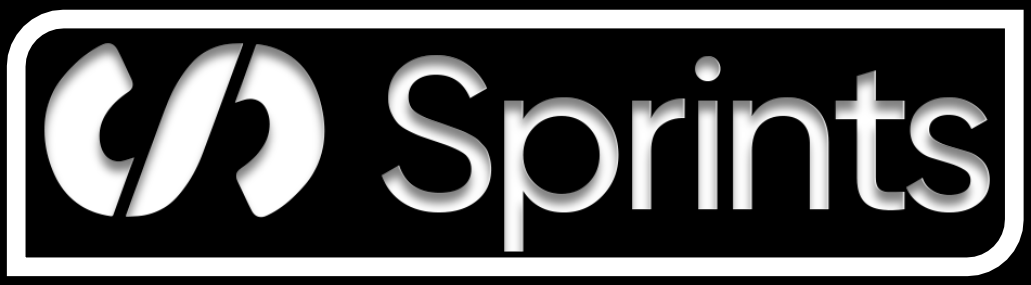


Small Operating System

Design

Bassel Yasser Mahmoud

2023

Sprints

Contents

Project Introduction

The Simple Operating System module has a capability to work with different Tasks, and how to manage and synchronize tasks with each other.

In this design Docs we'll discuss layered architecture, module description, drivers' documentation and UML on High Level Design.

We'll discuss also flowchart of each module, Pre-compiling configuration and Linking configuration on Low Level Design

High Level Design

Layered Architecture

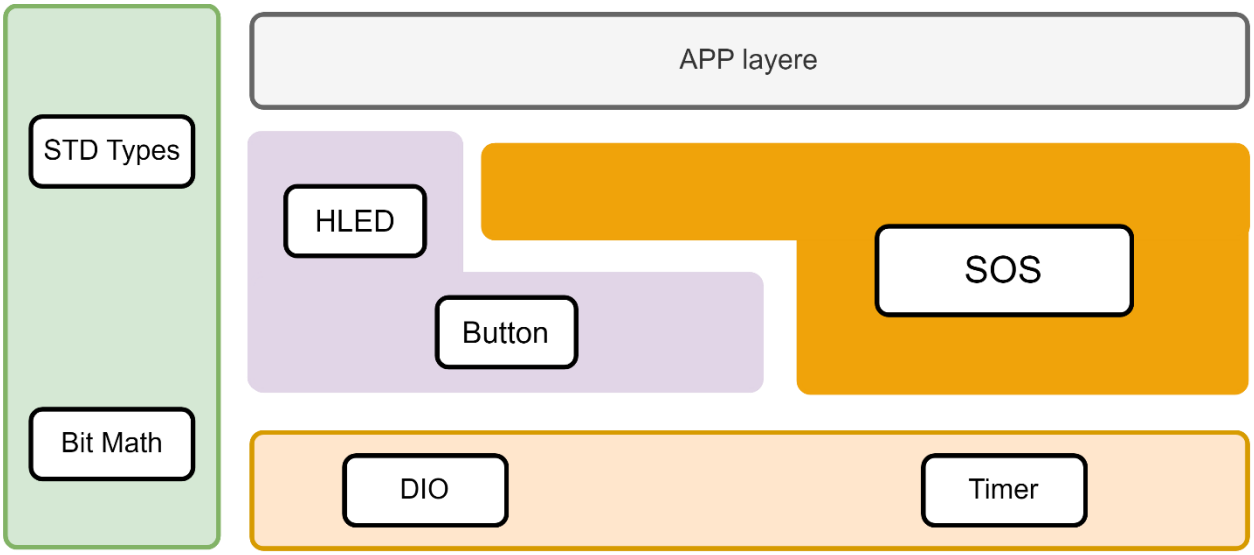
APP Layer: written in high level languages like java, C++, C# with rich GUI support. Application layer calls the middleware API in response to action by the user or an event.

SERVICE Layer: is a layer in an application that facilitates communication between the controller and the persistence layer, Faster and easy Integration with multiple applications, Light weight design to provide scalability.

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



Module Description

- **APP Layer**
  - **App:** 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.
- **SERVICE Layer**
  - **SOS:** Task management and synchronization.
- **HAL Layer**
  - **LED:** this led module configure selected pin as output and generate volt.
  - **Button:** Configure selected pin as input and pull up.
- **MCAL Layer**
  - **Timer:**
  - **DIO:** Configure DIO pins as INPUT or OUTPUT, HIGH or LOW, and get pin status.
- **COMMON Layer**
  - **std_types:** having basic standard types like (Uint32_t, Uint8_t, …..).
  - **bit_math:** Consist of bit manipulation like (SetBit, ClrBit, GetBit, ..).

