SOS

By.

Ahmed Atef

Sprints.ai

0

Project Introduction 2

Components 2

Main Application Flow 2

High Level Design 3

Layered Architecture 3

- Application Layer: 3
- SERVICE Layer: 3
- HAL Layer: 3
- MCAL Layer: 3
- Microcontroller: 3

Sequence Diagram (FOR HQ ->S.D) 4

State Machine Diagram 5

Class Diagram 6

Drivers' Documentation (APIs) 7

Project Introduction

the project delivers a small operating system that has a scheduler to do system tasks depending on their priority with defined Periodicity

Components

- ATmega32 microcontroller
- 2 LEDS
- 2 Buttons

Main Application Flow

Implement an application that calls the SOS module and use 2 tasks -

Task 1: Toggle LED_0 (Every 300 Milli-Seconds)

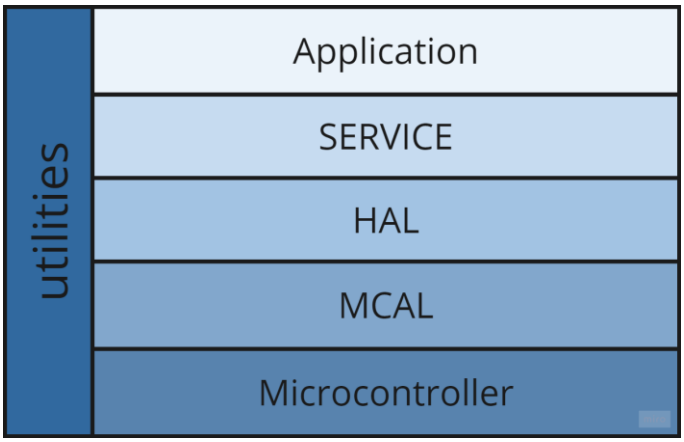
Task 2: Toggle LED_1 (Every 500 Milli-Seconds)

Make sure that these tasks occur periodically and forever

When pressing PBUTTON0, the SOS will stop

When Pressing PBUTTON1, the SOS will run

High Level Design



Layered Architecture

- ### Application Layer:

This is the topmost layer of the software stack, which contains the actual application logic. It interacts with the lower layers to perform its tasks. It is responsible for implementing the desired functionality of the system.

- ### SERVICE Layer:

This layer plays a key role in abstracting and encapsulating the low-level hardware details provided by the MCAL. It provides a set of services, functions, and interfaces that enable the application layer to interact with the underlying hardware

- ### HAL Layer:

This layer provides an abstraction for external devices connected to the microcontroller. The HAL layer provides interface to access external devices and hides the implementation details from the application layer.

