SOS Module

DESIGN DOCUMENT

ASSINED BY:

MOHAMED SAYAD

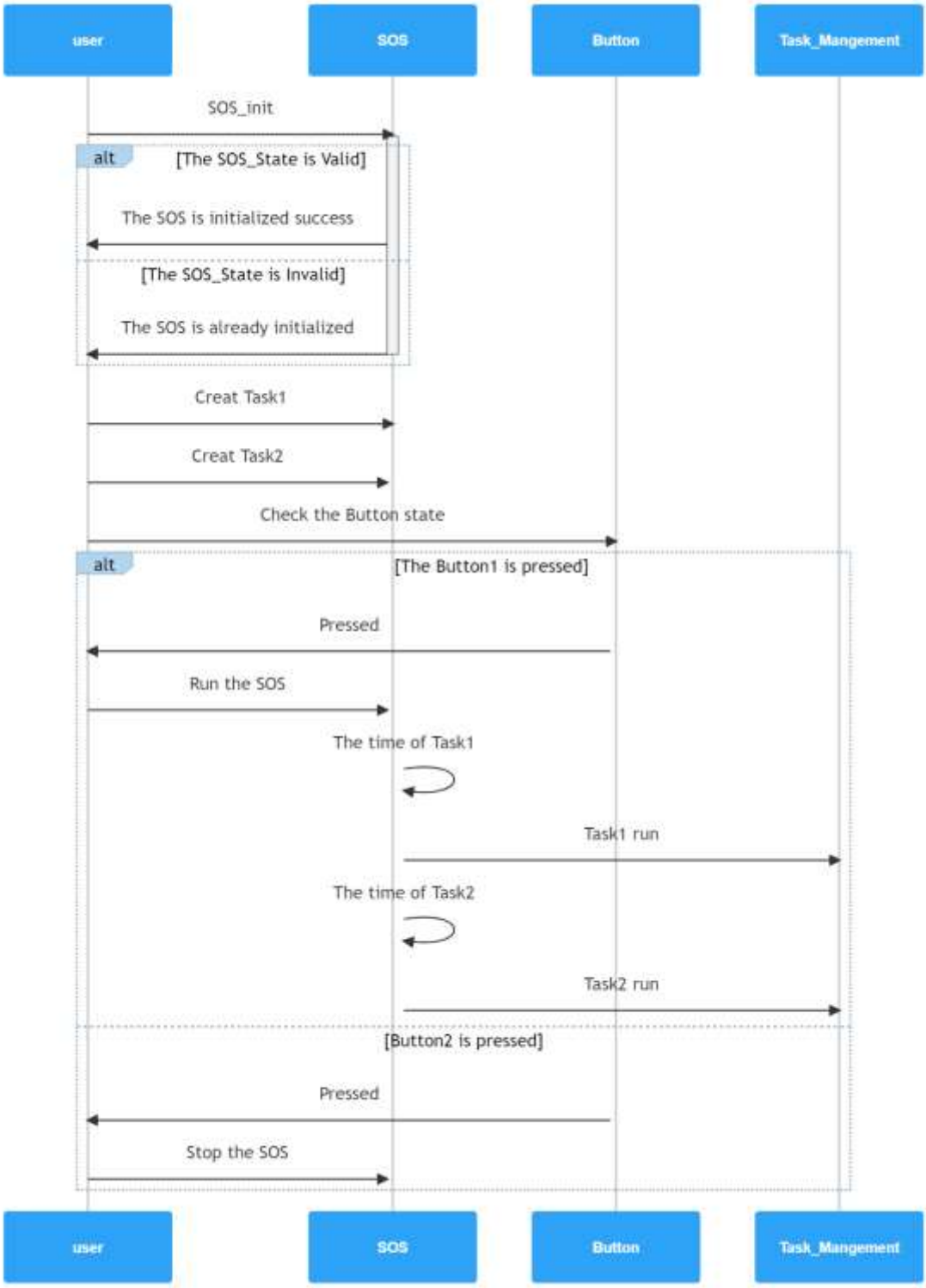
Mohamed Sayed
SPRINTS | MAADI

Table of contents

Contents

Sprints

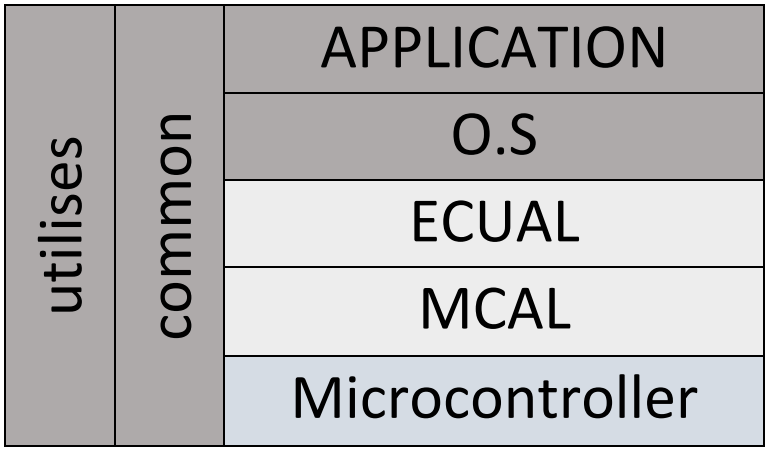
Firstly: Project Sequence Diagram:



Secondly: Layered architecture:

1- Microcontroller

2- MCAL
3- ECUAL
4- SERVICES
5- COMMON
6- Operating System
7- Application



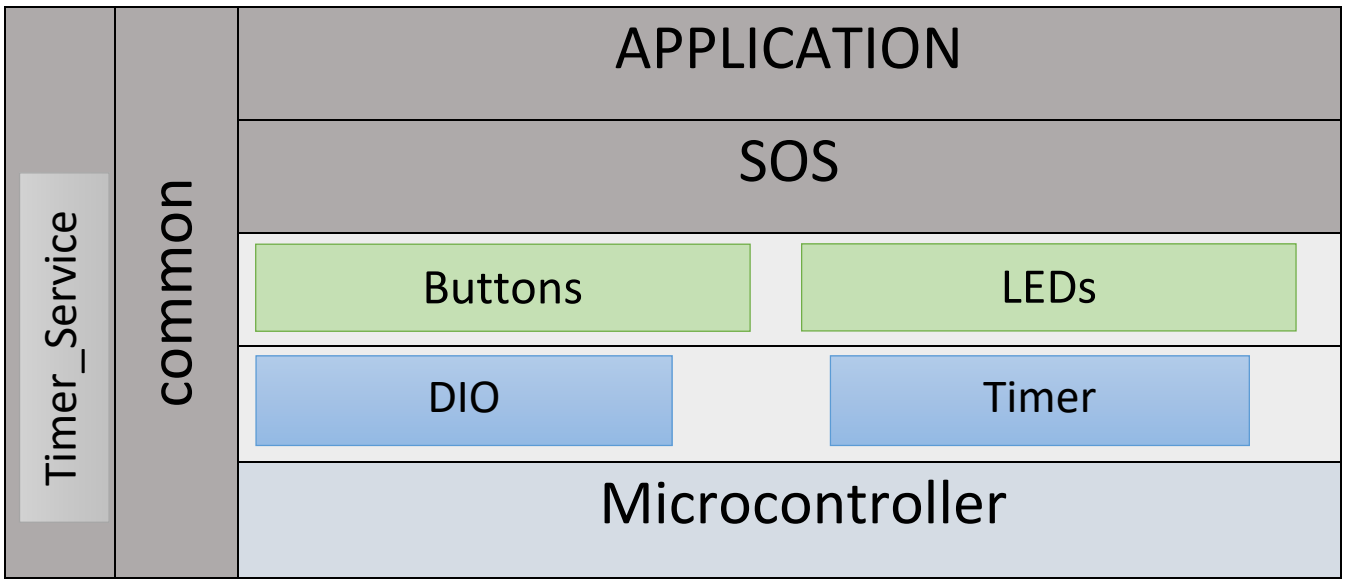
Thirdly: System modules:

1- Specify system modules/drivers:

- DIO, Timer
- LEDs, Buttons
- APPLICATION

2- Assign each module to its related layer:

- By drawing



MCAL APIs

Fourthly: APIs:

DIO APIs

- void DIO_InitPin (PIn_name pin ,PIN_Status status);
- void DIO_init (void);
- void DIO_WRitePin (PIn_name pin ,Voltage_type s);
- Voltage_type DIO_ReadPin(PIn_name pin);
- void DIO_WritePort(PORT_Type Port,u8 data);

Configuration file:

```
#include "DIO_INTERFACE.h"
const PIN_Status Pin_StatusArray [

OUTPUT,        //PINA0
OUTPUT,        //PINA1
OUTPUT,        //PINA2
OUTPUT,        //PINA3
INFREE,        //PINA4
INFREE,        //PINA5
INFREE,        //PINA6
INFREE,        //PINA7
OUTPUT,        //PINB0
OUTPUT,        //PINB1
OUTPUT,        //PINB2
OUTPUT,        //PINB3
OUTPUT,        //PINB4  //SS
OUTPUT,        //PINB5 //MOSI
OUTPUT,        //PINB6  //MISO
OUTPUT,        //PINB7 //SCK
OUTPUT,        //PINC0
OUTPUT,        //PINC1
OUTPUT,        //PINC2
OUTPUT,        //PINC3
OUTPUT,        //PINC4
OUTPUT,        //PINC5
OUTPUT,        //PINC6
OUTPUT,        //PINC7
OUTPUT,        //PIND0
OUTPUT,        //PIND1
INPUT,         //PIND2
INPUT,         //PIND3
INPUT,        //PIND4
INPUT,         //PIND5
INPUT,         //PIND6
INPUT         //PIND7
};
```

