Small OS Design

By Mahmoud Sarhan

Github Repo : https://github.com/MahmoudMatarawy/small-os

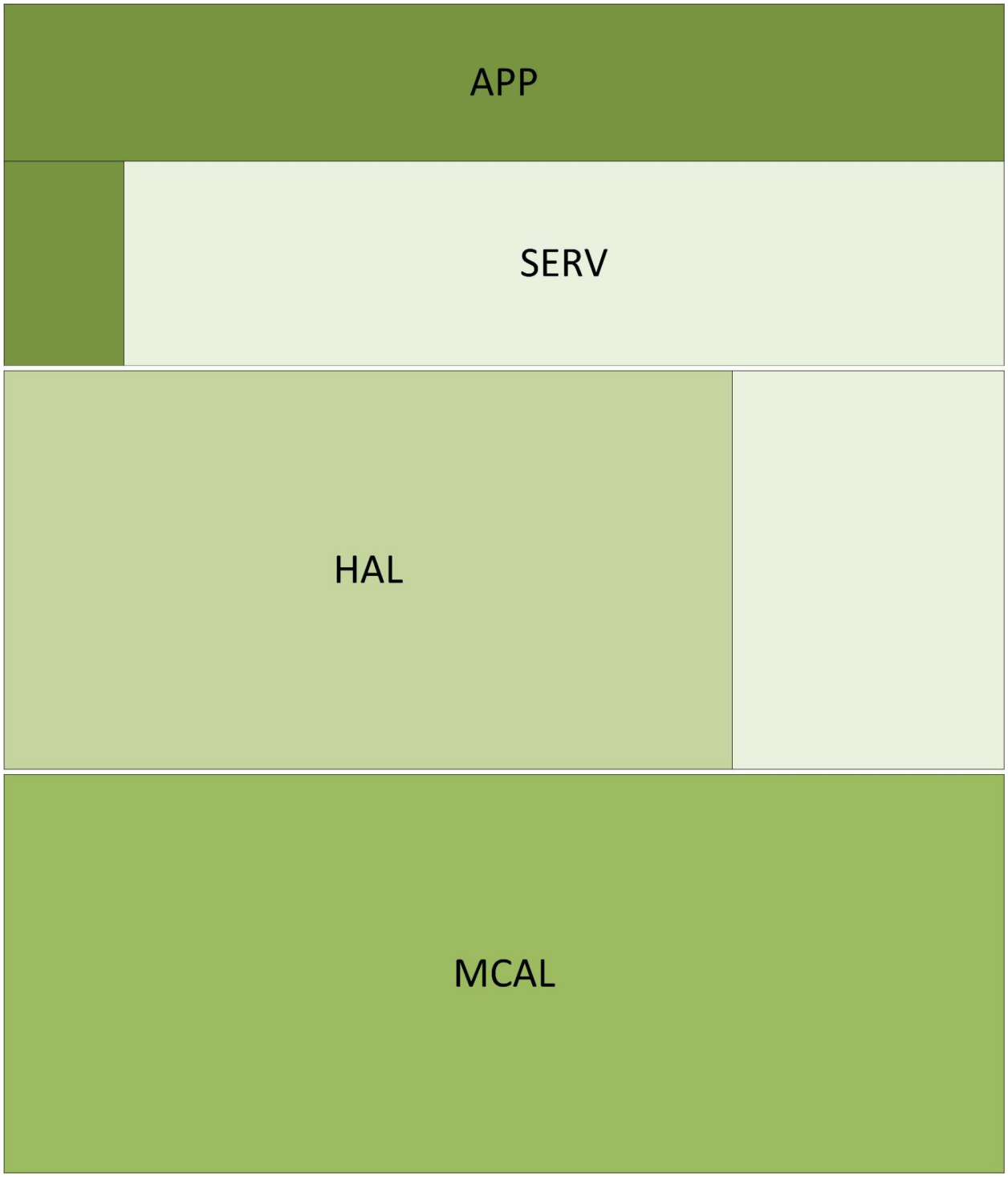
Table of content:

4. SERV APIs
    3.4.1 : SOS API
        3.4.1.1 : Flowchart
        3.4.1.2: Type definitions
        3.4.1.3: Services
5. APP APIs
    3.5.1 : APP API
        3.4.1.1 : Flowchart
        3.4.1.2: Type definitions
        3.4.1.3: Services

1 : Detailed Requirements

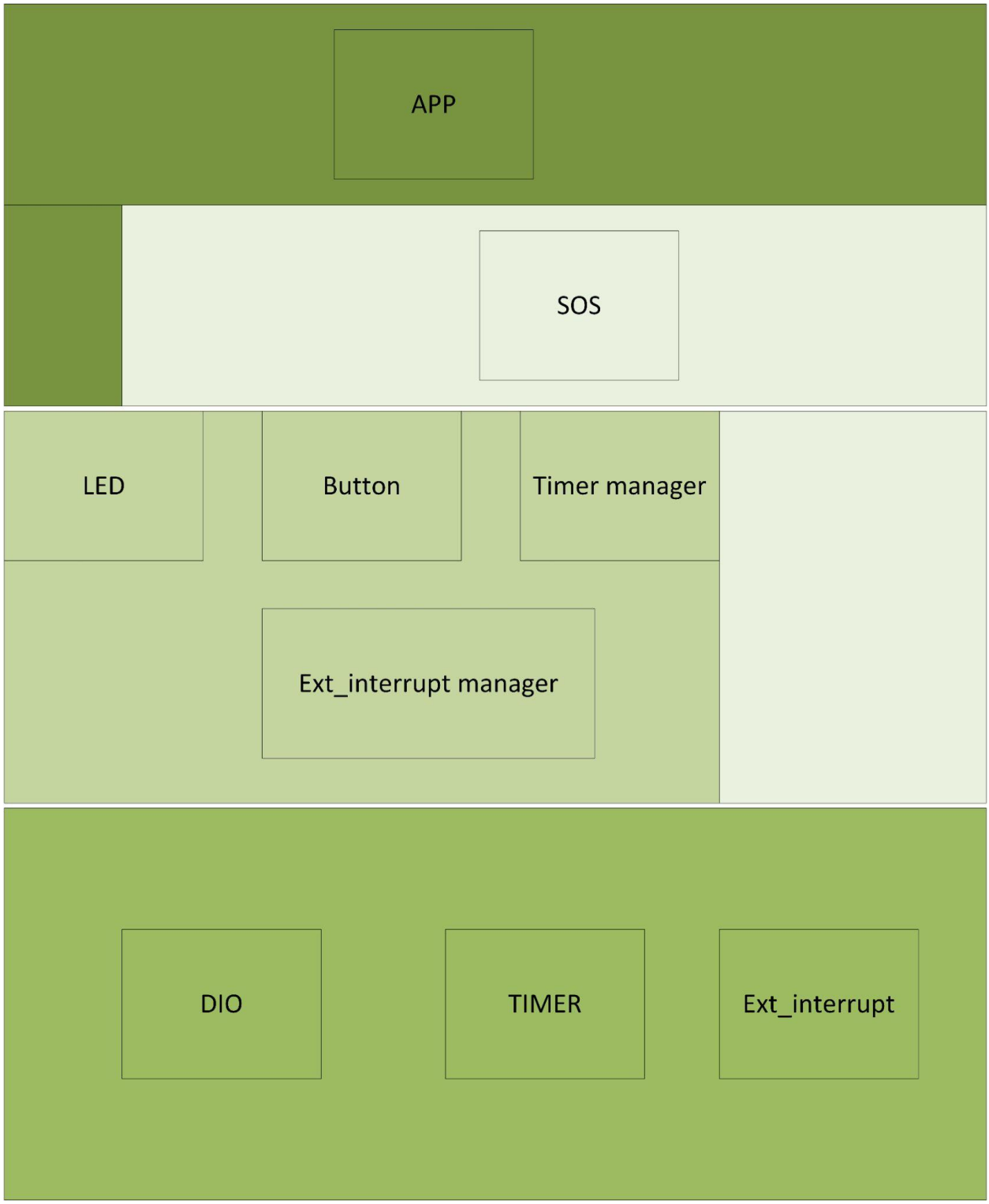
1. Implement an application that calls the SOS module and use 2 tasks
1. Task 1: Toggle LED_0 (Every 300 Milliseconds)
2. Task 2: Toggle LED_1 (Every 500 Milliseconds)
2. Make sure that these tasks occur periodically and forever
3. When pressing PBUTTON0, the SOS will stop
4. When Pressing PBUTTON1, the SOS will run