Drivers' documentation

APP

APP_vidInit

Service name	APP_vidInit
Description	This Function Make Modules Initialization
Syntax	void APP_vidInit (void)
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	void
Parameters (out)	None
Return	void
Available via	app.h

APP_vidStart

Service name	APP_vidStart
Description	This Function Start the Application.
Syntax	void APP_vidStart (void)
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	void
Parameters (out)	None
Return	void
Available via	app.h

SERVICE

SOS module

sos_Init

Service name	sos_Init
Description	This Function Initialize timer
Syntax	`enu_system_status_t sos_Init(void)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	`void`
Parameters (out)	None
Return	*`SOS_STATUS_SUCCESS` in case of Successful Operation*
	`SOS_STATUS_INVALID_STATE` in case that this function is already Initialized
	`SOS_STATUS_FAILED` in case happens any failer
Available via	sos_interface.h

sos_deInit

Service name	sos_deInit
Description	This Function De-Initialize timer
Syntax	`enu_system_status_t sos_deInit(void)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	`void`
Parameters (out)	None
Return	*SOS_STATUS_SUCCESS in case of Successful Operation*
	SOS_STATUS_INVALID_STATE in case that this function is already De-Initialized or not Initialized previously
	SOS_STATUS_FAILED in case happens any failer
Available via	sos_interface.h

sos_create_task

Service name	sos_create_task
Description	This Function create tasks
Syntax	`enu_system_status_t sos_create_task (str_task_t* ptr_str_task)`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	**ptr_str_task:** address of struct instance of task
Parameters (out)	None
Return	*SOS_STATUS_SUCCESS in case of Successful Operation*
	SOS_STATUS_FAILED in case happens any failer
Available via	sos_interface.h

sos_delete_task

Service name	sos_delete_task
Description	This Function delete tasks
Syntax	`enu_system_status_t sos_delete_task (str_task_t* ptr_str_task)`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	**ptr_str_task:** address of struct instance of task
Parameters (out)	None
Return	*SOS_STATUS_SUCCESS in case of Successful Operation*
	SOS_STATUS_FAILED in case happens any failer
Available via	sos_interface.h

sos_modify_task

Service name	sos_modify_task
Description	This Function modify tasks
Syntax	`enu_system_status_t sos_modify_task (str_task_t* ptr_src_str_task, str_task_t* ptr_mod_str_task)`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	**ptr_src_str_task:** address of struct instance of task to be modified **ptr_mod_str_task:** address of struct instance of modified task
Parameters (out)	None
Return	*SOS_STATUS_SUCCESS in case of Successful Operation*
	SOS_STATUS_FAILED in case happens any failer
Available via	sos_interface.h

sos_run

Service name	sos_run
Description	This Function start schedular
Syntax	`enu_system_status_t sos_run(void)`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	`void`
Parameters (out)	None
Return	*SOS_STATUS_SUCCESS in case of Successful Operation*
	SOS_STATUS_FAILED in case happens any failer
Available via	sos_interface.h

sos_disable

Service name	sos_disable
Description	This Function disable schedular
Syntax	`enu_system_status_t sos_disable(void)`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	`void`
Parameters (out)	None
Return	*`SOS_STATUS_SUCCESS` in case of Successful Operation*
	`SOS_STATUS_FAILED` in case happens any failer
Available via	sos_interface.h

HAL

HLED module

HLed_Init

Service name	HLed_Init
Description	This Function Init LED dio pin as output
Syntax	`enu_ledError_t HLed_Init (enu_pin en_pinNum)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**en_pinNum: dio pin selection**
Parameters (out)	None
Return	*LED_OK: in case of successful operation*
	LED_NOK: in case of failer operation
Available via	hled.h

HLed_on

Service name	HLed_on
Description	This Function give LED pin logic 1
Syntax	`enu_ledError_t HLed_on (enu_pin en_pinx);`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**en_pinNum: dio pin selection**
Parameters (out)	None
Return	*LED_OK: in case of successful operation*
	LED_NOK: in case of failer operation
Available via	hled.h

HLed_off

Service name	HLed_off
Description	This Function give LED pin logic 0
Syntax	`enu_ledError_t HLed_off (enu_pin en_pinx)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**en_pinNum: dio pin selection**
Parameters (out)	None
Return	*LED_OK: in case of successful operation*
	LED_NOK: in case of failer operation
Available via	hled.h

HLed_toggle

Service name	HLed_toggle
Description	This Function Change previous state of LED pin
Syntax	`enu_ledError_t HLed_toggle (enu_pin en_pinx)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**en_pinNum: dio pin selection**
Parameters (out)	None
Return	*LED_OK: in case of successful operation*
	LED_NOK: in case of failer operation
Available via	hled.h

Button module

HButton_Init

Service name	HButton_Init
Description	This Function Initialize button DIO pin as input and pull up
Syntax	`enu_buttonError_t HButton_Init (enu_pin en_pinx)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**en_pinx: DIO pin number**
Parameters (out)	None
Return	*BUTTON_OK: in case of successful operation*
	BUTTON_NOK: in case of failer operation
Available via	button.h

