Small OS Design

By Team 4

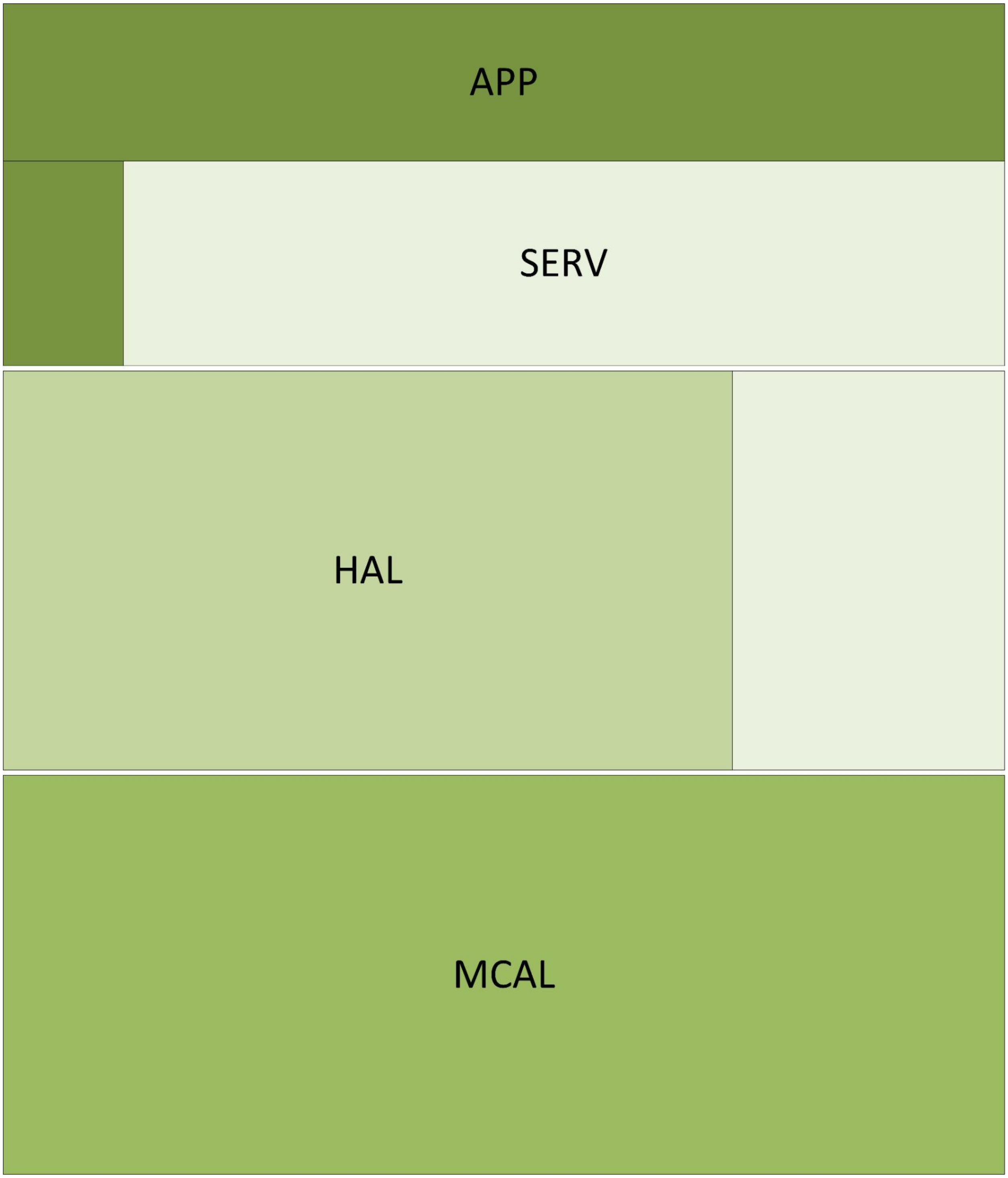
GitHub Repo: https://github.com/MahmoudMatarawy/small-os
PDF:
https://github.com/MahmoudMatarawy/small-os/blob/main/design%20doc/Small%20OS%20Design.pdf

Table of content:

1 : Detailed Requirements

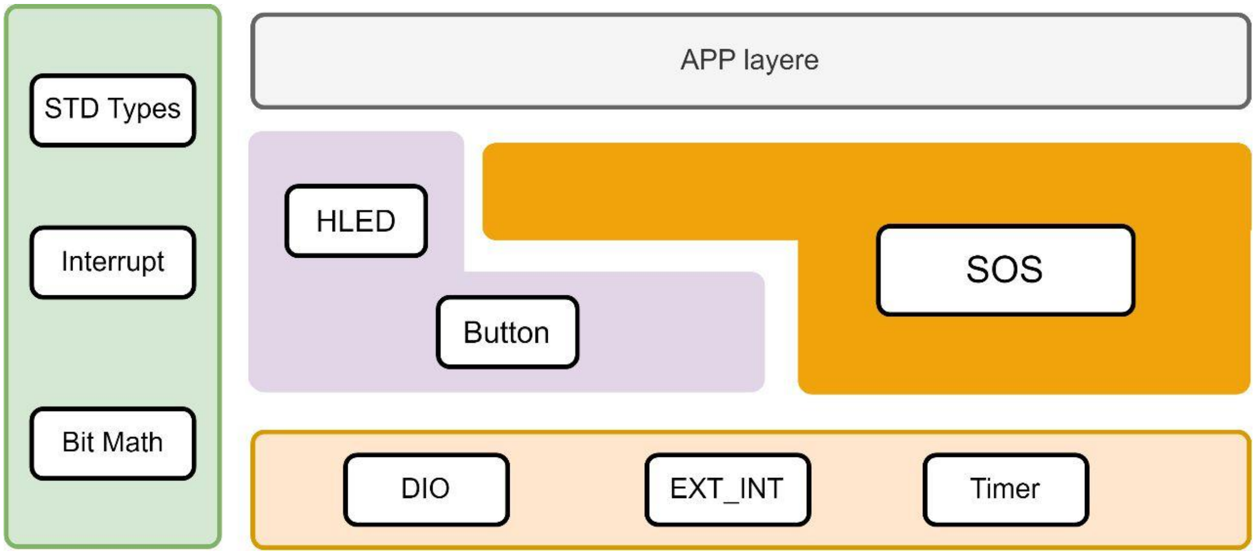
1. Implement an application that calls the SOS module and use 2 tasks
1. Task 1: Toggle LED_0 (Every 300 Milliseconds)
2. Task 2: Toggle LED_1 (Every 500 Milliseconds)
2. Make sure that these tasks occur periodically and forever
3. When pressing PBUTTON0, the SOS will stop
4. When Pressing PBUTTON1,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

