Sprints

Small Operating System
Design Document

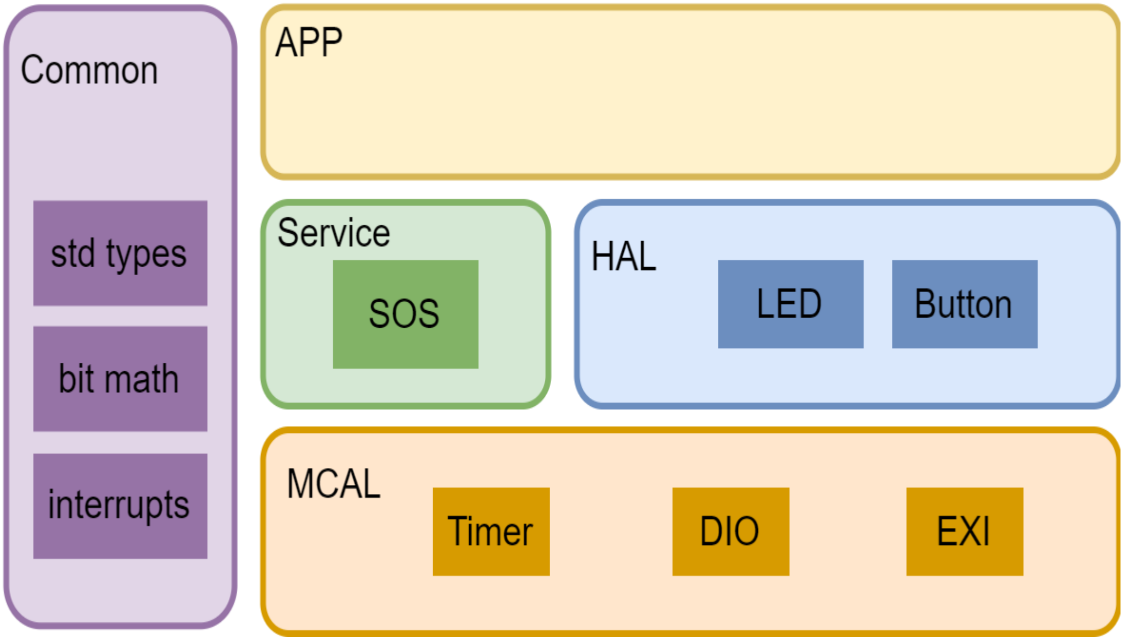
By:

Alaa Hisham

1. Project Introduction

2. High Level Design

2.1. Layered Architecture



2.2. Module Description

2.2.1. OS

2.2.1.1. SOS Module

A simple module that implements a minimal OS scheduler with a timer and no dispatcher. The OS is priority based and so it allows multiple tasks to execute concurrently by selecting which task to run next based on predefined task periodicity and priority.

2.2.2. HAL

2.2.2.1. LED Module

A simple module to interface with an LED through a number of APIs that provide functionalities like initializing the LED, turning it on/off or toggling it.

2.2.2.2. Button Module

A simple module to provide an interface with a push button. It is used to detect the user input by providing APIs for reading the button state and for debouncing the input signal.

2.2.3. MCAL

2.2.3.1. DIO Module

The DIO (Digital Input/Output) module controls the DIO peripheral to interface with different digital devices through APIs that provide functionalities to read, write and toggle different microcontroller pins.

2.2.3.2. EXI Module

This module allows the microcontroller to respond immediately to different external events on the external interrupt pins.

2.2.3.3. Timer Module

This module provides different timing mechanisms for development to create time based routines or control frequency of output signals through providing APIs that deal with different timer modes.

2.3. Driver Documentation

2.3.1. OS

SOS APIs

Function Name	***sos_init***
Syntax	enu_system_status_t_ sos_init (void);
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters	None
Return	*SOS_STATUS_SUCCESS:* initialized successfully
	SOS_STATUS_INVALID_STATE: sos already initialized

Function Name	***sos_deinit***
Syntax	enu_system_status_t_ sos_deinit (void);
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters	None
Return	*SOS_STATUS_SUCCESS:* deinitialized successfully
	SOS_STATUS_INVALID_STATE: sos already deinitialized or was not initialized before

Function Name	***sos_create_task***
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* pointer to task structure
Parameters (out)	None
Parameters (in, out)	None
Return	*SOS_STATUS_SUCCESS:* Task created successfully
	SOS_STATUS_REPEATED_PRIORITY: Given task priority is already assigned to another task
	SOS_STATUS_INVALID_TASK: when a null ptr is passed to function