HButton_ExtIntInit

Service name	HButton_ExtIntInit
Description	This Function Initialize button as external interrupt
Syntax	`enu_buttonError_t HButton_ExtIntInit (enu_pin en_pinx)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**en_pinx: DIO pin number**
Parameters (out)	None
Return	*BUTTON_OK: in case of successful operation*

	BUTTON_NOK: in case of failer operation
Available via	button.h

HButton_getPinVal

Service name	**HButton_getPinVal**
Description	This Function Get button state
Syntax	`enu_buttonError_t HButton_getPinVal (enu_pin en_pinx, Uint8_t* pu8_refVal)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**en_pinx:** DIO pin number
Parameters (out)	**pu8_refVal:** address of variable which button state to be stored
Return	*BUTTON_OK: in case of successful operation*
	BUTTON_NOK: in case of failer operation
Available via	button.h

Timer module

enuTimer2_init

Service name	enuTimer2_init
Description	This Function Initialize Timer mode
Syntax	`enu_timerStatus_t enuTimer2_init (enu_timerMode_t enTimerMode)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**enTimerMode:** Timer mode selection
Parameters (out)	**None**
Return	*TIMER_OK: in case of successful operation*
	TIMER_NOK: in case of failer operation
Available via	timer_interface.h

u8Timer2_setPrescallar

Service name	u8Timer2_setPrescallar
Description	This Function set timer prescaller
Syntax	`enu_timerStatus_t u8Timer2_setPrescallar (enu_timerPrescalar_t Copy_enPrescal)`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	**Copy_enPrescal:** Timer prescaller selection
Parameters (out)	**None**
Return	*TIMER_OK: in case of successful operation*
	TIMER_NOK: in case of failer operation
Available via	timer_interface.h

vidTimer2_OvfIrqEnable

Service name	vidTimer2_OvfIrqEnable
Description	This Function enable timer interrupt
Syntax	`enu_timerStatus_t vidTimer2_OvfIrqEnable(void)`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	`void`
Parameters (out)	`None`
Return	*TIMER_OK: in case of successful operation*
	TIMER_NOK: in case of failer operation
Available via	timer_interface.h

vidTimer2_OvfIrqDisable

Service name	vidTimer2_OvfIrqDisable
Description	This Function disable timer interrupt
Syntax	`enu_timerStatus_t vidTimer2_OvfIrqDisable(void)`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	`void`
Parameters (out)	`None`
Return	*TIMER_OK: in case of successful operation*
	TIMER_NOK: in case of failer operation
Available via	timer_interface.h

vidTimer2_start

Service name	vidTimer2_start
Description	This Function start timer or start counter
Syntax	`enu_timerStatus_t vidTimer2_start(void)`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	`void`
Parameters (out)	`None`
Return	*TIMER_OK: in case of successful operation*
	TIMER_NOK: in case of failer operation
Available via	timer_interface.h

vidTimer2_stop

Service name	vidTimer2_stop
Description	This Function stop timer or stop counter
Syntax	`enu_timerStatus_t vidTimer2_stop(void)`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	`void`
Parameters (out)	`None`
Return	*TIMER_OK: in case of successful operation*
	TIMER_NOK: in case of failer operation
Available via	timer_interface.h

u8Timer2_setTime_ms

Service name	u8Timer2_setTime_ms
Description	This Function set time in milli second
Syntax	`enu_timerStatus_t u8Timer2_setTime_ms (Uint32_t u32_time_ms)`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	**u32_time_ms:** copy of time in milli second
Parameters (out)	**None**
Return	*TIMER_OK: in case of successful operation*
	TIMER_NOK: in case of failer operation
Available via	timer_interface.h

vidTimer2_setcbf_OVF

Service name	vidTimer2_setcbf_OVF
Description	This Function set call back function
Syntax	`void vidTimer2_setcbf_OVF (cbf_t cbf)`
Sync/Async	Asynchronous
Reentrancy	Non-Reentrant
Parameters (in)	**cbf:** copy of function address
Parameters (out)	**None**
Return	*TIMER_OK: in case of successful operation*
	TIMER_NOK: in case of failer operation
Available via	timer_interface.h

DIO module

DIO_s8SETPinDir

Service name	DIO_s8SETPinDir
Description	This Function Initialize Pin Direction Input or Output
Syntax	`Sint8_t DIO_s8SETPinDir (enu_pin enPinCopy, enu_dir enPortDir)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**enPinCopy:** Select pin and port [DIO_PINA_0,……] **enPortDir:** Select Pin direction [INPUT, OUTPUT]
Parameters (out)	None
Return	*DIO_OK: in case of successful operation*
	DIO_NOK: in case of failer operation
Available via	dio_Interface.h