2 : Layered architecture



3 : System modules

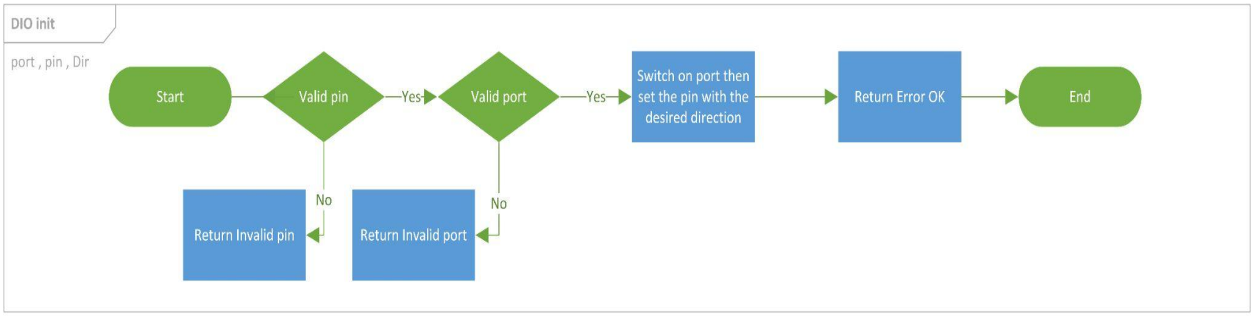
3.1: Module architecture

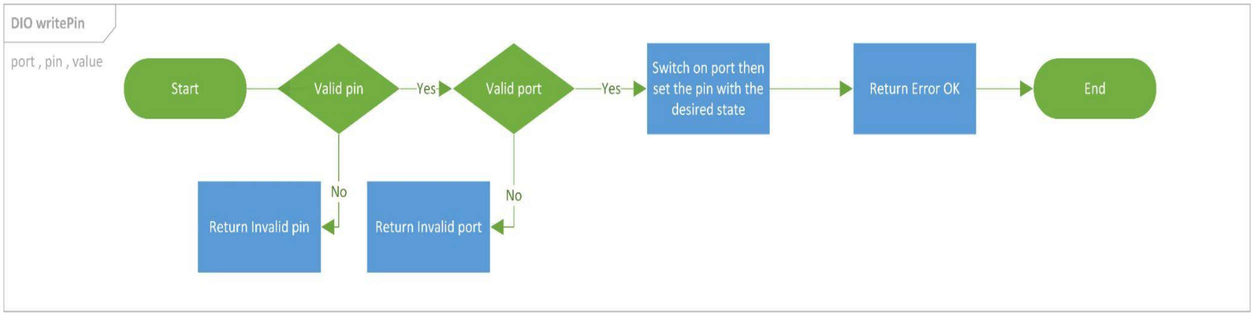


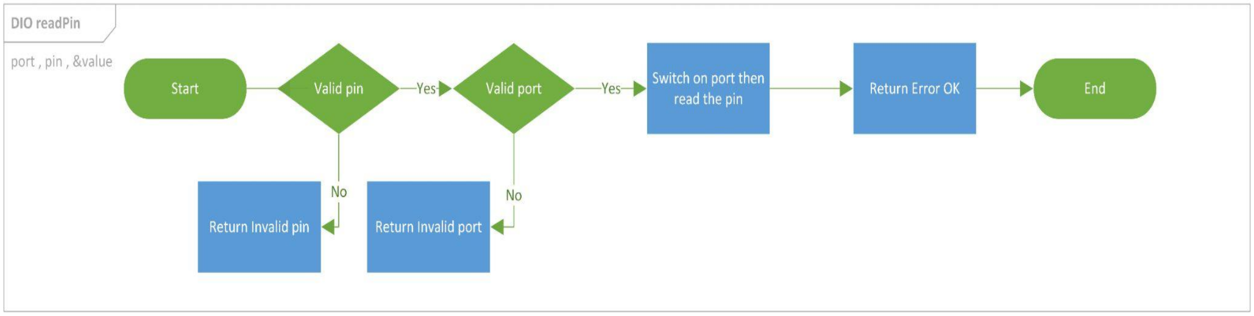
3.2: MCAL APIs

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

<table>
<tr><td>Name</td><td>u8_en_dioErrorsType</td></tr>
<tr><td>Type</td><td>Enumeration</td></tr>
<tr><td>Range</td><td><table>
<tr><td>DIO_E_OK</td><td>0x00</td><td>DIO error OK</td></tr>
<tr><td>DIO_InvalidPin</td><td>0x01</td><td>DIO error, invalid pin number.</td></tr>
<tr><td>DIO_InvalidPort</td><td>0x02</td><td>DIO error, invalid port number.</td></tr>
</table></td></tr>
<tr><td>Description</td><td>u8_en_dioErrors</td></tr>
<tr><td>Available via</td><td>dio.h</td></tr>
</table>

