Obstacle avoidance car Design

Team 4

|  |
| --- |
| Arafa Arafa |
| Bassel Yasser Mahmoud |
| Mahmoud Sarhan |
| Youssef Ahmed Abbas |

# 

# **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: EXTINT API  
 3.2.2.1: Flowchart  
 3.2.2.2: Services affecting the hardware unit

1. HAL APIs  
   3.3.1 : Timer Manager 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 : keypad API  
    3.3.3.1 : Flowchart  
    3.3.3.2 : Type definitions  
    3.3.3.3 : Services  
   3.3.4 : Car control API  
    3.3.4.1 : Flowchart  
    3.3.4.2 : Type definitions  
    3.3.4.3 : Services  
    3.3.5 : Button API  
    3.3.5.1 : Flowchart  
    3.3.5.2 : Type definitions  
    3.3.5.3 : Services

3.3.7: LCD API  
 3.3.7.1: Flowchart  
 3.3.7.2: Type definitions

3.3.8: Ultrasonic API  
 3.3.8.1: Flowchart  
 3.3.8.2: Services affecting the hardware unit

3.3.9: HEXTINT API  
 3.3.9.1: Flowchart  
 3.3.9.2: Services affecting the hardware unit

1. 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. 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

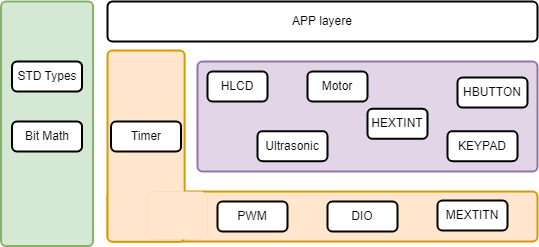
# **2 : Layered architecture**

**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.

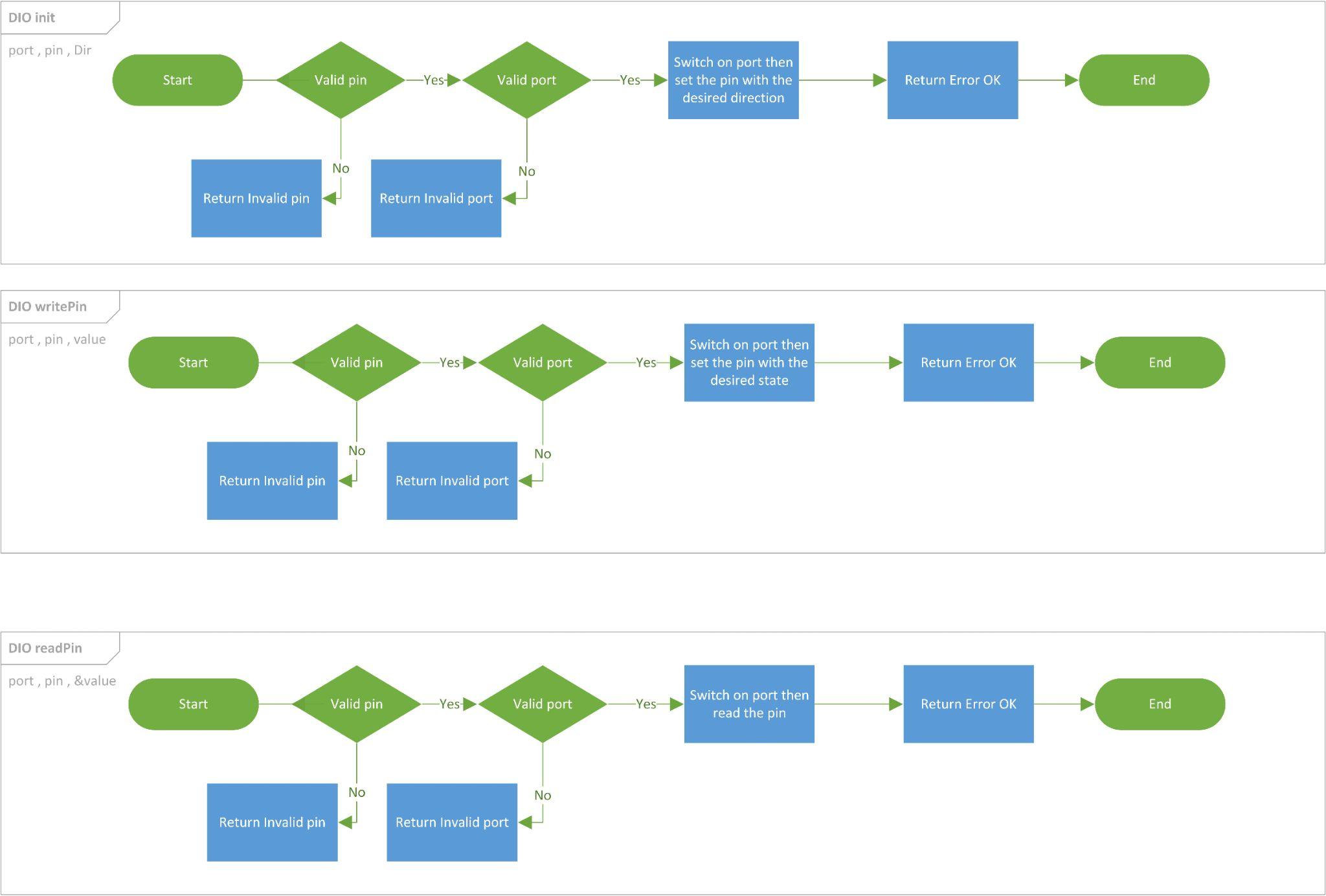
**Common Layer:** is the layer which consists of BIT\_MATH and STD types