DIO_s8SETPinVal

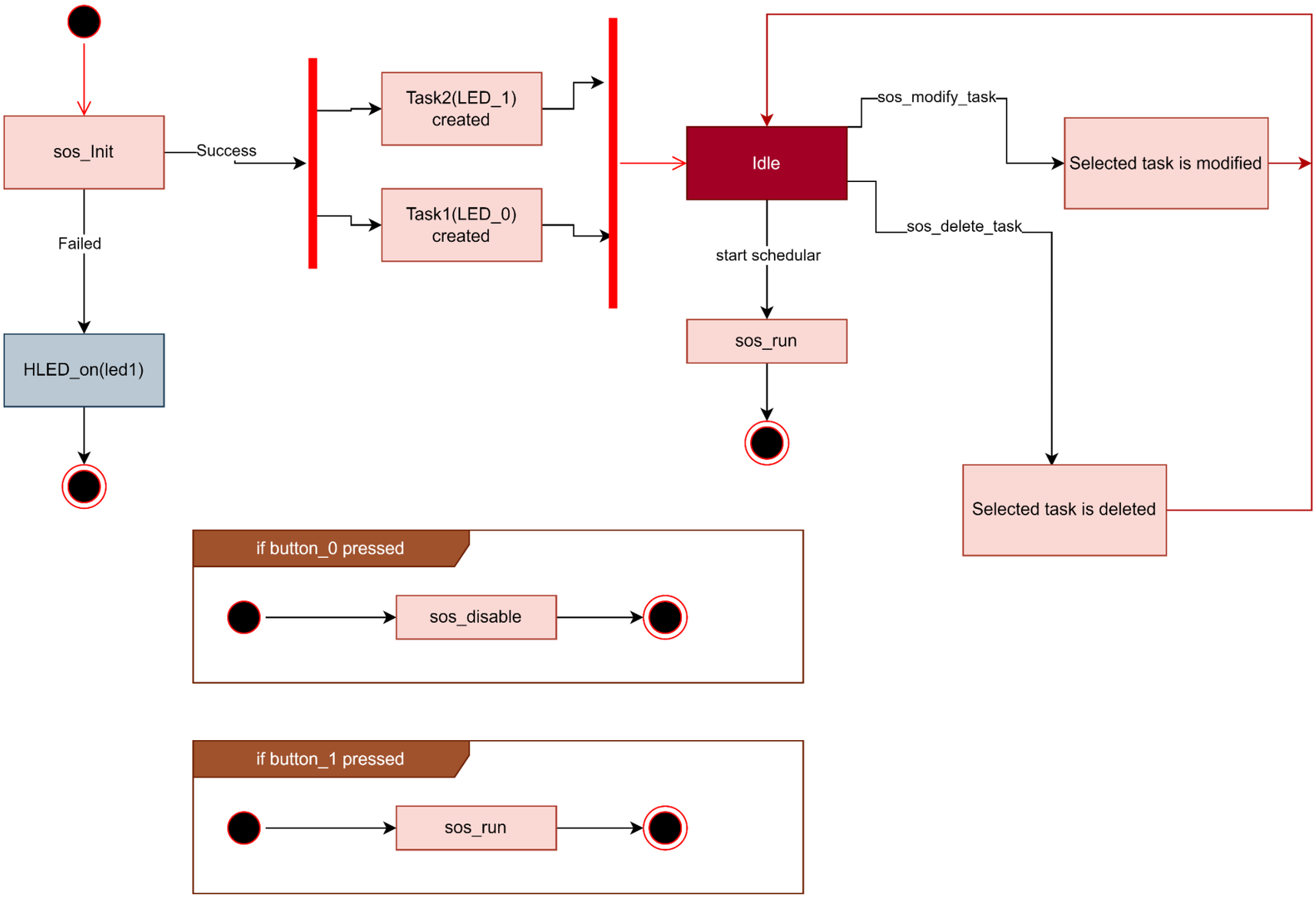
Service name	DIO_s8SETPinVal
Description	This Function Initialize Pin Value High or Low
Syntax	`Sint8_t DIO_s8SETPinVal (enu_pin enPinCopy, enu_dir enPortVal)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**enPinCopy:** Select pin and port [DIO_PINA_0,……] **enPortDir:** Select Pin Value [HIGH, LOW]
Parameters (out)	None
Return	*DIO_OK: in case of successful operation*
	DIO_NOK: in case of failer operation
Available via	dio_Interface.h