2 : Layered architecture



3 : System modules

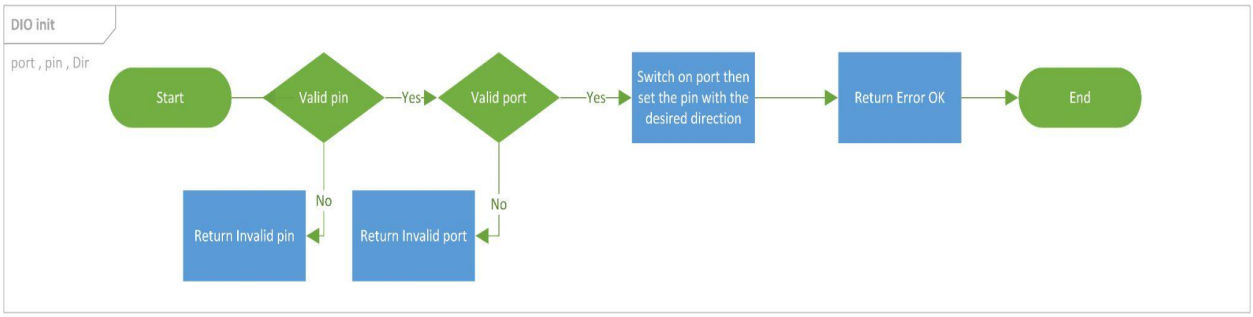
3.1: Module architecture

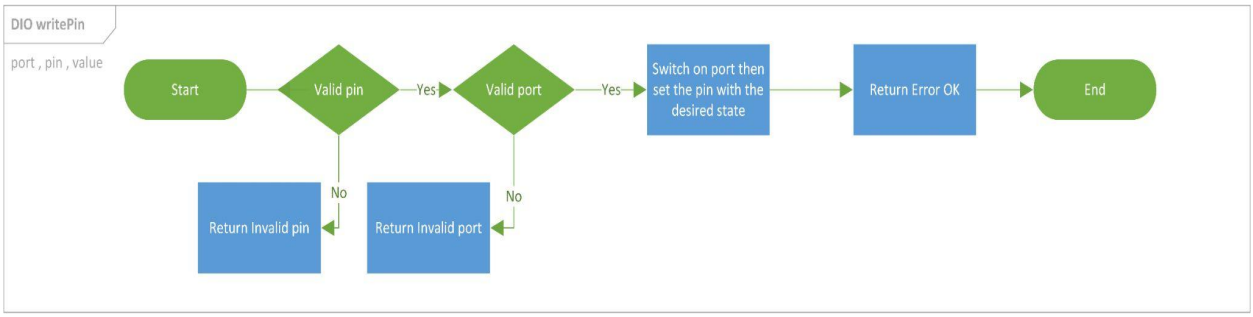


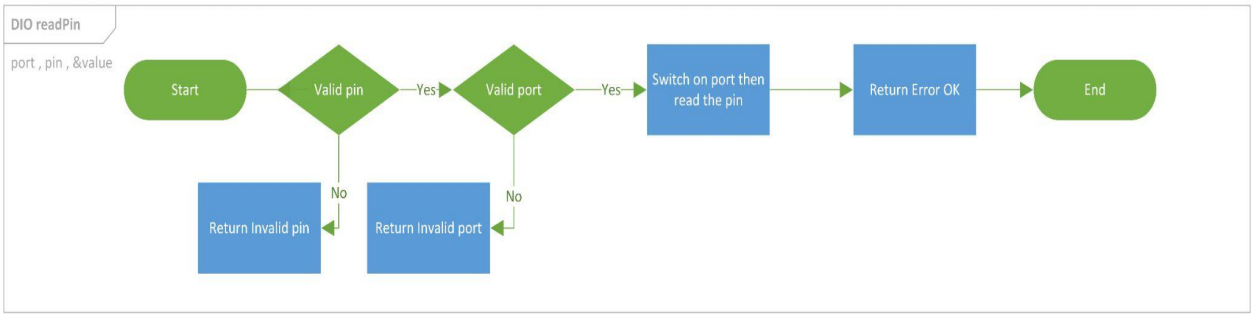
3.2: MCAL APIs

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

<table>
<tr><td></td><td>value</td><td>Pointer to store the level</td><td>STD_HIGH</td></tr>
<tr><td></td><td></td><td></td><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></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 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>Parameters (in)</td><td>Port, pin</td><td colspan="2">Channel ID</td></tr>
<tr><td></td><td>state</td><td>Value to be set</td><td>STD_HIGH</td></tr>
<tr><td></td><td></td><td></td><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></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 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

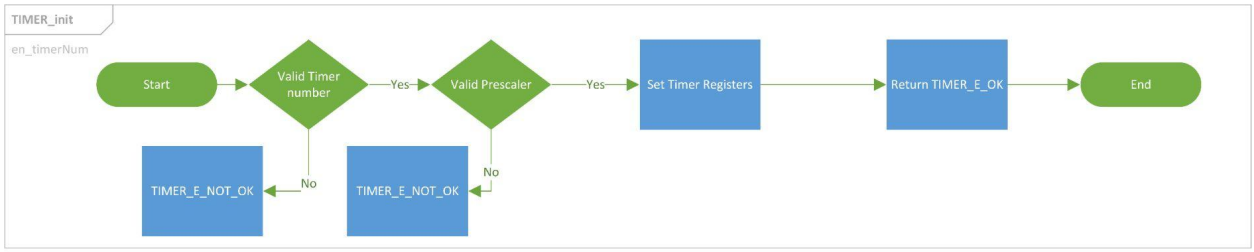
Service name	DIO_init

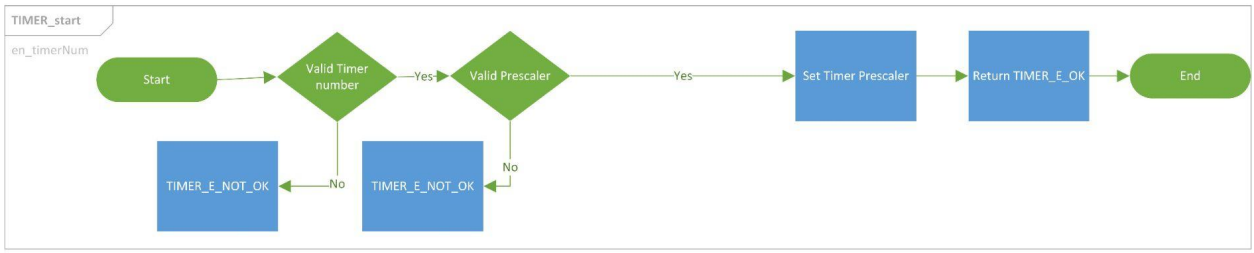
<table>
<tr><td>Syntax</td><td colspan="3">u8_en_dioErrors DIO_init (
en_dioPortsType port,
en_dioPinsType pin,
u8_en_dioDirType direction
);</td></tr>
<tr><td rowspan="3">Parameters (in)</td><td>Port, pin</td><td colspan="2">Channel ID</td></tr>
<tr><td rowspan="2">direction</td><td rowspan="2">Value 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 rowspan="3">DIO_Errors</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 sets the Direction of the pin</td></tr>
</table>

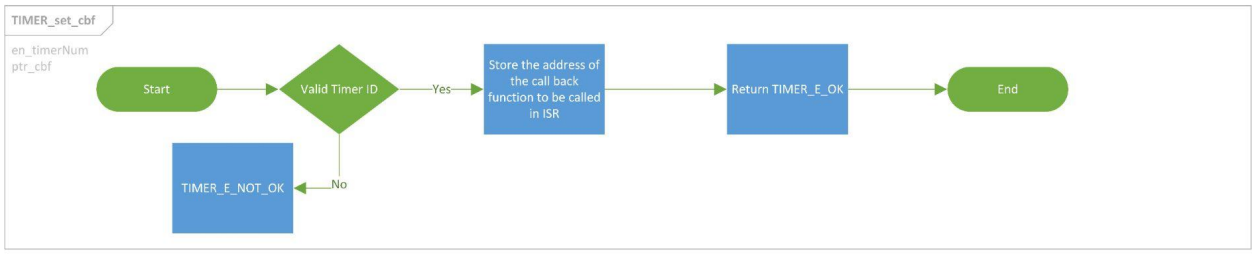
- This function shall return DIO_InvalidPin if pin number is invalid
- This function shall return DIO_InvalidPort if port number is invalid.