## **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

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | | |

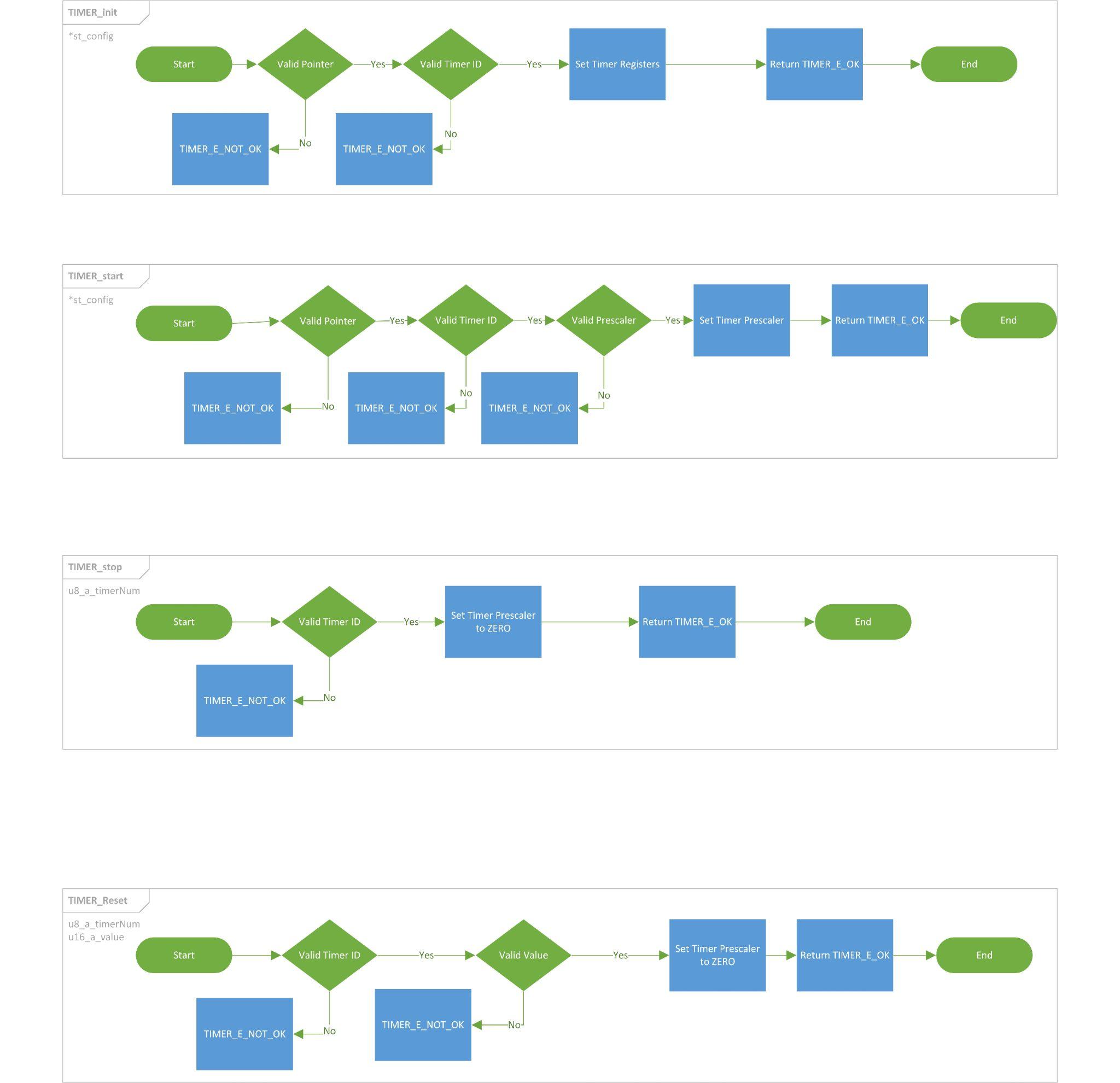
* This function shall return DIO\_InvalidPin if pin number is invalid.
* This function shall return DIO\_InvalidPort if port number is invalid.
* DIO\_init

|  |  |  |  |
| --- | --- | --- | --- |
| 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:

* 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

|  |  |  |
| --- | --- | --- |
| Service name | TIMER\_setCallBack | |
| Syntax | u8\_en\_timerErrorsType TIMER\_setCallBack (  void(\*a\_timerCallBack)(void),  u8\_en\_timerNumberType u8\_a\_timerNum  ); | |
| Parameters (in) | \*a\_timerCallBack | Pointer to the Callback function |
| u8\_a\_timerNum | Timer ID |
| Return | |  |  | | --- | --- | | u8\_en\_timerErrorsType | TIMER\_E\_OK | | TIMER\_E\_NOT\_OK | | |
| Description | This Function reset the TIMER | |

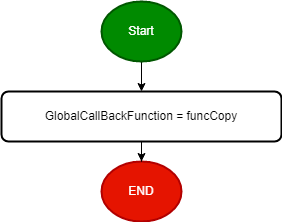
* This function shall return TIMER\_E\_NOK if a\_timerCallBack is NULL
* This function shall return TIMER\_E\_NOK if u8\_a\_timerNum is invalid.

### 3.2.3: ExtInt API:

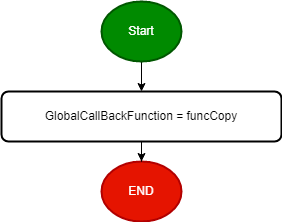
#### 3.2.3.1 :Flowcharts:

#### **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** | |
| 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 for Int 1 | |

## **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 | |

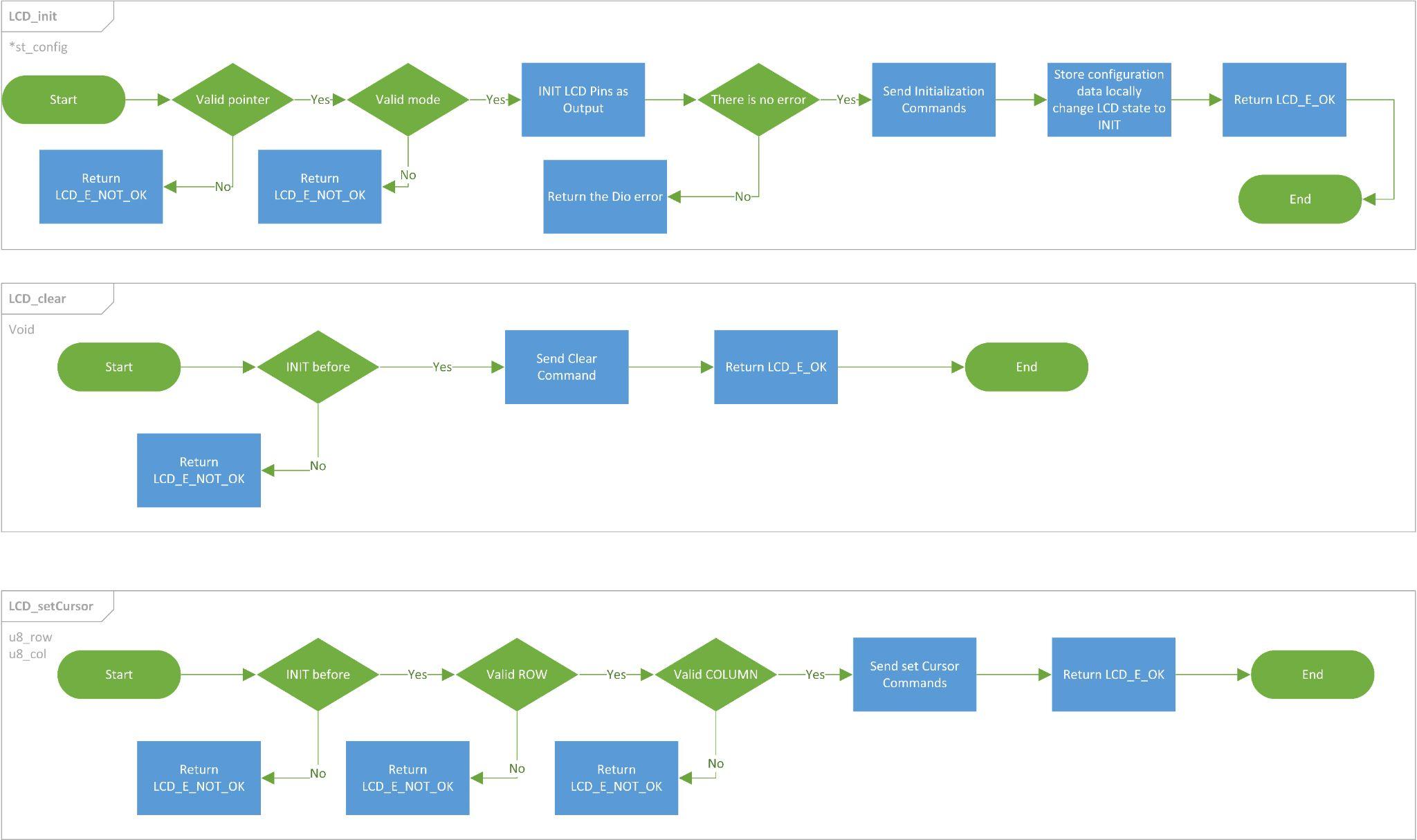
* This function shall return TIMER\_E\_NOK if u8\_en\_timerNum is invalid
* TIMER\_Manager\_reset

