Moving Car System Design

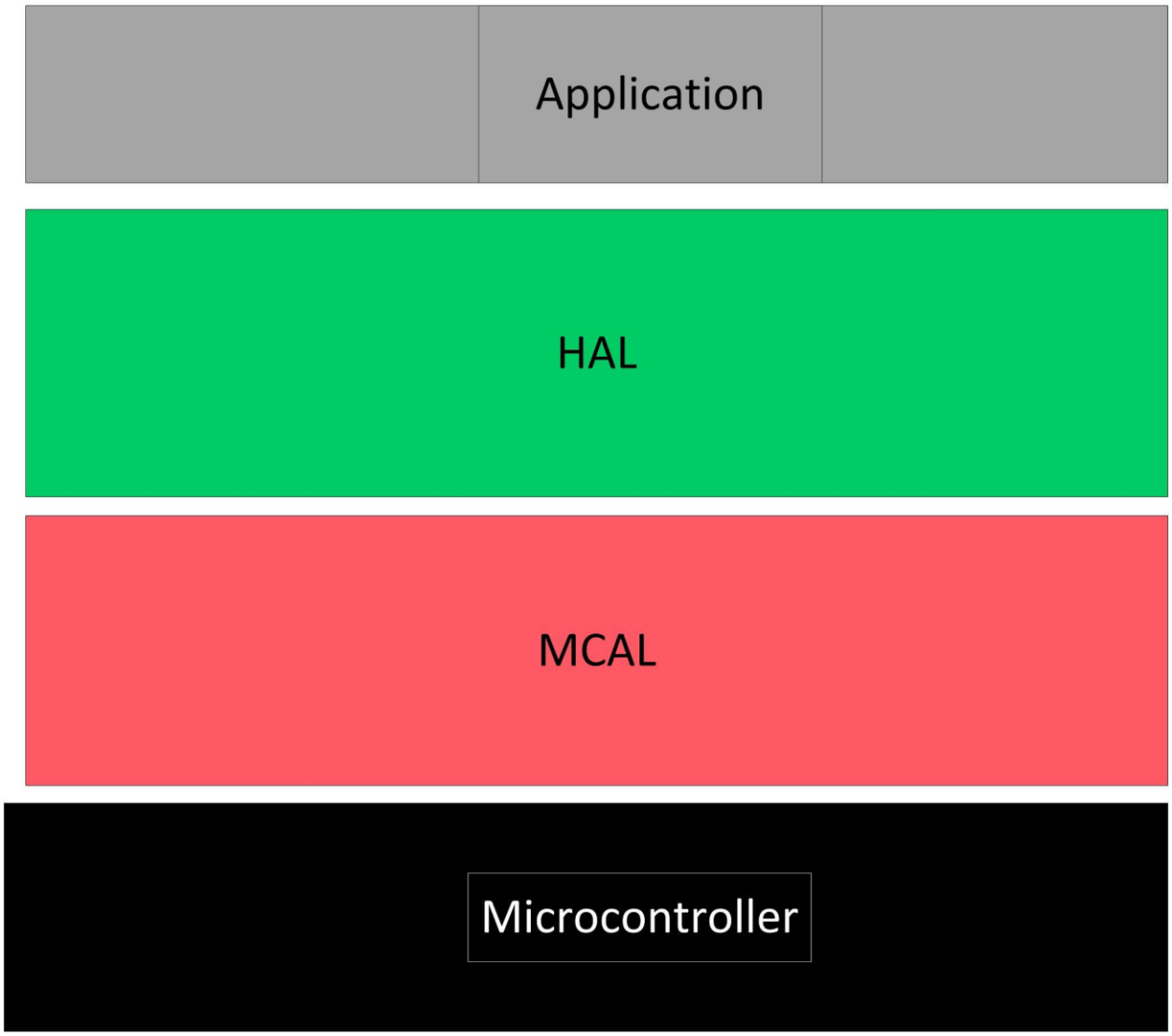
By : Team 4

Table Of Content:

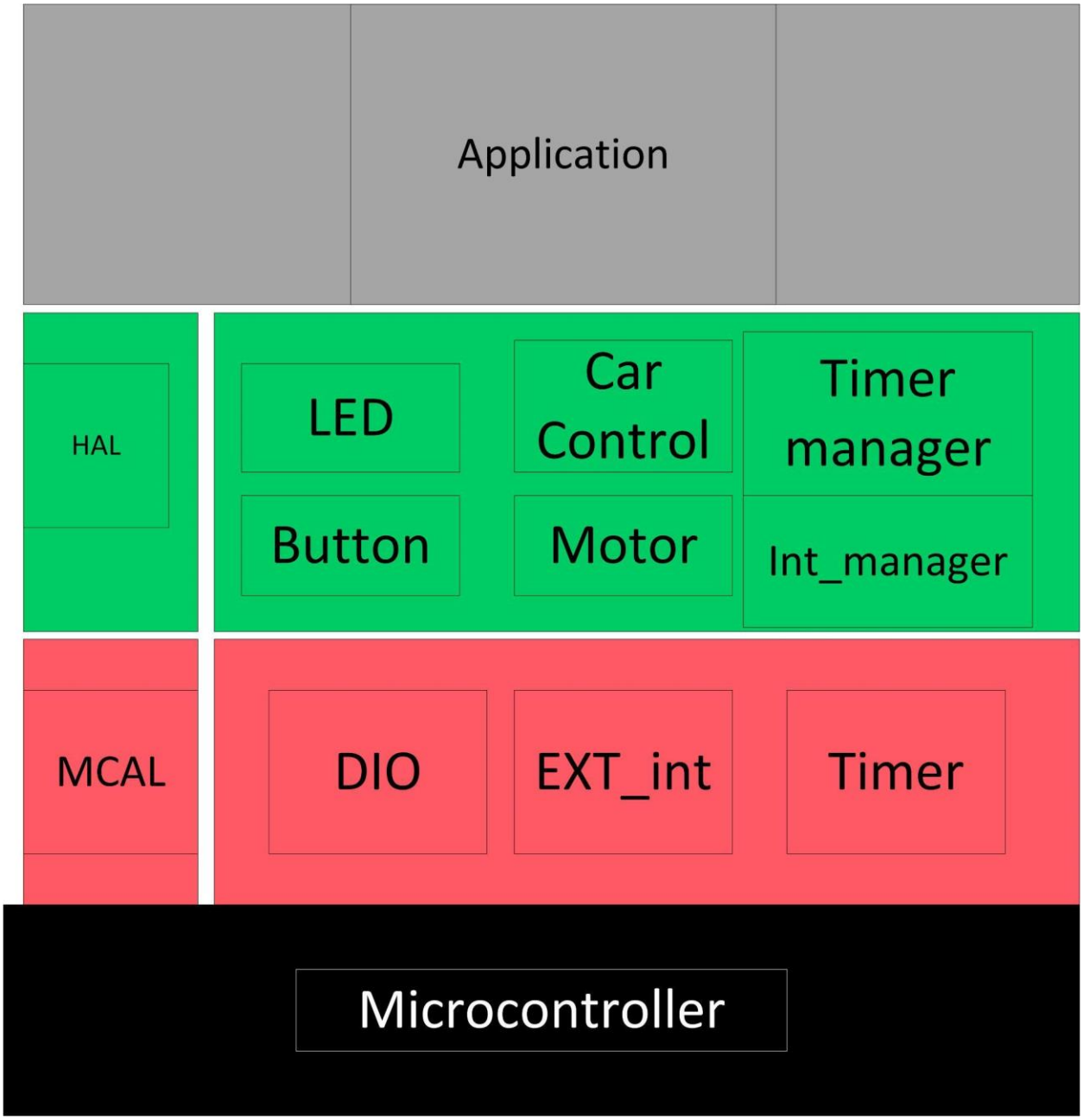
1 : Detailed Requirements

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



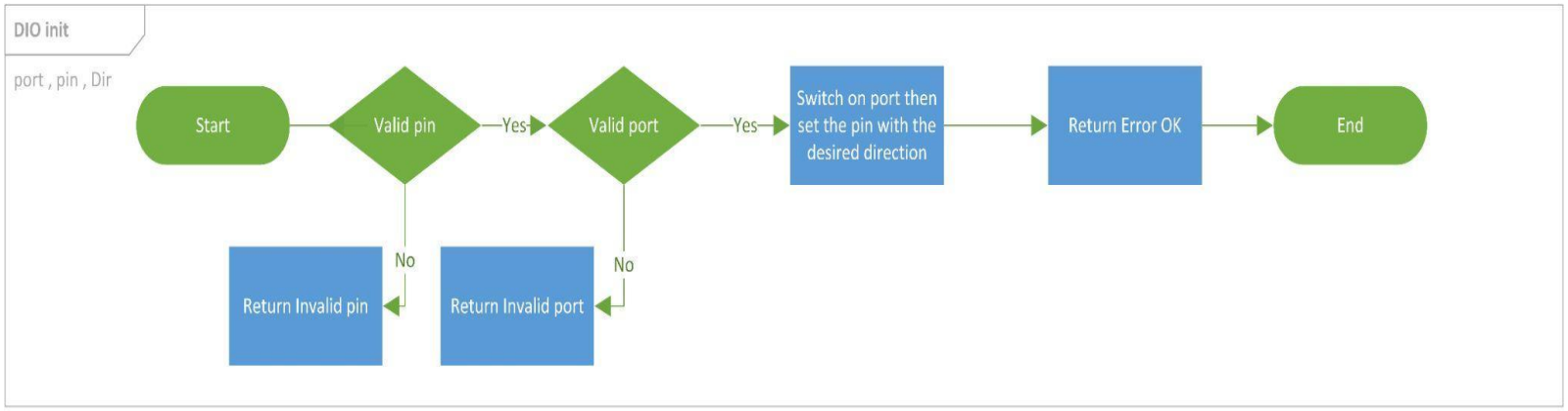
3 : System modules

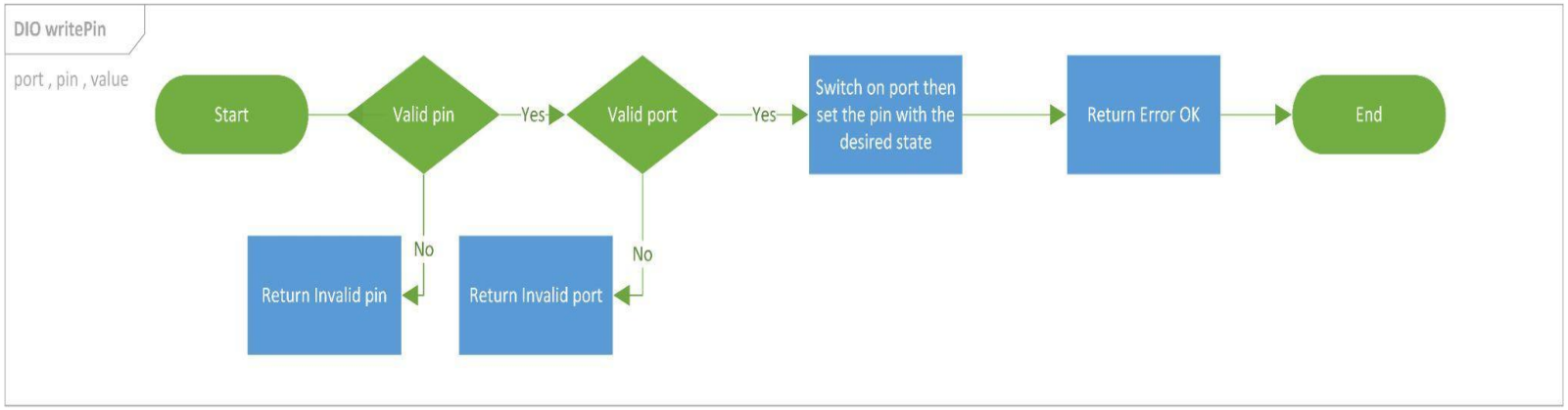


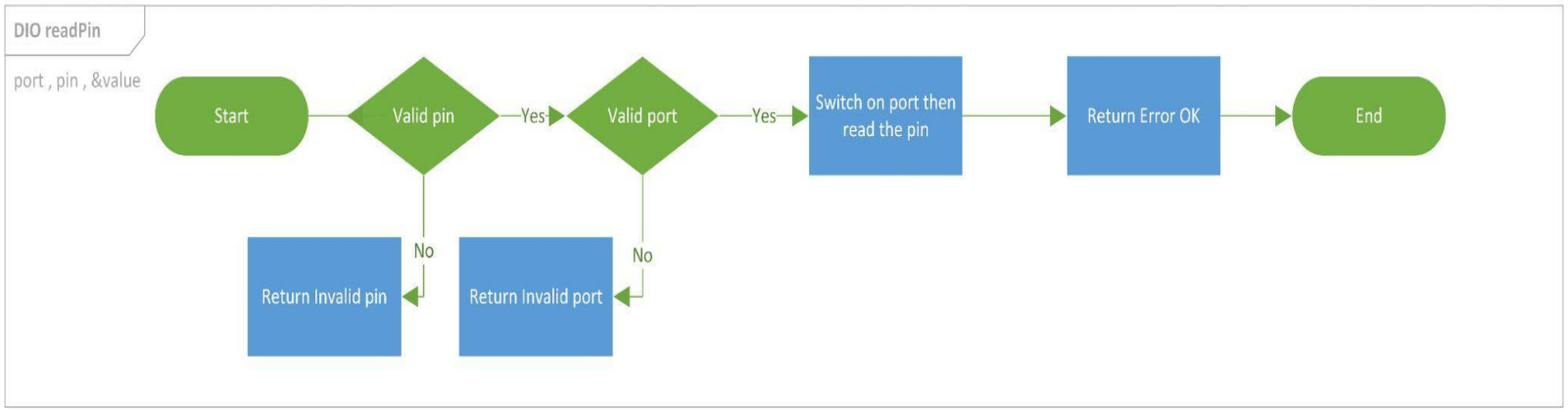
3.2 : MCAL APIs

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

<table>
<tr><td>Service name</td><td colspan="3">DIO_readPIN</td></tr>
<tr><td>Syntax</td><td colspan="3">u8_en_dioErrors DIO_readPIN (
en_dioPortsType port,
en_dioPinsType pin,
uint8_t* value
);</td></tr>
<tr><td rowspan="3">Parameters (in)</td><td>Port, pin</td><td colspan="2">Channel ID</td></tr>
