# Small Operating System

With Preemptive Priority Based Schedular

V1.0

Sharpel Malak
Sprints.ai

# Table of Content

# Introduction

**A** small operating system with a priority based preemptive scheduler based on time-triggered.

## Detailed Requirements

### Specifications

| sos_init function, this function will initialize the SOS database. | |
|---|---|
| **Function Name** | sos_init |
| **Syntax** | enu_system_status_t sos_init (void) |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non Reentrant |
| **Parameters (in):** | None |
| **Parameters (out):** | None |
| **Parameters (in, out):** | None |
| **Return:** | SOS_STATUS_SUCCESS: In case of Successful Operation. |
| | SOS_STATUS_INVALID_STATE: In case The SOS is already Initialized. |

| sos_deinit function, this function will reset the SOS database to invalid values | |
|---|---|
| **Function Name** | sos_deinit |
| **Syntax** | enu_system_status_t sos_deinit (void) |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non Reentrant |
| **Parameters (in):** | None |
| **Parameters (out):** | None |
| **Parameters (in, out):** | None |
| **Return:** | SOS_STATUS_SUCCESS: In case of Successful Operation. |
| | SOS_STATUS_INVALID_STATE: In case<br>The SOS is already De-Initialized or was not initialized previously |

| sos_create_task API, this API will create a new task and add it to the SOS database | |
|---|---|
| **Function Name** | sos_create_task |
| **Syntax** | enu_system_status_t sos_create_task(enu_task_priority_id_t enu_task_priority_id,str_tasks_config_t *str_tasks_config |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non Reentrant |
| **Parameters (in):** | enu_task_priority_id : Allocate task in order based on priority Id<br>str_tasks_config : Holds all task info(periodicity,referance,args) |
| **Parameters (out):** | None |
| **Parameters (in, out):** | None |
| **Return:** | SOS_STATUS_SUCCESS: In case of Successful Operation. |
| | SOS_NULL_ARGS: In case of Null poiters |
| | SOS_TASK_PERIODICITY_UNKNOWN : case undefined periodicity |
| | SOS_TASK_DUBLICATED_PIRIORITY : case of duplicated priority |

| **sos_delete_task API, this API will delete an existing task from the SOS database** | |
|---|---|
| **Function Name** | sos_delete_task |
| **Syntax** | enu_system_status_t sos_delete_task(enu_task_priority_id_t enu_task_priority_id) |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non Reentrant |
| **Parameters (in):** | enu_task_priority_id  : search for task in database |
| **Parameters (out):** | None |
| **Parameters (in, out):** | None |
| **Return:** | SOS_STATUS_SUCCESS: In case of Successful Operation. |
| | SOS_TASK_PRIRORITY_ERROR: In case of wrong priority id |
| | SOS_TASK_NOT_FOUND : in case of not found task |

| **sos_modify_task API, this API will modify existing task parameters in the SOS database** | |
|---|---|
| **Function Name** | sos_modify_task |
| **Syntax** | enu_system_status_t sos_modify_task(enu_task_priority_id_t enu_task_priority_id,str_tasks_config_t *str_tasks_config) |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non Reentrant |
| **Parameters (in):** | enu_task_priority_id : search for task in database<br>str_tasks_config : Holds all task info(periodicity,referance,args) |
| **Parameters (out):** | None |
| **Parameters (in, out):** | None |
| **Return:** | SOS_STATUS_SUCCESS: In case of Successful Operation. |
| | SOS_NULL_ARGS: In case of Null poiters |
| | SOS_TASK_PERIODICITY_UNKNOWN : case undefined periodicity |
| | SOS_TASK_NOT_FOUND : in case of not found task |

| **sos_run API, this API will run the small scheduler** | |
|---|---|
| **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_NO_TASKS_TO_RUN: In case of Empty Database. |

