Obstacle avoidance car Design

Team 4

Arafa Arafa
Bassel Yasser Mahmoud
Mahmoud Sarhan
Youssef Ahmed Abbas

Table Of Content:

1 : Detailed Requirements

System Requirements:

1. The car starts initially from 0 speed
2. The default rotation direction is to the right
3. Press PB2 to start or stop the robot
4. After Pressing Start:
    1. The LCD will display a centered message in line 1 “Set Def. Rot.”
    2. The LCD will display the selected option in line 2 “Right”
    3. The robot will wait for 5 seconds to choose between Right and Left
        1. When PB1 is pressed once, the default rotation will be Left and the LCD line 2 will be updated
        2. When PB1 is pressed again, the default rotation will be Right and the LCD line 2 will be updated
        3. For each press the default rotation will changed and the LCD line 2 is updated
        4. After the 5 seconds the default value of rotation is set
    4. The robot will move after 2 seconds from setting the default direction of rotation.
    5. For No obstacles or object is far than 70 centimeters:
        1. The robot will move forward with 30% speed for 5 seconds
        2. After 5 seconds it will move with 50% speed as long as there was no object or objects are located at more than 70 centimeters distance
        3. The LCD will display the speed and moving direction in line 1: “Speed:00% Dir: F/B/R/S”, F: forward, B: Backwards, R: Rotating, and S: Stopped
        4. The LCD will display Object distance in line 2 “Dist.: 000 Cm”
    6. For Obstacles located between 30 and 70 centimeters
        1. The robot will decrease its speed to 30%
        2. LCD data is updated
    7. For Obstacles located between 20 and 30 centimeters
        1. The robot will stop and rotates 90 degrees to right/left according to the chosen configuration
        2. The LCD data is updated
    8. For Obstacles located less than 20 centimeters
        1. The robot will stop, move backwards with 30% speed until distance is greater than 20 and less than 30
        2. The LCD data is updated
        3. Then preform point 8
    9. Obstacles surrounding the robot (Bonus)

1. If the robot rotated for 360 degrees without finding any distance greater than 20 it will stop
2. LCD data will be updated.
3. The robot will frequently (each 3 seconds) check if any of the obstacles was removed or not and move in the direction of the furthest object

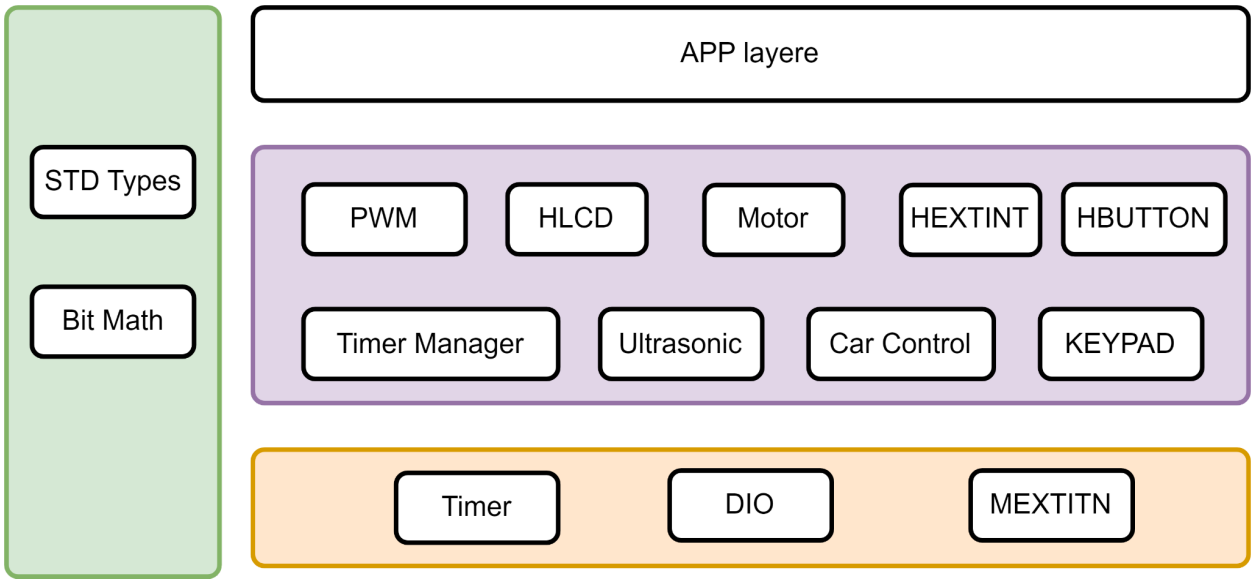
2 : Layered architecture

APP Layer: written in high level languages like java, C++, C# with rich GUI support. Application layer calls the middleware api in response to action by the user or an event.

HAL Layer: are a way to provide an interface between hardware and software so applications can be device independent.

MCAL Layer: is a software module that directly accesses on-chip MCU peripheral modules and external devices that are mapped to memory, and makes the upper software layer independent of the MCU. Details of the MCAL software module are shown below.

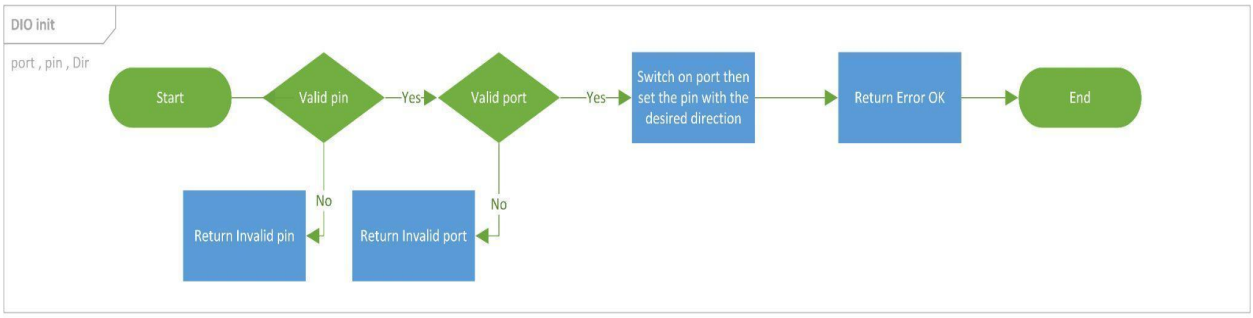
Common Layer: is the layer which consists of BIT_MATH and STD types

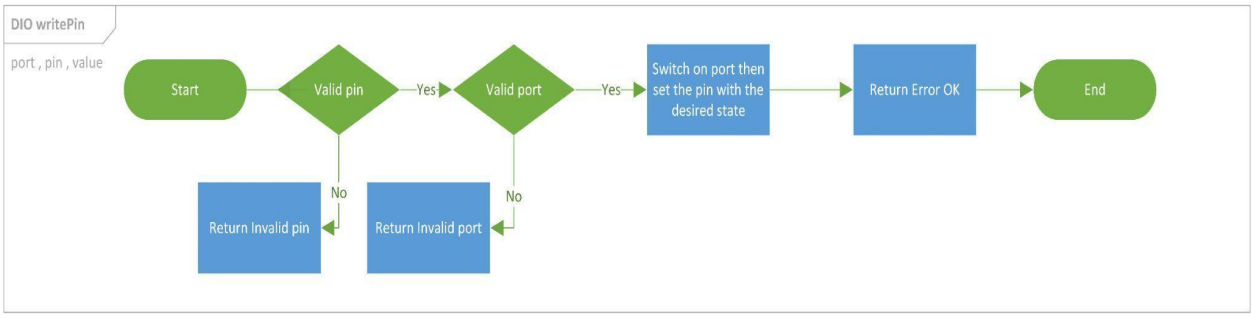


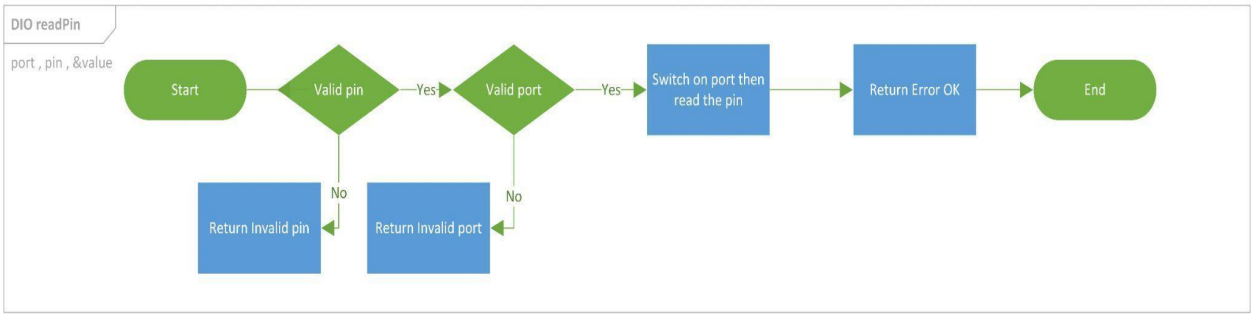
3.2: MCAL APIs

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_dioErrorsType		
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>u8_en_dioErrors</td><td>DIO_E_OK</td></tr><tr><td rowspan="2"></td><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

<table>
<tr><td>Service name</td><td colspan="3">DIO_writePIN</td></tr>
<tr><td>Syntax</td><td colspan="3">u8_en_dioErrors DIO_writePIN (
en_dioPortsType port, en_dioPinsType pin,
u8_en_dioLevelType state
);</td></tr>
<tr><td rowspan="3">Parameters (in)</td><td>Port, pin</td><td colspan="2">Channel ID</td></tr>
<tr><td rowspan="2">state</td><td rowspan="2">Value to be set</td><td>STD_HIGH</td></tr>
<tr><td>STD_LOW</td></tr>
<tr><td>Return</td><td colspan="3"><table><tr><td>u8_en_dioErrors</td><td>DIO_E_OK</td></tr><tr><td rowspan="2"></td><td>DIO_InvalidPin</td></tr><tr><td>DIO_InvalidPort</td></tr></table></td></tr>
<tr><td>Description</td><td colspan="3">This Function sets 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.

