external interrupt configuration

\*

\* -Input Output parameters -

\*-1-it is interrupt number (ext\_interrupt\_no\_t ext\_interrupt\_no)

\*-2-it is the interrupt condition (EDGE\_detection\_t EDGE\_detection)

\*

\* -Return-

\* u8\_en\_interruptErrorType

\*

\* -Return cases-

\*-1- (INT\_E\_OK) if there is something wrong

\*-2- (INT\_E\_NOK) otherwise

\*/

u8\_en\_interruptErrorType ext\_interrupt\_init(ext\_interrupt\_no\_t ext\_interrupt\_no, EDGE\_detection\_t EDGE\_detection);

/\*

\* -Description-

\*-this function enable external interrupt depend on external interrupt number

\*

\* -Input Output parameters -

\*-1-it is interrupt number (ext\_interrupt\_no\_t ext\_interrupt\_no)

\*

\*

\* -Return-

\* u8\_en\_interruptErrorType

\*

\* -Return cases-

\*-1- (INT\_E\_OK) if there is something wrong

\*-2- (INT\_E\_NOK) otherwise

\*/

u8\_en\_interruptErrorType ext\_interrupt\_enable(ext\_interrupt\_no\_t ext\_interrupt\_no);

/\*

\* -Description-

\*-this function disable external interrupt depend on external interrupt number

\*

\* -Input Output parameters -

\*-1-it is interrupt number (ext\_interrupt\_no\_t ext\_interrupt\_no)

\*

\*

\* -Return-

\* u8\_en\_interruptErrorType

\*

\* -Return cases-

\*-1- (INT\_E\_OK) if there is something wrong

\*-2- (INT\_E\_NOK) otherwise

\*/

u8\_en\_interruptErrorType ext\_interrupt\_disable(ext\_interrupt\_no\_t ext\_interrupt\_no);

/\*

\* -Description-

\*-this function set callback function to external interrupt

\*

\* -Input Output parameters -

\*-1-it is pointer to call back function (void (\*func)(void))