DIO_s8GETPinVal

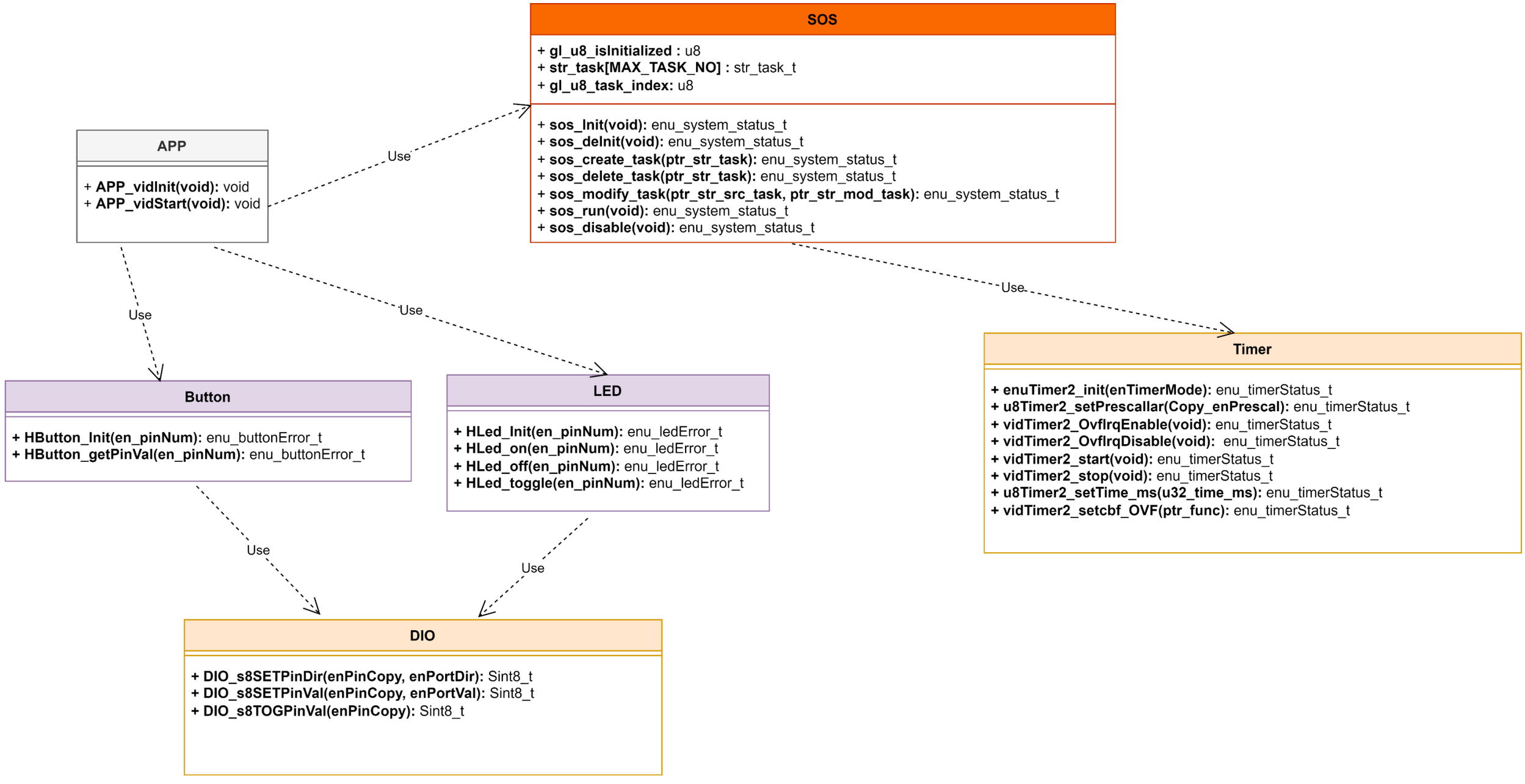
Service name	DIO_s8GETPinVal
Description	This Function Get value from selected pin
Syntax	`Sint8_t DIO_s8GETPinVal (enu_pin enPinCopy, Uint8_t* pu8Val)`
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	**enPinCopy:** Select pin and port [DIO_PINA_0,……]
Parameters (out)	**pu8Val:** Address of variable which pin status to be stored
Return	*DIO_OK: in case of successful operation*
	DIO_NOK: in case of failer operation
Available via	dio_Interface.h