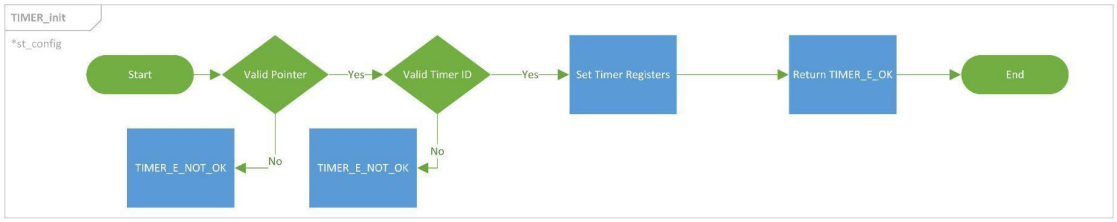
- DIO_init

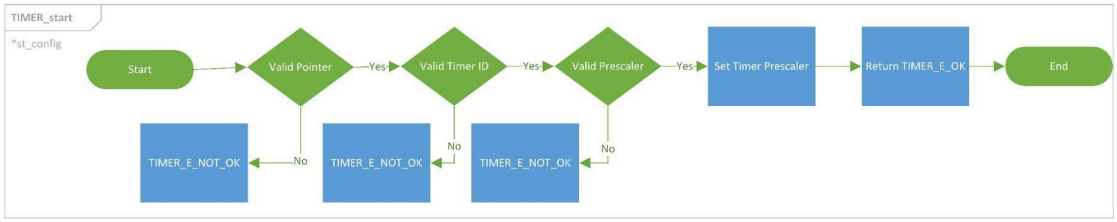
<table>
<tr><td>Service name</td><td colspan="3">DIO_init</td></tr>
<tr><td>Syntax</td><td colspan="3">u8_en_dioErrors DIO_init (
en_dioPortsType port,
en_dioPinsType pin,
u8_en_dioDirType direction
);</td></tr>
<tr><td rowspan="3">Parameters (in)</td><td>Port, pin</td><td colspan="2">Channel ID</td></tr>
<tr><td rowspan="2">direction</td><td rowspan="2">Value to be set</td><td>STD_INPUT</td></tr>
<tr><td>STD_OUTPUT</td></tr>
<tr><td>Return</td><td colspan="3"><table><tr><td>DIO_Errors</td><td>DIO_E_OK</td></tr><tr><td></td><td>DIO_InvalidPin</td></tr><tr><td></td><td>DIO_InvalidPort</td></tr></table></td></tr>
<tr><td>Description</td><td colspan="3">This Function sets the Direction of the pin</td></tr>
</table>

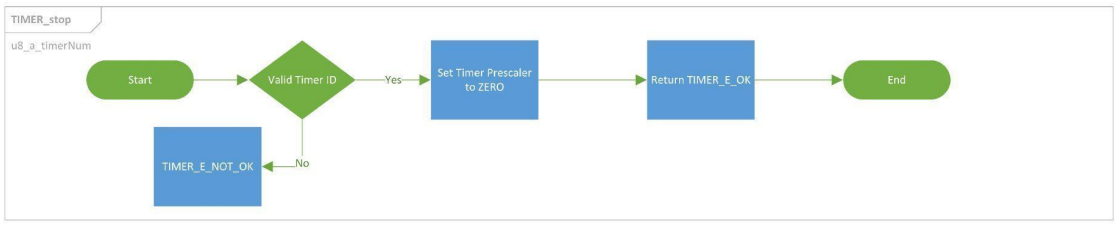
- This function shall return DIO_InvalidPin if pin number is invalid
- This function shall return DIO_InvalidPort if port number is invalid.

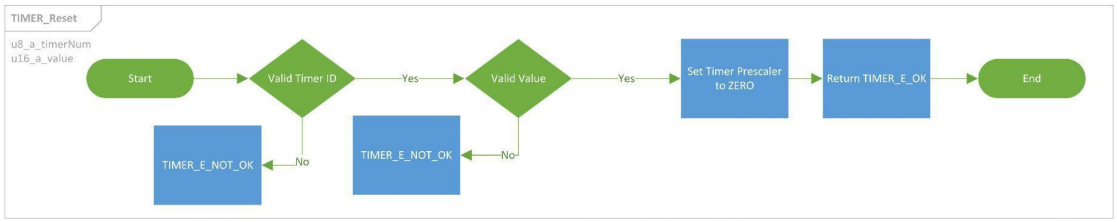
3.2.2: Timer API:

3.2.2.1 :Flowcharts:









3.2.2.2 : Type definitions:

- st_timerConfigType

Name	st_timerConfigType
Type	Structure
Range	Shall contain required timer configuration
Description	st_timerConfigType
Available via	timer_types.h

- u8_en_timerErrorsType

Name	u8_en_timerErrorsType		
Type	Enumeration		
Range	TIMER_E_OK	0x00	Timer error OK
	TIMER_E_NOT_OK	0x03	Timer error
Description	u8_en_timerErrorsType		
Available via	timer_types.h		

- u8_en_timerPrescalerType

Name	u8_en_timerPrescalerType
Type	Enumeration
Range	Shall Contain all Prescaler values
Description	u8_en_timerPrescalerType
Available via	timer_types.h

- u8_en_timerNumberType

Name	u8_en_timerNumberType
Type	Enumeration
Range	Shall Contain all Timers IDs
Description	u8_en_timerNumberType
Available via	timer_types.h

3.2.2.3 : Services affecting the hardware unit

- TIMER_init

Service name	TIMER_init	
Syntax	u8_en_timerErrorsType TIMER_init (st_timerConfigType* st_config);	
Parameters (in)	st_config	Pointer to the configuration structure
Return	u8_en_timerErrorsType	TIMER_E_OK TIMER_E_NOT_OK
Description	This Function Initialize TIMER module	

  - This function shall return TIMER_E_NOK if st_config is NULL
  - This function shall return TIMER_E_NOK if any of the configuration elements is invalid.

- TIMER_start

Service name	TIMER_start	
Syntax	u8_en_timerErrorsType TIMER_start (st_timerConfigType* st_config);	
Parameters (in)	st_config	Pointer to the configuration structure
Return	u8_en_timerErrorsType	TIMER_E_OK

		TIMER_E_NOT_OK
Description	This Function start TIMER	

- This function shall return TIMER_E_NOK if st_config is NULL
- This function shall return TIMER_E_NOK if any of the configuration elements is invalid.

- TIMER_stop

Service name	TIMER_stop	
Syntax	u8_en_timerErrorsType TIMER_stop (u8_en_timerNumberType u8_a_timerNum);	
Parameters (in)	u8_a_timerNum	Pointer to the configuration structure
Return	u8_en_timerErrorsType	TIMER_E_OK TIMER_E_NOT_OK
Description	This Function stop TIMER	

- This function shall return TIMER_E_NOK if u8_a_timerNum is invalid

- TIMER_reset

Service name	TIMER_reset	
Syntax	u8_en_timerErrorsType TIMER_reset (st_timerConfigType* st_config);	
Parameters (in)	st_config	Timer ID
Return	u8_en_timerErrorsType	TIMER_E_OK TIMER_E_NOT_OK
Description	This Function reset the TIMER	

- This function shall return TIMER_E_NOK if st_config is NULL
- This function shall return TIMER_E_NOK if any of the configuration elements is invalid.

- TIMER_setCallBack

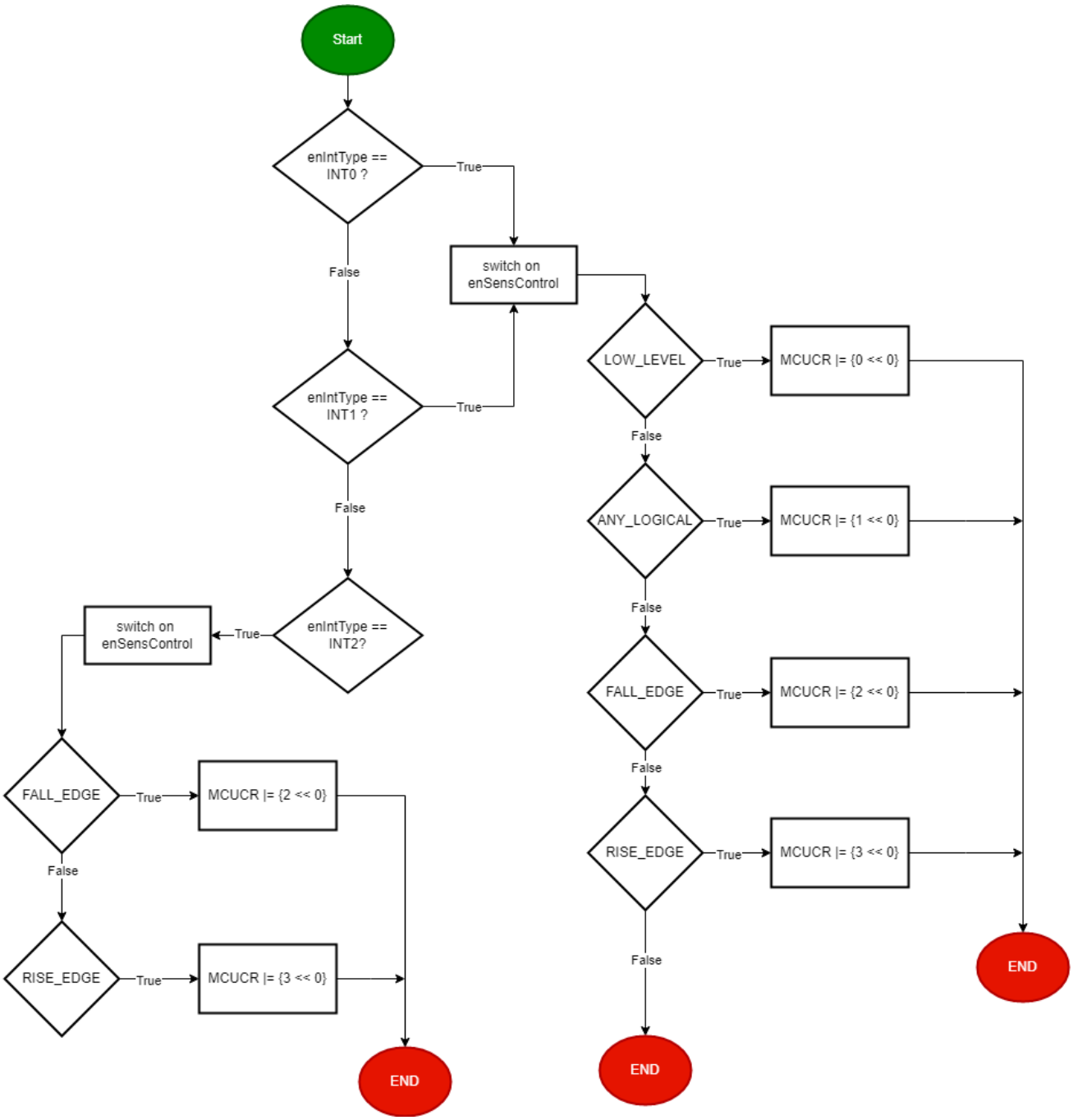
<table>
<tr><td>Service name</td><td colspan="2">TIMER_setCallBack</td></tr>
<tr><td>Syntax</td><td colspan="2">u8_en_timerErrorsType TIMER_setCallBack (
void(*a_timerCallBack)(void),
u8_en_timerNumberType u8_a_timerNum
);</td></tr>
<tr><td rowspan="2">Parameters (in)</td><td>*a_timerCallBack</td><td>Pointer to the Callback function</td></tr>
<tr><td>u8_a_timerNum</td><td>Timer ID</td></tr>
<tr><td>Return</td><td colspan="2"><table><tr><td rowspan="2">u8_en_timerErrorsType</td><td>TIMER_E_OK</td></tr><tr><td>TIMER_E_NOT_OK</td></tr></table></td></tr>
<tr><td>Description</td><td colspan="2">This Function reset the TIMER</td></tr>
</table>

- This function shall return TIMER_E_NOK if a_timerCallBack is NULL
- This function shall return TIMER_E_NOK if u8_a_timerNum is invalid.