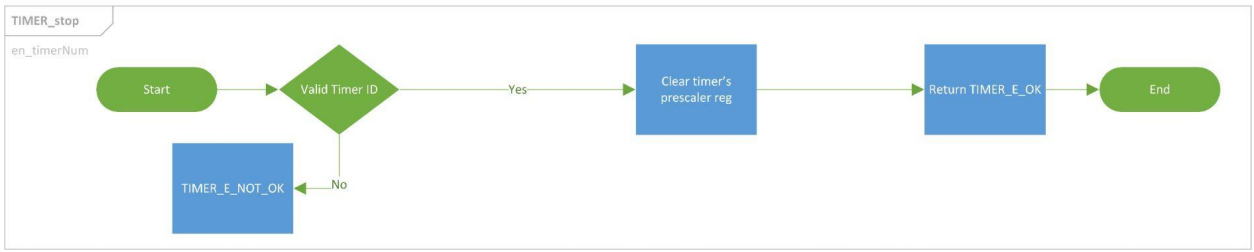
3.2.2: TIMER API:

3.2.2.1 :Flowcharts:









3.2.2.2 : Type definitions:

- st_timer_config_t

Name	st_timer_config_t
Type	Structure
Range	Shall contain required Timers configuration
Description	st_timer_config_t
Available via	timer_cfg.h

- u8_timerErrors_t

Name	u8_timerErrors_t		
Type	Enumeration		
Range	TIMER_E_OK	0x00	Timer error OK
	TIMER_E_NOT_OK	0x03	Timer error
Description	u8_timerErrors_t		
Available via	timer_types.h		

- en_timer_num_t

Name	en_timer_num_t
Type	Enumeration
Range	Shall contain all timers IDs
Description	en_timer_num_t
Available via	timer_types.h

- en_timer_clock_t

Name	en_timer_clock_t

Type	Enumeration
Range	Shall contain all timers prescalers
Description	en_timer_clock_t
Available via	timer_types.h

- en_timer_interrupt_feature_t

Name	en_timer_interrupt_feature_t
Type	Enumeration
Range	Shall contain enable and disable interrupt feature
Description	en_timer_interrupt_feature_t
Available via	timer_types.h

3.2.2.3 : Services affecting the hardware unit

- TIMER_init

<table>
<tr><td>Service name</td><td colspan="2">TIMER_init</td></tr>
<tr><td>Syntax</td><td colspan="2">u8_timerErrors_t TIMER_init(
en_timer_num_t en_timerNum
);</td></tr>
<tr><td>Parameters (in)</td><td>en_timerNum</td><td>Timer number</td></tr>
<tr><td>Return</td><td colspan="2"><table><tr><td rowspan="2">u8_timerErrors_t</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 Initialize timer module</td></tr>
</table>

- TIMER_start

Service name	TIMER_start	
Syntax	u8_timerErrors_t TIMER_start(en_timer_num_t en_timerNum);	
Parameters (in)	en_timerNum	Timer number
Return	u8_timerErrors_t	TIMER_E_OK TIMER_E_NOT_OK
Description	This Function starts the timer	

- TIMER_stop

Service name	TIMER_stop	
Syntax	u8_timerErrors_t TIMER_stop(en_timer_num_t en_timerNum);	
Parameters (in)	en_timerNum	Timer number
Return	u8_timerErrors_t	TIMER_E_OK TIMER_E_NOT_OK
Description	This Function stops the timer	

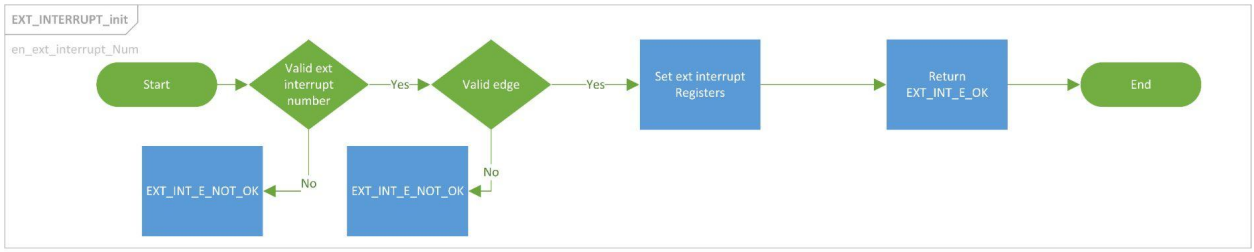
- TIMER_set_cbf

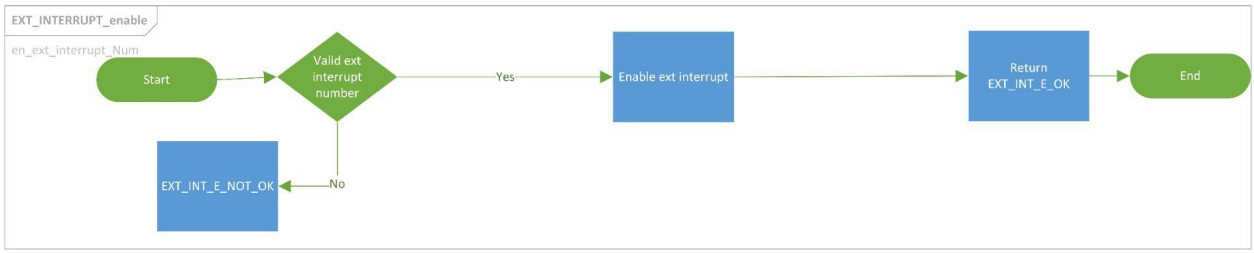
Service name	TIMER_set_cbf	
Syntax	u8_timerErrors_t TIMER_set_cbf(en_timer_num_t en_timerNum, timerCallBack callBackFunction_ptr);	
Parameters (in)	en_timerNum	Timer number
	callBackFunction_ptr	Pointer to the call back function

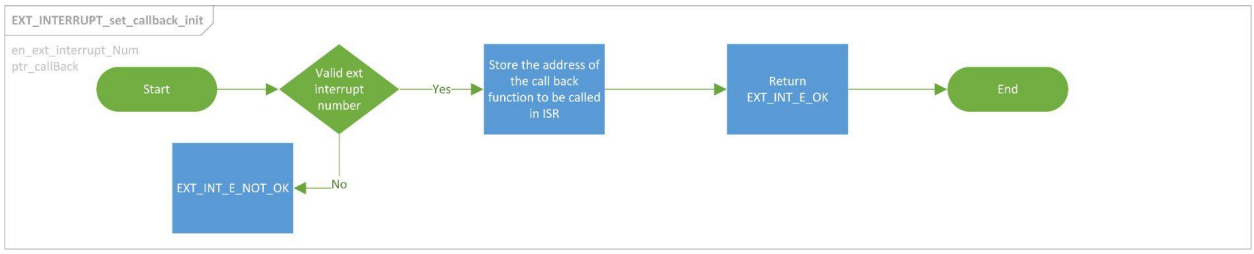
<table>
<tr><td>Return</td><td><table>
<tr><td rowspan="2">u8_timerErrors_t</td><td>TIMER_E_OK</td></tr>
<tr><td>TIMER_E_NOT_OK</td></tr>
</table></td></tr>
<tr><td>Description</td><td>This Function starts the timer</td></tr>
</table>

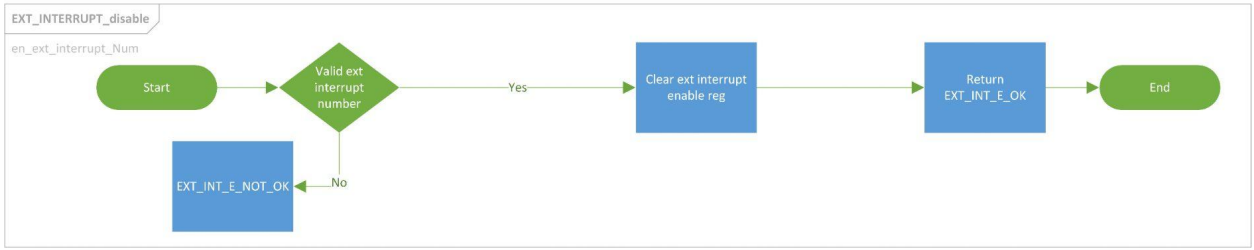
3.2.2: External interrupt API:

3.2.2.1 :Flowcharts:









3.2.2.2 : Type definitions:

- u8_interruptError_t

Name	u8_interruptError_t		
Type	Enumeration		
Range	EXT_INT_E_OK	0x00	Ext Interrupt error OK
	EXT_INT_E_NOK	0x04	Ext Interrupt error
Description	u8_interruptError_t		
Available via	ext_interrupt_types.h		

- en_ext_interrupt_num_t

Name	en_ext_interrupt_num_t
Type	Enumeration
Range	Shall contain all external interrupts IDs
Description	en_ext_interrupt_num_t
Available via	ext_interrupt_types.h