- 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

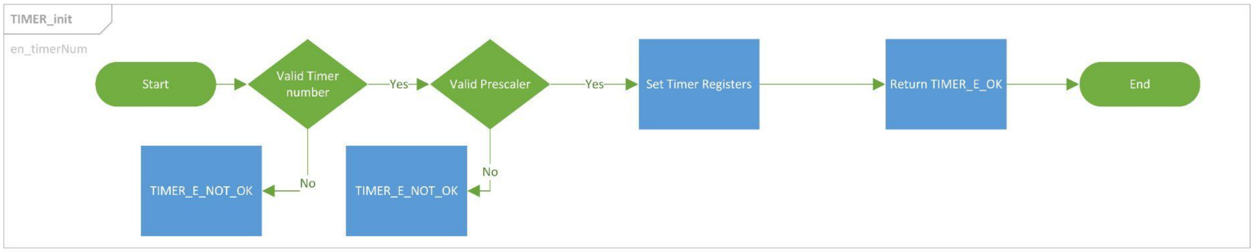
Service name	DIO_init
Syntax	u8_en_dioErrors DIO_init (en_dioPortsType port, en_dioPinsType pin, u8_en_dioDirType direction

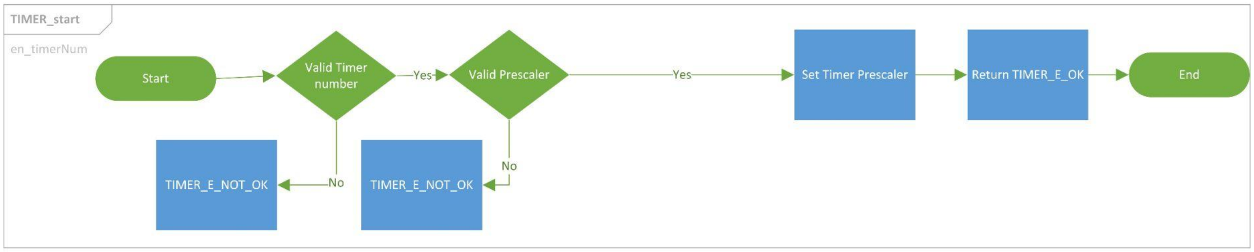
<table>
<tr><td></td><td colspan="3">);</td></tr>
<tr><td rowspan="3">Parameters (in)</td><td>Port, pin</td><td colspan="2">Channel ID</td></tr>
<tr><td rowspan="2">direction</td><td rowspan="2">Value to be set</td><td>STD_INPUT</td></tr>
<tr><td>STD_OUTPUT</td></tr>
<tr><td>Return</td><td colspan="3"><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>

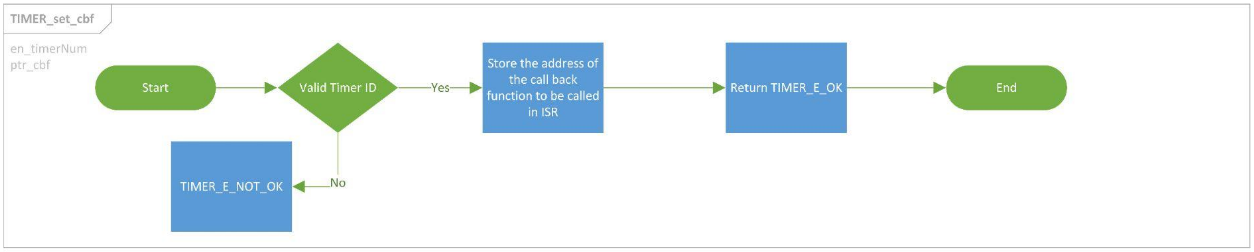
- This function shall return DIO_InvalidPin if pin number is invalid
- This function shall return DIO_InvalidPort if port number is invalid.

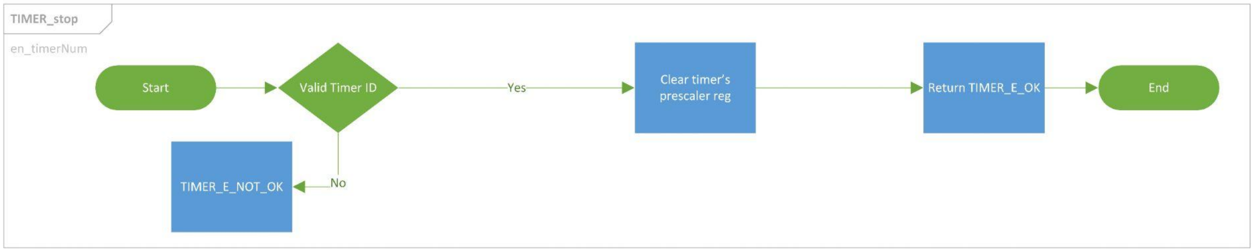
3.2.2: TIMER API:

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

<table>
<tr><td>Name</td><td>u8_timerErrors_t</td></tr>
<tr><td>Type</td><td>Enumeration</td></tr>
<tr><td>Range</td><td><table><tr><td>TIMER_E_OK</td><td>0x00</td><td>Timer error OK</td></tr><tr><td>TIMER_E_NOT_OK</td><td>0x03</td><td>Timer error</td></tr></table></td></tr>
<tr><td>Description</td><td>u8_timerErrors_t</td></tr>
<tr><td>Available via</td><td>timer_types.h</td></tr>
</table>

- 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

Service name	TIMER_init	
Syntax	u8_timerErrors_t TIMER_init(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 Initialize timer module	