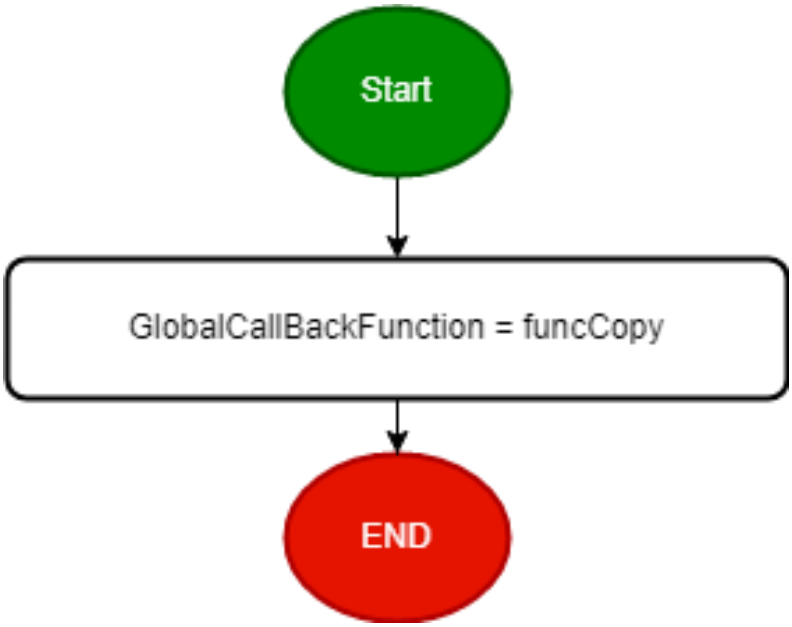
3.2.3: ExtInt API:

3.2.3.1 :Flowcharts:

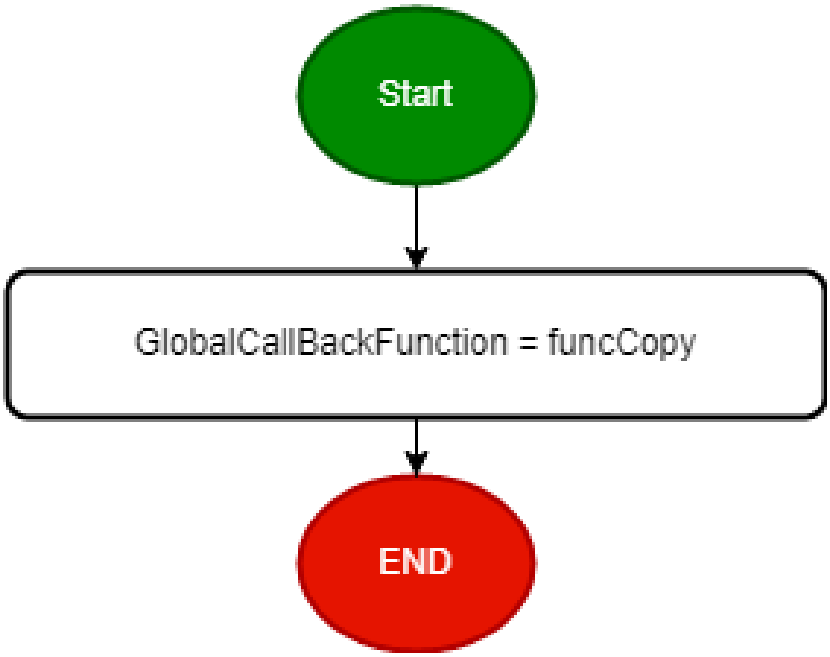
Uint8_t vidExtInt_init (enu_int_type_t, enu_sns_ctrl_t)



```
void vidCallBackFunc (ptr_func funcCopy)
```



```
void vidCallBackFuncInt1(ptr_func funcCopy);
```



3.2.3.2 : Services affecting the hardware unit

Uint8_t vidExtInt_init (en_int_type_t, en_sns_ctrl_t);

Service name	vidExtInt_init	
Parameters (in)	en_int_type_t	Interrupt type [INT0, INT1. INT2]
	en_sns_ctrl_t	snsCtrl : Sense Control {ANY_LOGICAL, FALL_EDGE, RISE_EDGE}
Return	Uint8_t	MEXTINT_OK MEXTINT_NOK
Description	External Interrupt Initialization	

Uint8_t vidCallBackFunc (ptr_func funcCopy);

Service name	vidCallBackFunc	
Parameters (in)	ptr_func	Pointer to function
Return	Uint8_t	MEXTINT_OK MEXTINT_NOK
Description	Take pointer to function to be executed in ISR when it fires	

Uint8_t vidCallBackFuncInt1(ptr_func funcCopy);

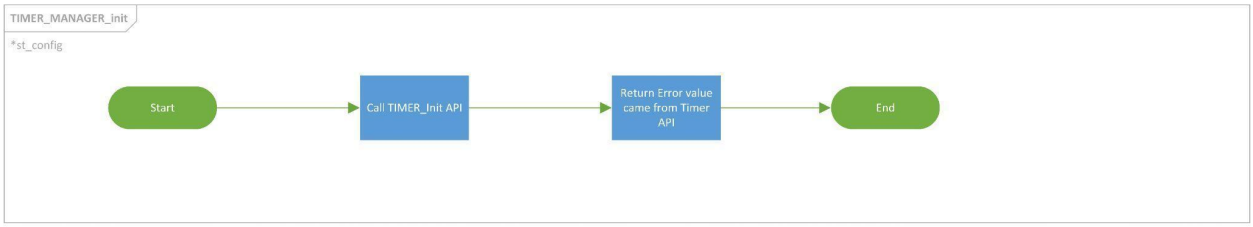
Service name	vidCallBackFuncInt1

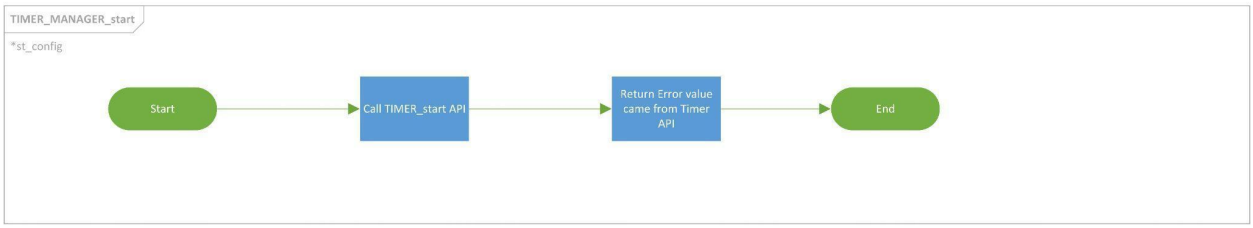
<table>
<tr><td>Parameters (in)</td><td>ptr_func</td><td>Pointer to function</td></tr>
<tr><td>Return</td><td colspan="2"><table><tr><td>Uint8_t</td><td>MEXTINT_OK</td></tr><tr><td></td><td>MEXTINT_NOK</td></tr></table></td></tr>
<tr><td>Description</td><td colspan="2">Take pointer to function to be executed in ISR when it fires for Int 1</td></tr>
</table>

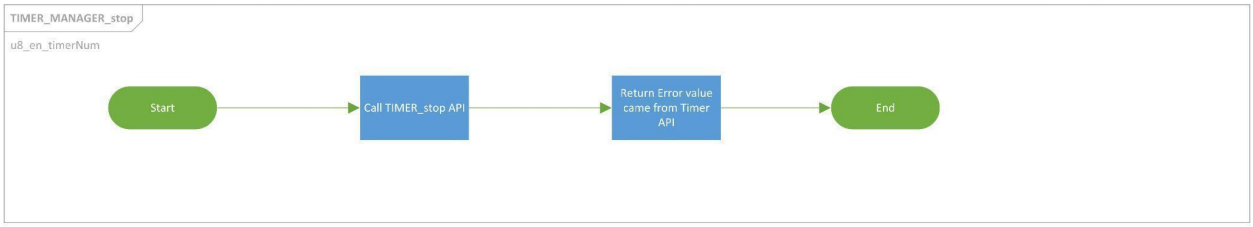
3.3 : HAL APIs

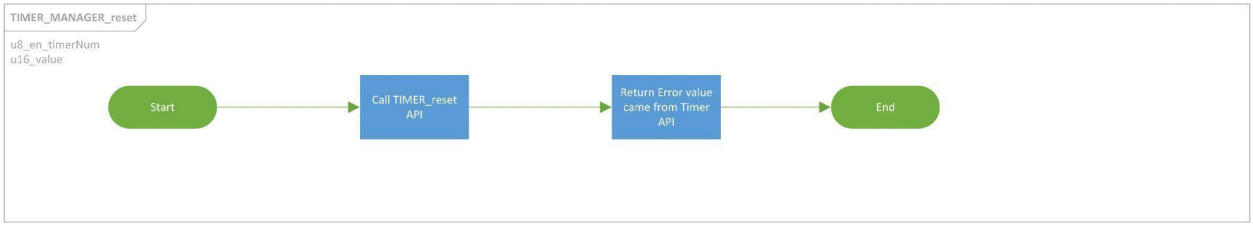
3.3.1: Timer Manager API:

3.3.1.1 :Flowcharts:









3.3.1.2 : Type definitions:

Imported from Timer Module

3.3.1.3 : Services affecting the hardware unit

- TIMER_Manager_init

Service name	TIMER_Manager_init	
Syntax	u8_en_timerErrorsType TIMER_Manager_init (st_timerConfigType* st_config);	
Parameters (in)	st_config	Pointer to the configuration structure
Return	u8_en_timerErrorsType	TIMER_E_OK TIMER_E_NOT_OK
Description	This Function Initialize TIMER module	

- This function shall return TIMER_E_NOK if st_config is NULL
- This function shall return TIMER_E_NOK if any of the configuration elements is invalid.