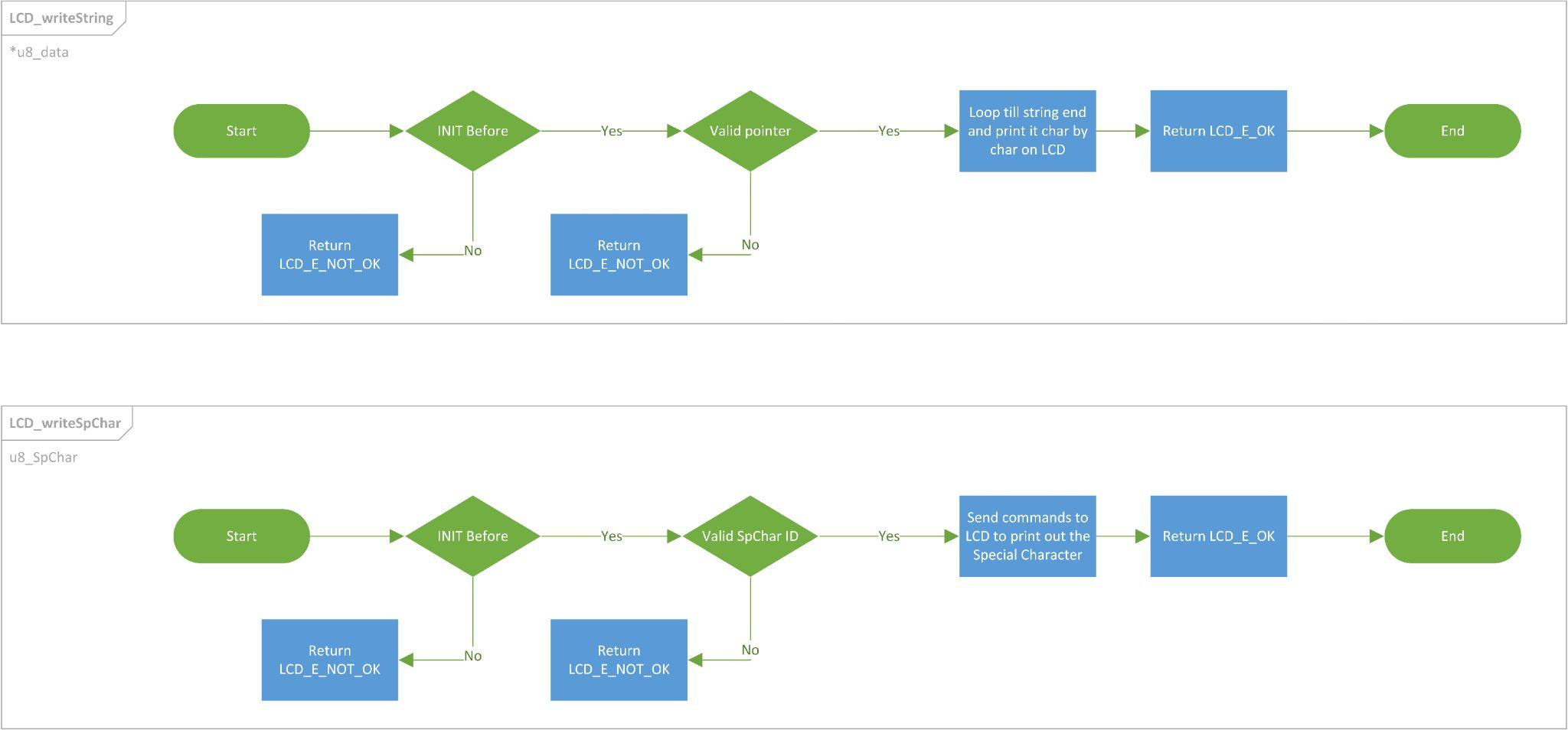
|  |  |  |
| --- | --- | --- |
| Service name | TIMER\_Manager\_reset | |
| Syntax | u8\_en\_timerErrorsType TIMER\_Manager\_reset (  st\_timerConfigType\* st\_config  ); | |
| Parameters (in) | st\_config | Pointer to the configuration structure |
| Return | |  |  | | --- | --- | | u8\_en\_timerErrorsType | 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:

* st\_lcdConfigType

|  |  |
| --- | --- |
| Name | st\_lcdConfigType |
| Type | Structure |
| Range | Shall contain required LCD configuration |
| Description | st\_lcdConfigType |
| Available via | lcd.h |

* u8\_en\_lcdErrorsType

|  |  |
| --- | --- |
| Name | u8\_en\_lcdErrorsType |
| Type | Enumeration |
| Range | |  |  |  | | --- | --- | --- | | LCD\_E\_OK | 0x00 | LCD error OK | | LCD\_E\_NOT\_OK | 0x05 | LCD error | |
| Description | u8\_en\_lcdErrorsType |
| Available via | lcd.h |

* u8\_en\_lcdModeType

|  |  |
| --- | --- |
| Name | u8\_en\_lcdModeType |
| Type | Enumeration |
| Range | |  |  |  | | --- | --- | --- | | LCD\_4\_BIT\_MODE | 0x00 | LCD 4-bit mode | | LCD\_8\_BIT\_MODE | 0x01 | LCD 8-bit mode | | LCD\_INVALID\_MODE | 0X02 | LCD invalid mode | |
| Description | u8\_en\_lcdModeType |
| Available via | lcd.h |

* u8\_en\_lcdSpCharType

|  |  |
| --- | --- |
| Name | u8\_en\_lcdSpCharType |
| Type | Enumeration |
| Range | Shall contain all special characters IDs |
| Description | u8\_en\_lcdSpCharType |
| Available via | lcd.h |

#### 3.3.2.3 : Services affecting the hardware unit

* LCD\_init

|  |  |  |
| --- | --- | --- |
| Service name | LCD\_init | |
| Syntax | u8\_en\_lcdErrorsType LCD\_init (  st\_lcdConfigType\* st\_config  ); | |
| Parameters (in) | st\_config | Pointer to the configuration structure |
| Return | |  |  | | --- | --- | | u8\_en\_lcdErrorsType | LCD\_E\_OK | | LCD\_E\_NOT\_OK | | |
| Description | This Function Initialize LCD module | |

* This function shall return LCD\_E\_NOK if st\_config is NULL
* This function shall return LCD\_E\_NOK if any of the configuration elements is invalid.
* LCD\_clear

|  |  |
| --- | --- |
| Service name | LCD\_clear |
| Syntax | u8\_en\_lcdErrorsType LCD\_clear (  void  ); |
| Parameters (in) | None |
| Return | |  |  | | --- | --- | | u8\_en\_lcdErrorsType | LCD\_E\_OK | | LCD\_E\_NOT\_OK | |
| Description | This Function Clear LCD |

* LCD\_setCursor

|  |  |  |
| --- | --- | --- |
| Service name | LCD\_setCursor | |
| Syntax | u8\_en\_lcdErrorsType LCD\_setCursor (  uint8\_t u8\_row,  uint8\_t u8\_col  ); | |
| Parameters (in) | u8\_row | The desired row to set cursor |
| u8\_col | The desired column to set cursor |
| Return | |  |  | | --- | --- | | u8\_en\_lcdErrorsType | LCD\_E\_OK | | LCD\_E\_NOT\_OK | | |
| Description | This Function sets the cursor location on LCD | |