- 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	

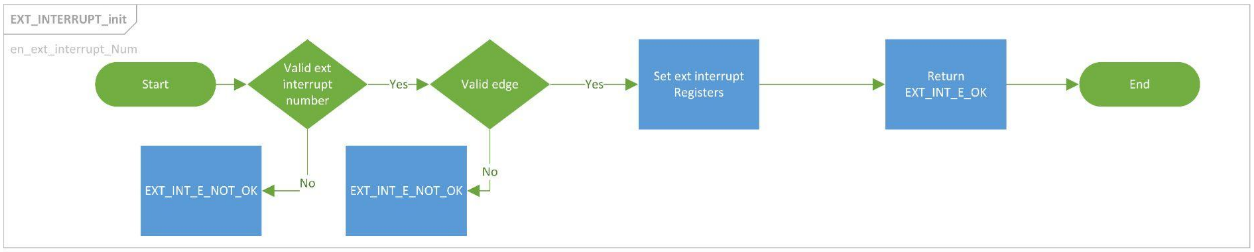
- TIMER_set_cbf

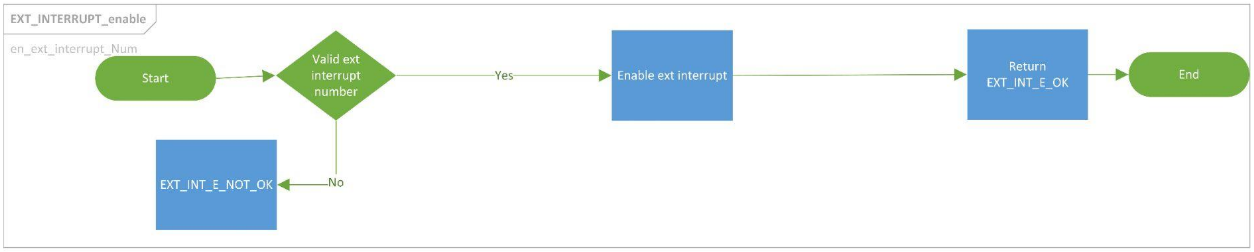
Service name	TIMER_set_cbf	
Syntax	u8_timerErrors_t TIMER_set_cbf(en_timer_num_t en_timerNum, timerCallBack callBackFunction_ptr);	
Parameters (in)	en_timerNum	Timer number
	callBackFunction_ptr	Pointer to the call back function

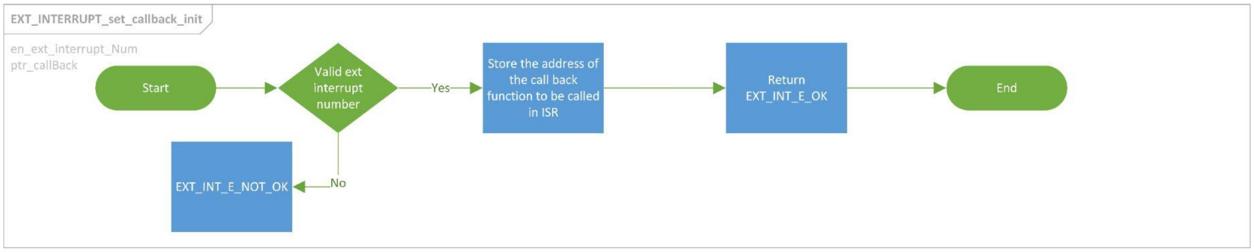
Return	u8_timerErrors_t	TIMER_E_OK
		TIMER_E_NOT_OK
Description	This Function starts the timer	

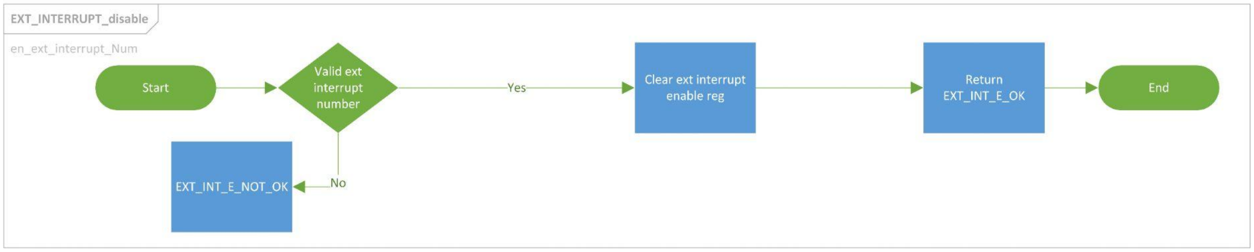
3.2.2: External interrupt API:

3.2.2.1 :Flowcharts:






EXT_INTERRUPT_set_callback_init
en_ext_interrupt_Num
ptr_callBack
Start
Valid ext interrupt number
Yes
Store the address of the call back function to be called in ISR
Return EXT_INT_E_OK
End
No
EXT_INT_E_NOT_OK



3.2.2.2 : Type definitions:

- u8_interruptError_t

<table>
<tr><td>Name</td><td>u8_interruptError_t</td></tr>
<tr><td>Type</td><td>Enumeration</td></tr>
<tr><td>Range</td><td><table>
<tr><td>EXT_INT_E_OK</td><td>0x00</td><td>Ext Interrupt error OK</td></tr>
<tr><td>EXT_INT_E_NOK</td><td>0x04</td><td>Ext Interrupt error</td></tr>
</table></td></tr>
<tr><td>Description</td><td>u8_interruptError_t</td></tr>
<tr><td>Available via</td><td>ext_interrupt_types.h</td></tr>
</table>