Function Name	***sos_delete_task***
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* pointer to task to delete
Parameters (out)	None
Parameters (in, out)	None
Return	*SOS_STATUS_SUCCESS:* Task deleted successfully
	SOS_STATUS_INVALID_Task: passed null ptr
	SOS_TASK_DOES_NOT_EXIST: no task of given priority exists in DB

Function Name	***sos_modify_task***
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* Reference to task
Parameters (out)	None
Parameters (in, out)	None
Return	*SOS_STATUS_SUCCESS:* Task modified successfully
	SOS_STATUS_INVALID_Task: passed null ptr
	SOS_TASK_DOES_NOT_EXIST: no task of given priority exists in DB

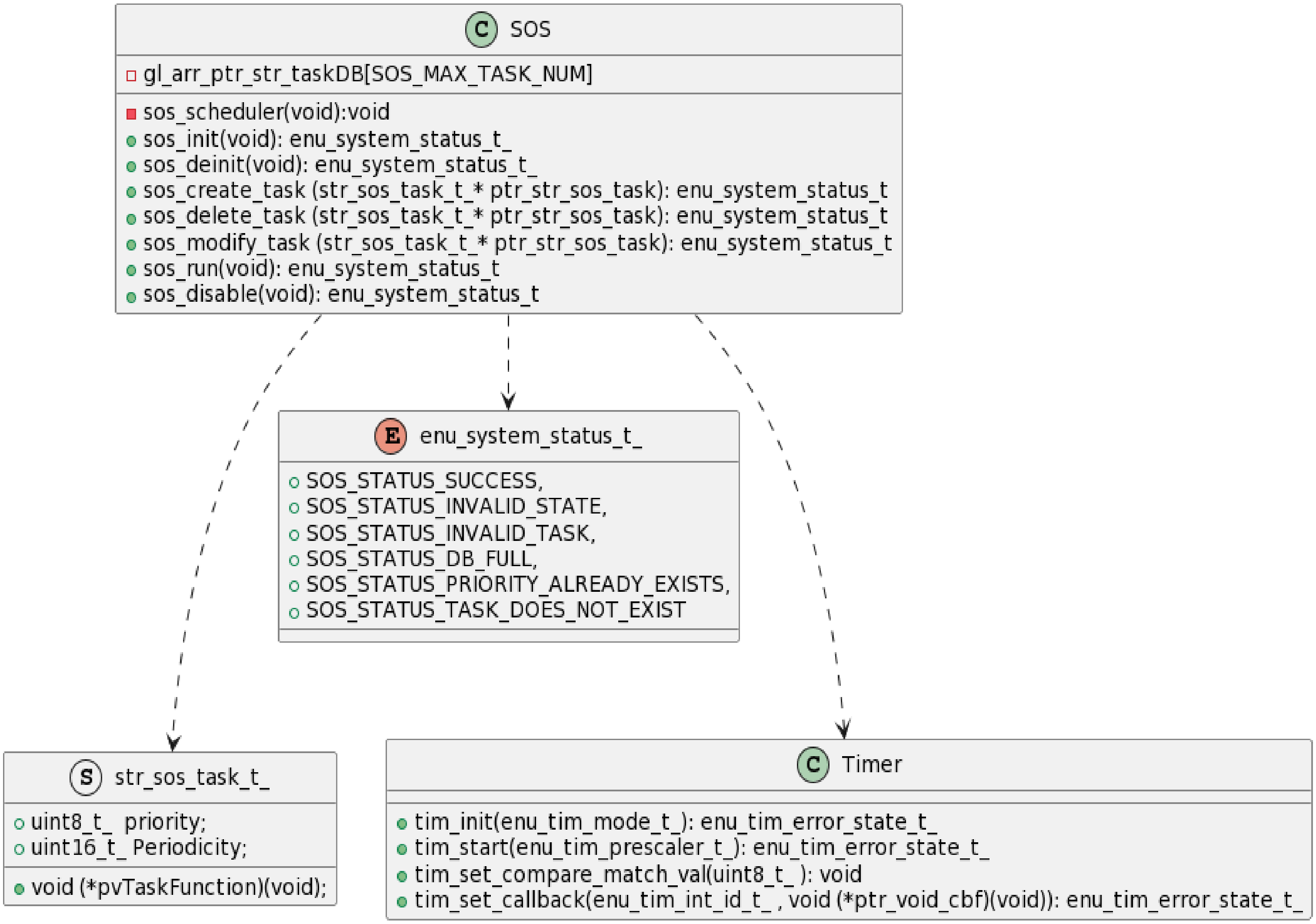
Function Name	***sos_run***
Syntax	enu_system_status_t_ sos_run(void);
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	None
Parameters (out)	None
Parameters (in, out)	None
Return	*SOS_STATUS_SUCCESS:* Successful Operation
	SOS_STATUS_INVALID_STATE: sos already running/not initialized