- en_edge_detection_t

Name	en_edge_detection_t
Type	Enumeration
Range	Shall contain all external interrupts edge detection cases
Description	en_edge_detection_t
Available via	ext_interrupt_types.h

3.2.2.3 : Services affecting the hardware unit

- EXT_INTERRUPT_init

Service name	EXT_INTERRUPT_init	
Syntax	u8_interruptError_t EXT_INTERRUPT_init(en_ext_interrupt_num_t en_ext_interrupt_num);	
Parameters (in)	en_timerNum	Ext interrupt number
Return	u8_interruptError_t	EXT_INT_E_OK EXT_INT_E_NOK
Description	This Function Initialize external interrupt module	

- EXT_INTERRUPT_enable

Service name	EXT_INTERRUPT_enable	
Syntax	u8_interruptError_t EXT_INTERRUPT_enable(en_ext_interrupt_num_t en_ext_interrupt_num);	
Parameters (in)	en_timerNum	Ext interrupt number
Return	u8_interruptError_t	EXT_INT_E_OK EXT_INT_E_NOK
Description	This Function enables external interrupt	

- EXT_INTERRUPT_disable

Service name	EXT_INTERRUPT_disable	
Syntax	u8_interruptError_t EXT_INTERRUPT_disable(en_ext_interrupt_num_t en_ext_interrupt_num);	
Parameters (in)	en_timerNum	Ext interrupt number
Return	u8_interruptError_t	EXT_INT_E_OK EXT_INT_E_NOK

Description	This Function disables external interrupt

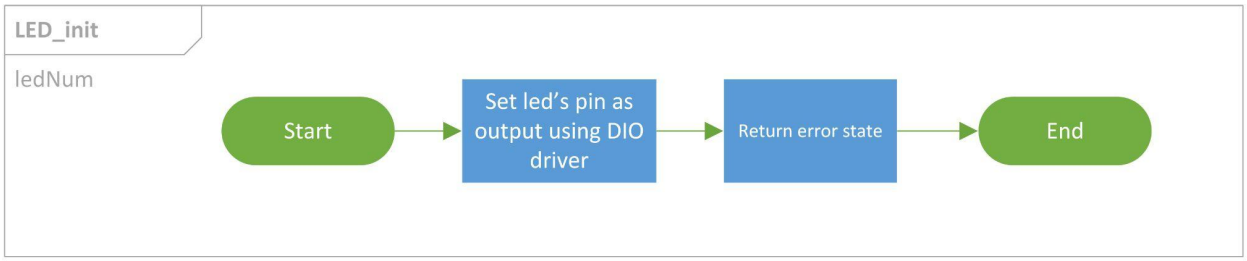
- EXT_INTERRUPT_set_callback_init

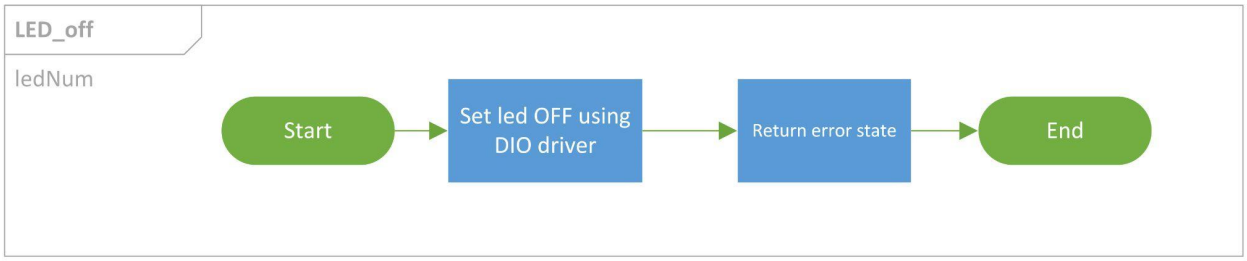
Service name	EXT_INTERRUPT_set_callback_init	
Syntax	u8_interruptError_t EXT_INTERRUPT_set_callback_init(en_ext_interrupt_num_t en_ext_interrupt_num, void(*callback)(void));	
Parameters (in)	en_timerNum	Ext interrupt number
	callback	Pointer to the call back function
Return	u8_interruptError_t	EXT_INT_E_OK EXT_INT_E_NOK
Description	This Function saves call back pointer to call it in ISR	

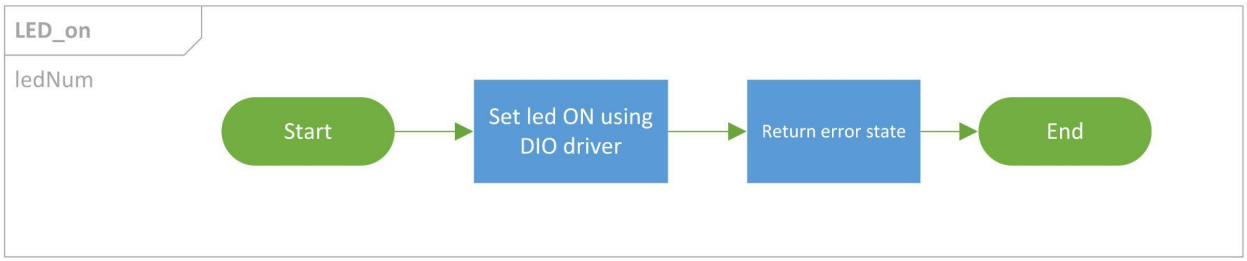
3.3: HAL APIs

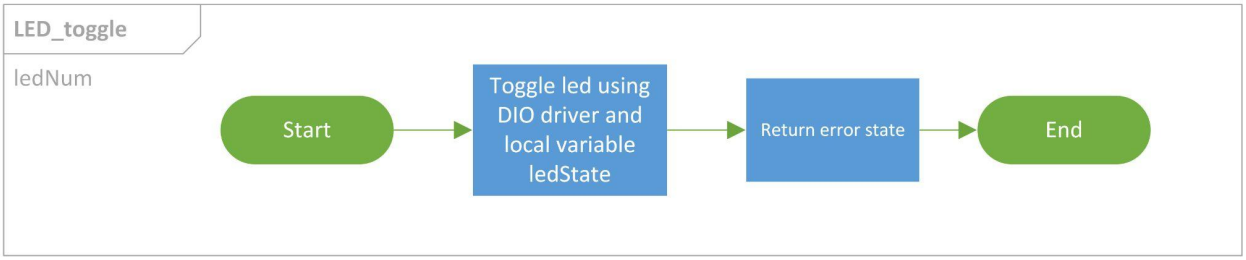
3.3.1: LED API:

3.3.1.1 :Flowcharts:









3.3.1.2 : Type definitions:

- st_ledConfig_t

Name	st_ledConfig_t
Type	Structure
Range	Shall contain required LED configuration
Description	st_ledConfig_t
Available via	led_cfg.h

- u8_ledError_t

Name	u8_ledError_t		
Type	Enumeration		
Range	LED_ERROR_OK	0x00	LED error OK
	LED_ERROR_NOT_OK	0x05	LED error
Description	u8_ledError_t		
Available via	led.h		

- en_ledNum_t

Name	en_ledNum_t		
Type	Enumeration		
Range	LED_0	0x00	LED_0
	LED_1	0x01	LED_1
Description	en_ledNum_t		
Available via	led.h		

3.3.1.3 : Services affecting the hardware unit

- LED_init

Service name	LED_init	
Syntax	u8_ledError_t LED_init(en_ledNum_t ledNum);	
Parameters (in)	ledNum	Led number
Return	u8_ledError_t	LED_ERROR_OK LED_ERROR_NOT_OK
Description	This Function Initialize LED module	

- LED_on

Service name	LED_on	
Syntax	u8_ledError_t LED_on(en_ledNum_t ledNum);	
Parameters (in)	ledNum	Led number
Return	u8_ledError_t	LED_ERROR_OK LED_ERROR_NOT_OK
Description	This Function turn on LED	

- LED_off

Service name	LED_off	
Syntax	u8_ledError_t LED_off(en_ledNum_t ledNum);	
Parameters (in)	ledNum	Led number
Return	u8_ledError_t	LED_ERROR_OK LED_ERROR_NOT_OK