\*

\* -Return-

\* u8\_en\_interruptErrorType

\*

\* -Return cases-

\*-1- (INT\_E\_OK) if there is something wrong

\*-2- (INT\_E\_NOK) otherwise

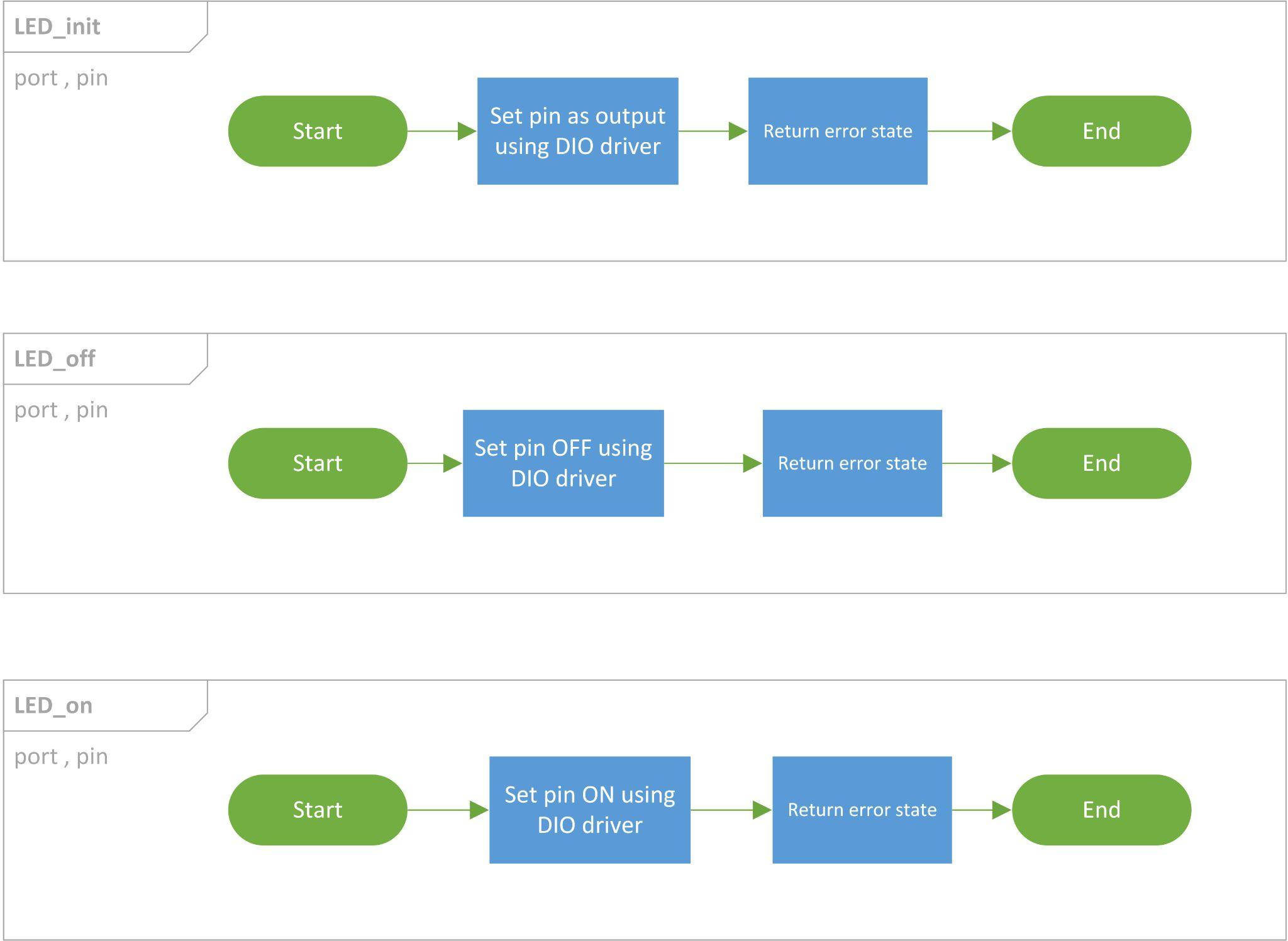
\*/

u8\_en\_interruptErrorType ext\_interrupt\_set\_callback\_init(ext\_interrupt\_no\_t ext\_interrupt\_no ,void(\*callback)(void));

## **3.3 : HAL APIs**

### 3.3.1 : LED API:

#### 3.3.1.1 : Flowcharts:



#### 3.3.1.2 : Type definitions:

* u8\_en\_ledStateType

|  |  |  |
| --- | --- | --- |
| Name | u8\_en\_ledStateType | |
| Type | Enumeration | |
| Range | LED\_OFF | 0x00 |
| LED\_ON | 0x01 |
| Description | u8\_en\_ledStateType | |
| Available via | led\_types.h | |

* u8\_en\_ledErrorType

|  |  |  |
| --- | --- | --- |
| Name | u8\_en\_ledErrorType | |
| Type | Enumeration | |
| Range | LED\_OK | 0x00 |
| LED\_InvalidPin | 0x01 |
| LED\_InvalidPort | 0x02 |
| Description | u8\_en\_ledErrorType | |
| Available via | led\_types.h | |

* st\_ledConfigType

|  |  |
| --- | --- |
| Name | st\_ledConfigType |
| Type | structure |
| Description | st\_ledConfigType |
| Available via | led\_types.h |

#### 3.3.1.3 : Services affecting the hardware unit:

* LED\_off

|  |  |  |
| --- | --- | --- |
| Service name | LED\_off | |
| Syntax | u8\_en\_ledErrorType LED\_off(  uint8\_t u8\_a\_port,  uint8\_t u8\_a\_pin  ); | |
| Parameters (in) | Port, pin | Channel ID |
| Return | |  |  | | --- | --- | | u8\_en\_ledErrorType | LED\_OK | | LED\_InvalidPin | | LED\_InvalidPort | | |
| Description | This Function sets the level of the pin to low | |

* This function shall return DIO\_InvalidPin if pin number is invalid.
* This function shall return DIO\_InvalidPort if port number is invalid.
* LED\_on

|  |  |  |
| --- | --- | --- |
| Service name | LED\_on | |
| Syntax | u8\_en\_ledErrorType LED\_on(  uint8\_t u8\_a\_port,  uint8\_t u8\_a\_pin  ); | |
| Parameters (in) | Port, pin | Channel ID |
| Return | |  |  | | --- | --- | | u8\_en\_ledErrorType | LED\_OK | | LED\_InvalidPin | | LED\_InvalidPort | | |
| Description | This Function sets the level of the pin | |