- TIMER_Manager_start

Service name	TIMER_Manager_start	
Syntax	u8_en_timerErrorsType TIMER_Manager_start (st_timerConfigType* st_config);	
Parameters (in)	st_config	Pointer to the configuration structure
Return	u8_en_timerErrorsType	TIMER_E_OK TIMER_E_NOT_OK
Description	This Function start TIMER	

- This function shall return TIMER_E_NOK if st_config is NULL
- This function shall return TIMER_E_NOK if any of the configuration elements is invalid.

- TIMER_Manager_stop

Service name	TIMER_Manager_stop
Syntax	u8_en_timerErrorsType TIMER_Manager_stop (u8_en_timerNumberType u8_en_timerNum);

Parameters (in)	u8_en_timerNum	Timer ID
Return	u8_en_timerErrorsType	TIMER_E_OK TIMER_E_NOT_OK
Description	This Function stop TIMER	

- This function shall return TIMER_E_NOK if u8_en_timerNum is invalid

- TIMER_Manager_reset

Service name	TIMER_Manager_reset	
Syntax	u8_en_timerErrorsType TIMER_Manager_reset (st_timerConfigType* st_config);	
Parameters (in)	st_config	Pointer to the configuration structure
Return	u8_en_timerErrorsType	TIMER_E_OK TIMER_E_NOT_OK
Description	This Function reset the TIMER with the initial value	

- This function shall return TIMER_E_NOK if st_config is NULL
- This function shall return TIMER_E_NOK if any of the configuration elements is invalid.

3.3.2: LCD API:

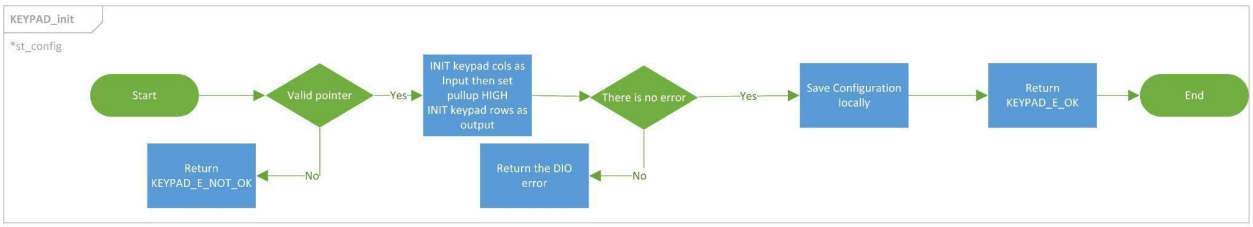
3.3.2.1 :Flowcharts:

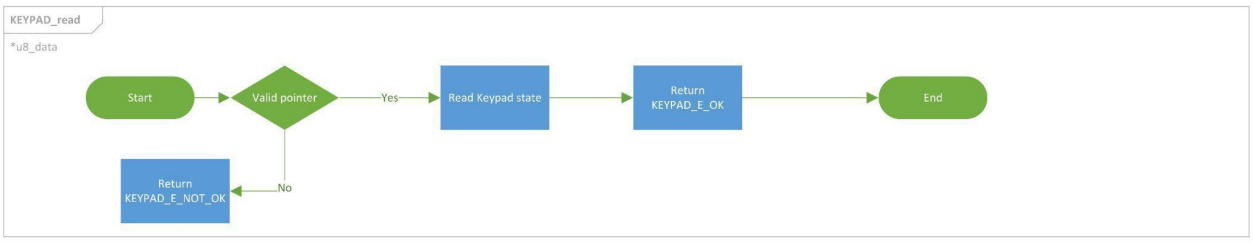
3.3.2.2 : Type definitions:

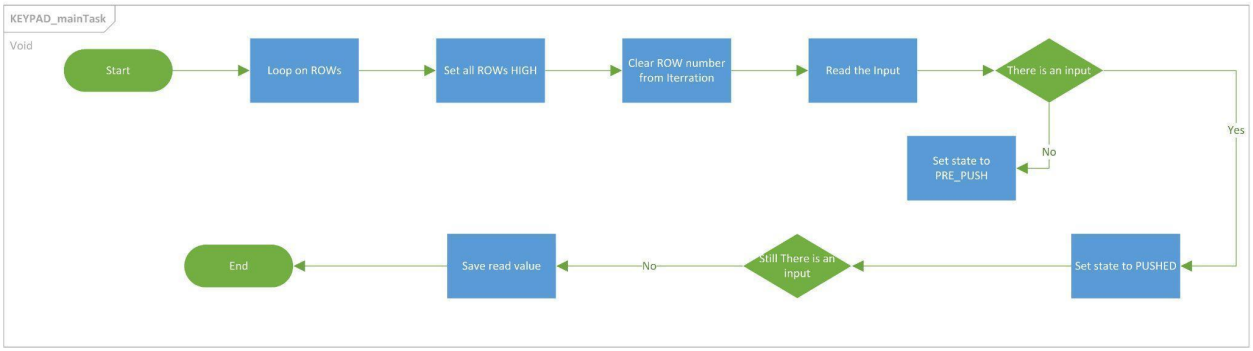
3.3.2.3 : Services affecting the hardware unit

3.3.3 : Keypad API :

3.3.3.1 :Flowcharts:







3.3.3.2 : Type definitions:

- st_keypadConfigType

Name	st_keypadConfigType
Type	Structure
Range	Shall contain required Keypad configuration
Description	st_keypadConfigType

Available via	keypad.h

- u8_en_keypadErrorsType

Name	u8_en_keypadErrorsType		
Type	Enumeration		
Range	KEYPAD_E_OK	0x00	Keypad error OK
	KEYPAD_E_NOT_OK	0x07	Keypad error
Description	u8_en_keypadErrorsType		
Available via	keypad.h		

3.3.3.3 : Services affecting the hardware unit

- KEYPAD_init

Service name	KEYPAD_init	
Syntax	u8_en_keypadErrorsType KEYPAD_init (st_keypadConfigType* st_config);	
Parameters (in)	st_config	Pointer to the configuration structure
Return	u8_en_keypadErrorsType	KEYPAD_E_OK
		KEYPAD_E_NOT_OK
Description	This Function Initialize Keypad module	

- This function shall return KEYPAD_E_NOK if st_config is NULL
- This function shall return KEYPAD_E_NOK if any of the configuration elements is invalid.
- This function shall return DIO_E_NOT_OK if failed to initialize the pin direction to be OUTPUT or INPUT

- KEYPAD_read

Service name	KEYPAD_read	
Syntax	u8_en_keypadErrorsType KEYPAD_read (uint8_t * u8_data);	
Parameters (in)	u8_data	Pointer to variable where to store value read from keypad
Return	u8_en_keypadErrorsType	KEYPAD_E_OK KEYPAD_E_NOT_OK
Description	This Function read Keypad	

3.3.4 : Car Control API :

3.3.4.1: Flowcharts:

3.3.4.2 : Type definitions:

- st_carControlConfigType

Name	st_carControlConfigType
Type	Structure
Range	Shall contain required car motors configuration
Description	st_carControlConfigType
Available via	car_control.h

- u8_en_carControlErrorsType

Name	u8_en_carControlErrorsType		
Type	Enumeration		
Range	CAR_E_OK	0x00	CAR error OK
	CAR_E_NOT_OK	0x07	CAR error
Description	u8_en_carControlErrorsType		
Available via	car_control.h		

3.3.4.2 : Services affecting the hardware unit

- CAR_init

Service name	CAR_init	
Syntax	u8_en_carControlErrorsType CAR_init (st_carControlConfigType* st_config_R, st_carControlConfigType* st_config_L);	
Parameters (in)	st_config	Pointer to the configuration structure
Return	u8_en_carControlErrorsType	CAR_E_OK
		CAR_E_NOT_OK
Description	This Function Initialize car module	

- CAR_moveForward

Service name	• CAR_moveForward
Syntax	u8_en_carControlErrorsType CAR_moveForward(st_carControlConfigType* st_config_R, st_carControlConfigType* st_config_L

	);	
Parameters (in)	st_config	Pointer to the configuration structure
Return	u8_en_carControlErrorsType	CAR_E_OK CAR_E_NOT_OK
Description	This Function moving the car forward	

- CAR_turnRight

Service name	CAR_turnRight	
Syntax	u8_en_carControlErrorsType CAR_turnRight(st_carControlConfigType* st_config_R, st_carControlConfigType* st_config_L);	
Parameters (in)	st_config	Pointer to the configuration structure
Return	u8_en_carControlErrorsType	CAR_E_OK CAR_E_NOT_OK
Description	This Function turn the car right	

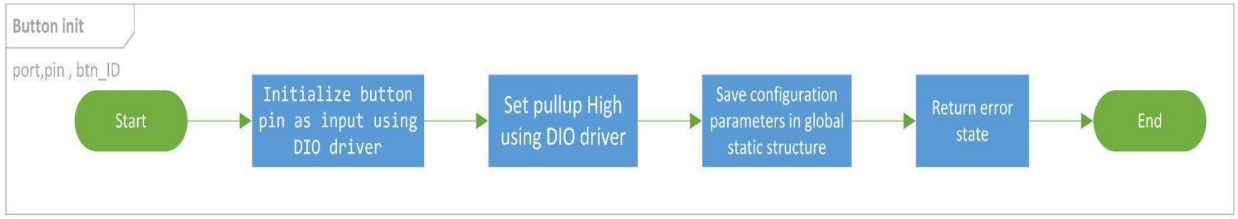
- CAR_turnLeft

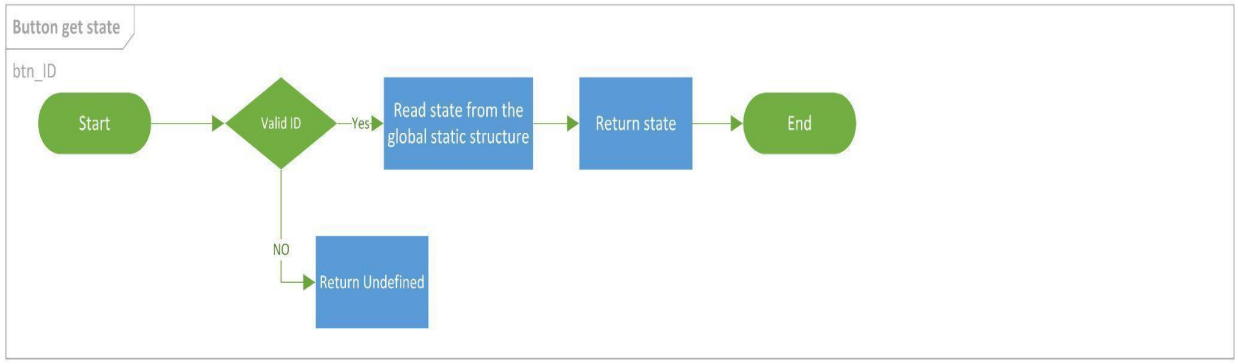
Service name	CAR_turnLeft	
Syntax	u8_en_carControlErrorsType CAR_turnLeft(st_carControlConfigType* st_config_R, st_carControlConfigType* st_config_L);	
Parameters (in)	st_config	Pointer to the configuration structure