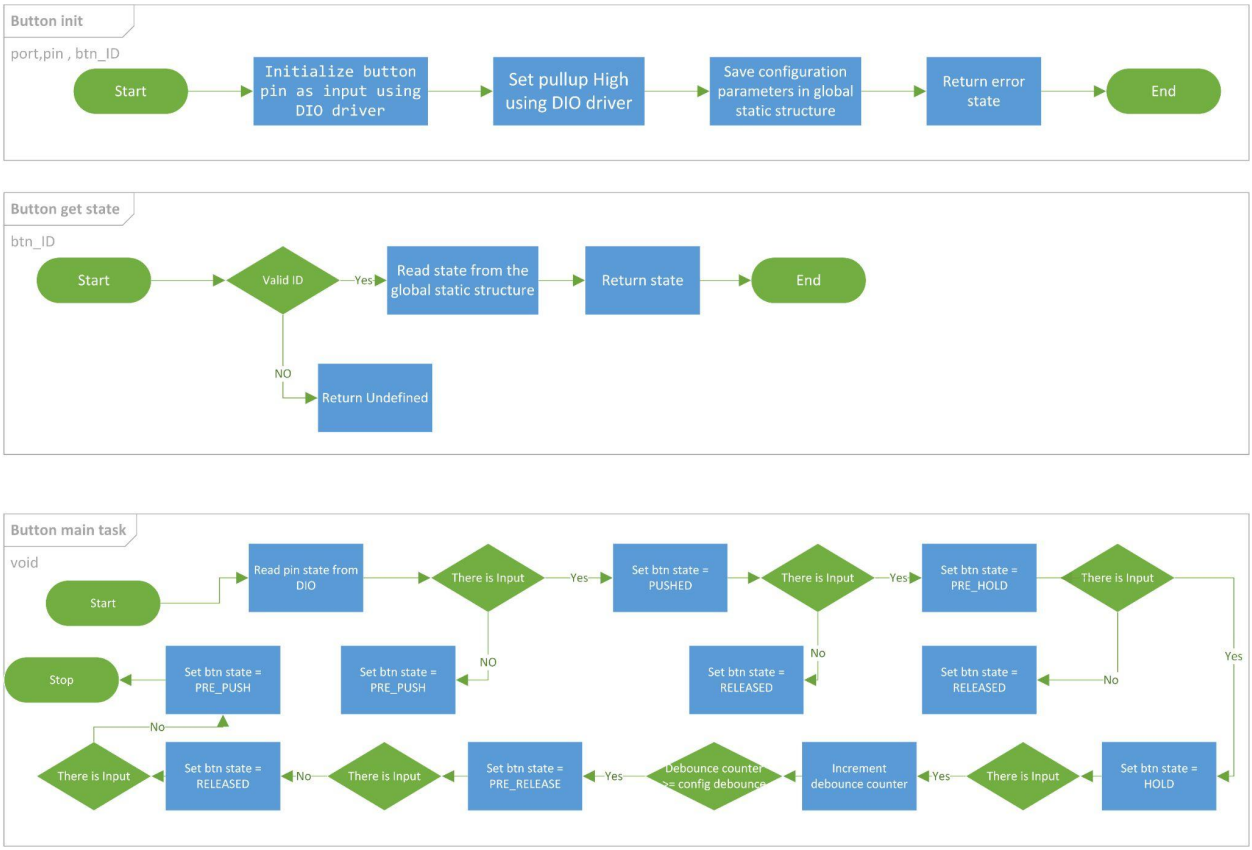
Description	This Function turn off LED

- LED_toggle

Service name	LED_toggle	
Syntax	u8_ledError_t LED_toggle(en_ledNum_t ledNum);	
Parameters (in)	ledNum	Led number
Return	u8_ledError_t	LED_ERROR_OK LED_ERROR_NOT_OK
Description	This Function toggles LED	

3.3.2: Button API:

3.3.2.1 :Flowcharts:



3.3.2.2 : Type definitions:

- st_btnConfig_t

Name	st_btnConfig_t
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

- en_btnLevel_t

Name	en_btnLevel_t
Type	Enumeration

Range	BT_PUSH_LEVEL | 0x00 | Push Level BT_RELEASE_LEVEL | 0x01 | Release Level
Description	Button Level Enum
Available via	button_types.h

- en_btnState_t

Name	en_btnState_t
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

- en_btnId_t

Name	en_btnId_t
Type	Enumeration
Range	Button_Start | 0x00 | Start Button
Description	Button ID Enum
Available via	button_types.h

3.3.2.3 : Services affecting the hardware unit

- BUTTON_getState

<table>
<tr><td>Service name</td><td colspan="2">BUTTON_getState</td></tr>
<tr><td>Syntax</td><td colspan="2">en_btnState_t BUTTON_getState(
en_btnId_t en_btnId
);</td></tr>
<tr><td>Parameters (in)</td><td>en_btnId</td><td>Start 0x00</td></tr>
<tr><td>Return</td><td colspan="2"><table><tr><td rowspan="7">en_btnState_t</td><td>BT_PRE_PUSH</td></tr><tr><td>BT_PUSHED</td></tr><tr><td>BT_PRE_HOLD</td></tr><tr><td>BT_HOLD</td></tr><tr><td>BT_PRE_RELEASE</td></tr><tr><td>BT_RELEASED</td></tr><tr><td>BT_UNDEFINED</td></tr></table></td></tr>
<tr><td>Description</td><td colspan="2">This Function gets the Button state.</td></tr>
</table>

- Button_Main_Task

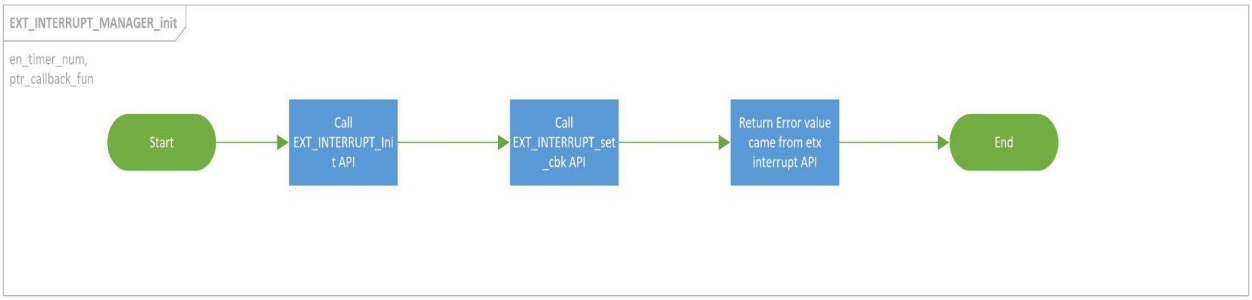
Service name	BUTTON_Main_Task
Syntax	void BUTTON_Main_Taskt(void);
Parameters (in)	NONE
Return	NONE
Description	This Function update all button states Shall call periodic

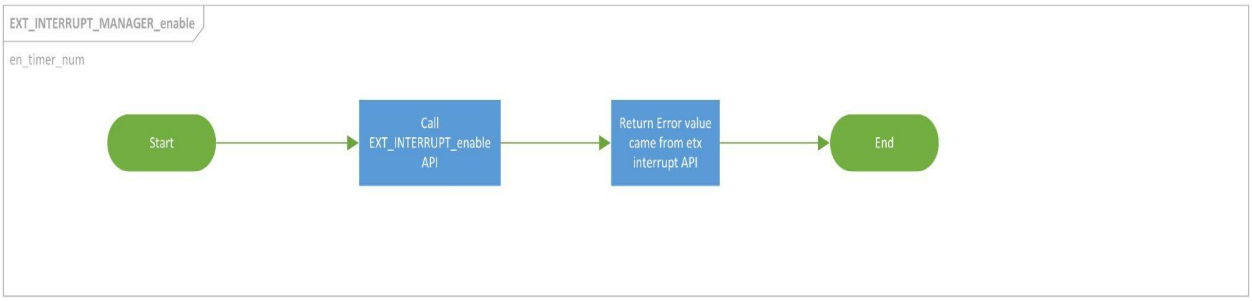
- BUTTON_init

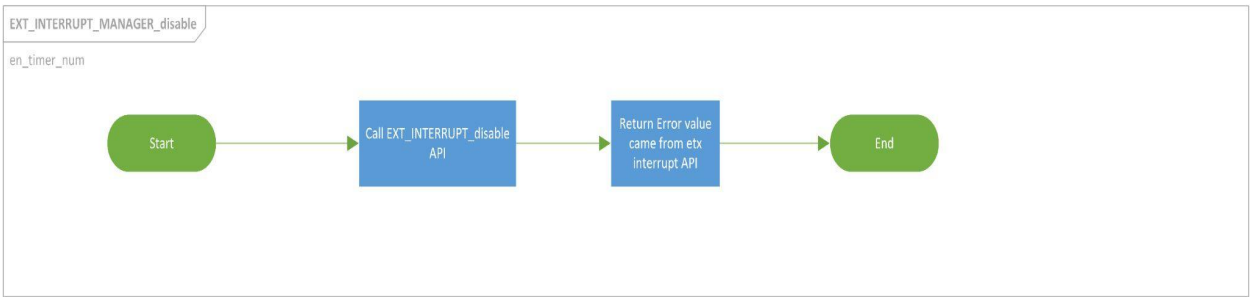
<table>
<tr><td>Service name</td><td colspan="2">BUTTON_init</td></tr>
<tr><td>Syntax</td><td colspan="2">en_btnState_t BUTTON_init(
en_btnId_t en_btnId
);</td></tr>
<tr><td>Parameters (in)</td><td>en_btnId</td><td>Start 0x00</td></tr>
<tr><td>Return</td><td colspan="2"><table><tr><td rowspan="2">en_btnState_t</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>