* This function shall return DIO\_InvalidPin if pin number is invalid.
* This function shall return DIO\_InvalidPort if port number is invalid.
* LED\_init

|  |  |  |
| --- | --- | --- |
| Service name | LED\_init | |
| Syntax | u8\_en\_ledErrorType LED\_init(  uint8\_t u8\_a\_port,  uint8\_t u8\_a\_pin  ); | |
| Parameters (in) | Port, pin | Channel ID |
| Return | |  |  | | --- | --- | | u8\_en\_ledErrorType | LED\_OK | | LED\_InvalidPin | | LED\_InvalidPort | | |
| Description | This Function sets the Direction of the led pin as output | |

* This function shall return LED\_InvalidPin if pin number is invalid
* This function shall return LED\_InvalidPort if port number is invalid.

### 3.3.2 : Motor API:

#### 3.3.2.1 : Flowcharts:

#### 3.3.2.2 : Type definitions:

typedef struct ST\_motor\_t

{

en\_dioPortsType port;

en\_dioPinsType pin\_num1;

en\_dioPinsType pin\_num2;

}ST\_motor\_t;

typedef enum EN\_motor\_error\_t

{

MOTOR\_OK,

MOTOR\_NOK

}EN\_motor\_error\_t;

#### 3.3.2.3 : Services affecting the hardware unit:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* description: this function used to init the motor as output

\* input : pointer to structure which have port and two pin number

\* return :MOTOR\_OK or MOTR\_NOK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

EN\_motor\_error\_t MOTOR\_INIT(const ST\_motor\_t\* motor);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* description : this function used to move the motor forward

\* input : pointer to structure which have port and two pin number and speed of motor

\* return :MOTOR\_OK or MOTR\_NOK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

EN\_motor\_error\_t MOTOR\_FORWARD(const ST\_motor\_t\* motor);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* description : this function used to move the motor backward

\* input : pointer to structure which have port and two pin number and speed of motor

\* return :MOTOR\_OK or MOTR\_NOK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

EN\_motor\_error\_t MOTOR\_BACKWARD(const ST\_motor\_t\* motor);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* description : this function used to stop the motor

\* input : pointer to structure which have port and two pin number

\* return :MOTOR\_OK or MOTR\_NOK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

EN\_motor\_error\_t MOTOR\_STOP(const ST\_motor\_t\* motor);

### 3.3.3 : Car Control API :

#### 3.3.3.1 : Flowcharts:

#### 3.3.3.2 : Type definitions:

typedef enum EN\_car\_error\_t

{

CAR\_OK,

CAR\_NOK,

}EN\_car\_error\_t;

#### 3.3.3.3 : Services affecting the hardware unit

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*description : used to initlize the two motor as output

\*input : this function take two pointers to motor structure

\*return : MOTOR\_OK, MOTOR\_NOK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

EN\_car\_error\_t CAR\_INIT(const ST\_motor\_t\* motor\_1,const ST\_motor\_t\* motor\_2);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*description : used to move the car forward by specific speed

\*input : this function take two pointers to motor structure and speed of the car

\*return : MOTOR\_OK, MOTOR\_NOK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

EN\_car\_error\_t CAR\_FORWARD(const ST\_motor\_t\* motor\_1,const ST\_motor\_t\* motor\_2);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*description: used to reverse the car to the right

\*input : this function take two pointers to motor structure and speed of the car

\*return : MOTOR\_OK, MOTOR\_NOK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

EN\_car\_error\_t CAR\_REVERSE\_RIGHT(const ST\_motor\_t\* motor\_1,const ST\_motor\_t\* motor\_2);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*description : used to stop the car

\*input : this function take two pointers to motor structure

\*return : MOTOR\_OK, MOTOR\_NOK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

EN\_car\_error\_t CAR\_STOP(const ST\_motor\_t\* motor\_1,const ST\_motor\_t\* motor\_2);