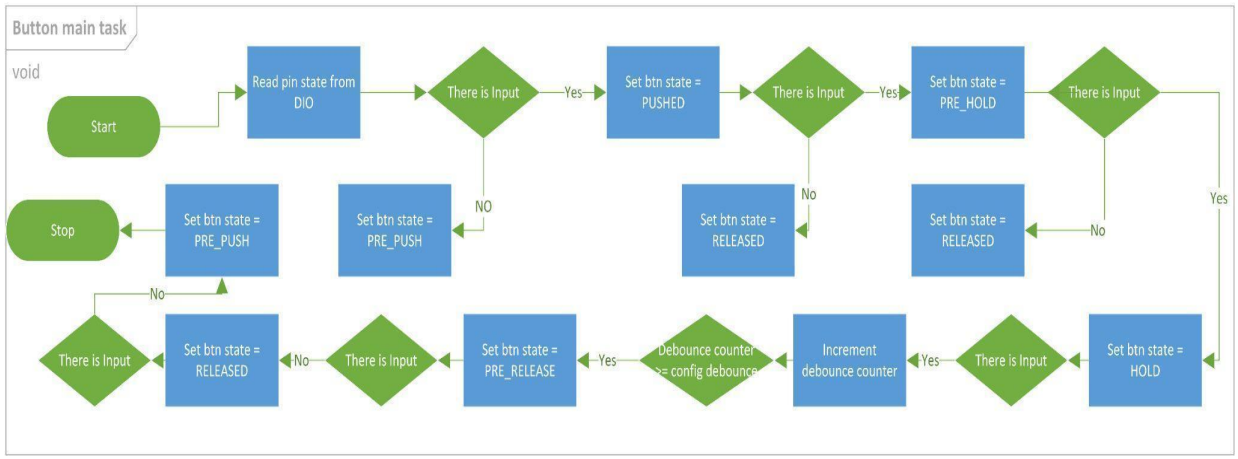
Return	u8_en_carControlErrorsType	CAR_E_OK
		CAR_E_NOT_OK
Description	This Function turn the car left	

3.3.5: Button API:

3.3.5.1: Flowcharts:







3.3.5.2 : 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		

3.3.5.2 : 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

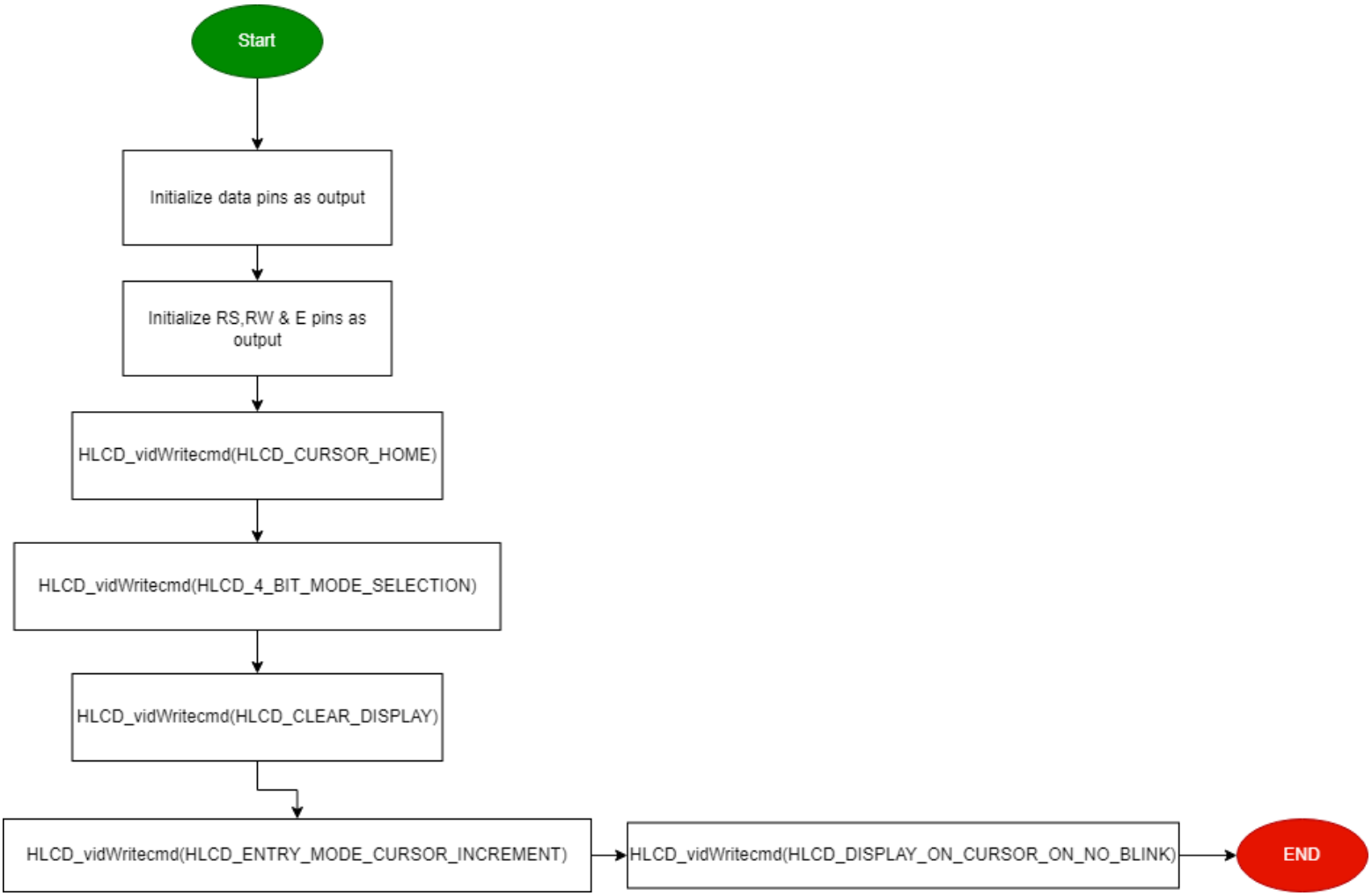
- BUTTON_init

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	

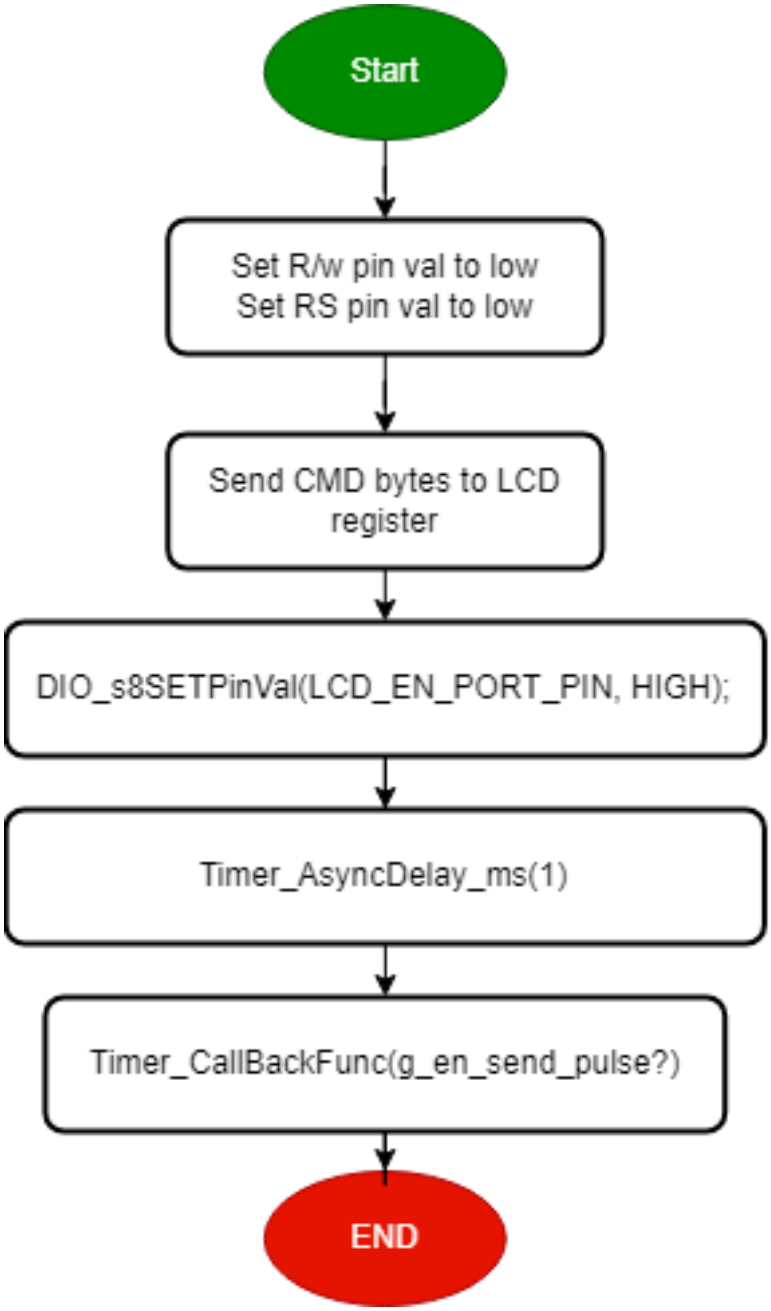
3.3.6 : LCD API :

3.3.6.1 :Flowcharts:

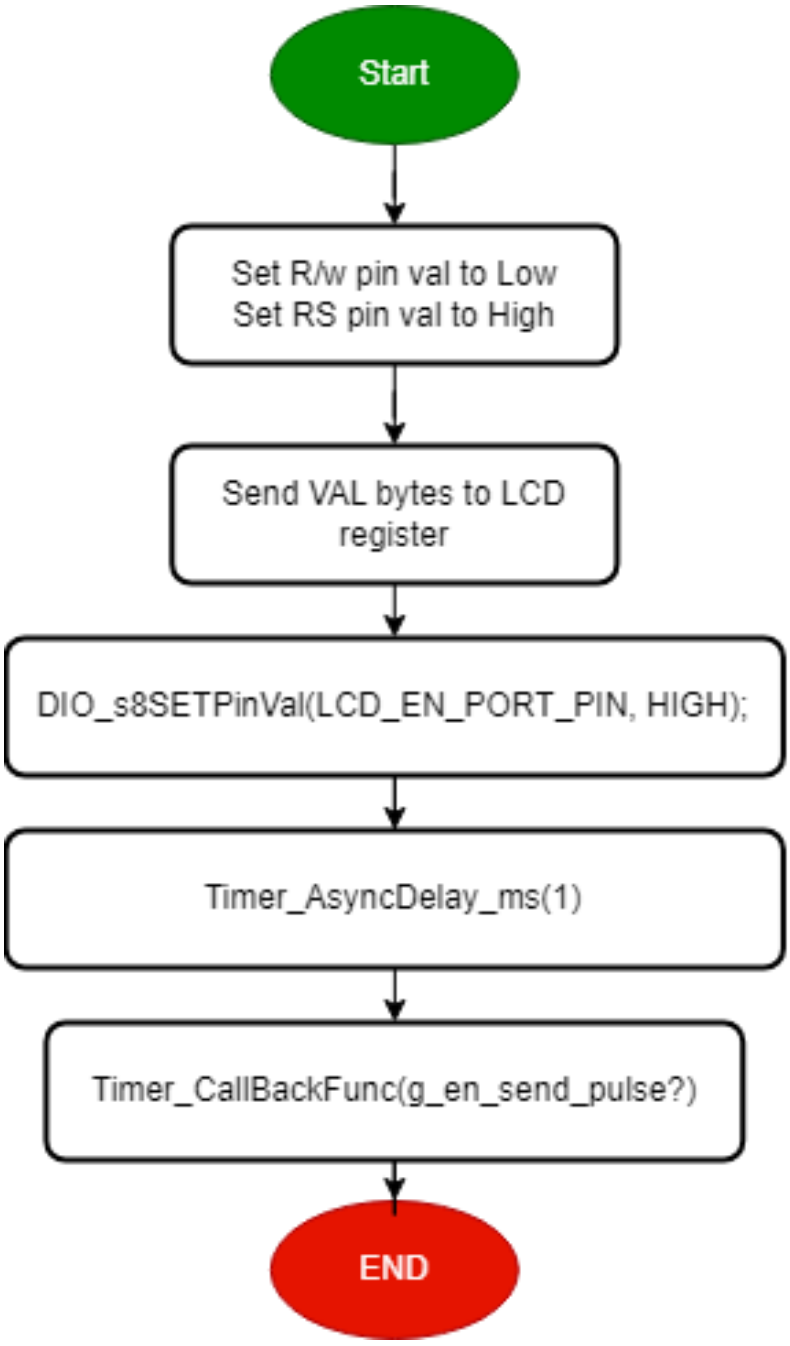
`lcdErrorsType` **HLCD_vidInit(void)**



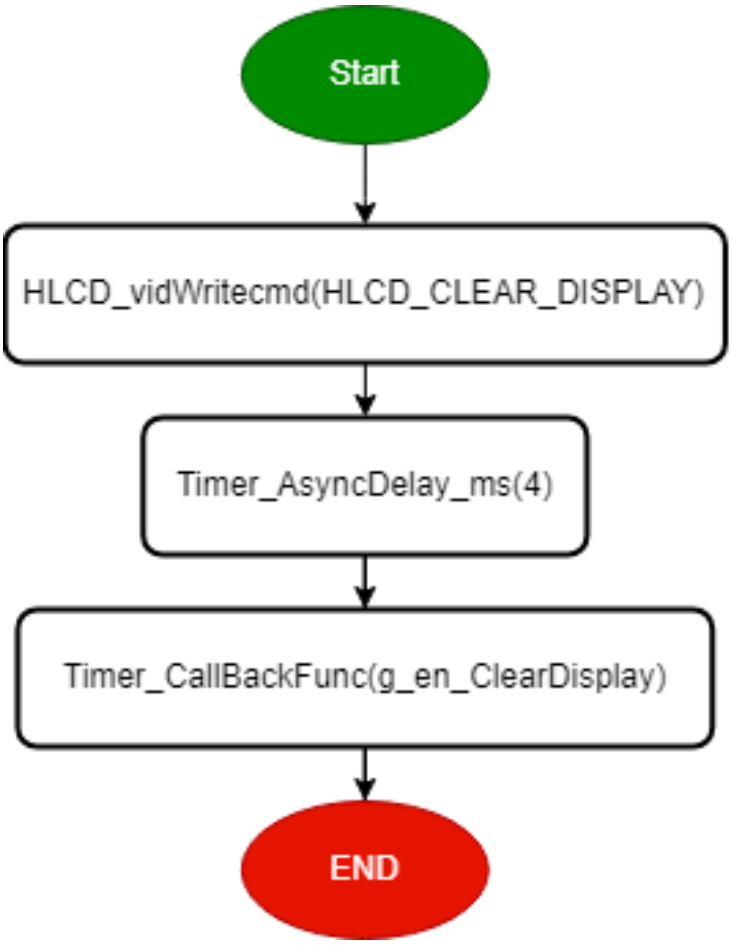
`lcdErrorsType` **HLCD _vidWritecmd (Uint8_t u8commandCopy)**



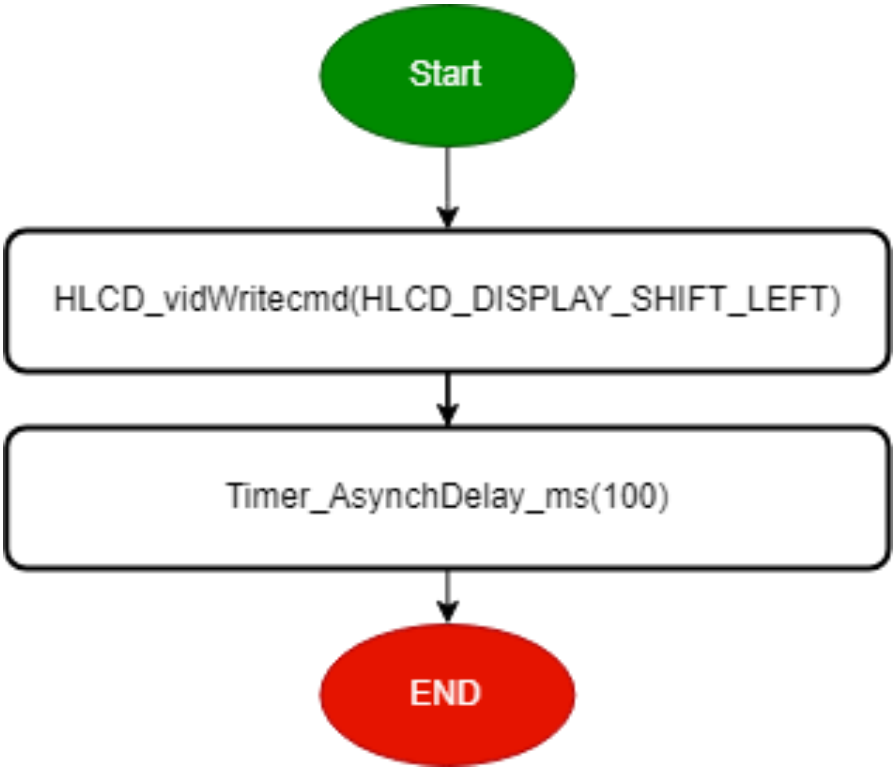
`lcdErrorsType` **HLCD_vidWriteChar (Uint8_t u8CharCopy)**



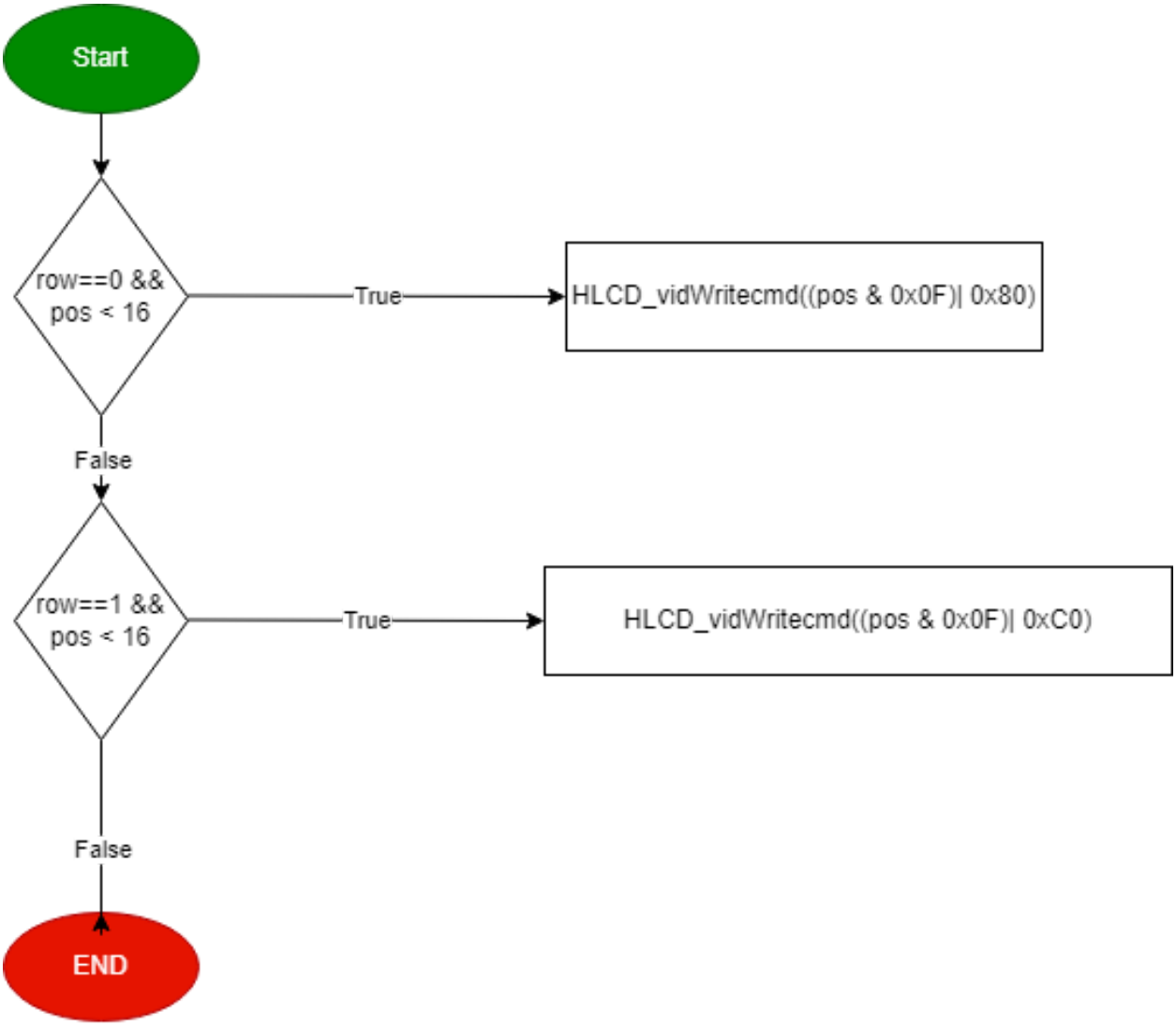
`lcdErrorsType` **HLCD_ClrDisplay(void)**