3.3.3: External interrupt manager API:

3.3.3.1 :Flowcharts:







3.3.3.2 : Type definitions:

Imported from External interrupt Module

3.3.3.3 : Services affecting the hardware unit

- EXT_INTERRUPT_MANAGER_init

Service name	EXT_INTERRUPT_MANAGER_init
Syntax	u8_interruptError_t EXT_INTERRUPT_MANAGER_init(en_ext_interrupt_num_t en_ext_interrupt_num, void(*callback)(void));

Parameters (in)	en_timerNum	Ext interrupt number
	callback	Pointer to the call back function
Return	u8_interruptError_t	EXT_INT_E_OK EXT_INT_E_NOK
Description	This Function Initialize external interrupt module	

- EXT_INTERRUPT_MANAGER_enable

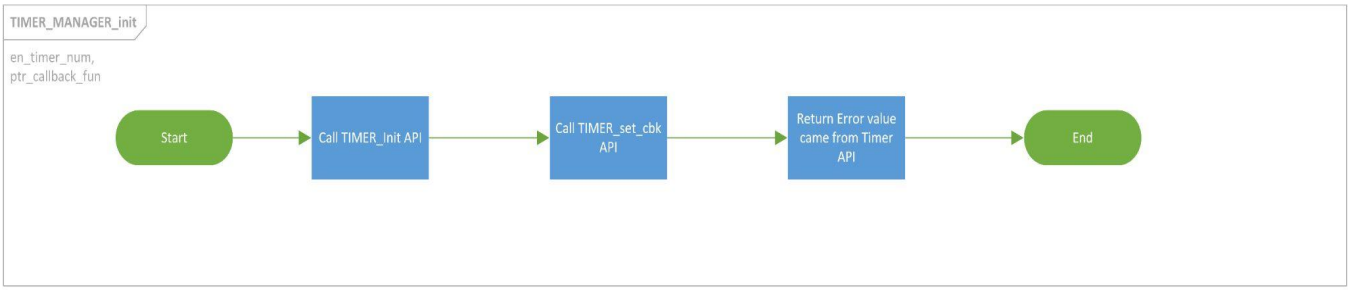
Service name	EXT_INTERRUPT_MANAGER_enable	
Syntax	u8_interruptError_t EXT_INTERRUPT_MANAGER_enable(en_ext_interrupt_num_t en_ext_interrupt_num);	
Parameters (in)	en_timerNum	Ext interrupt number
Return	u8_interruptError_t	EXT_INT_E_OK EXT_INT_E_NOK
Description	This Function enables external interrupt	

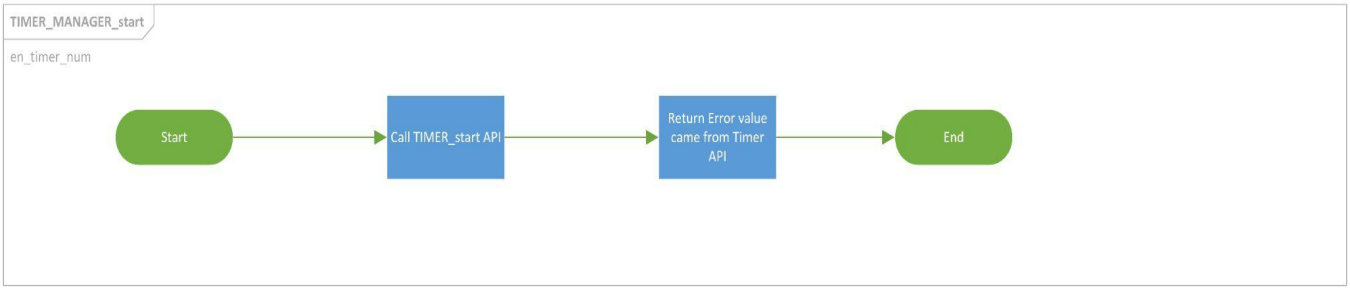
- EXT_INTERRUPT_MANAGER_disable

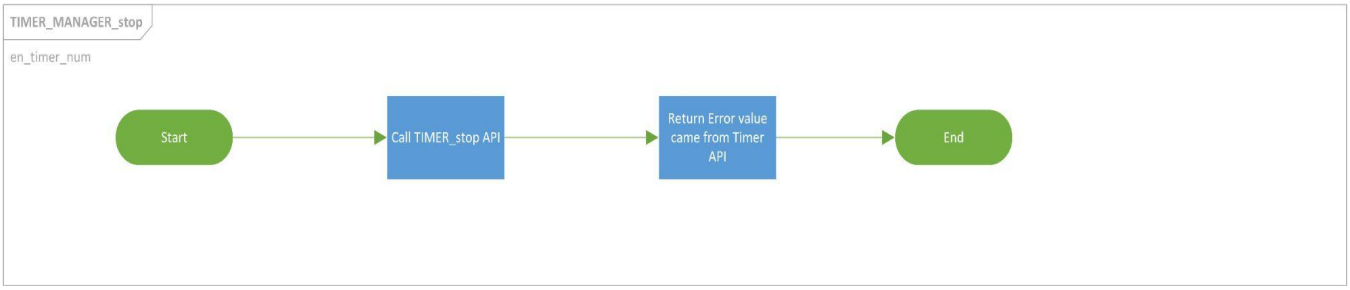
Service name	EXT_INTERRUPT_MANAGER_disable	
Syntax	u8_interruptError_t EXT_INTERRUPT_disable(en_ext_interrupt_num_t en_ext_interrupt_num);	
Parameters (in)	en_timerNum	Ext interrupt number
Return	u8_interruptError_t	EXT_INT_E_OK EXT_INT_E_NOK
Description	This Function disables external interrupt	

3.3.4: Timer manager API:

3.3.4.1 :Flowcharts:







3.3.4.2 : Type definitions:

Imported from Timer Module

3.3.4.3 : Services affecting the hardware unit

- TIMER_MANAGER_init

Service name	TIMER_MANAGER_init
Syntax	u8_timerErrors_t TIMER_MANAGER_init(en_timer_num_t en_timerNum,

	timerCallBack callBackFunction_ptr);	
Parameters (in)	en_timerNum	Timer number
	callBackFunction_ptr	Pointer to the call back function
Return	u8_timerErrors_t	TIMER_E_OK TIMER_E_NOT_OK
Description	This Function initializes a timer module	

- TIMER_MANAGER_start

Service name	TIMER_MANAGER_start	
Syntax	u8_timerErrors_t TIMER_MANAGER_start(en_timer_num_t en_timerNum);	
Parameters (in)	en_timerNum	Timer number
Return	u8_timerErrors_t	TIMER_E_OK TIMER_E_NOT_OK
Description	This Function starts the timer	

- TIMER_MANAGER_stop

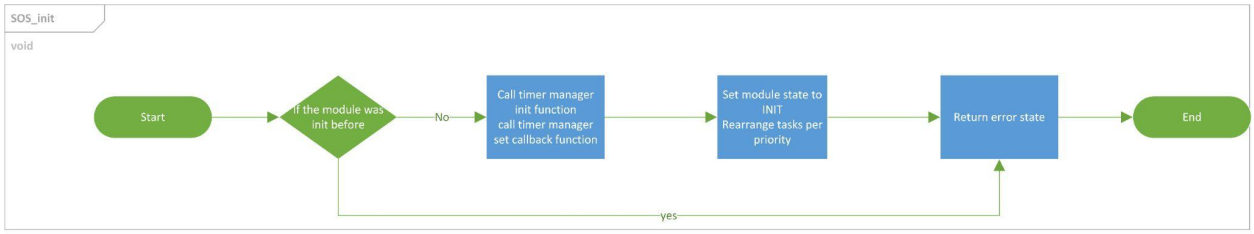
Service name	TIMER_MANAGER_stop	
Syntax	u8_timerErrors_t TIMER_MANAGER_stop(en_timer_num_t en_timerNum);	
Parameters (in)	en_timerNum	Timer number
Return	u8_timerErrors_t	TIMER_E_OK TIMER_E_NOT_OK