- 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	

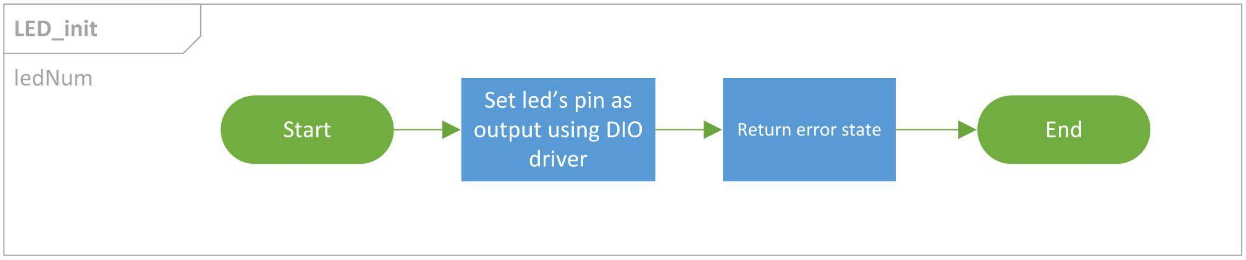
Syntax	u8_interruptError_t EXT_INTERRUPT_set_callback_init(
	en_ext_interrupt_num_t en_ext_interrupt_num,
	void(*callback)(void)
	);	

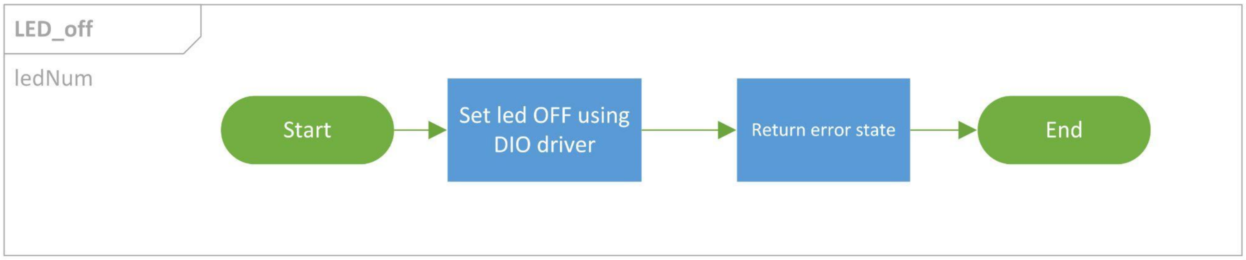
Parameters (in)	en_timerNum	Ext interrupt number
	callback	Pointer to the call back function
Return	u8_interruptError_t	EXT_INT_E_OK
		EXT_INT_E_NOK
Description	This Function saves call back pointer to call it in ISR	

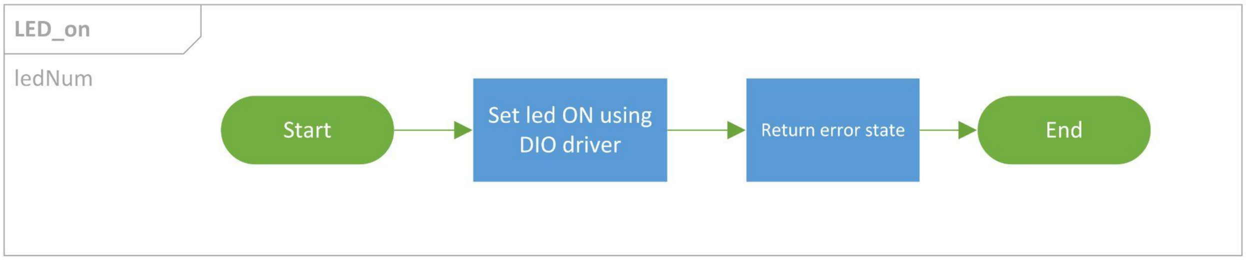
3.3: HAL APIs

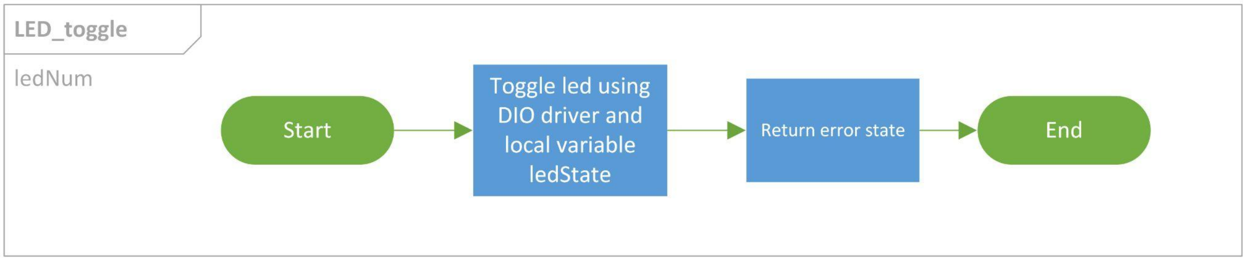
3.3.1: LED API:

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

<table>
<tr><td>Name</td><td>u8_ledError_t</td></tr>
<tr><td>Type</td><td>Enumeration</td></tr>
<tr><td>Range</td><td><table><tr><td>LED_ERROR_OK</td><td>0x00</td><td>LED error OK</td></tr><tr><td>LED_ERROR_NOT_OK</td><td>0x05</td><td>LED error</td></tr></table></td></tr>
<tr><td>Description</td><td>u8_ledError_t</td></tr>
<tr><td>Available via</td><td>led.h</td></tr>
</table>

- en_ledNum_t

<table>
<tr><td>Name</td><td>en_ledNum_t</td></tr>
<tr><td>Type</td><td>Enumeration</td></tr>
<tr><td>Range</td><td><table><tr><td>LED_0</td><td>0x00</td><td>LED_0</td></tr><tr><td>LED_1</td><td>0x01</td><td>LED_1</td></tr></table></td></tr>
<tr><td>Description</td><td>en_ledNum_t</td></tr>
<tr><td>Available via</td><td>led.h</td></tr>
</table>