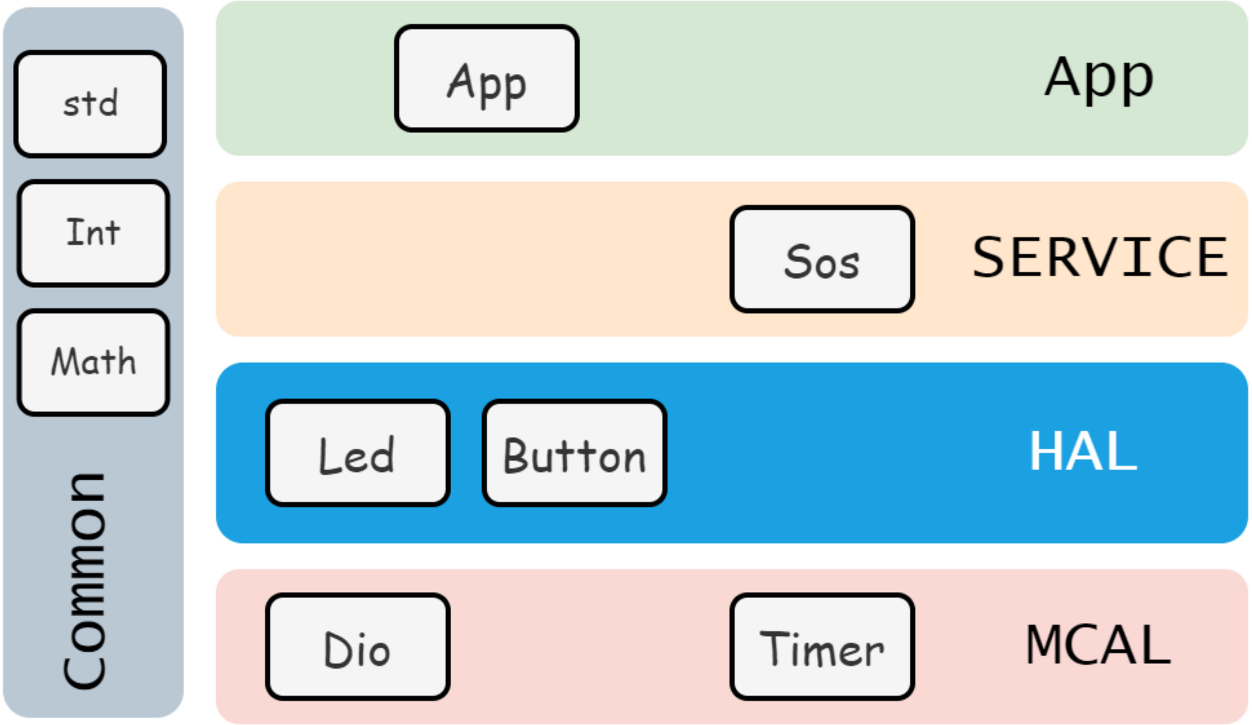
| sos_disable API, this API will stop the scheduler | |
|---|---|
| **Function Name** | sos_disable |
| **Syntax** | enu_system_status_t sos_disable(void) |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non Reentrant |
| **Parameters (in):** | None |
| **Parameters (out):** | None |
| **Parameters (in, out):** | None |
| **Return:** | SOS_STATUS_SUCCESS: In case of Successful Operation. |

### Main Application Flow

- Implement an application that calls the SOS module and use 4 tasks
  - Task 1: Toggle LED_0 (Every 3 Milli-Seconds)
  - Task 2: Toggle LED_1 (Every 5 Milli-Seconds)
- Make sure that these tasks occur periodically and forever
- When pressing **PBUTTON0**, the SOS will stop **(stop task)**
- When Pressing **PBUTTON1**, the SOS will run **(start task)**

## High Level Design

### Layered architecture



## Modules Descriptions

- Dio

  Stands for Digital Input/Output. It is an interface component that allows the system to send digital signals to devices. Also read signals from others

- Timer

  A timer is a specialized type of clock used for measuring specific time intervals

- Led

  This Module Controls Leds state in the program

- Button

  The push button module allows detection in states of high or low from the onboard momentary push button

- Sos

  Small operating  system that  manages all Application processes.