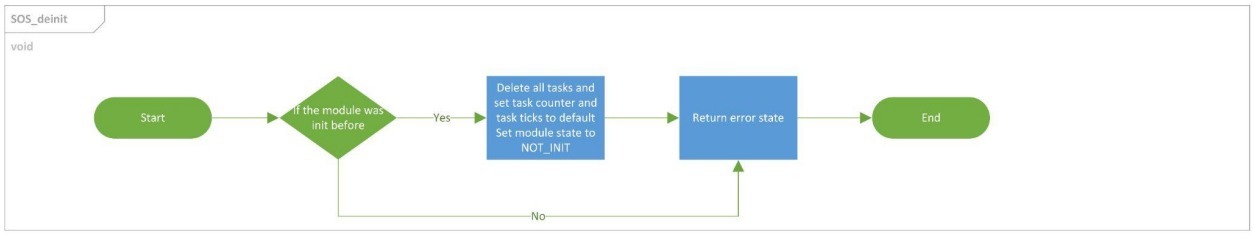
Description	This Function stops the timer

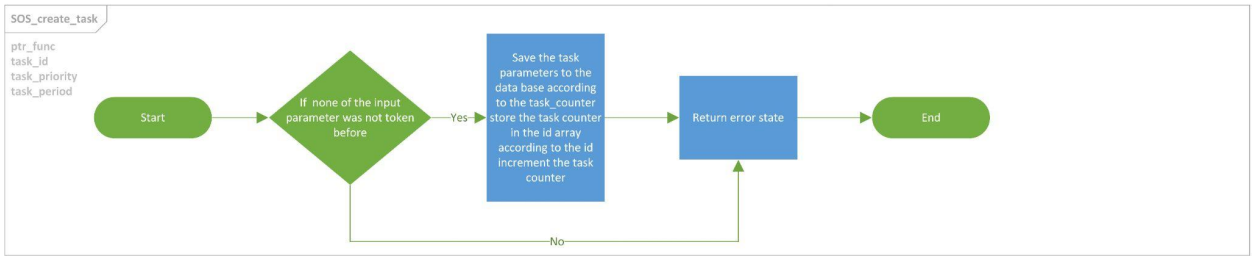
3.4: SERV APIs

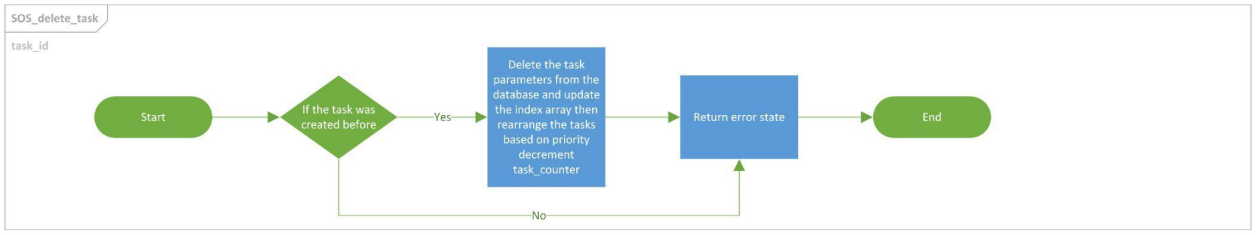
3.4.1: SOS API:

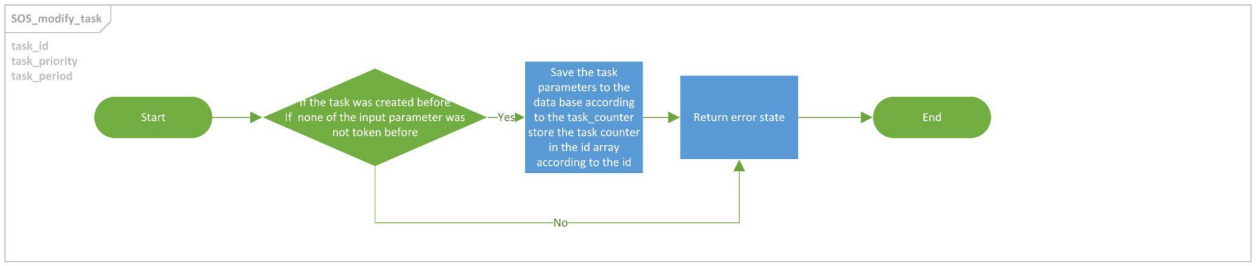
3.4.1.1 :Flowcharts:

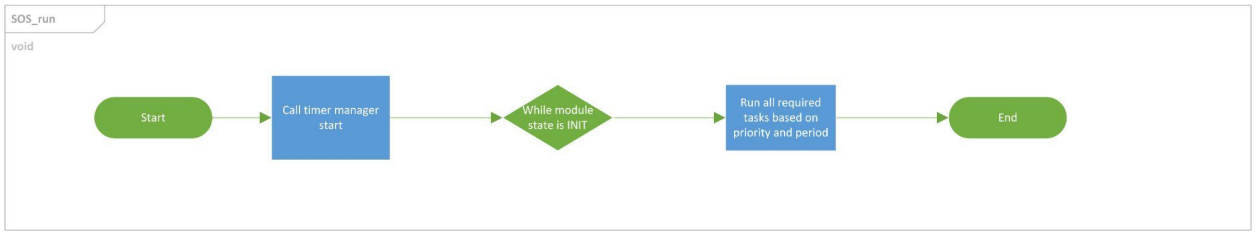


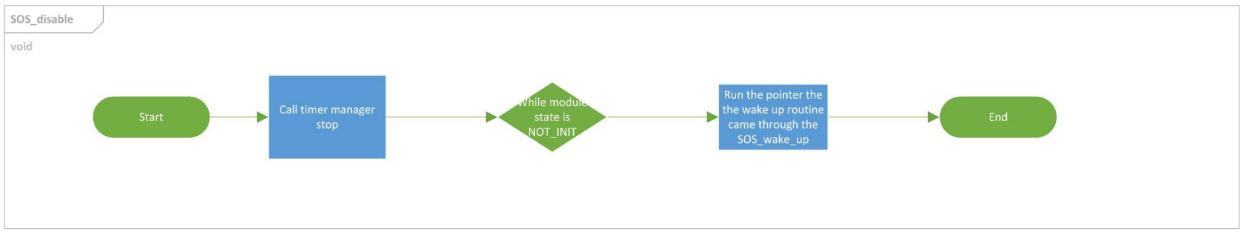


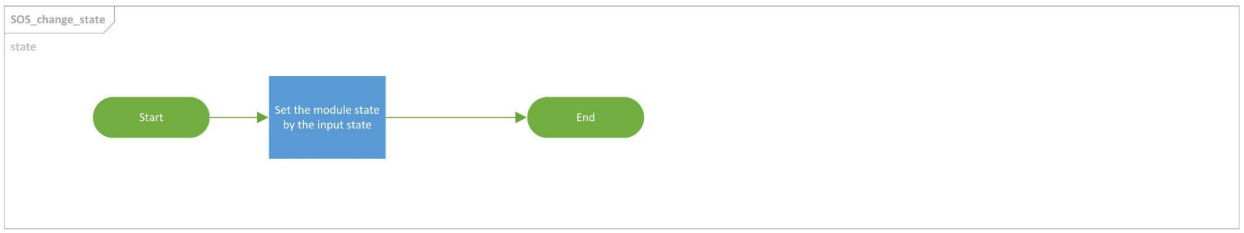


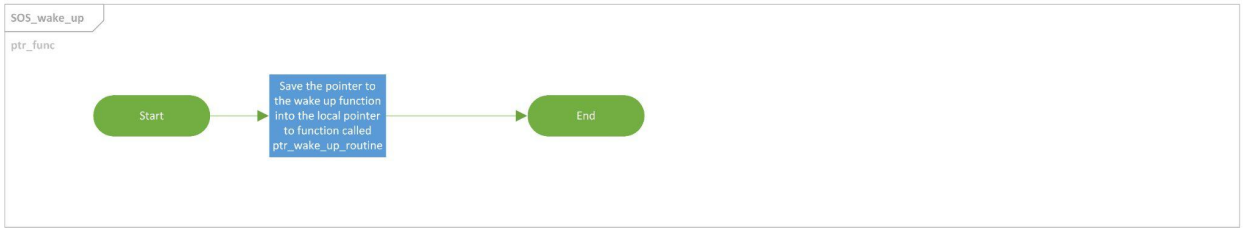

SOS_delete_task
task_id
Start
If the task was created before
Yes
Delete the task parameters from the database and update the index array then rearrange the tasks based on priority decrement task_counter
Return error state
End
No