### 3.3.4 : Timer Manager API :

#### 3.3.4.1 : Flowcharts:

#### 3.3.4.2 : Type definitions:

typedef void (\*Fptr) (void);

typedef struct

{

TimerType\_t timer\_num; /\* @ref TimerType\_t\*/

TimerMode\_t timer\_mode; /\* @ref TimerMode\_t\*/

*uint16\_t* timer\_InitialValue; /\* the pres-loaded value on Timer/Counter Register\*/

*uint16\_t* timer\_compare\_MatchValue; /\* the top value on Output Compare Register\*/

Fptr call\_back\_function; /\*pointer to function that take void and return nothing(void) ,should loaded with call-back function's address\*/

}TimerManger\_config\_t;

#### 3.3.4.3 : Services affecting the hardware unit

/\*

\* Description: Function to Initialize Timer Driver

\* - Working in Interrupt Mode

\* - set Timer initial value

\* - set Timer\_ID (Timer0, Timer1, Timer2)

\* - set Timer\_Mode(OverFlow, Compare,PWM)

\* - if using CTC mode choose Timer compare match value And choose output\_compare\_mode

\*

\*@param stPtr\_a\_TimerConfig :A Reference of the Timer configuration

\* @return Std\_ReturnType : status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType TIMER\_MANGER\_init(const TimerManger\_config\_t \*stPtr\_a\_TimerConfig);

/\*

\* Description : TIMER START COUNTING BY CONFIGURE THE TIMER CLOCK

\* @param en\_a\_timer\_clk :timer clock configuration with pres-scaler

en\_a\_timer\_num :timer channel : timer0,timer1,timer2

\* @return Std\_ReturnType: status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType TIMER\_MANGER\_start(const TimerClock\_t en\_a\_timer\_clock,const TimerType\_t en\_a\_timer\_num);

/\*

\* Description :Function to Halt the timer (stop)

\* @param the en\_a\_timer\_num timer type

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType TIMER\_MANGER\_stop(const TimerType\_t en\_a\_timer\_num);

### 3.3.5 : Interrupt Manager API :

#### 3.3.5.1 : Flowcharts:

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

#### 3.3.5.1 : Type definitions:

/\*

\*

\*

\* -ext\_interrupt\_config\_t datatype hold the external interrupt config

\* -Members-

\* -1- (ext\_interrupt\_no\_t ext\_interrupt\_no) number of external interrupt

\* -2- (EDGE\_detection\_t edge\_select) the state of line will fire the interrupt

\*

\*/

typedef struct

{

ext\_interrupt\_no\_t ext\_interrupt\_no;

EDGE\_detection\_t edge\_select;

}ext\_interrupt\_config\_t;

#### 3.3.5.1 : Services affecting the hardware unit

/\*

\* -Description-

\*-this function init the external interrupt configuration

\*

\* -Input Output parameters -

\*-1-it is configuration of external interrupt (ext\_interrupt\_config\_t \*ext\_interrupt\_config)

\*-2-it is pointer to callback function (void(\*callback)(void))

\*

\* -Return-

\* u8\_en\_interruptErrorType

\*

\* -Return cases-

\*-1- (INT\_E\_OK) if there is something wrong

\*-2- (INT\_E\_NOK) otherwise

\*/

u8\_en\_interruptErrorType ext\_init(ext\_interrupt\_config\_t \*ext\_interrupt\_config, void(\*callback)(void));

/\*

\* -Description-

\*-this function enable external interrupt depend on external interrupt configuration

\*

\* -Input Output parameters -

\*-1-it is configuration of external interrupt (ext\_interrupt\_config\_t \*ext\_interrupt\_config)

\*

\*

\* -Return-

\* u8\_en\_interruptErrorType

\*

\* -Return cases-

\*-1- (INT\_E\_OK) if there is something wrong

\*-2- (INT\_E\_NOK) otherwise

\*/

u8\_en\_interruptErrorType ext\_enable(uint8\_t u8\_intNum);

/\*

\* -Description-

\*-this function disable external interrupt depend on external interrupt configuration

\*

\* -Input Output parameters -

\*-1-it is configuration of external interrupt (ext\_interrupt\_config\_t \*ext\_interrupt\_config)