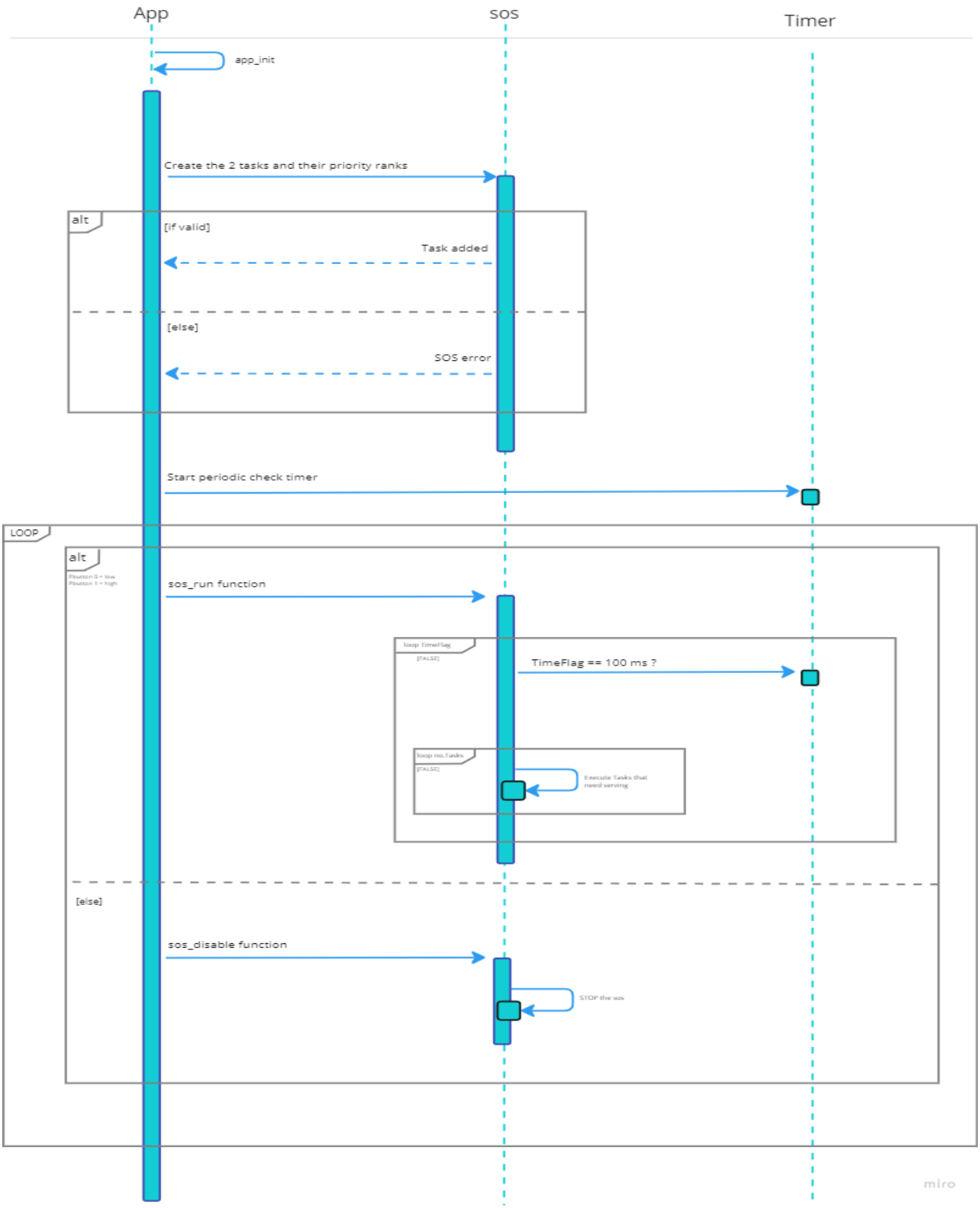
- ### MCAL Layer:

(Microcontroller Abstraction Layer): This layer provides an abstraction for the microcontroller hardware. It includes low-level drivers for peripherals. It hides the hardware details and provides a uniform interface to the upper layers.

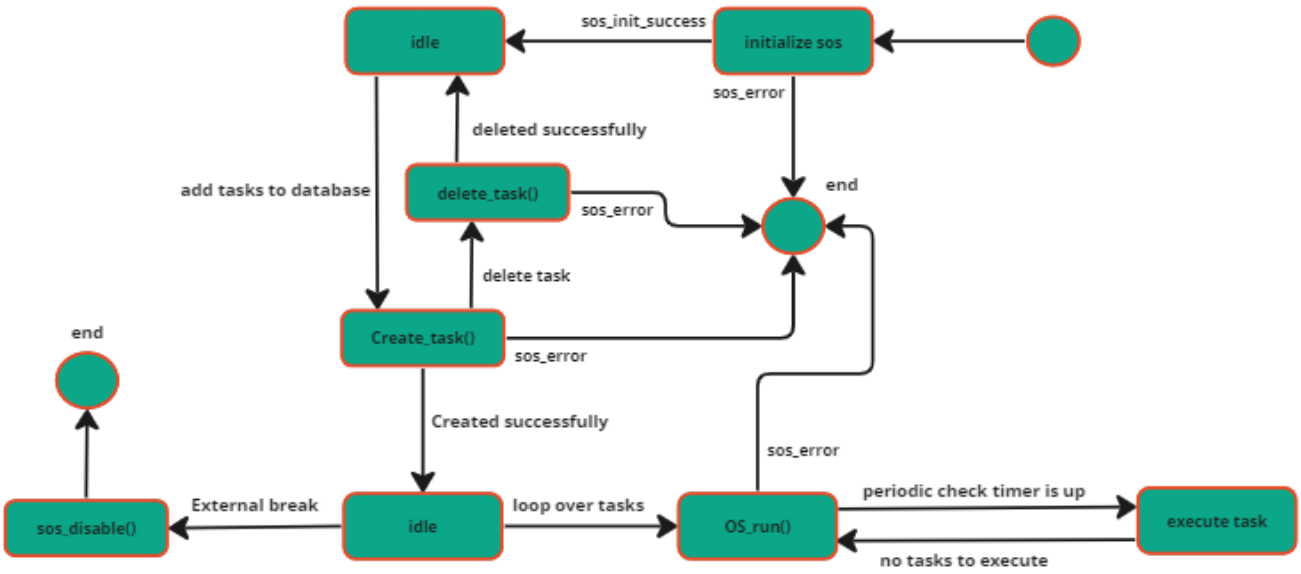
- ### Microcontroller:

This layer represents the physical hardware layer consisting of the microcontroller chip. The microcontroller is responsible for executing the code stored in its memory and controlling the behavior of the system.

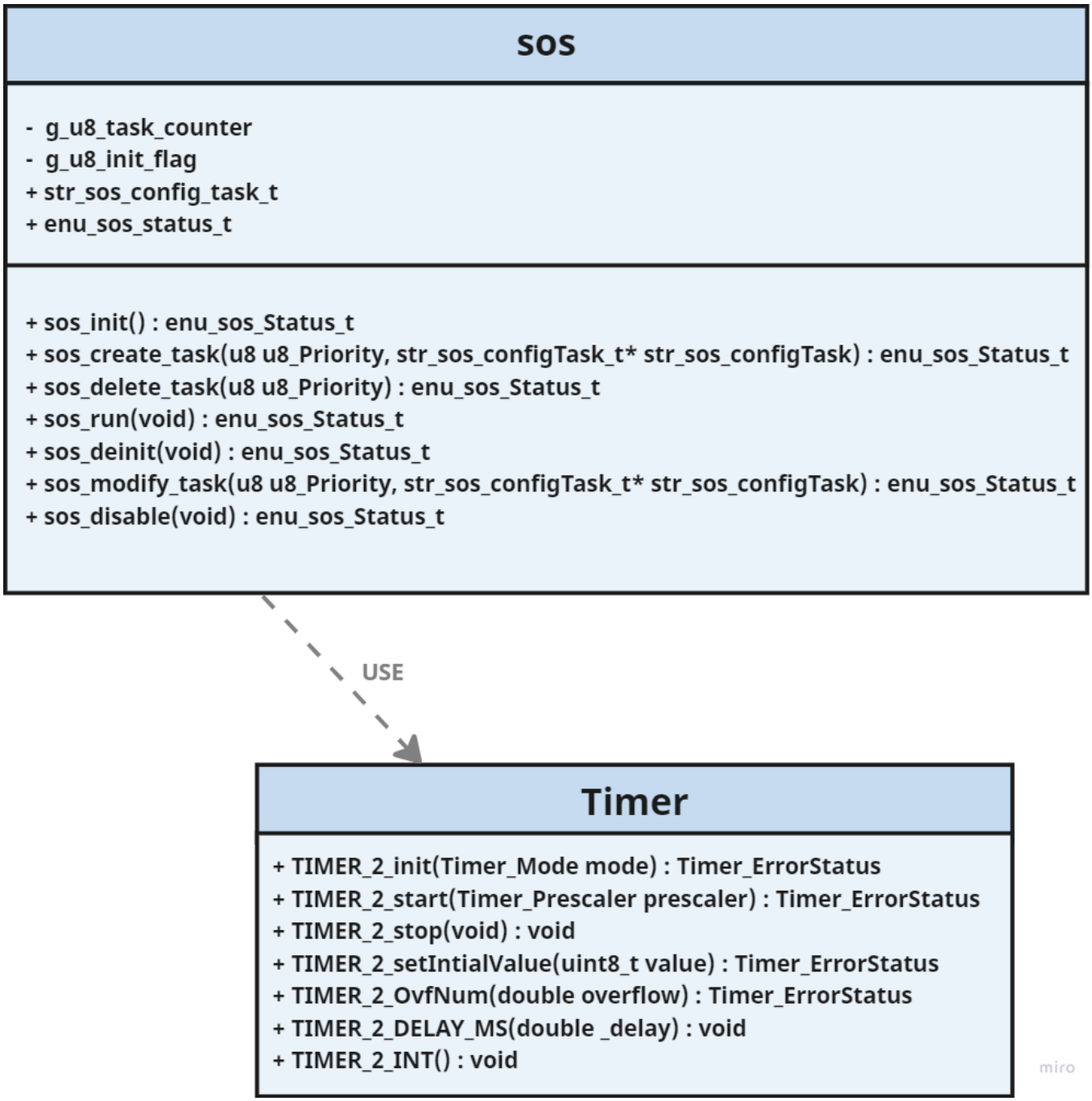
Sequence Diagram (FOR HQ ->S.D)