Sprints

Timer APIs

- void Timer1_Init(Timer1Mode_type mode);
- void Timer1_Start(Timer1Scaler_type scaler);
- void Timer1_OCRA1Mode(OC1A_Mode_type oc1a_mode);
- void Timer1_OVF_InterruptEnable(void);
- void Timer1_OVF_InterruptDisable(void);
- void Timer1_OCA_InterruptEnable(void);
- void Timer1_OCA_InterruptDisable(void);
- void Timer1_OVF_SetCallBack(void(*LocalFptr)(void));
- void Timer1_OCA_SetCallBack(void(*LocalFptr)(void));

Configuration file:

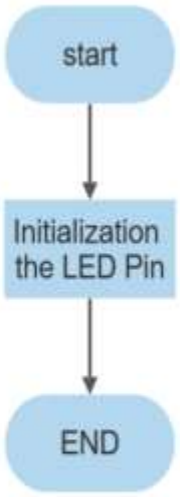
```
typedef enum{
    TIMER1_STOP=0,
    TIMER1_SCALER_1,
    TIMER1_SCALER_8,
    TIMER1_SCALER_64,
    TIMER1_SCALER_256,
    TIMER1_SCALER_1024,
    EXTERNAL0_FALLING,
    EXTERNAL0_RISING
}Timer1Scaler_type;
#define TIMER1_STOP()       TCCR1B&=0XF8
#define TIMER1_SET(value)   TCNT1=value
typedef enum
{
    TIMER1_NORMAL_MODE=0,
    TIMER1_CTC_ICR_TOP_MODE,
    TIMER1_CTC_OCRA_TOP_MODE,
    TIMER1_FASTPWM_ICR_TOP_MODE,
    TIMER1_FASTPWM_OCRA_TOP_MODE

}Timer1Mode_type;

typedef enum
{
    OCRA_DISCONNECTED=0,
    OCRA_TOGGLE,
    OCRA_NON_INVERTING,
    OCRA_INVERTING
}OC1A_Mode_type;
```

ECUAL APIs

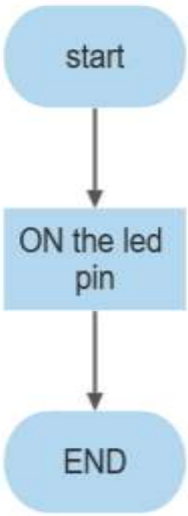
LEDs APIs

- **void LED_init(u8 led);**
- **void LED_ON (u8 LED);**
- **void LED_OFF (u8 LED);**
- **void LED_Toggle (u8 LED);**

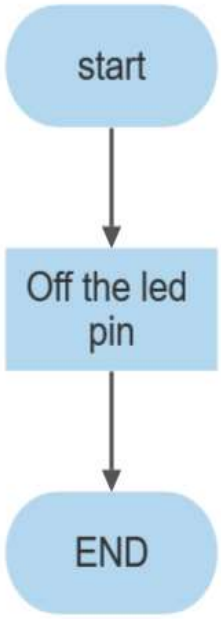
LEDs FLOWCHARTS:



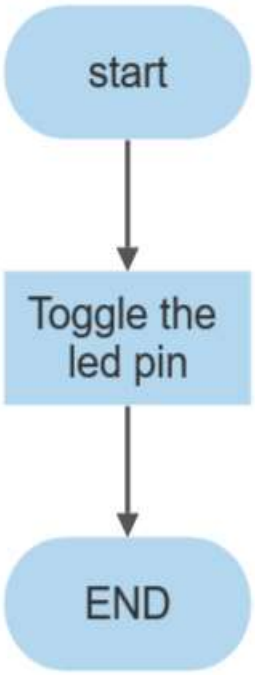
void LED_init(u8 led);



void LED_ON (u8 LED);



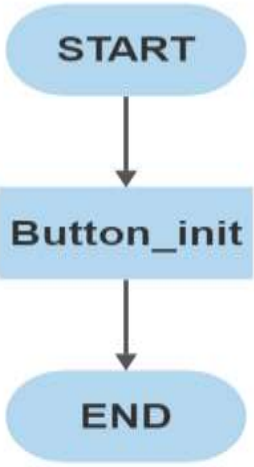
void LED_OFF (u8 LED);