Function Name	***sos_disable***
Syntax	enu_system_status_t_ sos_disable(void);
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Parameters (in)	None
Parameters (out)	None
Parameters (in, out)	None
Return	*SOS_STATUS_SUCCESS:* Successful Operation
	SOS_STATUS_INVALID_STATE: sos already disabled/ not running

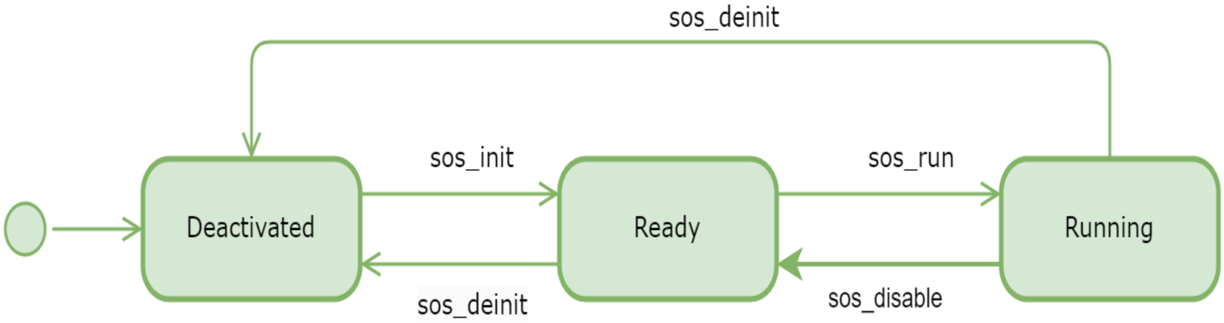
Function Name	***sos_scheduler***
Syntax	static void sos_scheduler(void);
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
None	None
Parameters (out)	None
Parameters (in, out)	None
Return	void