State Machine Diagram



Class Diagram



Drivers' Documentation (APIs)

```
typedef struct
{
	/* Selecting the periodicity of the selected task */
	u8 u8Periodicity;
	/* Assigning the pointer to the task function */
	void (*pfTask) (void);
 } strOSConfigTask_t;
```

```
typedef Enum{
	OS_OK,
	OS_ERROR,
	PRIORITY_EMPTY,
	PRIORITY_FULL,
	OS_INIT,
	OS_TASK_ADDED,
	OS_TASK_DELETED

      } enuOSErrorStatus_t;
```

Syntax	`enuOSErrorStatus_t OS_Init ()`
Description	A function to initialize the timer used in OS
Sync\Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	void
Parameters (out)	None

Return value	`typedef Enum{`
	`OS_OK,`
	`OS_ERROR,`
	`PRIORITY_EMPTY,`
	`PRIORITY_FULL,`
	`OS_INIT,`
	`OS_TASK_ADDED,`
	`OS_TASK_DELETED`
	`} enuOSErrorStatus_t;`

Syntax	`enuOSErrorStatus_t OS_CreateTask (u8 u8Priority, strOSConfigTask_t* strOSConfigTask_t)`
Description	A function to create a certain task
Sync\Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	*strOSConfigTask_t
Parameters (out)	None

Return value	`typedef Enum{`
	`OS_OK,`
	`OS_ERROR,`
	`PRIORITY_EMPTY,`
	`PRIORITY_FULL,`
	`OS_INIT,`
	`OS_TASK_ADDED,`
	`OS_TASK_DELETED`
	
	`} enuOSErrorStatus_t;`

Syntax	`enuOSErrorStatus_t OS_DeleteTask (u8 u8Priority)`
Description	`A function to delete a certain task`
Sync\Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	`*strOSConfigTask_t`
Parameters (out)	None

Return value	`typedef Enum{`
	`OS_OK,`
	`OS_ERROR,`
	`PRIORITY_EMPTY,`
	`PRIORITY_FULL,`
	`OS_INIT,`
	`OS_TASK_ADDED,`
	`OS_TASK_DELETED`
	`} enuOSErrorStatus_t;`

Syntax	`enuOSErrorStatus_t OS_modify_task (u8 u8Priority, strOSConfigTask_t* strOSConfigTask_t)`
Description	`A function to modify a certain task`
Sync\Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	`*strOSConfigTask_t`
Parameters (out)	None

Return value	`typedef Enum{`
	`OS_OK,`
	`OS_ERROR,`
	`PRIORITY_EMPTY,`
	`PRIORITY_FULL,`
	`OS_INIT,`
	`OS_TASK_ADDED,`
	`OS_TASK_DELETED`
	`} enuOSErrorStatus_t;`

Syntax	`void OS_Run(void)`
Description	`A function to run the OS`
Sync\Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	void
Parameters (out)	None
Return value	void

Syntax	`void OS_disable ()`
Description	`A function to disable running of OS`
Sync\Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	void
Parameters (out)	None
Return value	void