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:

- str_button_t

Name	str_button_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.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.h		

- en_btnState_t

Name	en_btnState_t		
Type	Enumeration		
Range	BT_PUSHED	0x00	Pushed Level
	BT_RELEASED	0x01	Released Level
Description	Button state Enum		
Available via	button.h		

3.3.2.3 : Services affecting the hardware unit

- BUTTON_get_state

Service name	BUTTON_get_state	
Syntax	Enu_button_return_state_t button_get_state(str_button_t *ptr_str_btn ,button_state_t *ptr_enu_btn_state);	
Parameters (in)	ptr_str_btn	Pointer to the configuration structure
	ptr_enu_btn_state	Pointer where to store the state
Return	Enu_button_return_state_t	BUTTON_E_OK
		BUTTON_E_NOK
Description	This Function init a button without an external interrupt	

- BUTTON_init

Service name	BUTTON_init	
Syntax	Enu_button_return_state_t button_init(str_button_t *ptr_str_btn);	
Parameters (in)	ptr_str_btn	Pointer to the configuration structure
Return	Enu_button_return_state_t	BUTTON_E_OK BUTTON_E_NOK
Description	This Function init a button without an external interrupt	

- BUTTON_with_int

Service name	BUTTON_with_int	
Syntax	Enu_button_return_state_t button_with_INT(str_button_t *ptr_str_btn , void (*func)(void));	
Parameters (in)	ptr_str_btn	Pointer to the configuration structure
	func	Pointer to the callback function
Return	Enu_button_return_state_t	BUTTON_E_OK BUTTON_E_NOK
Description	This Function init a button with an external interrupt	

- BUTTON_enable_INT

Service name	Button_enable_INT	
Syntax	Enu_button_return_state_t button_enable_INT(str_button_t *ptr_str_btn);	
Parameters (in)	ptr_str_btn	Pointer to the configuration structure
Return	Enu_button_return_state_t	BUTTON_E_OK BUTTON_E_NOK

Description	This Function enable a button with an external interrupt

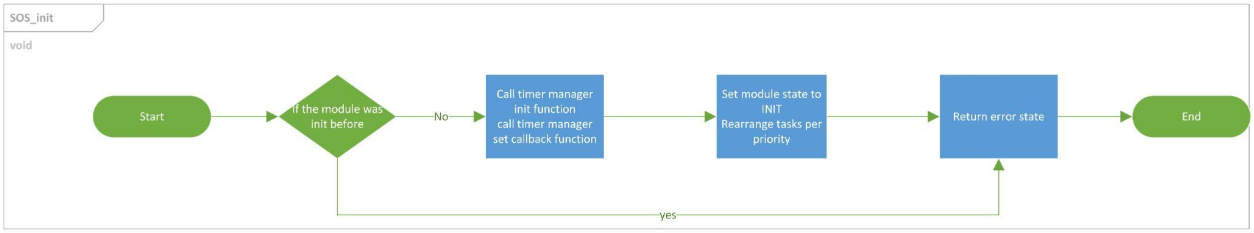
- BUTTON_disable_INT

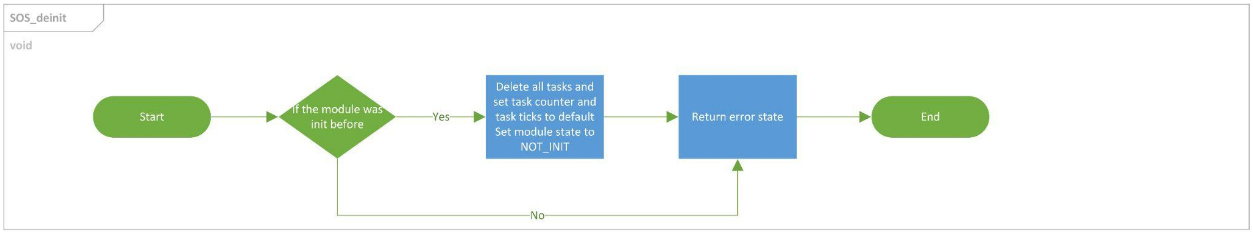
Service name	Button_disable_INT	
Syntax	Enu_button_return_state_t button_disable_INT(str_button_t *ptr_str_btn);	
Parameters (in)	ptr_str_btn	Pointer to the configuration structure
Return	Enu_button_return_state_t	BUTTON_E_OK BUTTON_E_NOK
Description	This Function disable a button with an external interrupt	

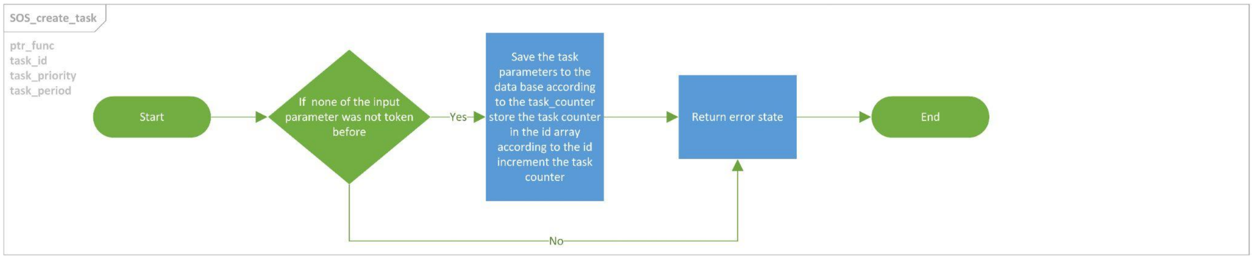
3.4: SERV APIs

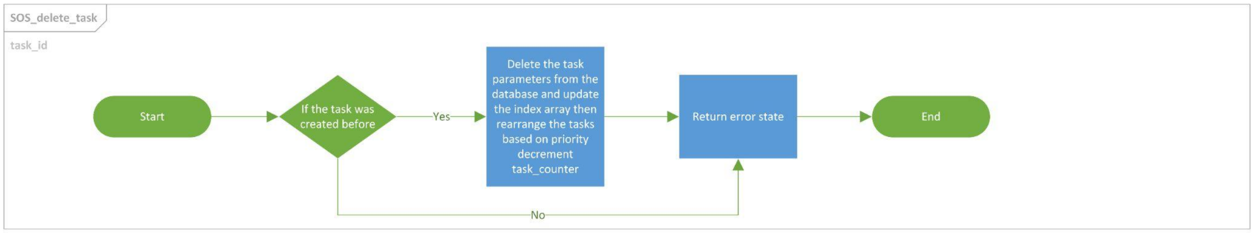
3.4.1: SOS API:

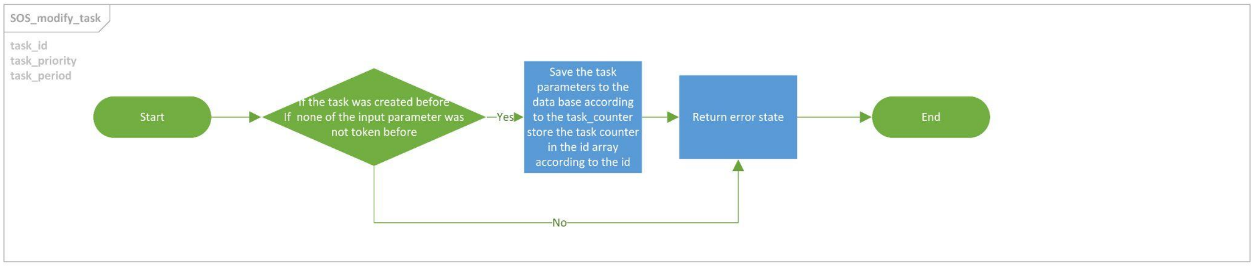
3.4.1.1 :Flowcharts:

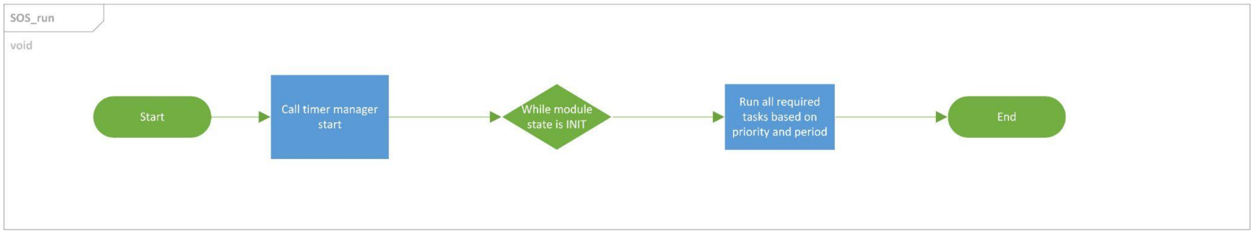

SOS_init
void
Start
If the module was init before
No
Call timer manager init function call timer manager set callback function
Set module state to INIT Rearrange tasks per priority
Return error state
End
yes

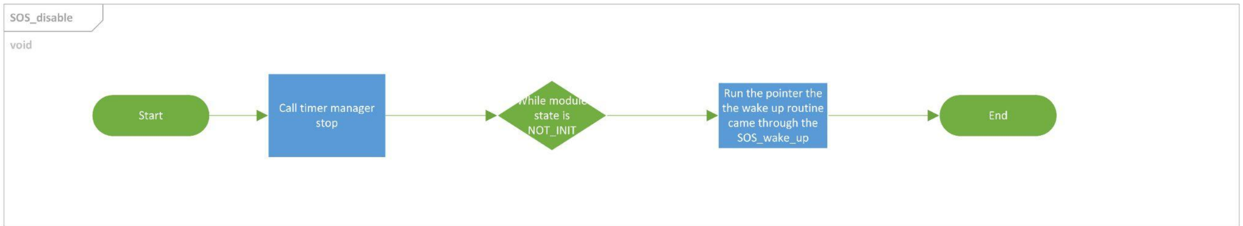

SOS_deinit
void
Start
If the module was init before
Yes
Delete all tasks and set task counter and task ticks to default Set module state to NOT_INIT
Return error state
End
No

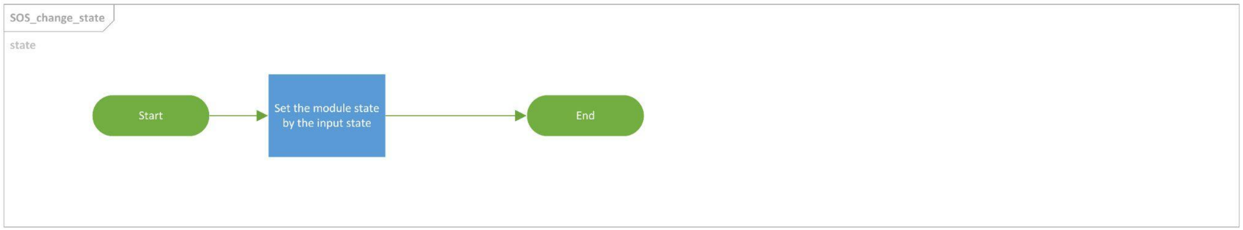

SOS_create_task
ptr_func
task_id
task_priority
task_period
Start
If none of the input parameter was not token before
Yes
Save the task parameters to the data base according to the task_counter store the task counter in the id array according to the id increment the task counter
Return error state
End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