<tr><td rowspan="2">value</td><td rowspan="2">Pointer to store the level</td><td>STD_HIGH</td></tr>
<tr><td>STD_LOW</td></tr>
<tr><td>Return</td><td colspan="3"><table><tr><td rowspan="3">u8_en_dioErrors</td><td>DIO_E_OK</td></tr><tr><td>DIO_InvalidPin</td></tr><tr><td>DIO_InvalidPort</td></tr></table></td></tr>
<tr><td>Description</td><td colspan="3">This Function gets the level of the pin</td></tr>
</table>

  - 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
```

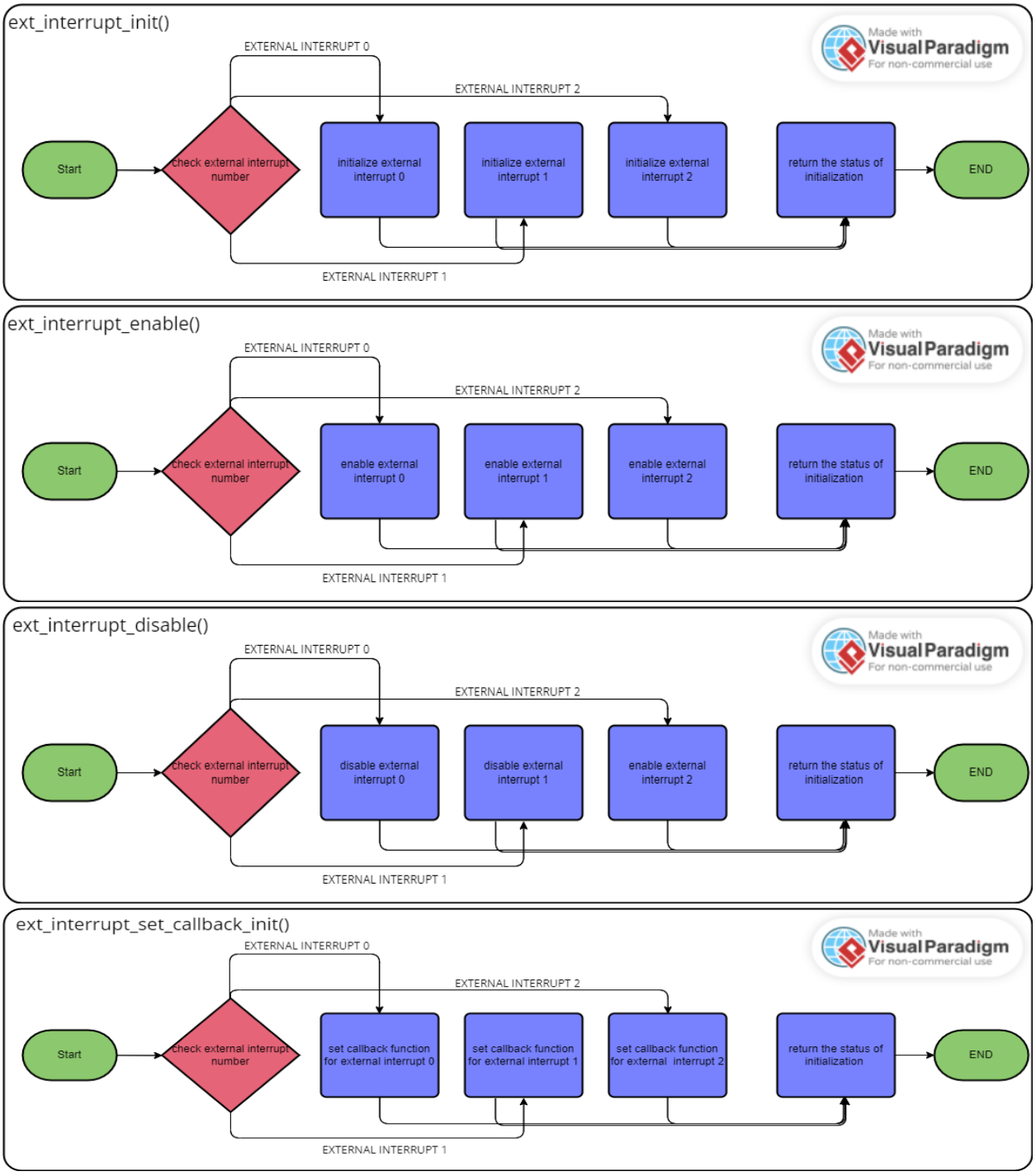
```
 *                - Choose Timer initial value