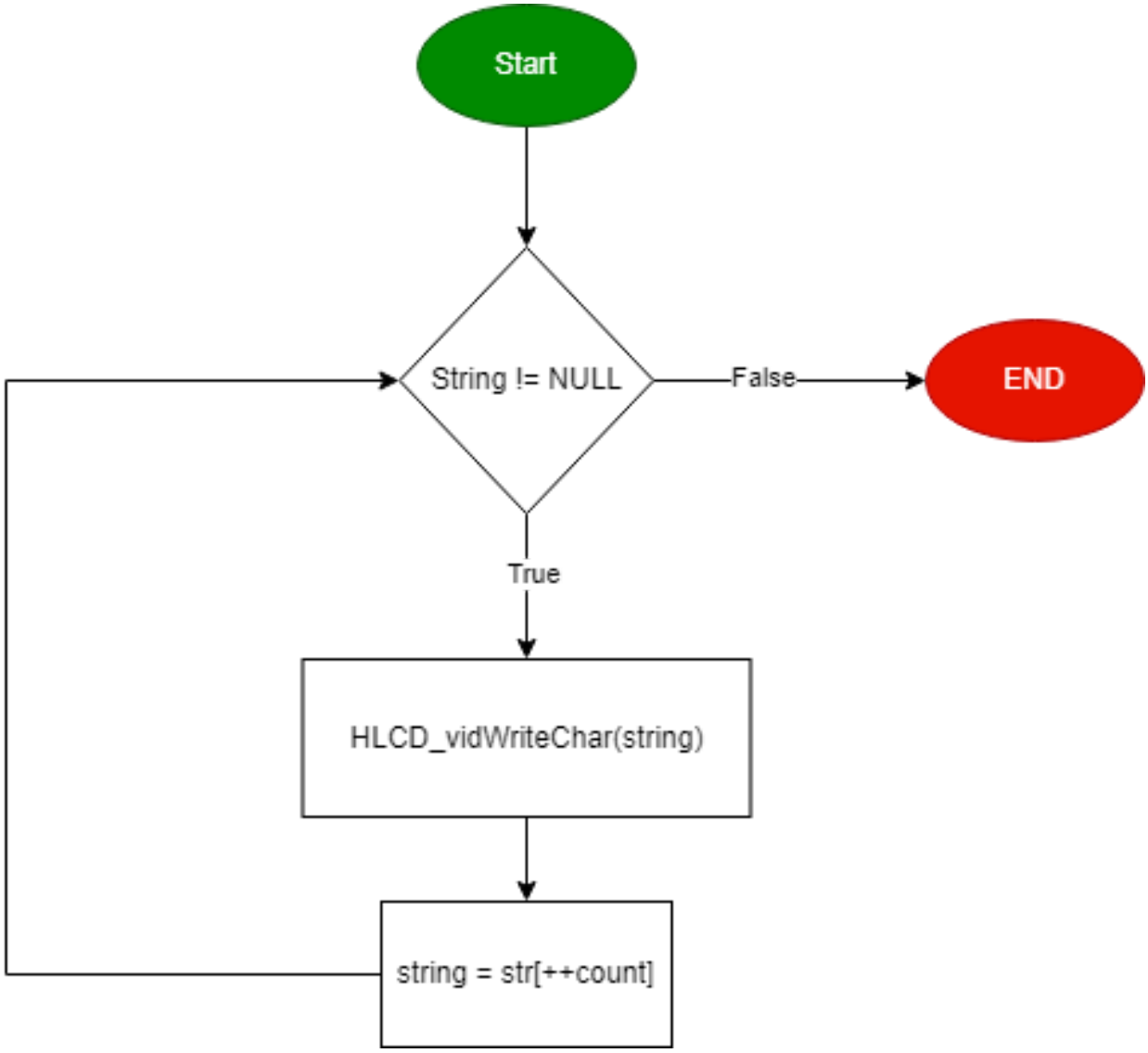
`lcdErrorsType` **HLCD_ShiftLeft(void)**



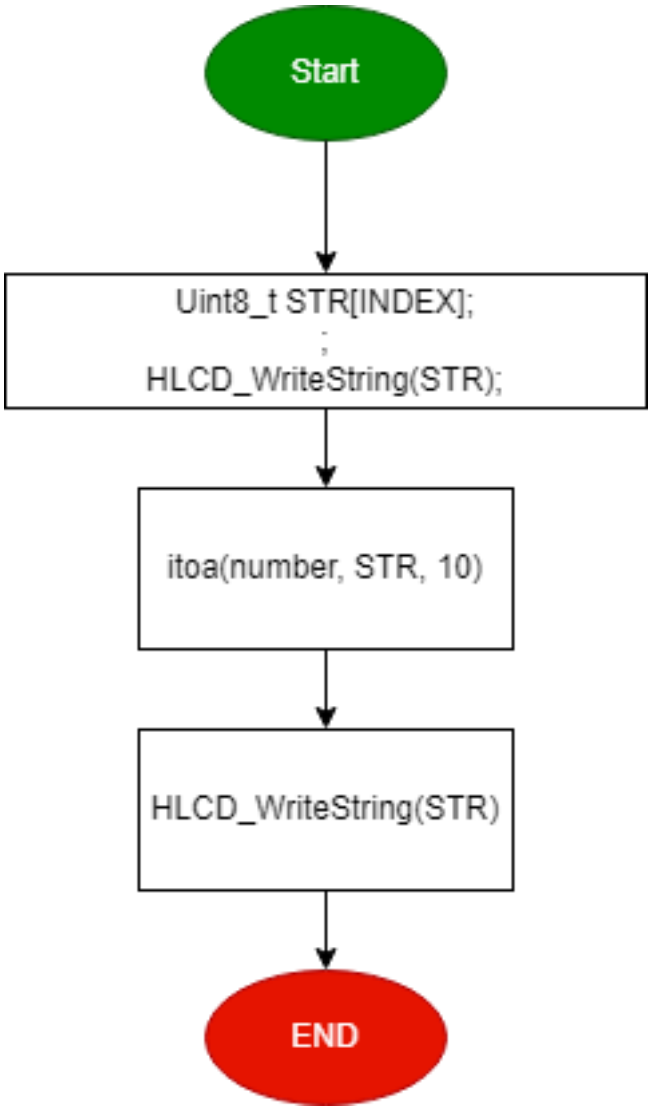
`lcdErrorsType` **HLCD_gotoXY (Uint8_t row, Uint8_t pos)**



`lcdErrorsType` **HLCD_WriteString (Uint8_t* str)**



`lcdErrorsType` **HLCD_WriteInt (Uint32_t number)**



3.3.6.2 : Services affecting the hardware unit

```
u8_en_lcdErrorsType HLCD_vidInit(void);
```

Service name	HLCD_vidInit	
Parameters (in)	**void**	
Return	u8_en_lcdErrorsType	LCD_E_OK
		LCD_E_NOT_OK
Description	func to set LCD initialization	

```
u8_en_lcdErrorsType HLCD_vidWritecmd(uint8_t u8commandCopy);
```

Service name	HLCD_vidWritecmd	
Parameters (in)	uint8_t	u8commandCopy --> take lcd cmd instructions from instruction table <https://components101.com/sites/default/files/component_datasheet/16x2%20LCD%20Datasheet.pdf>}
Return	u8_en_lcdErrorsType	LCD_E_OK
		LCD_E_NOT_OK
Description	func to configure some commands on lcd	

```
u8_en_lcdErrorsType HLCD_vidWriteChar(uint8_t u8CharCopy);
```

Service name	HLCD_vidWriteChar	
Parameters (in)	uint8_t	u8CharCopy -> take ascii code of char or char address on CGROM
Return	u8_en_lcdErrorsType	LCD_E_OK LCD_E_NOT_OK
Description	func to write char on lcd	

```
u8_en_lcdErrorsType HLCD_ClrDisplay(void);
```

Service name	HLCD_ClrDisplay	
Parameters (in)	**void**	
Return	u8_en_lcdErrorsType	LCD_E_OK LCD_E_NOT_OK
Description	func to clear anything on lcd	

```
u8_en_lcdErrorsType HLCD_ShiftLeft(void);
```

Service name	HLCD_ShiftLeft	
Parameters (in)	**void**	
Return	u8_en_lcdErrorsType	LCD_E_OK LCD_E_NOT_OK
Description	func to shift the lcd display from right to left	

```
u8_en_lcdErrorsType HLCD_gotoXY(uint8_t row, uint8_t pos);
```

Service name	HLCD_gotoXY	
Parameters (in)	uint8_t	row -> take row number 0 or 1

	uint8_t	pos -> take colom number from 0 ~ 16
Return	u8_en_lcdErrorsType	LCD_E_OK LCD_E_NOT_OK
Description	func to determine position which char print at this position on lcd ### NOTE : (2rows x 16coloms)	

```
u8_en_lcdErrorsType HLCD_WriteString(uint8_t* str);
```

Service name	HLCD_WriteString	
Parameters (in)	uint8_t*	str --> which take string as argument
Return	u8_en_lcdErrorsType	LCD_E_OK LCD_E_NOT_OK
Description	func to write string on lcd	

```
u8_en_lcdErrorsType HLCD_WriteInt(Uint32_t number);
```

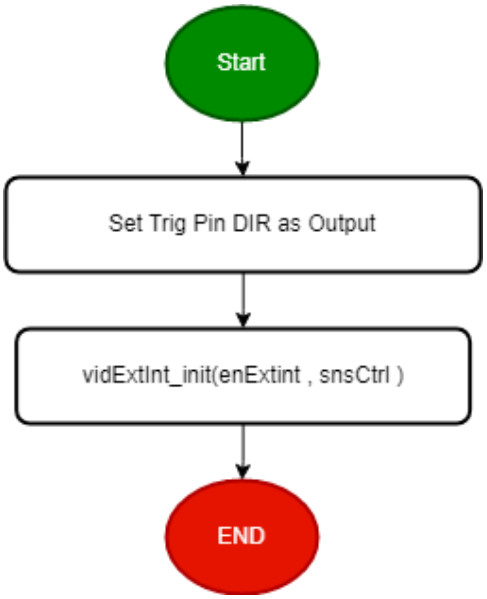
Service name	HLCD_WriteInt	
Parameters (in)	Uint32_t	number --> which take number as argument
Return	u8_en_lcdErrorsType	LCD_E_OK LCD_E_NOT_OK