\*

\*

\* -Return-

\* u8\_en\_interruptErrorType

\*

\* -Return cases-

\*-1- (INT\_E\_OK) if there is something wrong

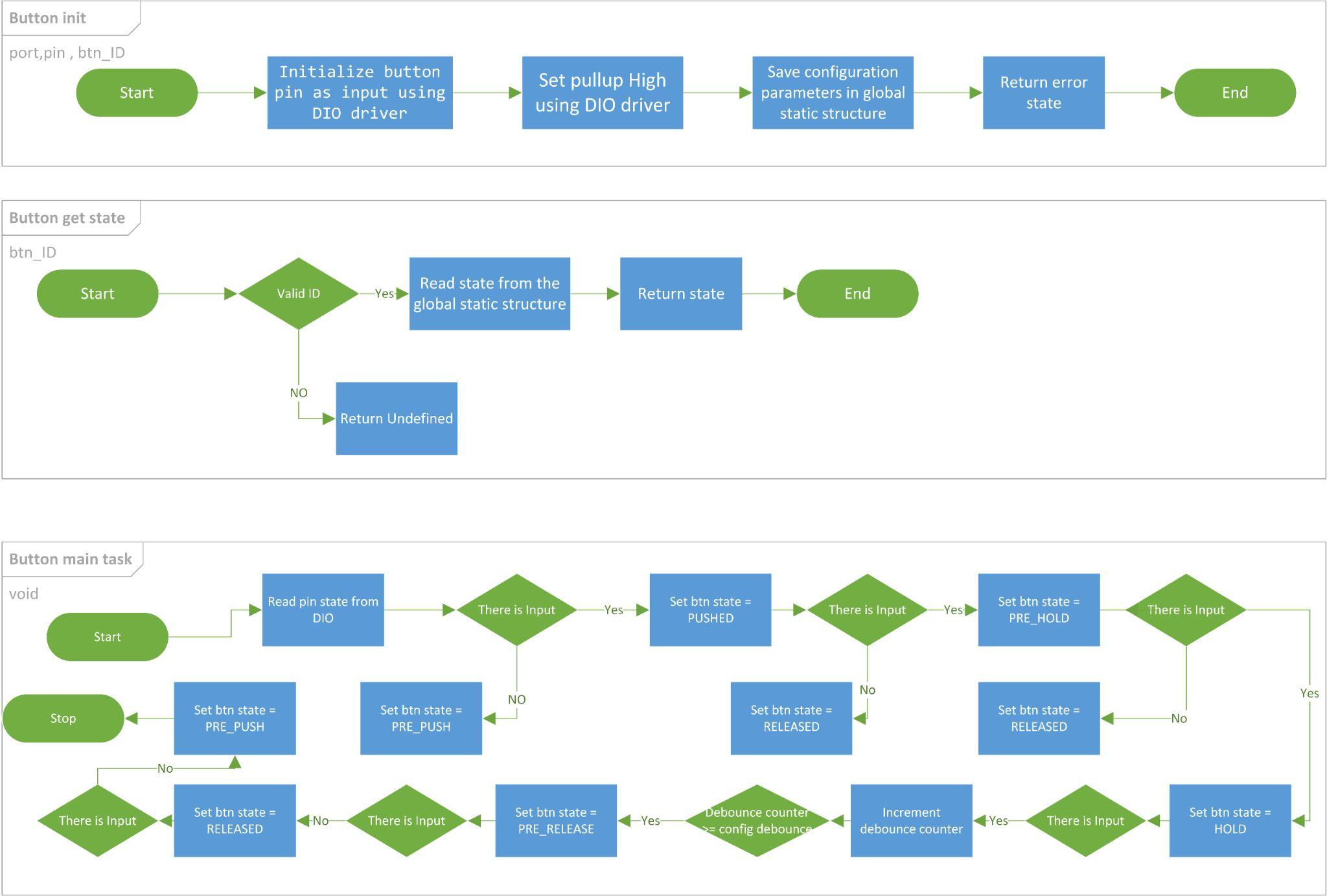
\*-2- (INT\_E\_NOK) otherwise

\*/

u8\_en\_interruptErrorType ext\_disable(uint8\_t u8\_intNum);