SOS_modify_task
task_id
task_priority
task_period
Start
If the task was created before
If none of the input parameter was not token before
Yes
Save the task parameters to the data base according to the task_counter store the task counter in the id array according to the id
Return error state
End
No


SOS_run
void
Start
Call timer manager start
While module state is INIT
Run all required tasks based on priority and period
End


SOS_disable
void
Start
Call timer manager stop
While module state is NOT_INIT
Run the pointer the the wake up routine came through the SOS_wake_up
End


SOS_change_state
state
Start
Set the module state by the input state
End


SOS_wake_up
ptr_func
Start
Save the pointer to the wake up function into the local pointer to function called ptr_wake_up_routine
End

3.4.1.2 : Type definitions:

- st_task_config_t

Name	st_task_config_t
Type	Structure
Description	This is the type of the data structure containing the overall configuration data for the SOS API
Available via	sos_types.h

- arr_st_gs_task_config

Name	arr_st_gs_task_config
Type	Array of Structures
Description	This is the type of the data structure containing the overall configuration data for the application tasks
Available via	sos.c

- u8_gs_arr_index_id

Name	u8_gs_arr_index_id
Type	Array
Description	This is the type of the data containing the overall IDs indexes data for the application tasks
Available via	sos.c

- enu_system_status_t

Name	enu_system_status_t		
Type	Enumeration		
Range	SOS_STATUS_SUCCESS	0x00	SOS error OK

	SOS_STATUS_INVALID_STATE | 0x07 | SOS error
Description	enu_system_status_t
Available via	sos_types.h

3.4.1.3 : Services affecting the hardware unit

- SOS_init

Service name	SOS_init
Syntax	enu_system_status_t SOS_init(void);
Parameters (in)	void
Return	enu_system_status_t: SOS_STATUS_SUCCESS / SOS_STATUS_INVALID_STATE
Description	This Function Initialize SOS module

- SOS_deinit

Service name	SOS_deinit
Syntax	enu_system_status_t SOS_deinit(void);
Parameters (in)	void
Return	enu_system_status_t: SOS_STATUS_SUCCESS / SOS_STATUS_INVALID_STATE
Description	This Function DeInitialize SOS module

- SOS_run

Service name	SOS_run

Syntax	enu_system_status_t SOS_run(void);
Parameters (in)	void
Return	void
Description	This Function Runs SOS module

- SOS_disable

Service name	SOS_disable
Syntax	enu_system_status_t SOS_disable(void);
Parameters (in)	void
Return	void
Description	This Function Disable SOS module

- SOS_change_state

Service name	SOS_change_state
Syntax	enu_system_status_t SOS_change_state(uint8_t u8_state);
Parameters (in)	State which to store in the SOS module state
Return	void
Description	This Function Change the state of the SOS module to switch between sos_run and sos_disable

- SOS_wake_up

Service name	SOS_wake_up
Syntax	enu_system_status_t SOS_wake_up(ptr_function_name_t ptr_function_name);
Parameters (in)	Ptr_function_name pointer to the wake-up routine
Return	void
Description	This Function to switch from sos_disable to sos_run

- SOS_delete_task

Service name	SOS_delete_task	
Syntax	enu_system_status_t SOS_delete_task(uint8_t u8_task_id);	
Parameters (in)	U8_task_id the task id to be deleted	
Return	enu_system_status_t	SOS_STATUS_SUCCESS
		SOS_STATUS_INVALID_STATE
Description	This Function deletes a task from SOS module	

- SOS_modify_task

Service name	SOS_modify_task	
Syntax	enu_system_status_t SOS_modify_task(uint8_t u8_task_id , uint8_t u8_task_periority,uint16_t u16_task_period);	
Parameters (in)	U8_task_id the task ID to be modified	
	U8_task_periority the new task priority	
	U16_task_period the new task period	
Return	enu_system_status_t	SOS_STATUS_SUCCESS
		SOS_STATUS_INVALID_STATE
Description	This Function modify a task in SOS module	

- SOS_create_task

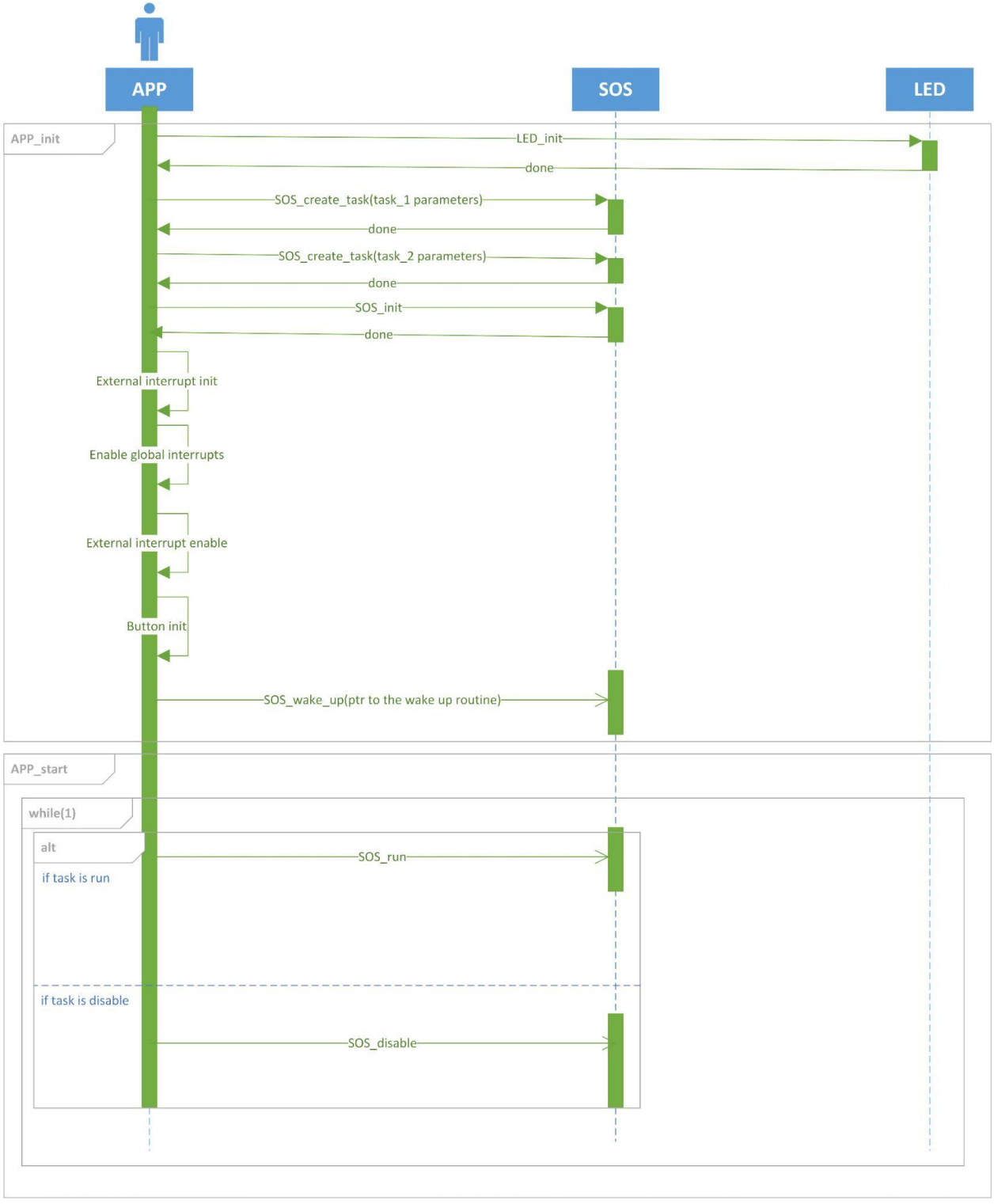
Service name	SOS_create_task
Syntax	enu_system_status_t SOS_create_task(ptr_function_name_t ptr_function_name , uint8_t u8_task_id , uint8_t u8_task_periority,uint16_t u16_task_period);
Parameters (in)	ptr_function_name, Pointer to the task

<table>
<tr><td></td><td>U8_task_id, the task ID to be created</td></tr>
<tr><td></td><td>U8_task_periority, the task priority</td></tr>
<tr><td></td><td>U16_task_period, the task period</td></tr>
<tr><td>Return</td><td><table><tr><td rowspan="2">enu_system_status_t</td><td>SOS_STATUS_SUCCESS</td></tr><tr><td>SOS_STATUS_INVALID_STATE</td></tr></table></td></tr>
<tr><td>Description</td><td>This Function creates a task in SOS module</td></tr>
</table>

3.5: APP APIs

3.5.1: APP API:

3.5.1.1 :seq diagram:



3.5.1.2 : 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