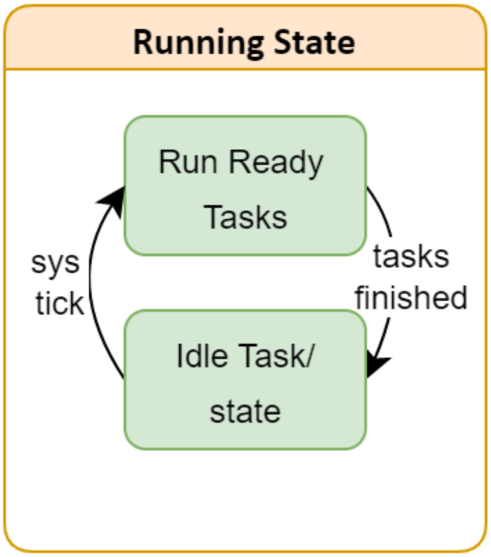
3. System Diagrams

3.1. SOS Module Class Diagram

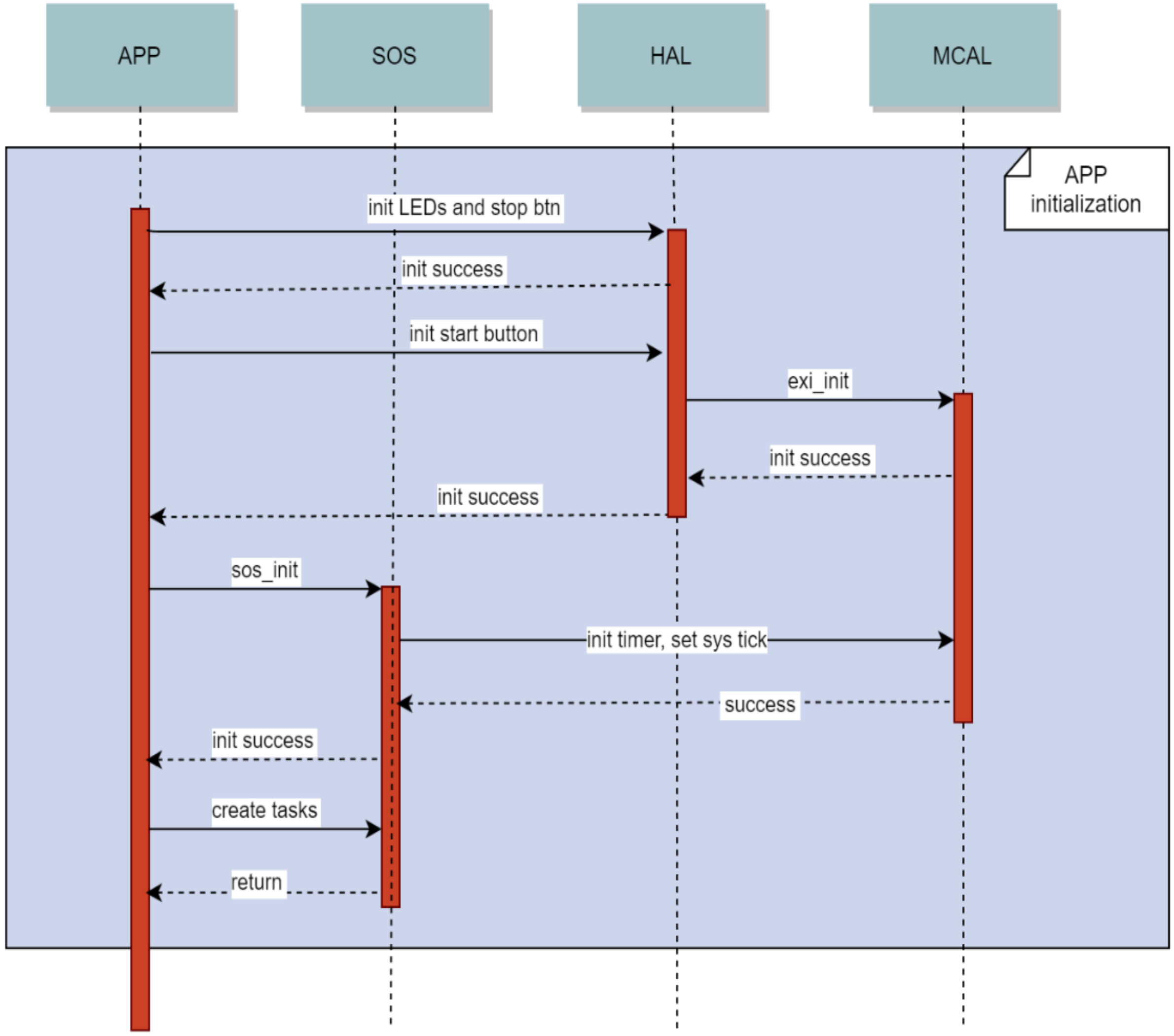

C SOS
gl_arr_ptr_str_taskDB[SOS_MAX_TASK_NUM]
sos_scheduler(void):void
sos_init(void): enu_system_status_t_
sos_deinit(void): enu_system_status_t_
sos_create_task (str_sos_task_t_* ptr_str_sos_task): enu_system_status_t
sos_delete_task (str_sos_task_t_* ptr_str_sos_task): enu_system_status_t
sos_modify_task (str_sos_task_t_* ptr_str_sos_task): enu_system_status_t
sos_run(void): enu_system_status_t
sos_disable(void): enu_system_status_t
E enu_system_status_t_
SOS_STATUS_SUCCESS,
SOS_STATUS_INVALID_STATE,
SOS_STATUS_INVALID_TASK,
SOS_STATUS_DB_FULL,
SOS_STATUS_PRIORITY_ALREADY_EXISTS,
SOS_STATUS_TASK_DOES_NOT_EXIST
S str_sos_task_t_
uint8_t_ priority;
uint16_t_ Periodicity;
void (*pvTaskFunction)(void);
C Timer
tim_init(enu_tim_mode_t_): enu_tim_error_state_t_
tim_start(enu_tim_prescaler_t_): enu_tim_error_state_t_
tim_set_compare_match_val(uint8_t_): void
tim_set_callback(enu_tim_int_id_t_ , void (*ptr_void_cbf)(void)): enu_tim_error_state_t_