 *                - 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:



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

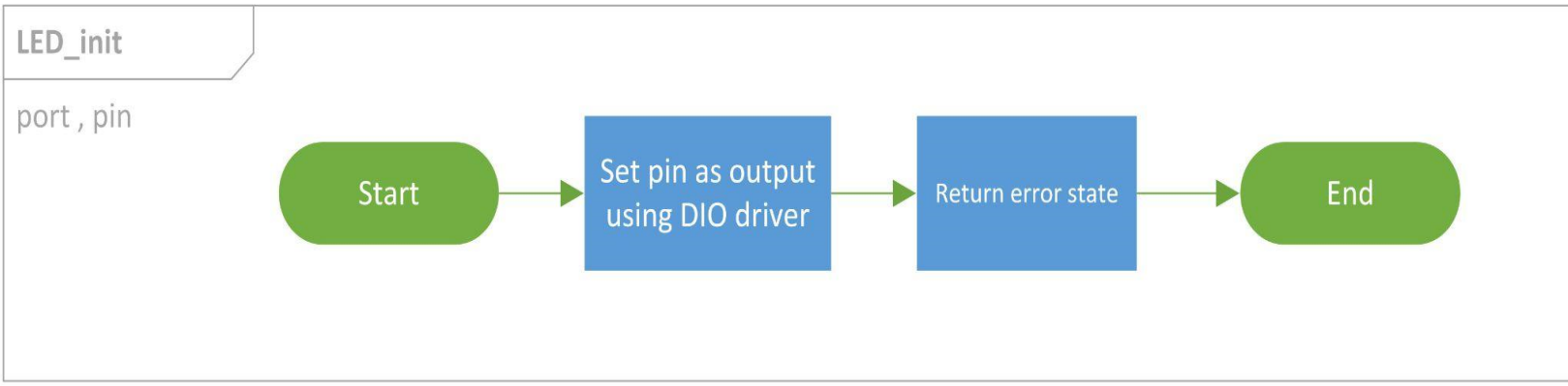
```
/*
*   -Description-
*-this function INIT the external interrupt configuration
*
*   -Input Output parameters -
*-1-it is interrupt number  (ext_interrupt_no_t ext_interrupt_no)
*-2-it is the interrupt condition   (EDGE_detection_t EDGE_detection)
*
*   -Return-
*   u8_en_interruptErrorType
*
*   -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);
```

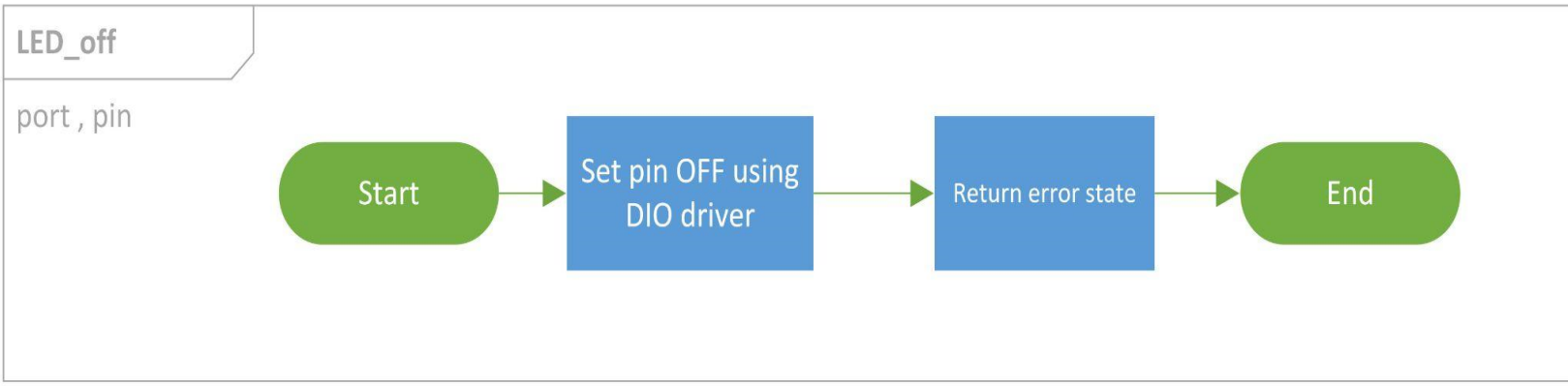
```c
/*
*   -Description-
*-this function disable external interrupt depend on external interrupt number
*
*   -Input Output parameters -
*-1-it is interrupt number  (ext_interrupt_no_t ext_interrupt_no)
*
*
*   -Return-
*   u8_en_interruptErrorType
*
*   -Return cases-
*-1- (INT_E_OK) if there is something 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));
```