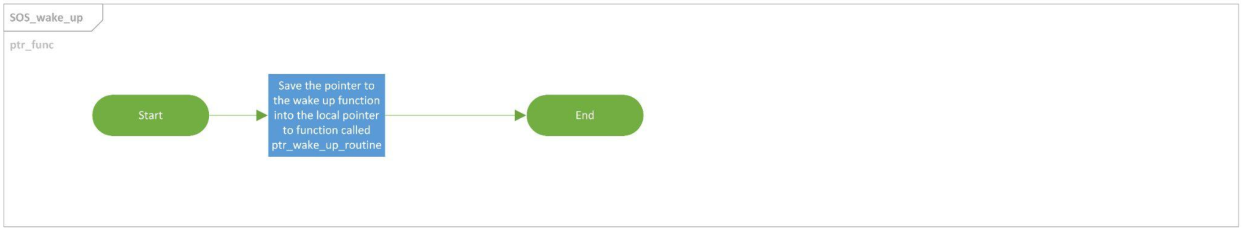

SOS_delete_task
task_id
Start
If the task was created before
Yes
Delete the task parameters from the database and update the index array then rearrange the tasks based on priority decrement task_counter
Return error state
End
No


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

<table>
<tr><td>Name</td><td>enu_system_status_t</td></tr>
<tr><td>Type</td><td>Enumeration</td></tr>
<tr><td>Range</td><td><table><tr><td>SOS_STATUS_SUCCESS</td><td>0x00</td><td>SOS error OK</td></tr></table></td></tr>
</table>

	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

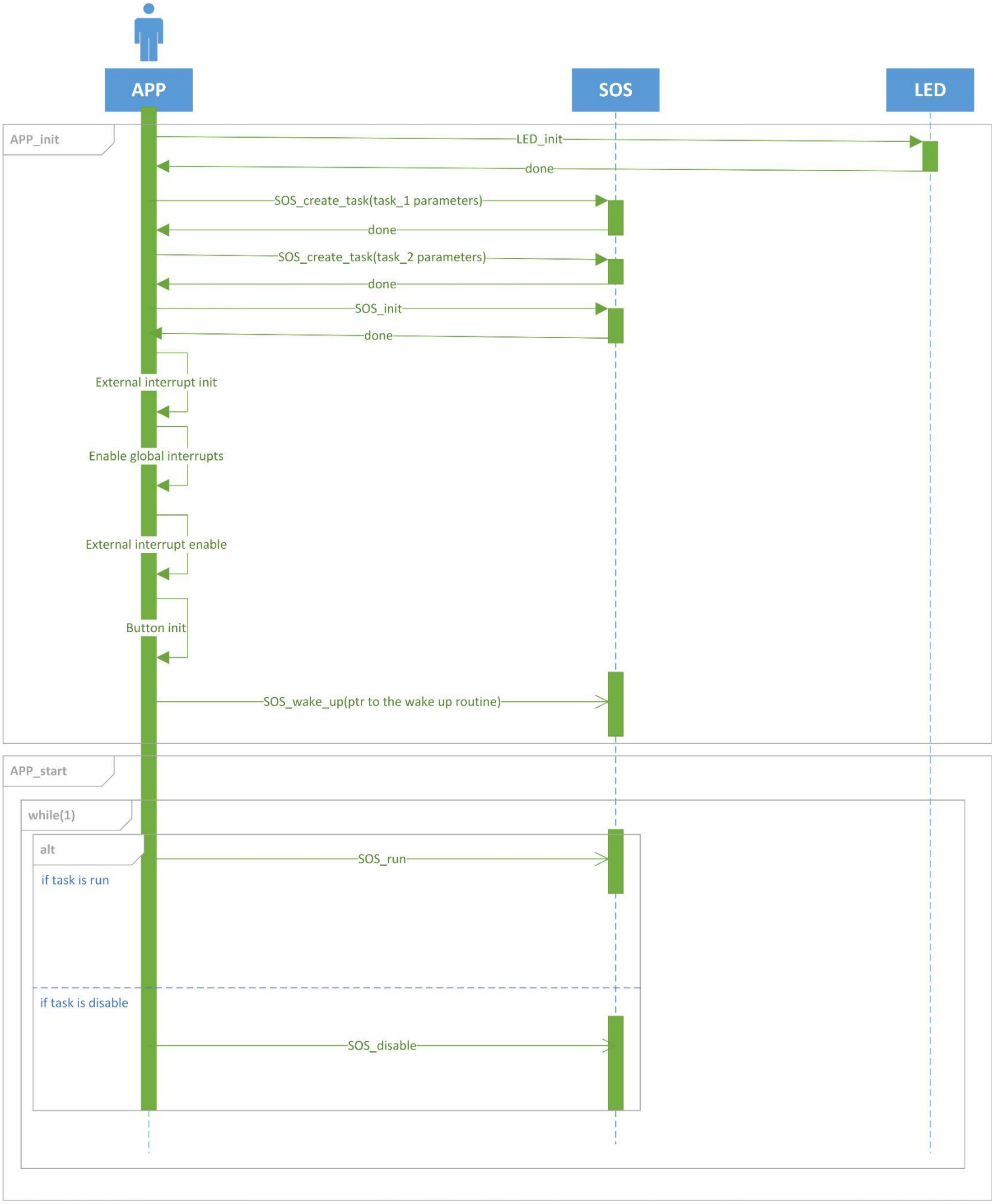
Service name	SOS_create_task
Syntax	enu_system_status_t SOS_create_task(ptr_function_name_t ptr_function_name , uint8_t u8_task_id , uint8_t u8_task_periority,uint16_t u16_task_period);
Parameters (in)	ptr_function_name, Pointer to the task

<table>
<tr><td rowspan="3"></td><td>U8_task_id, the task ID to be created</td></tr>
<tr><td>U8_task_periority, the task priority</td></tr>
<tr><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