* LCD\_writeString

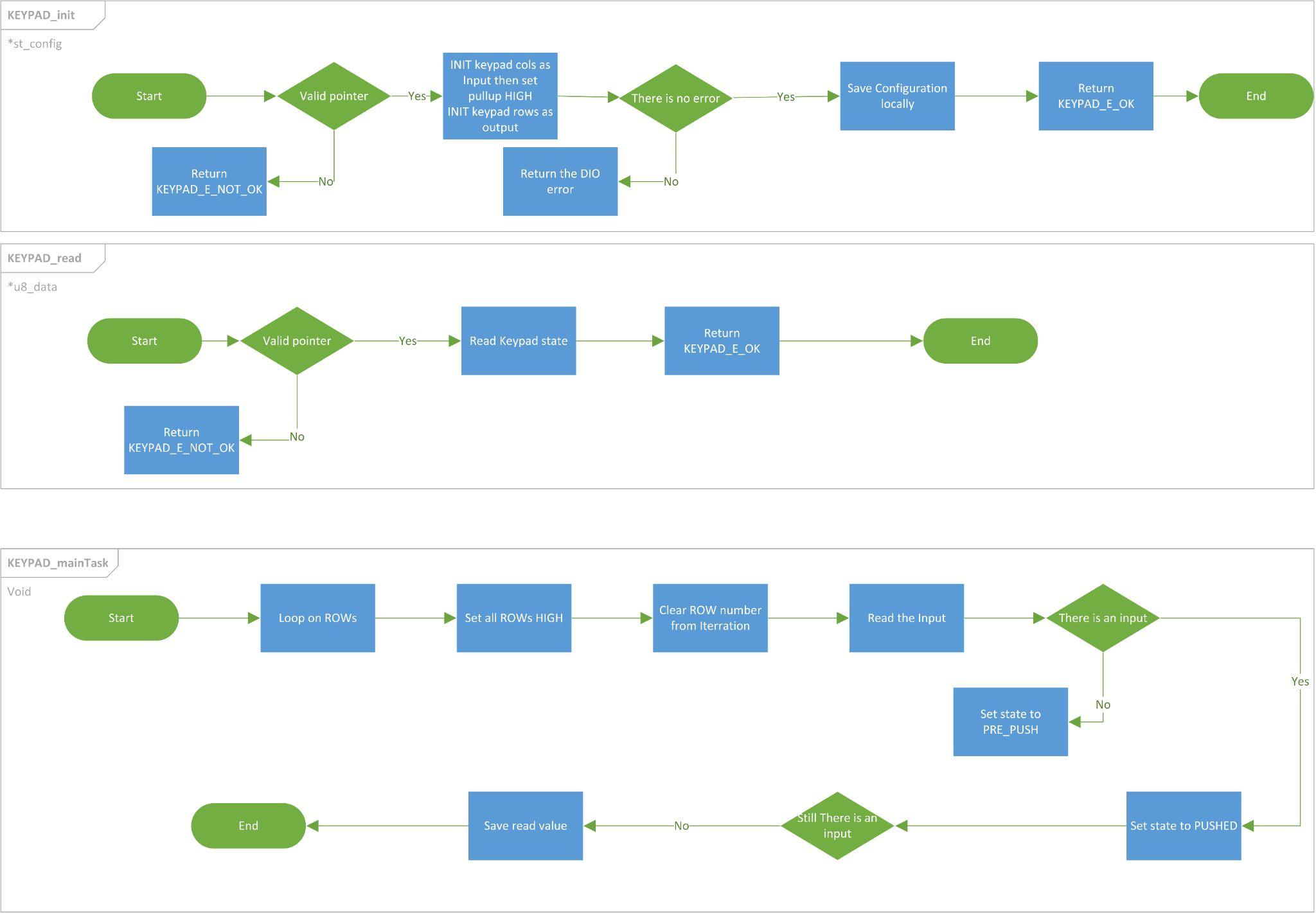
|  |  |  |
| --- | --- | --- |
| Service name | LCD\_writeString | |
| Syntax | u8\_en\_lcdErrorsType LCD\_writeString (  uint8\_t\* u8\_data  ); | |
| Parameters (in) | u8\_data | Pointer to string to it print on screen |
| Return | |  |  | | --- | --- | | u8\_en\_lcdErrorsType | LCD\_E\_OK | | LCD\_E\_NOT\_OK | | |
| Description | This Function write a string on LCD | |

* LCD\_writeSpChar

|  |  |  |
| --- | --- | --- |
| Service name | LCD\_writeSpChar | |
| Syntax | u8\_en\_lcdErrorsType LCD\_writeSpChar (  u8\_en\_lcdSpCharType u8\_SpChar  ); | |
| Parameters (in) | u8\_SpChar | Special character ID to it print on screen |
| Return | |  |  | | --- | --- | | u8\_en\_lcdErrorsType | LCD\_E\_OK | | LCD\_E\_NOT\_OK | | |
| Description | This Function write a special character on LCD | |