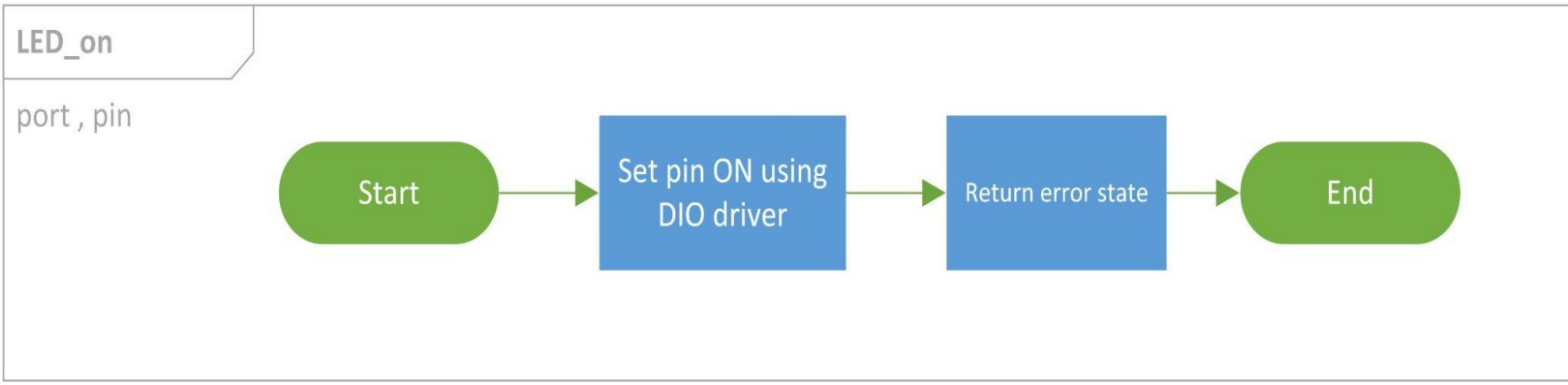
3.3 : HAL APIs

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;
```

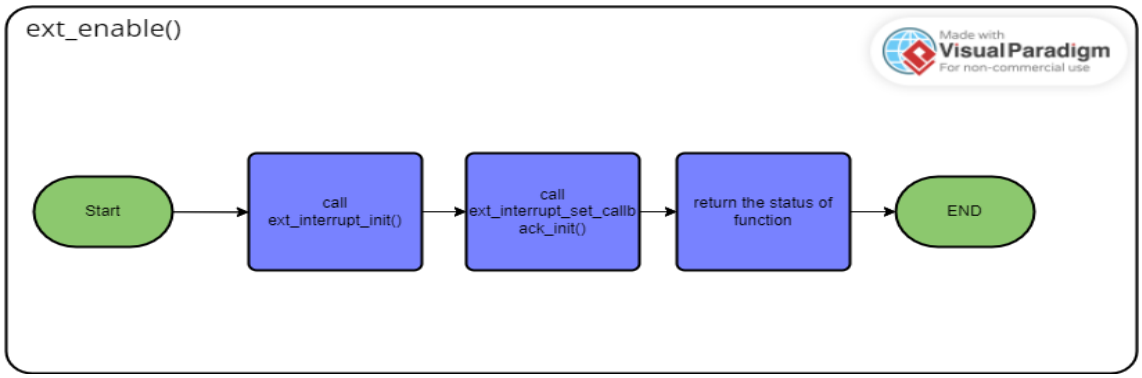
3.3.4.3 : Services affecting the hardware unit

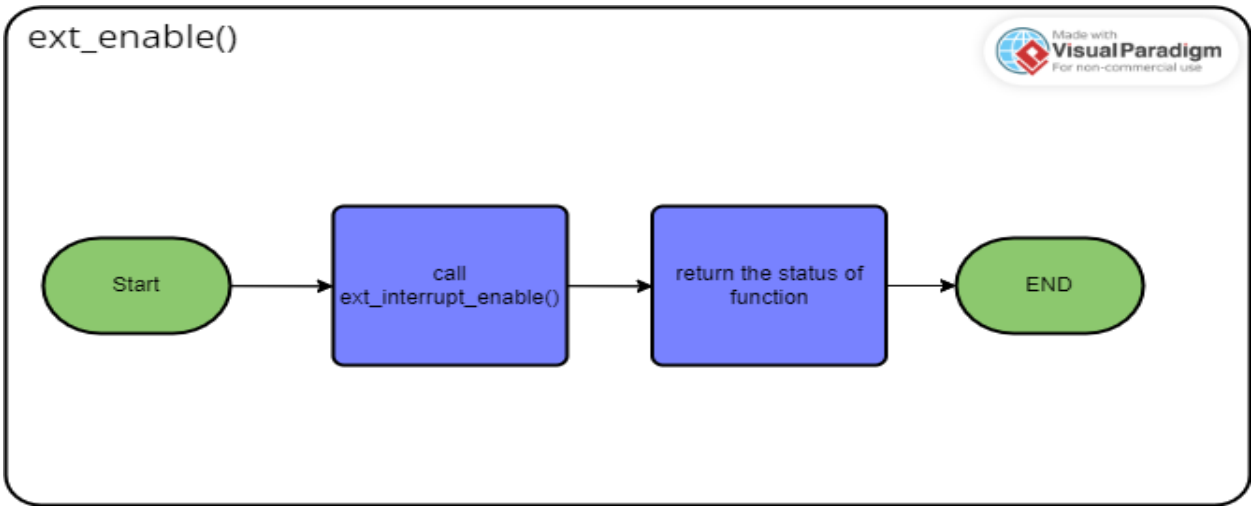
```
/*