- App

  Contain Main application Logic

## Drivers' documentation

- **Dio**

  Description : This function initialize PIN and set it's direction

  ARGS : take PIN Number and PORT Number and Direction (INPUT,OUTPUT)

  return : return DIO_OK if the PIN initializes correctly, DIO_NOT_OK otherwise

  EN_DIO_ERROR DIO_init(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber,EN_DIO_DIRECTION direction);

  Description : This function write on PIN and set it's level

  ARGS : take PIN Number and PORT Number and level (LOW,HIGH)

  return : return DIO_OK if the PIN level sets correctly, DIO_NOT_OK otherwise

  EN_DIO_ERROR DIO_write(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber,EN_DIO_LEVEL level);

Description : This function toggles PIN level

ARGS : take PIN Number and PORT Number

return : return DIO_OK if the PIN toggles correctly, DIO_NOT_OK otherwise

EN_DIO_ERROR DIO_toggle(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber);

Description : This function reads PIN level and store it in the variable

ARGS : take PIN Number and PORT Number and pointer to the variable

return : return DIO_OK if the PIN value stored correctly , DIO_NOT_OK otherwise

EN_DIO_ERROR DIO_read(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber, uint8_t * value);

- **Timer**

Description : This function initialize Timer 1 with CTC mode and enable interrupts

ARGS : void

return : void

void TIMER_ONE_init(void);

Description : This function starts Timer 1 with configured prescaller

ARGS : void

return : void

void TIMER_ONE_start(void);

Description : This function stops Timer 1

ARGS : void

return : void

void TIMER_ONE_stop(void);

Description : This function calculate number of ticks to achieve desired time and assign the value in compare register

ARGS : time in milliseconds

return : void

void TIMER_ONE_setDelay(uint16_t delay_ms);

Description : This function set call Back when ISR fired the call back function executes

ARGS : pointer to call back function

return : void

void TIMER_ONE_setCallBack(void(*ptr_func)(void));

- **Led**

Description : This function inits led as output

ARGS : pointer to struct (pin/port)

return : return LED_OK if the Led initialized correctly , LED_NOT_OKAY otherwise

enu_led_error_t LED_init(str_led_config_t *str_ptr_led_config);

Description : This function sent High to pin

ARGS : pointer to struct (pin/port)

return : return LED_OK if the Led turns high correctly , LED_NOT_OKAY otherwise

enu_led_error_t LED_on(str_led_config_t *str_ptr_led_config);

Description : This function sent Low to pin

ARGS : pointer to struct (pin/port)

return : return LED_OK if the Led turns Low correctly , LED_NOT_OKAY otherwise

enu_led_error_t LED_off(str_led_config_t *str_ptr_led_config);

Description : This function toggle pin state

ARGS : pointer to struct (pin/port)

return : return LED_OK if the Led toggled correctly , LED_NOT_OKAY otherwise

enu_led_error_t LED_toggle(str_led_config_t *str_ptr_led_config);

- **Button**

Description : This function initialize PIN and set it's direction as Input

ARGS : take PIN Number and PORT Number

return : return BTN_OK if the PIN initializes correctly, BTN_NOT_OK otherwise

EN_BTN_Error_t Button_init(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber);