### 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

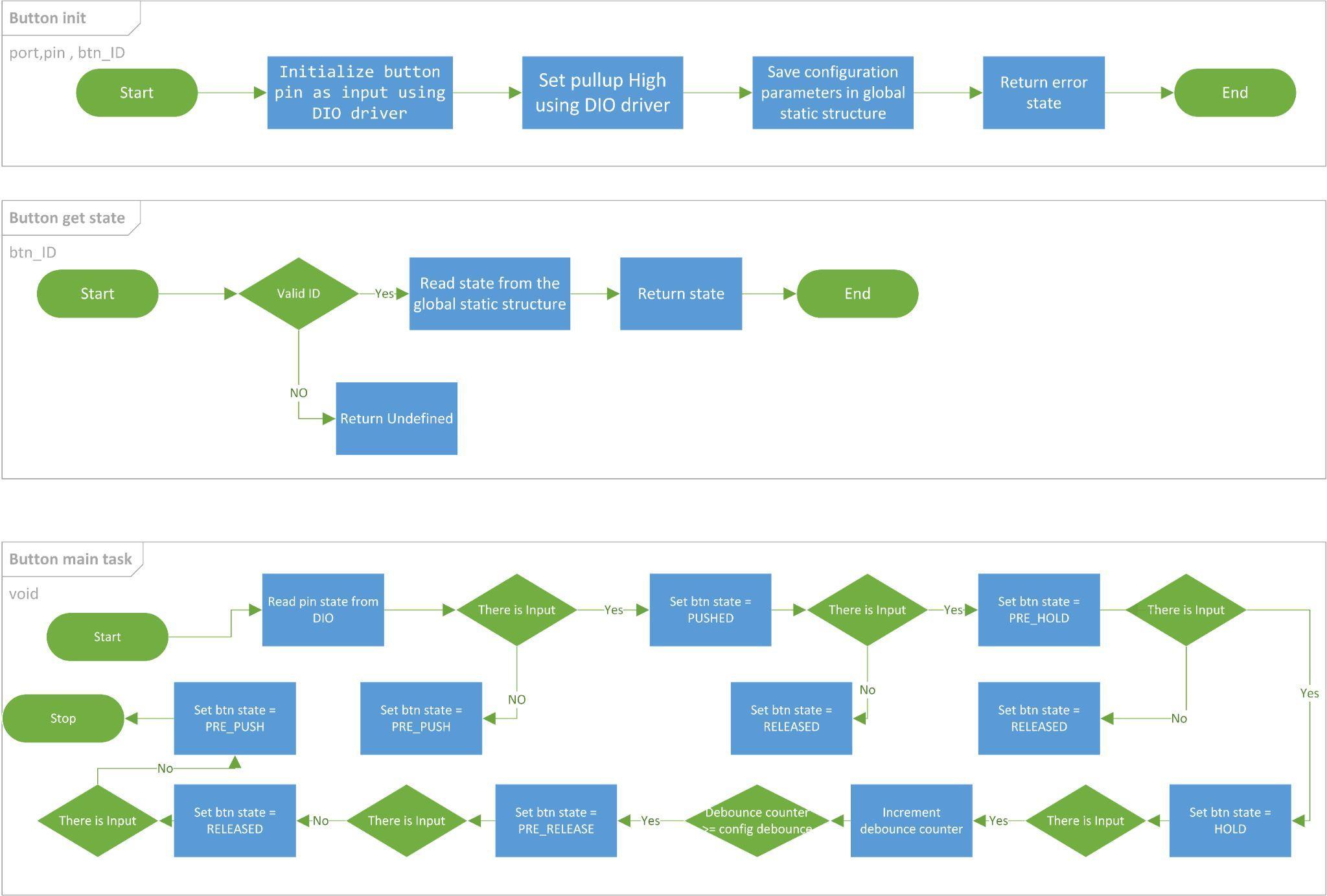
|  |  |  |
| --- | --- | --- |
| Service name | CAR\_turnRight | |
| Syntax | u8\_en\_carControlErrorsType CAR\_turnRight(  st\_carControlConfigType\* st\_config\_R,  st\_carControlConfigType\* st\_config\_L  ); | |
| Parameters (in) | st\_config | Pointer to the configuration structure |
| Return | |  |  | | --- | --- | | 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 | |

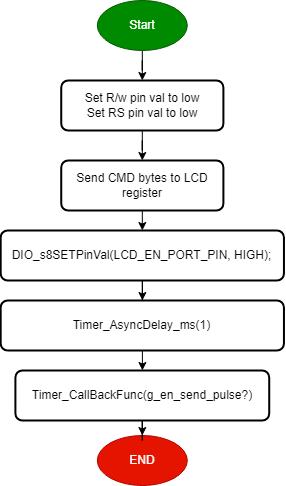
### 3.3.6 : LCD API :

#### 3.3.6.1 :Flowcharts:

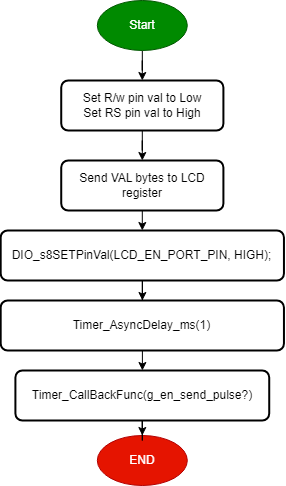
#### lcdErrorsType **HLCD\_vidInit(void)**

****

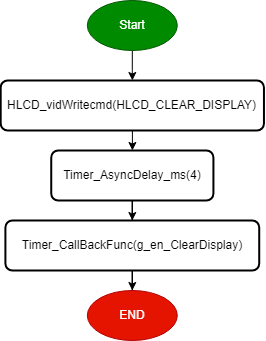
#### lcdErrorsType **HLCD \_vidWritecmd (Uint8\_t u8commandCopy)**