 * 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
```

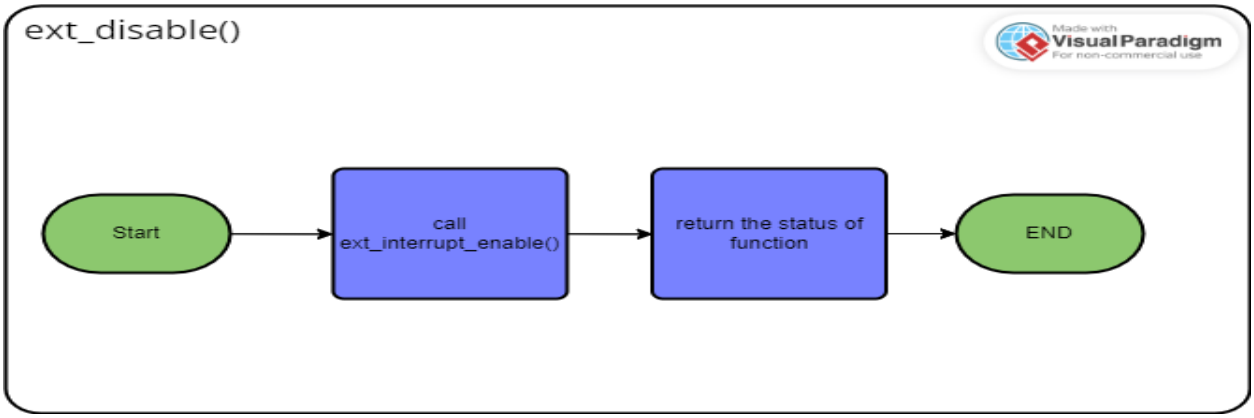
```
 * E_NOT_OK :the function has issues performing the function

 */

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:







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

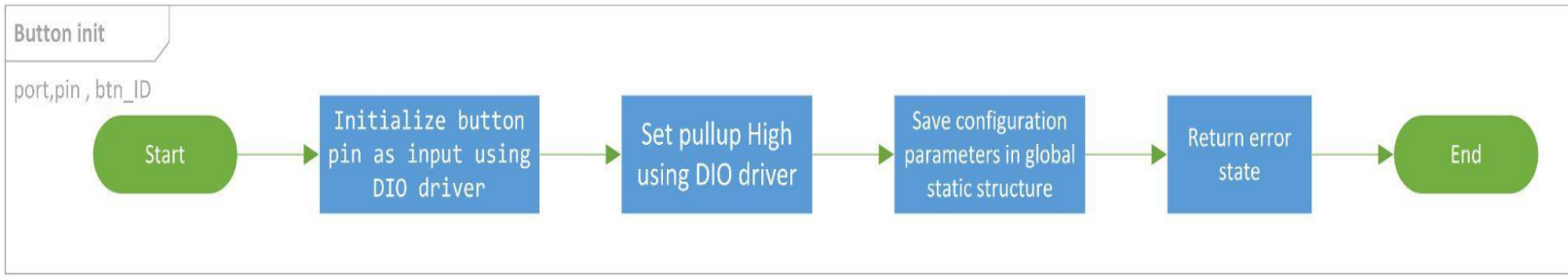
```
/*
*   -Description-
*-this function init the external interrupt configuration
*
*   -Input Output parameters -
*-1-it is configuration of external interrupt (ext_interrupt_config_t
*ext_interrupt_config)
*-2-it is pointer to callback function   (void(*callback)(void))
*
*   -Return-
*   u8_en_interruptErrorType
*
*   -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));
```

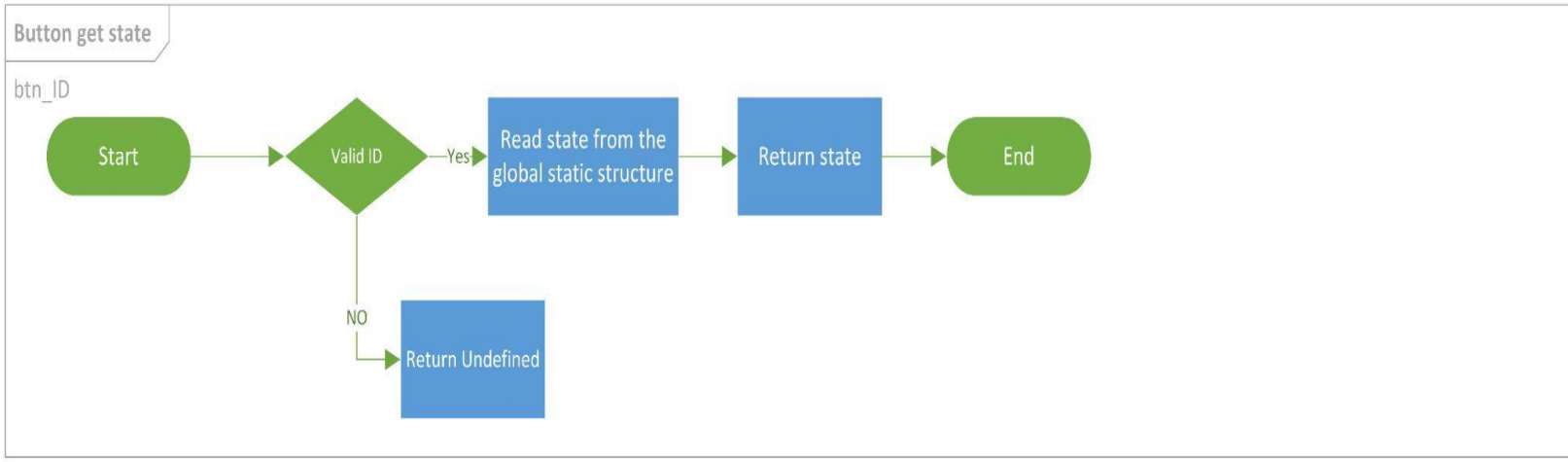
```
/*
*   -Description-
*-this function enable external interrupt depend on external interrupt configuration
*
*   -Input Output parameters -
*-1-it is configuration of external interrupt (ext_interrupt_config_t
*ext_interrupt_config)
*
*
*   -Return-
*   u8_en_interruptErrorType
*
*   -Return cases-
*-1- (INT_E_OK) if there is something wrong
*-2- (INT_E_NOK) otherwise
*/
u8_en_interruptErrorType ext_enable(uint8_t u8_intNum);
```