Description : This function Read PIN value and store it in variable

ARGS : take PIN Number and PORT Number and the address of the variable

return : return BTN_OK if the PIN read correctly, BTN_NOT_OK otherwise

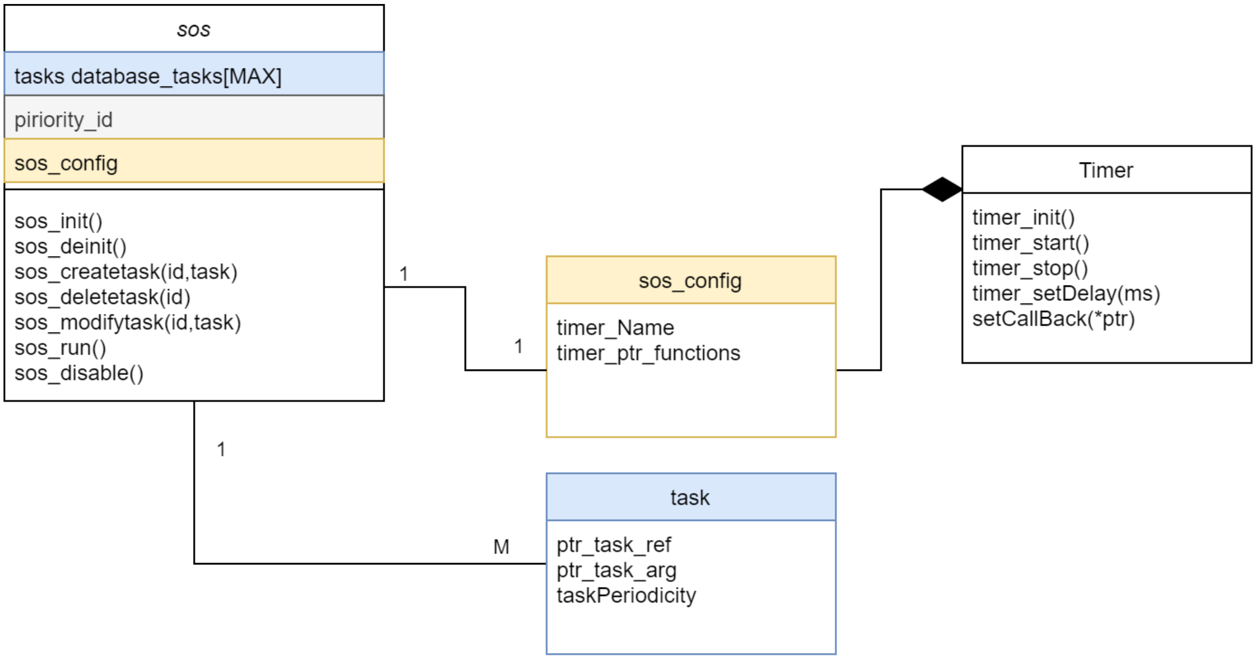
EN_BTN_Error_t Button_read(EN_DIO_PINS pinNumber,EN_DIO_PORTS portNumber,uint8_t *value);

- Sos

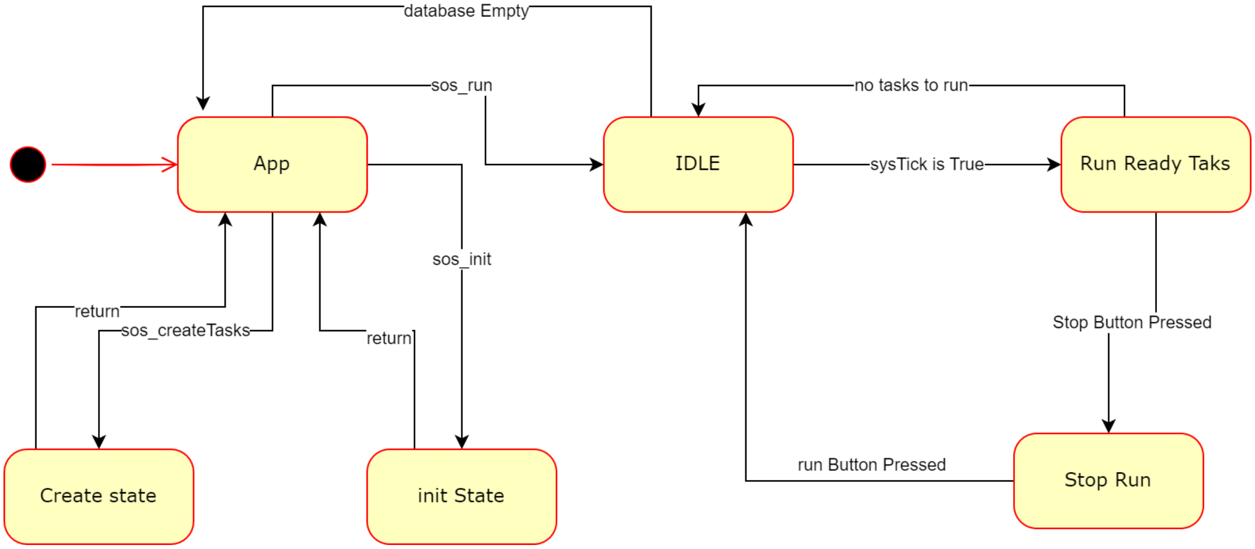
Go to Specifications section.