UML

State Machine

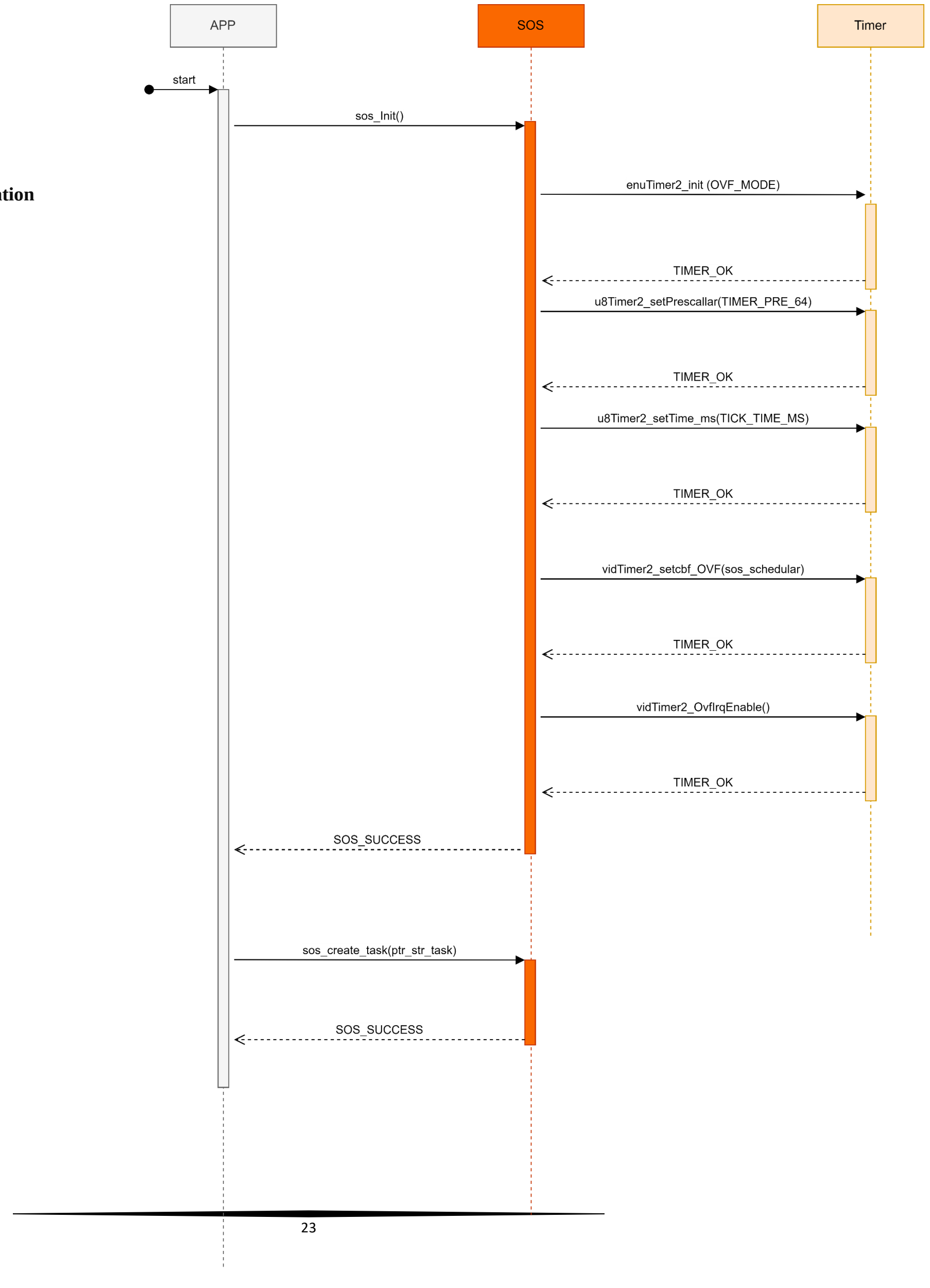


Class diagram

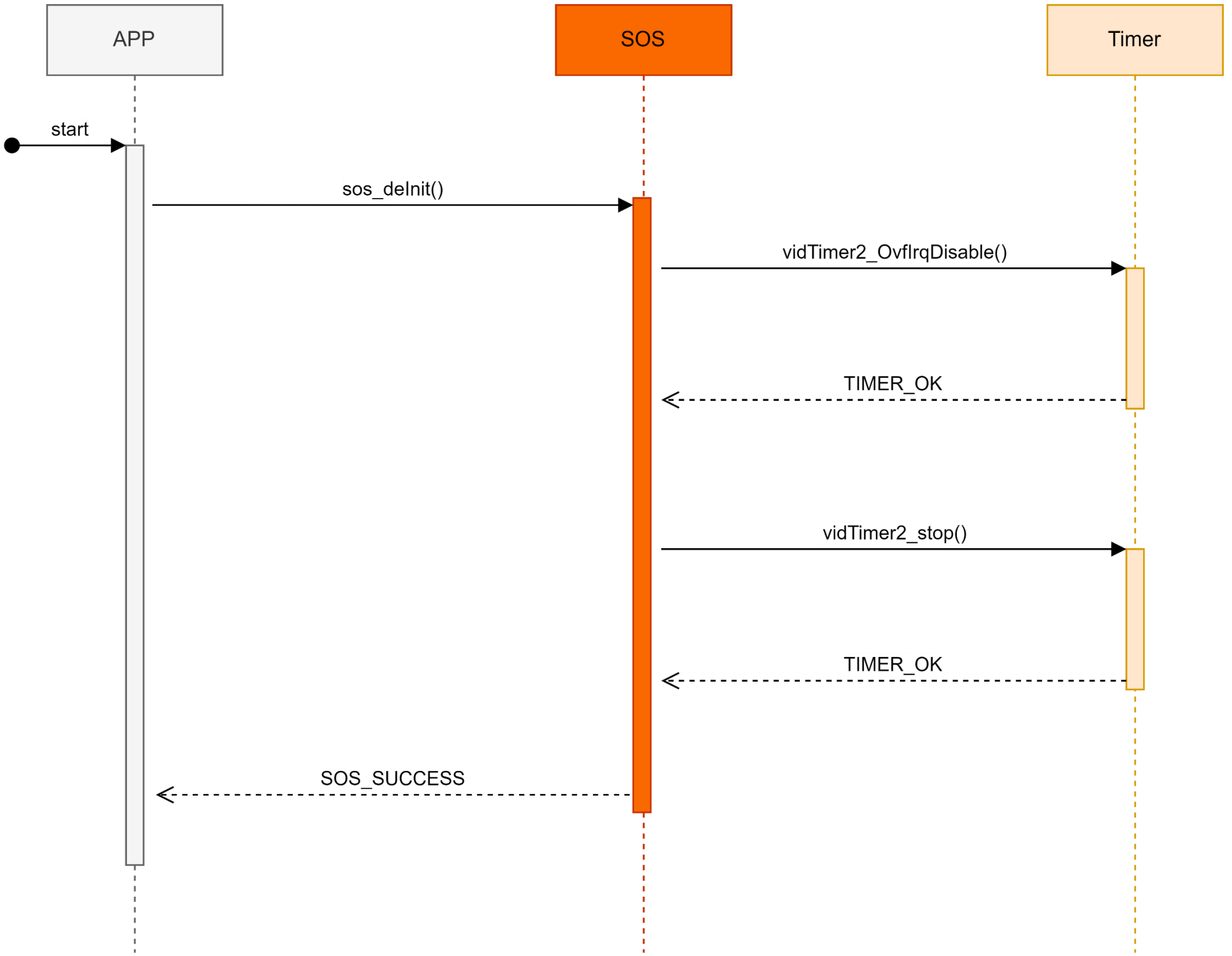


Sequence Diagram

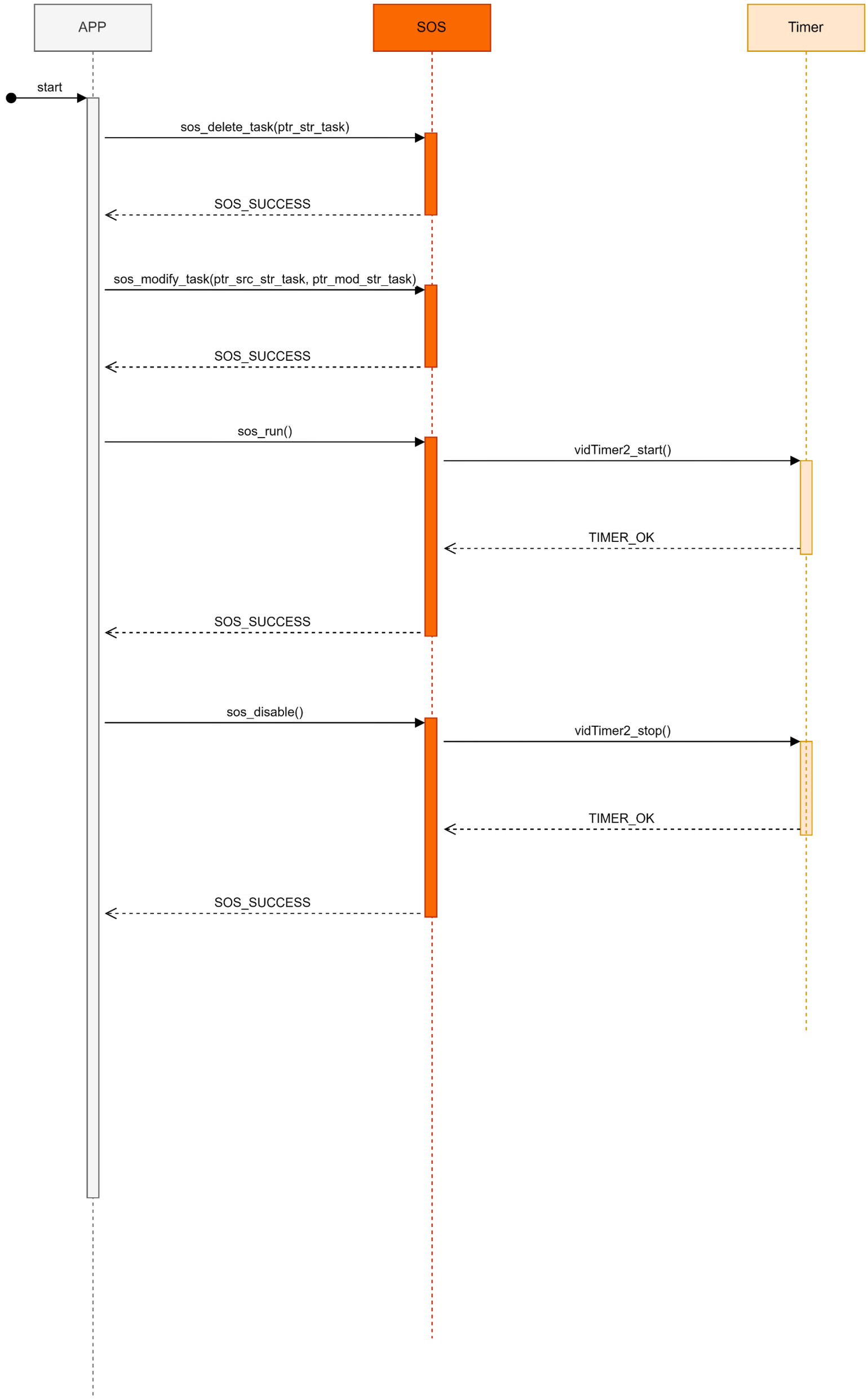
Task Initialization and Creation



Task De-Initialization



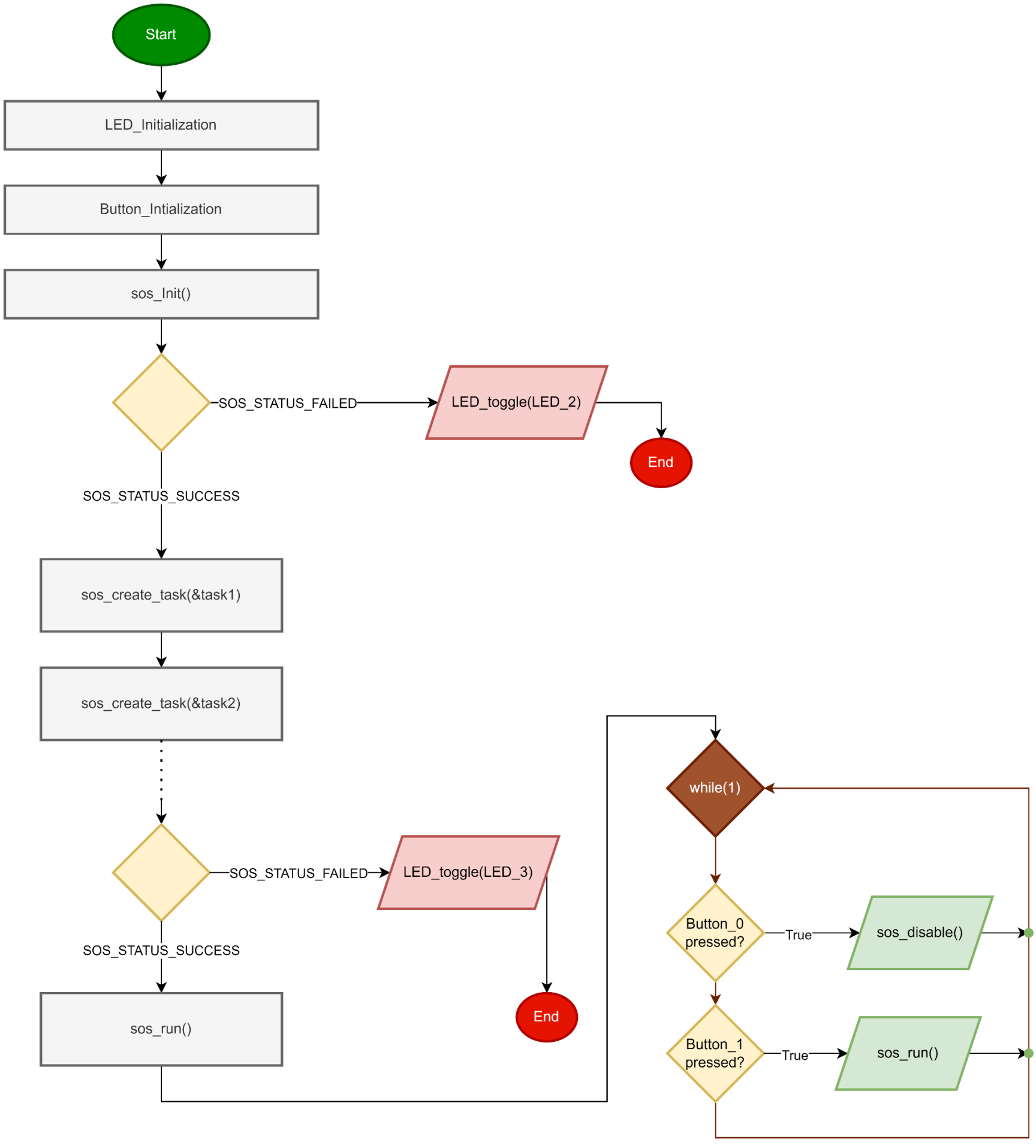
Task deletion, Task modification
SOS_run, SOS_disable Sequence diagram



Low Level Design

Flowchart

APP



Pre-compiling configuration

SERVICE

SOS module

MAX_TASK_NO

Name	MAX_TASK_NO
Type	MACRO
Range	Maximum task number which holded in buffer
Found in	sos_config.h

TICK_TIME_MS

Name	TICK_TIME_MS
Type	MACRO
Range	Tick time in milli second
Found in	sos_config.h

Linking Configuration

SERVICE

SOS module

str_task_t

Name	str_task_t
Type	Struct
Description	Struct containing configuration for each task
Configuration	*ptr_func*
	u8_task_id
	u8_task_priority
	u16_periodcty
	enu_taskMode
Found in	sos_Interface.h