### 3.3.6: Button API:

#### 3.3.6.1: Flowcharts:



#### 3.3.6.2 : Type definitions:

* st\_btnConfigType

|  |  |
| --- | --- |
| Name | st\_btnConfigType |
| Type | Structure |
| Description | This is the type of the external data structure containing the overall configuration data for the Button API |
| Available via | button\_types.h |

* u8\_en\_btnLevelType

|  |  |
| --- | --- |
| Name | u8\_en\_btnLevelType |
| Type | Enumeration |
| Range | |  |  |  | | --- | --- | --- | | BT\_PUSH\_LEVEL | 0x00 | Push Level | | BT\_RELEASE\_LEVEL | 0x01 | Release Level | |
| Description | Button Level Enum |
| Available via | button\_types.h |

* u8\_en\_btnStateType

|  |  |
| --- | --- |
| Name | u8\_en\_btnStateType |
| Type | Enumeration |
| Range | |  |  |  | | --- | --- | --- | | BT\_PRE\_PUSH | 0x00 | Pre Push Level | | BT\_PUSHED | 0x01 | Pushed Level | | BT\_PRE\_HOLD | 0x02 | Pre Hold Level | | BT\_HOLD | 0x03 | Hold Level | | BT\_PRE\_RELEASE | 0x04 | Pre Release Level | | BT\_RELEASED | 0x05 | Released Level | | BT\_UNDEFINED | 0x06 | Undefined | |
| Description | Button state Enum |
| Available via | button\_types.h |

* Button\_IdType

|  |  |
| --- | --- |
| Name | u8\_en\_btnIdType |
| Type | Enumeration |
| Range | |  |  |  | | --- | --- | --- | | Button\_Start | 0x00 | Start Button | |
| Description | Button ID Enum |
| Available via | button\_types.h |

#### 3.3.6.2 : Services affecting the hardware unit

* BUTTON\_getState

|  |  |  |
| --- | --- | --- |
| Service name | BUTTON\_getState | |
| Syntax | u8\_en\_btnStateType BUTTON\_getState(  u8\_en\_btnIdType en\_btnId  ); | |
| Parameters (in) | en\_btnId | Start 0x00 |
| Return | |  |  | | --- | --- | | Button\_StateTyp | BT\_PRE\_PUSH | | BT\_PUSHED | | BT\_PRE\_HOLD | | BT\_HOLD | | BT\_PRE\_RELEASE | | BT\_RELEASED | | BT\_UNDEFINED | | |
| Description | This Function gets the Button state. | |

## 

* button\_Main\_Task

|  |  |
| --- | --- |
| Service name | button\_Main\_Task |
| Syntax | void button\_Main\_Taskt(  void  ); |
| Parameters (in) | NONE |
| Return | NONE |
| Description | This Function update all button states  Shall call periodic |