# UML

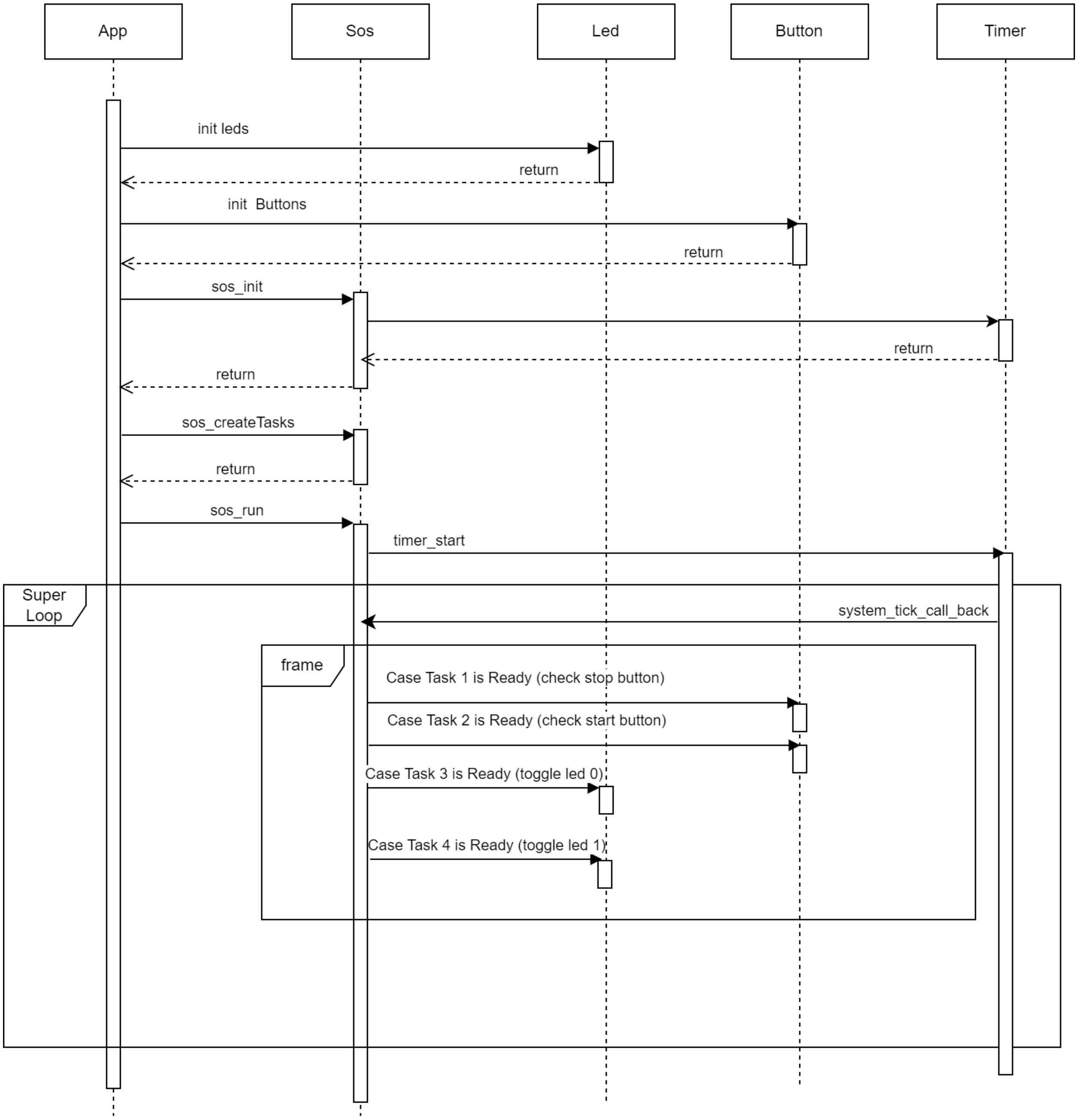
## Sos Class Diagram



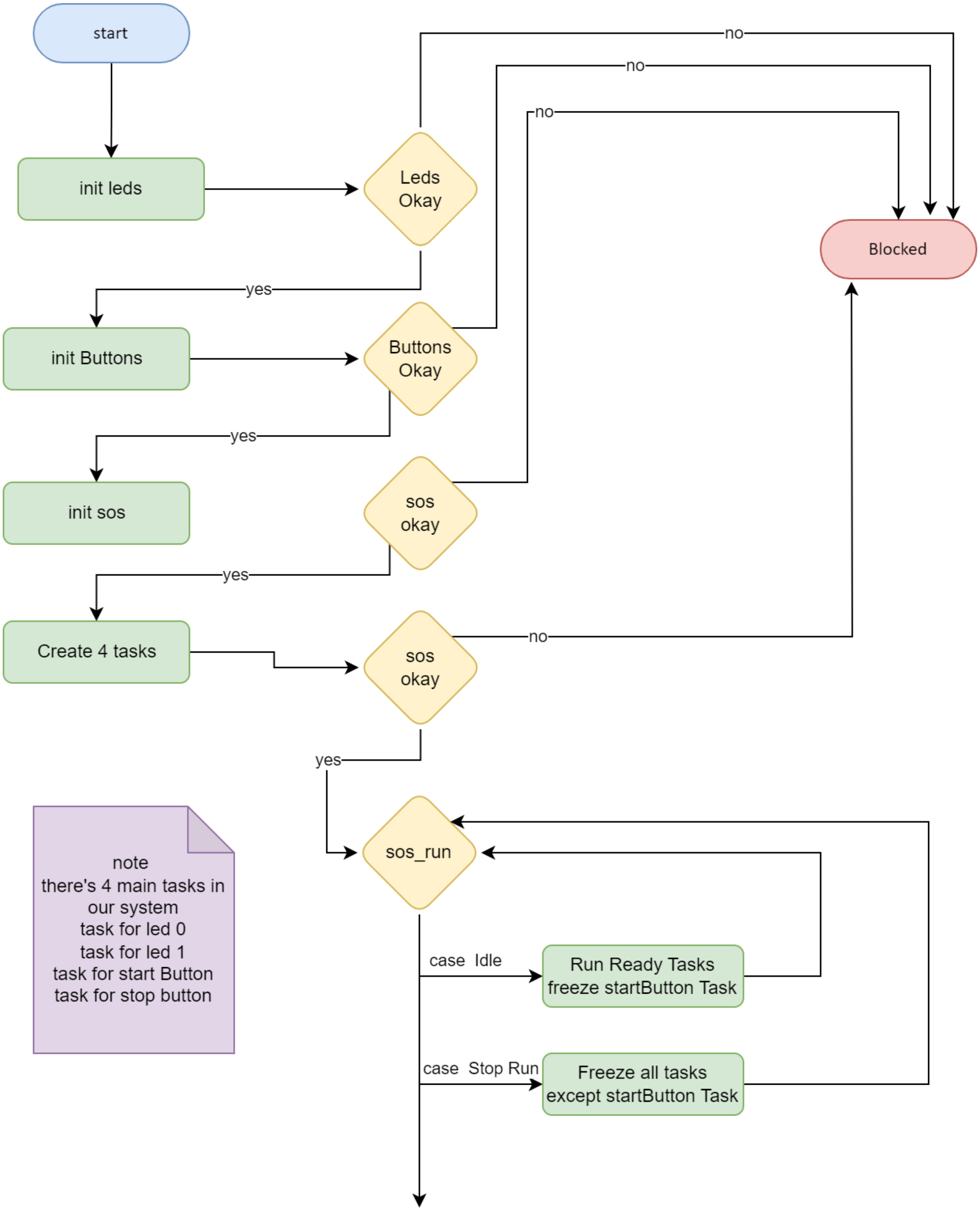
## Sos State Machine



## Sequence diagram


App
Sos
Led
Button
Timer
init leds
return
init  Buttons
return
sos_init
return
return
sos_createTasks
return
sos_run
timer_start
Super Loop
system_tick_call_back
frame
Case Task 1 is Ready (check stop button)
Case Task 2 is Ready (check start button)
Case Task 3 is Ready (toggle led 0)
Case Task 4 is Ready (toggle led 1)

## App Flowchart



## To Be Done ( Low Level Design )

- Flowcharts
- Pre-compiling configurations
- Linking configurations