Description	func to write integer number on lcd

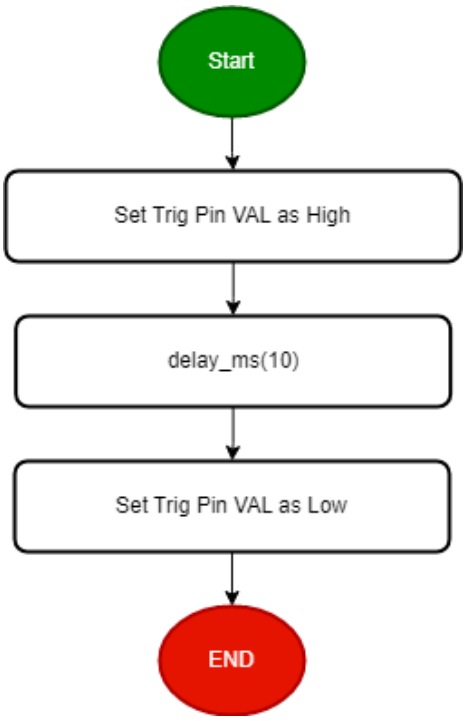
3.3.7 : Ultrasonic API :

3.3.7.1 :Flowcharts:

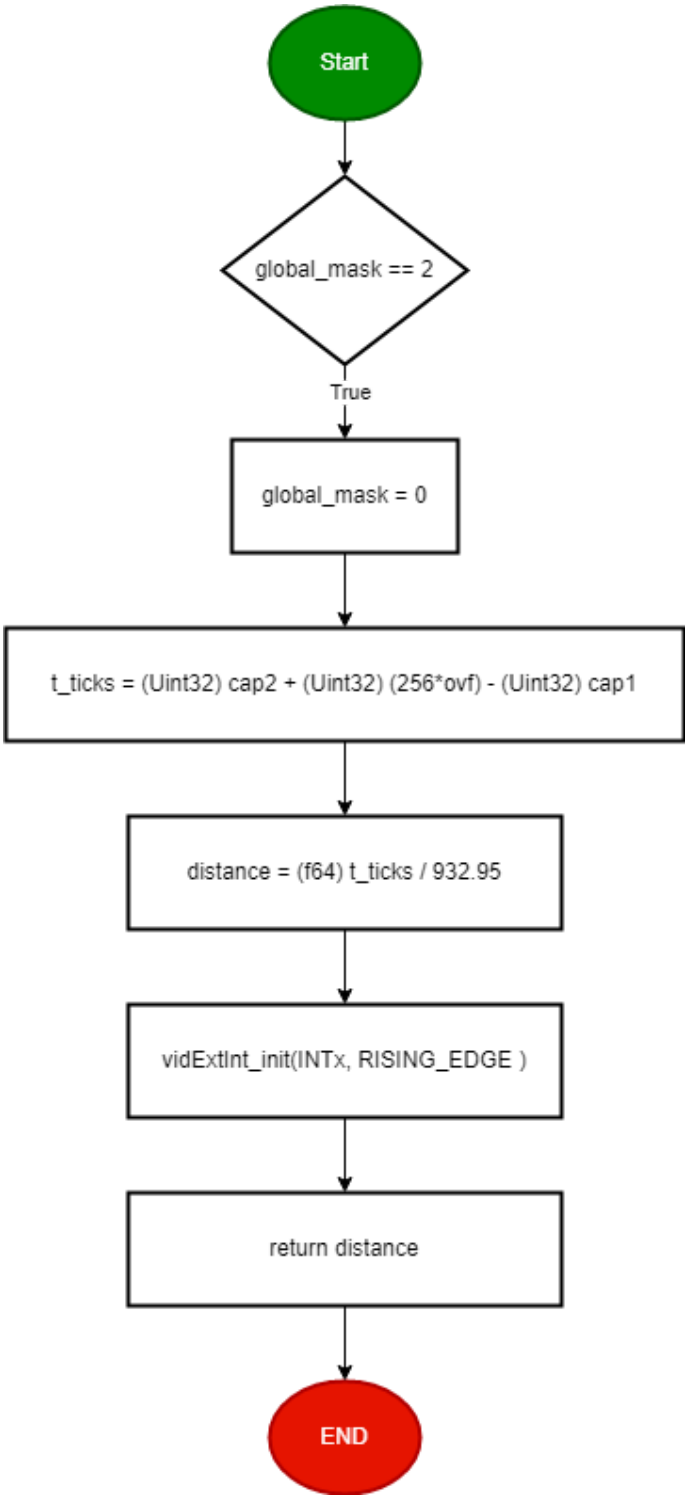
void HULTRASONIC_vidInit (enu_int_type_t enExtint, enu_sns_ctrl_t snsCtrl)



void HULTRASONIC_vidTrigger(**void**)



Uint8_t HULTRASONIC_u8Read(void)



3.3.7.2 : Services affecting the hardware unit

void HULTRASONIC_vidInit (en_int_type_t enExtint, en_sns_ctrl_t snsCtrl);

Service name	HULTRASONIC_vidInit	
Parameters (in)	en_int_type_t	Interrupt type [INT0, INT1. INT2]
	en_sns_ctrl_t	snsCtrl : Sense Control {ANY_LOGICAL, FALL_EDGE, RISE_EDGE}
Return	void	
Description	Set Echo pin as input Set trig pin as output Initialize external interrupt Initialize timer2	

void HULTRASONIC_vidTrigger(void);

Service name	HULTRASONIC_vidTrigger
Parameters (in)	void
Return	void
Description	Sending pulse

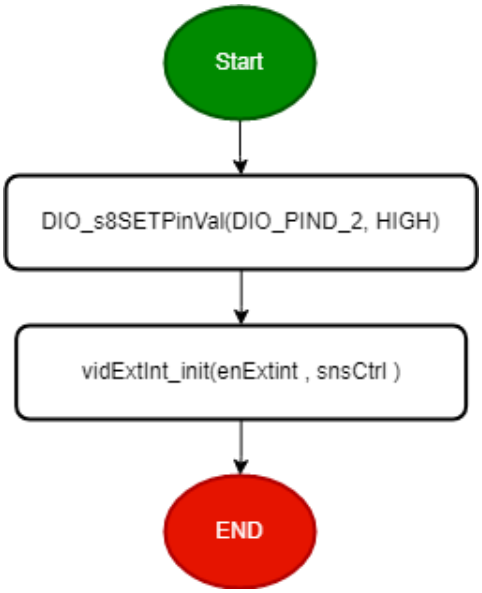
Uint8_t HULTRASONIC_u8Read(void);

Service name	HULTRASONIC_u8Read	
Parameters (in)	void	
Return	Uint8_t	Distance from Ultrasonic sensor
Description	Read distance from ultrasonic sensor	

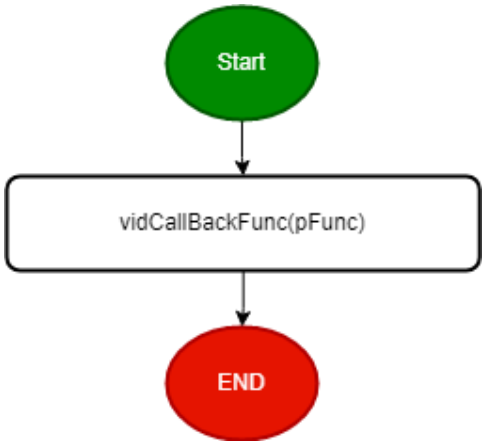
3.3.8 : HEXTINT API :

3.3.8.1 :Flowcharts:

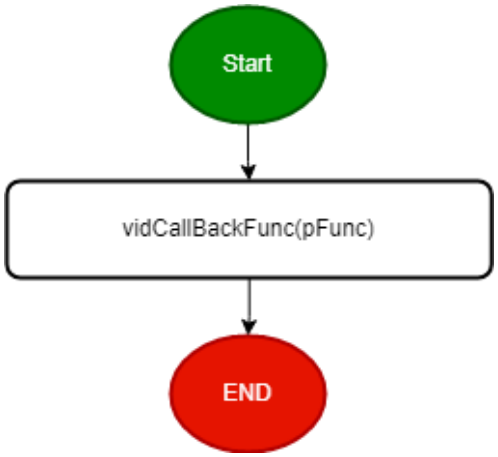
enu_HExtIntError_t HExtInt_enInit (enu_int_type_t enExtint, enu_sns_ctrl_t snsCtrl)



enu_HExtIntError_t HExtInt_enCBF (ptr_func pFunc)



enu_HExtIntError_t HExtInt_enCBFInt1(**ptr_func** pFunc)



3.3.8.2 : Services affecting the hardware unit

enu_HExtIntError_t HExtInt_enInit (enu_int_type_t enExtint, enu_sns_ctrl_t snsCtrl);

Service name	**HExtInt_enInit**	
Parameters (in)	**enu_int_type_t**	Interrupt type [INT0, INT1. INT2]
	enu_sns_ctrl_t	snsCtrl : Sense Control {ANY_LOGICAL, FALL_EDGE, RISE_EDGE}
Return	**enu_HExtIntError_t**	HEXTINT_OK HEXTINT_NOK
Description	External Interrupt Initialization	

enu_HExtIntError_t HExtInt_enCBF (ptr_func pFunc);

Service name	**HExtInt_enCBF**	
Parameters (in)	**ptr_func**	Pointer to function
Return	**enu_HExtIntError_t**	HEXTINT_OK HEXTINT_NOK
Description	Take pointer to function to be executed in ISR when it fires	

enu_HExtIntError_t HExtInt_enCBFInt1(ptr_func pFunc);

Service name	**HExtInt_enCBFInt1**

Parameters (in)	**ptr_func**	Pointer to function
Return	**enu_HExtIntError_t**	HEXTINT_OK HEXTINT_NOK
Description	Take pointer to function to be executed in ISR when it fires for ExtInt_1	

3.3.9 : PWM API :

3.4.1.3 : Services affecting the hardware unit

3.4 : App APIs

3.4.1 : APP API :

3.4.1.3 : Services affecting the hardware unit

- APP_vidStart

Service name	APP_vidStart
Syntax	void APP_vidStart(void);
Description	This Function Start the Application.
Available via	app.h

- APP_vidInit

Service name	APP_vidInit
Syntax	void APP_vidInit(void);
Description	This Function Initialize used Modules
Available via	app.h