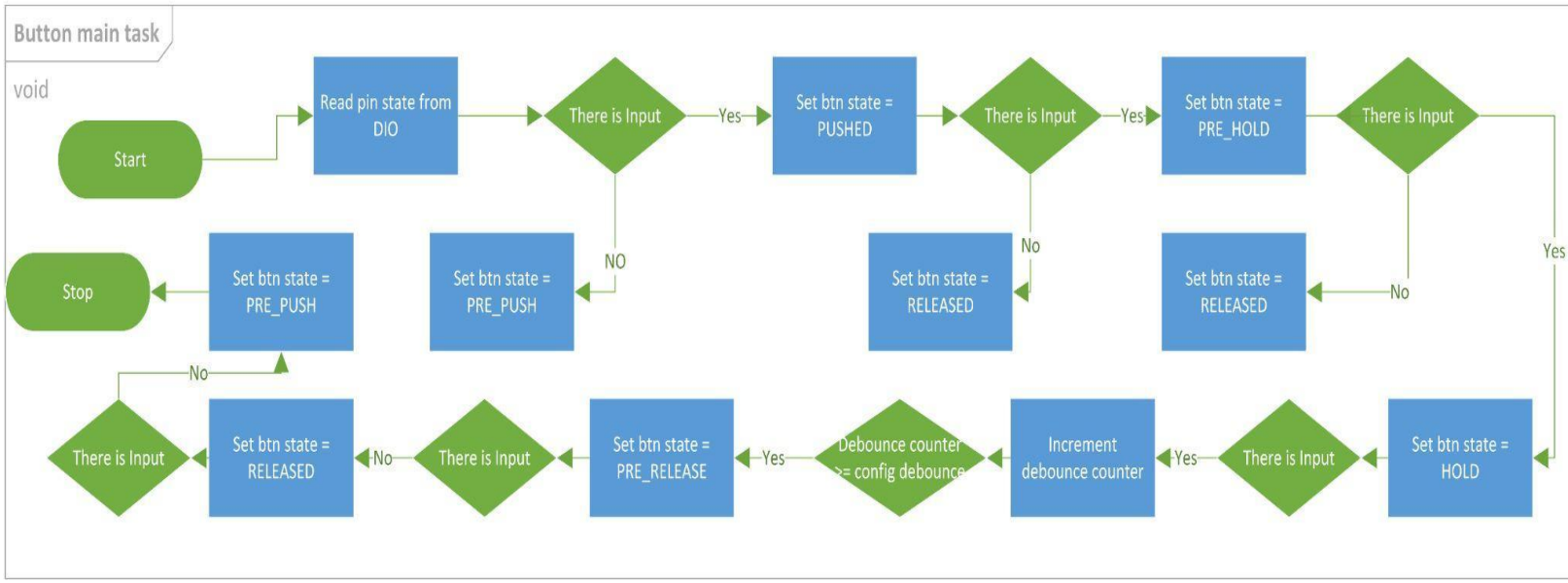
```
/*
*   -Description-
*-this function disable external interrupt depend on external interrupt configuration
*
*   -Input Output parameters -
*-1-it is configuration of external interrupt (ext_interrupt_config_t
*ext_interrupt_config)
*
*
*   -Return-
*   u8_en_interruptErrorType
*
*   -Return cases-
*-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

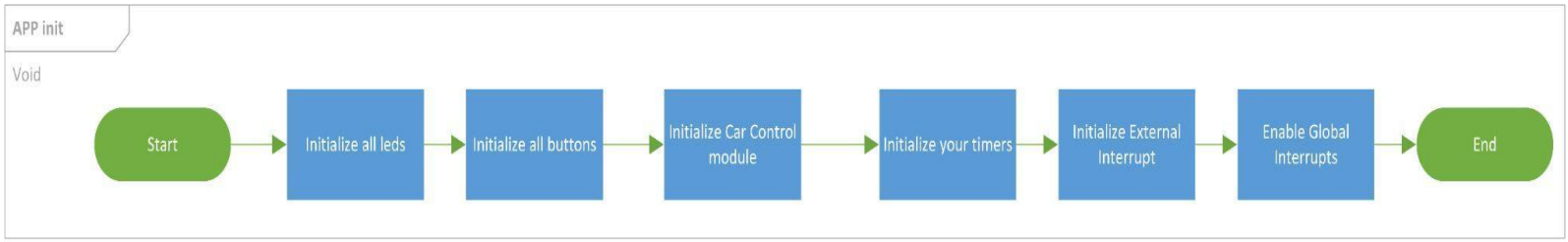
Service name	BUTTON_init
Syntax	u8_en_btnStateType BUTTON_init(uint8_t u8_a_port, uint8_t u8_a_pin, u8_en_btnIdType en_btnId);

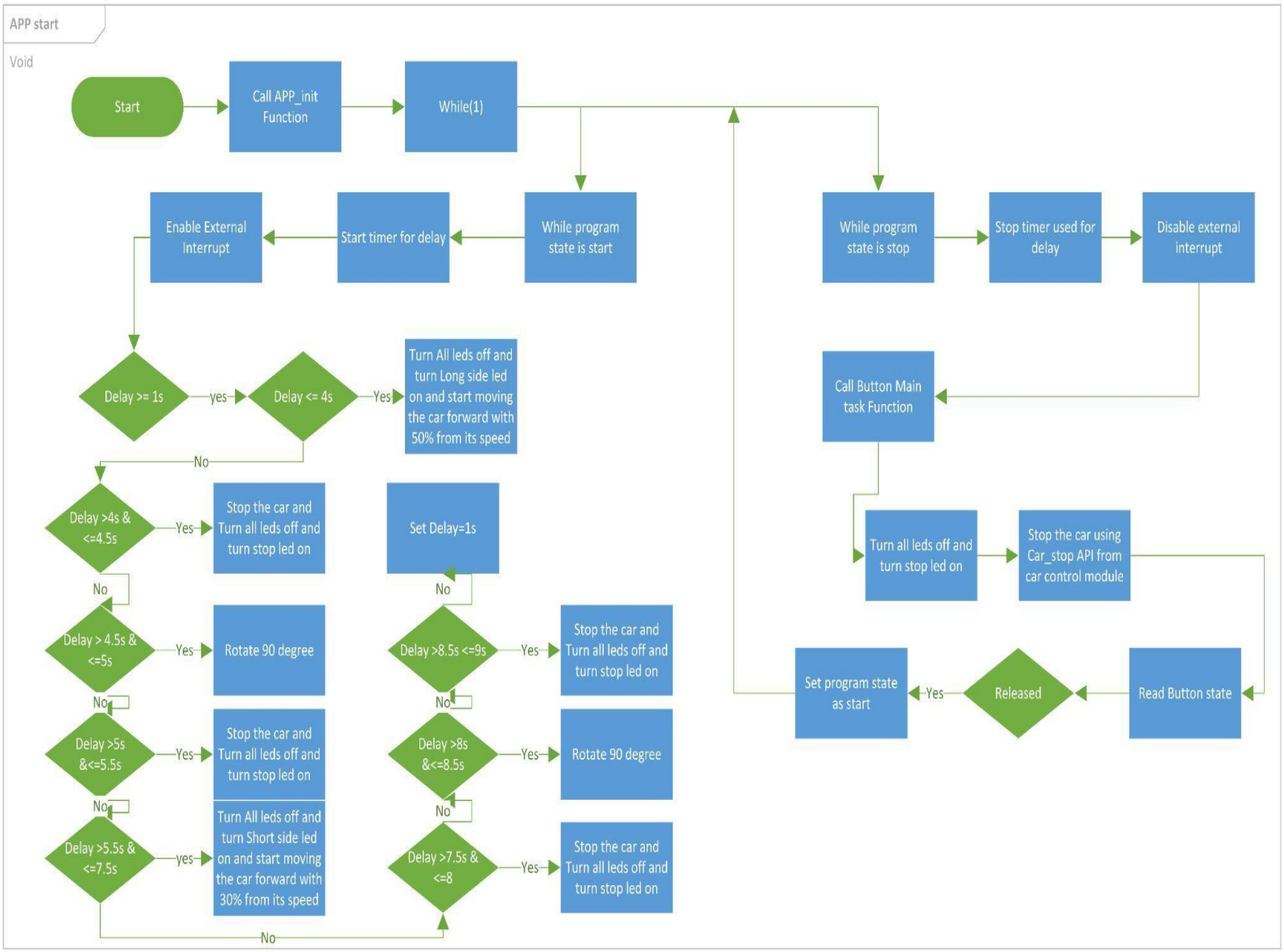
<table>
<tr><td rowspan="2">Parameters (in)</td><td>Port, pin</td><td>Channel ID</td></tr>