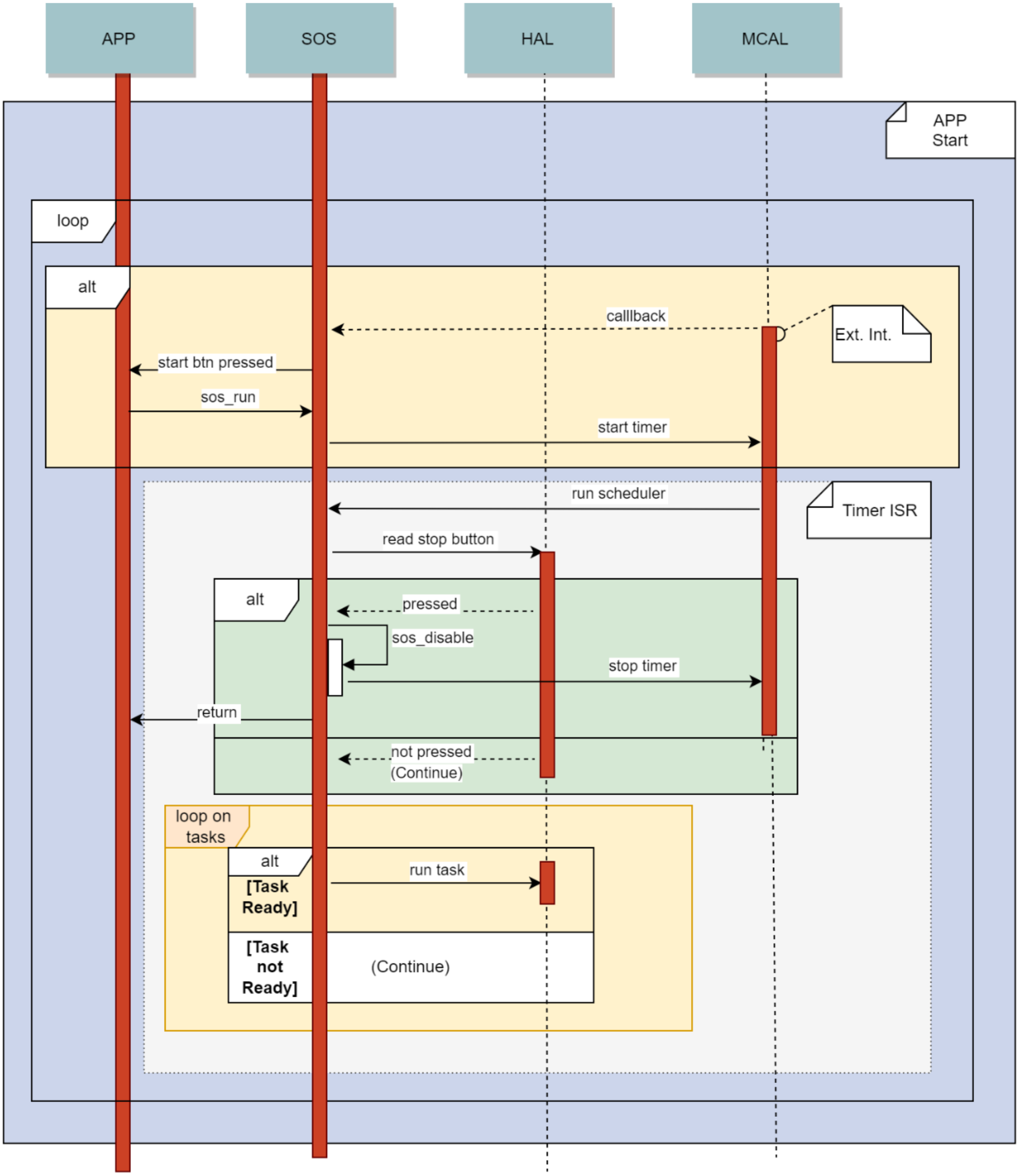
3.2. SOS Module State Machine





3.3. Application Sequence Diagram




APP
SOS
HAL
MCAL
APP Start
loop
alt
calllback
Ext. Int.
start btn pressed
sos_run
start timer
run scheduler
Timer ISR
read stop button
alt
pressed
sos_disable
stop timer
return
not pressed
(Continue)
loop on tasks
alt
run task
[Task Ready]
[Task not Ready]
(Continue)