#### lcdErrorsType **HLCD\_vidWriteChar (Uint8\_t u8CharCopy)**



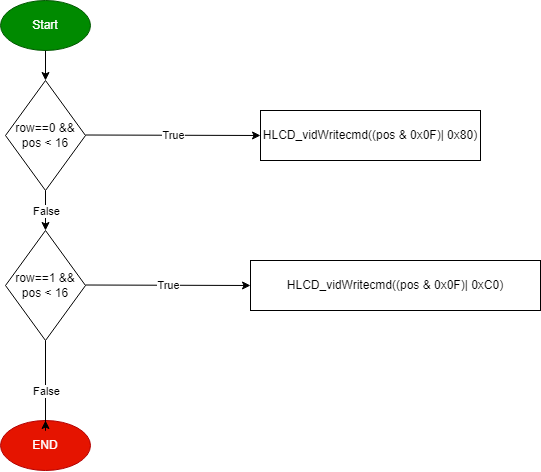
#### lcdErrorsType **HLCD\_ClrDisplay(void)**



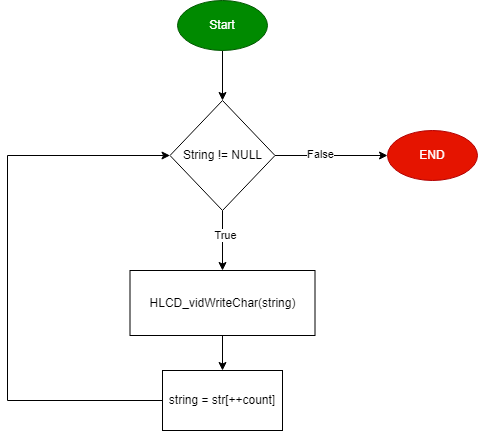
#### lcdErrorsType **HLCD\_ShiftLeft(void)**



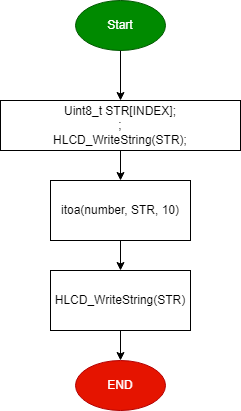
#### lcdErrorsType **HLCD\_gotoXY (Uint8\_t row, Uint8\_t pos)**

****

#### 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 | |

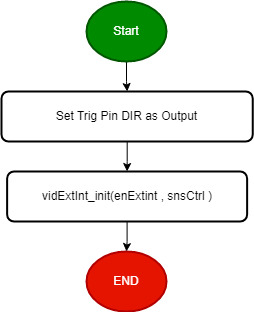
u8\_en\_lcdErrorsType HLCD\_WriteInt(Uint32\_t number);

|  |  |  |
| --- | --- | --- |
| 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 | |

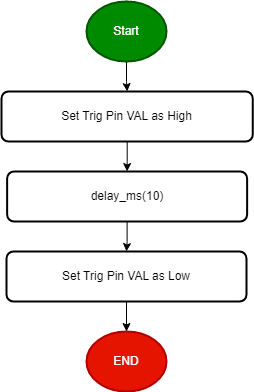
### 3.3.7 : Ultrasonic API :

#### 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 | v  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  void |
| Description | Sending pulse |

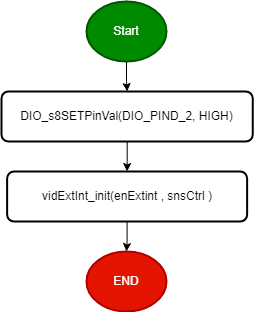
**Uint8\_t HULTRASONIC\_u8Read(void);**

|  |  |
| --- | --- |
| Service name | **HULTRASONIC\_u8Read** |
| Parameters (in) | **void** |
| Return | |  |  | | --- | --- | | **Uint8\_t** | Distance from Ultrasonic sensor | |
| Description | Read distance from ultrasonic sensor |

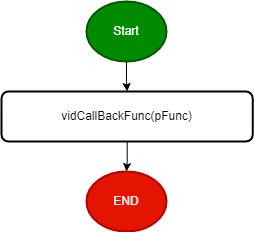
### 3.3.8 : HEXTINT API :

#### 3.3.8.1 :Flowcharts:

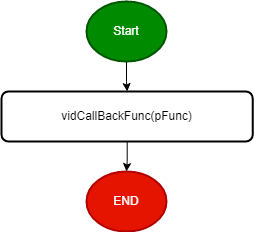
#### **enu\_HExtIntError\_t HExtInt\_enInit (enu\_int\_type\_t enExtint, enu\_sns\_ctrl\_t snsCtrl)**

****

#### **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 | |

## **3.4 : 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 |