<tr><td>en_btnId</td><td>Start 0x00</td></tr>
<tr><td>Return</td><td colspan="2"><table><tr><td rowspan="2">Button_StateTyp</td><td>BT_PRE_PUSH</td></tr><tr><td>BT_UNDEFINED</td></tr></table></td></tr>
<tr><td>Description</td><td colspan="2">This Function sets the Direction of the button pin as input</td></tr>
</table>

3.4 : App APIs

3.4.1: App API:

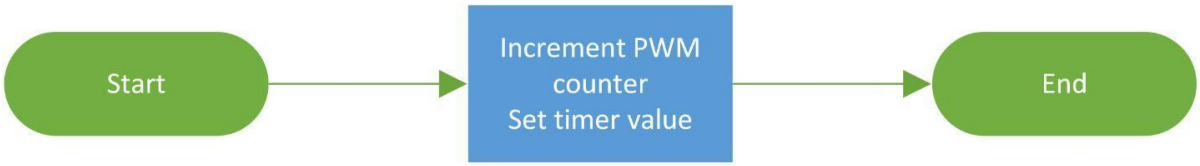
3.4.1.1: Flowcharts:





Timer used for PWM Handeler

Void

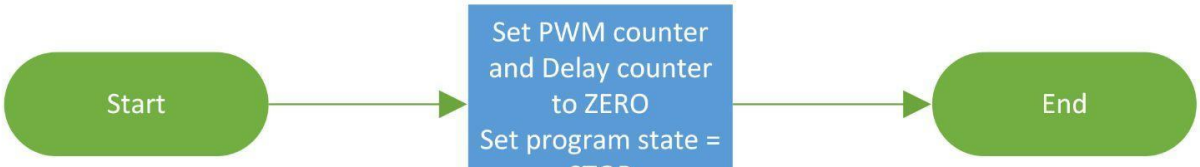


Timer used for Delay Handeler

Void

External Interrupt Handeler

Void



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