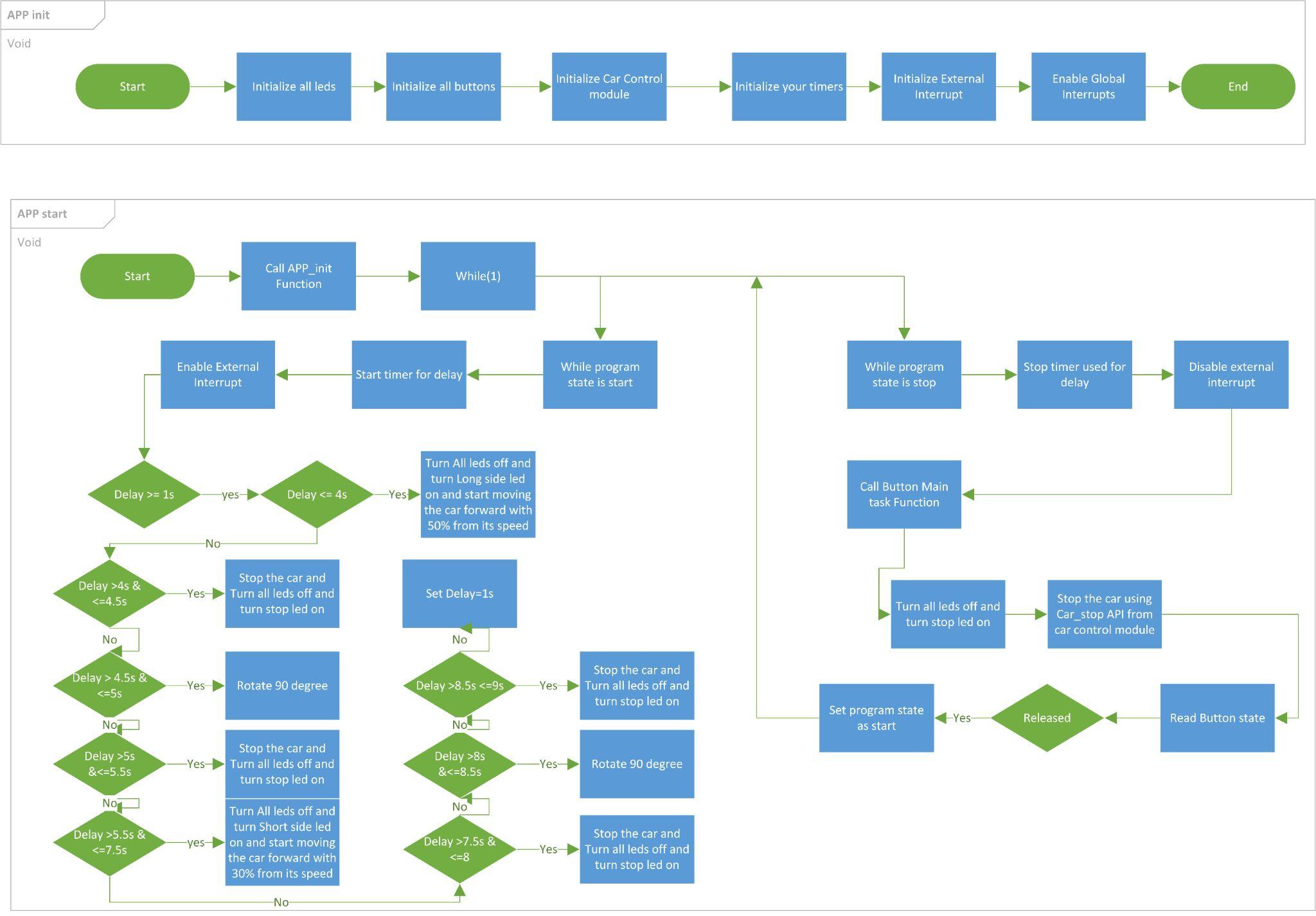
* BUTTON\_init

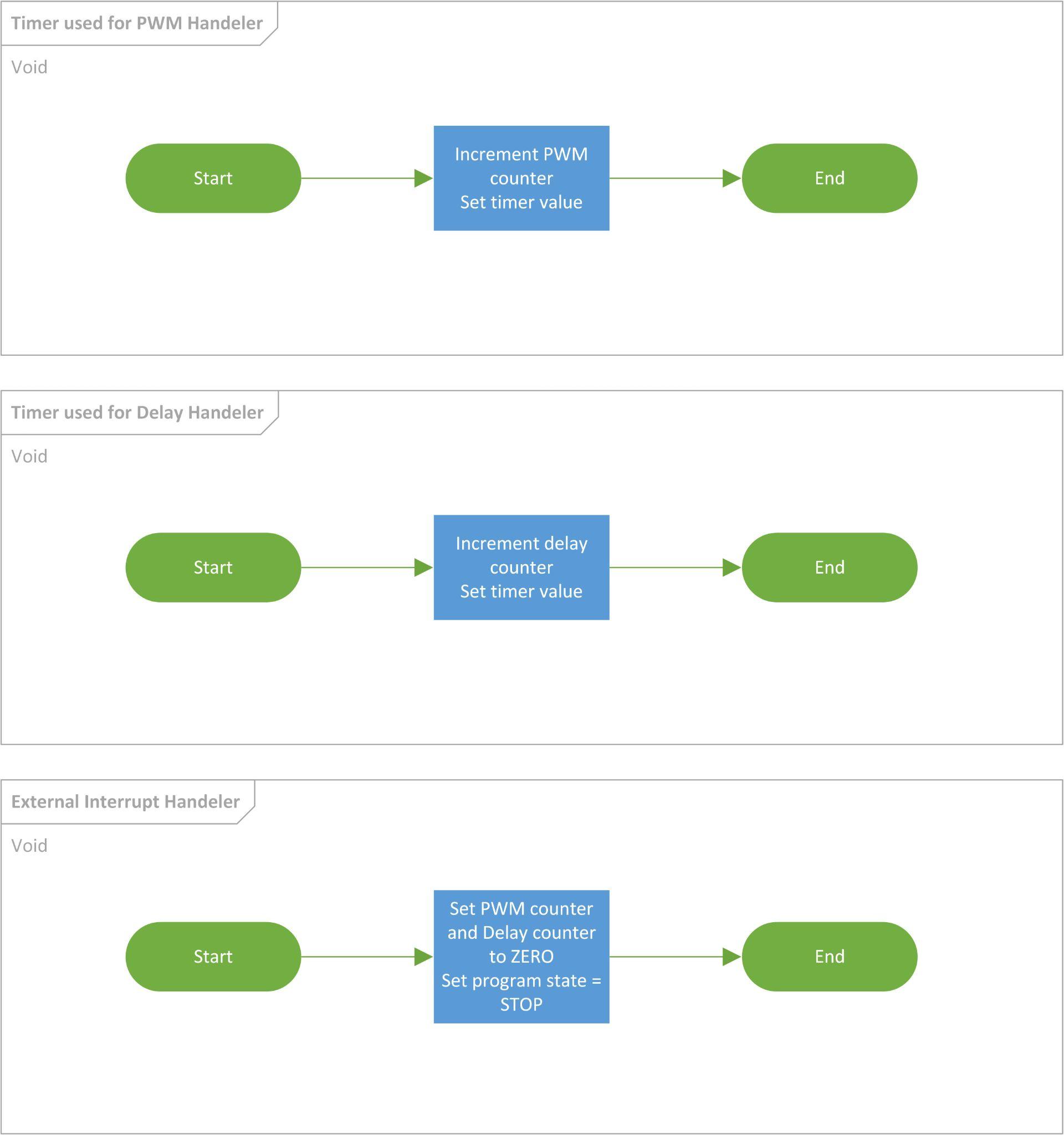
|  |  |  |
| --- | --- | --- |
| Service name | BUTTON\_init | |
| Syntax | u8\_en\_btnStateType BUTTON\_init(  uint8\_t u8\_a\_port,  uint8\_t u8\_a\_pin,  u8\_en\_btnIdType en\_btnId  ); | |
| Parameters (in) | Port, pin | Channel ID |
| en\_btnId | Start 0x00 |
| Return | |  |  | | --- | --- | | Button\_StateTyp | BT\_PRE\_PUSH | | BT\_UNDEFINED | | |
| Description | This Function sets the Direction of the button pin as input | |

## **3.4 : App APIs**

### 3.4.1: App API:

#### 3.4.1.1: Flowcharts:





#### 3.4.1.2 : Type definitions:

* u8\_en\_PROGRAM\_STATE

|  |  |
| --- | --- |
| Name | u8\_en\_PROGRAM\_STATE |
| Type | Enumeration |
| Range | |  |  |  | | --- | --- | --- | | BTN\_START | 0x00 | Program start | | BTN\_STOP | 0x01 | Program stop | |
| Description | u8\_en\_PROGRAM\_STATE |
| Available via | app.h |

* u8\_en\_ledIdType

|  |  |
| --- | --- |
| Name | u8\_en\_ledIdType |
| Type | Enumeration |
| Range | |  |  |  | | --- | --- | --- | | LED\_SHORT\_SIDE | 0x00 | Short Side LED | | LED\_LONG\_SIDE | 0x01 | Long Side LED | | LED\_STOP | 0x02 | Stop LED | | LED\_ROTATE | 0x03 | Rotate LED | |
| Description | u8\_en\_ledIdType |
| Available via | app.h |

#### 3.4.1.3 : Services affecting the hardware unit

* APP\_start

|  |  |
| --- | --- |
| Service name | APP\_start |
| Syntax | void APP\_start(void); |
| Description | This Function Start the Application. |
| Available via | app.h |

* APP\_init

|  |  |
| --- | --- |
| Service name | APP\_init |
| Syntax | void APP\_init(void); |
| Description | This function initialize all drivers used in the application. |
| Available via | app.c |

## 