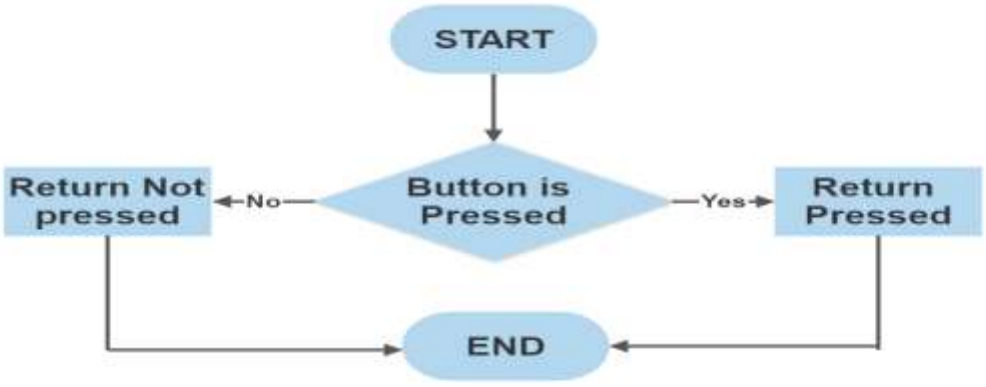
void LED_Toggle (u8 LED);

Buttons APIs

- **void button_init(PIn_name pin);**
- **Button_Status Button_Check(u8 Button);**



void Button_Init(void);



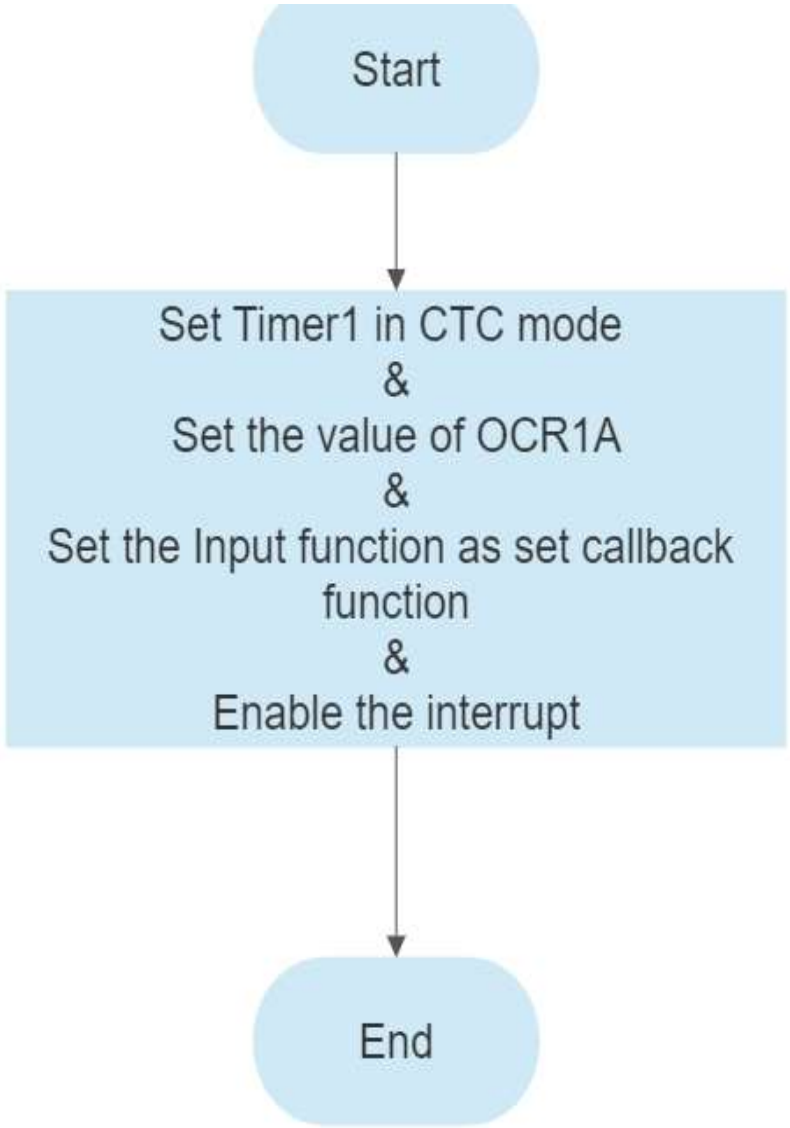
Button_Status Button_Check(u8 Button);

SECVICE APIs

Timer_Service APIs

- **void Timer1_SetInterruptTime_ms (u16 time,void(*LocalFptr)(void));**
- **void Timer1_SetInterruptTime_us (u16 time,void(*LocalFptr)(void));**
- **void Timer1_SetInterruptTime_s (u16 time,void(*LocalFptr)(void));**

FLOWCHARTS:



SOS APIs

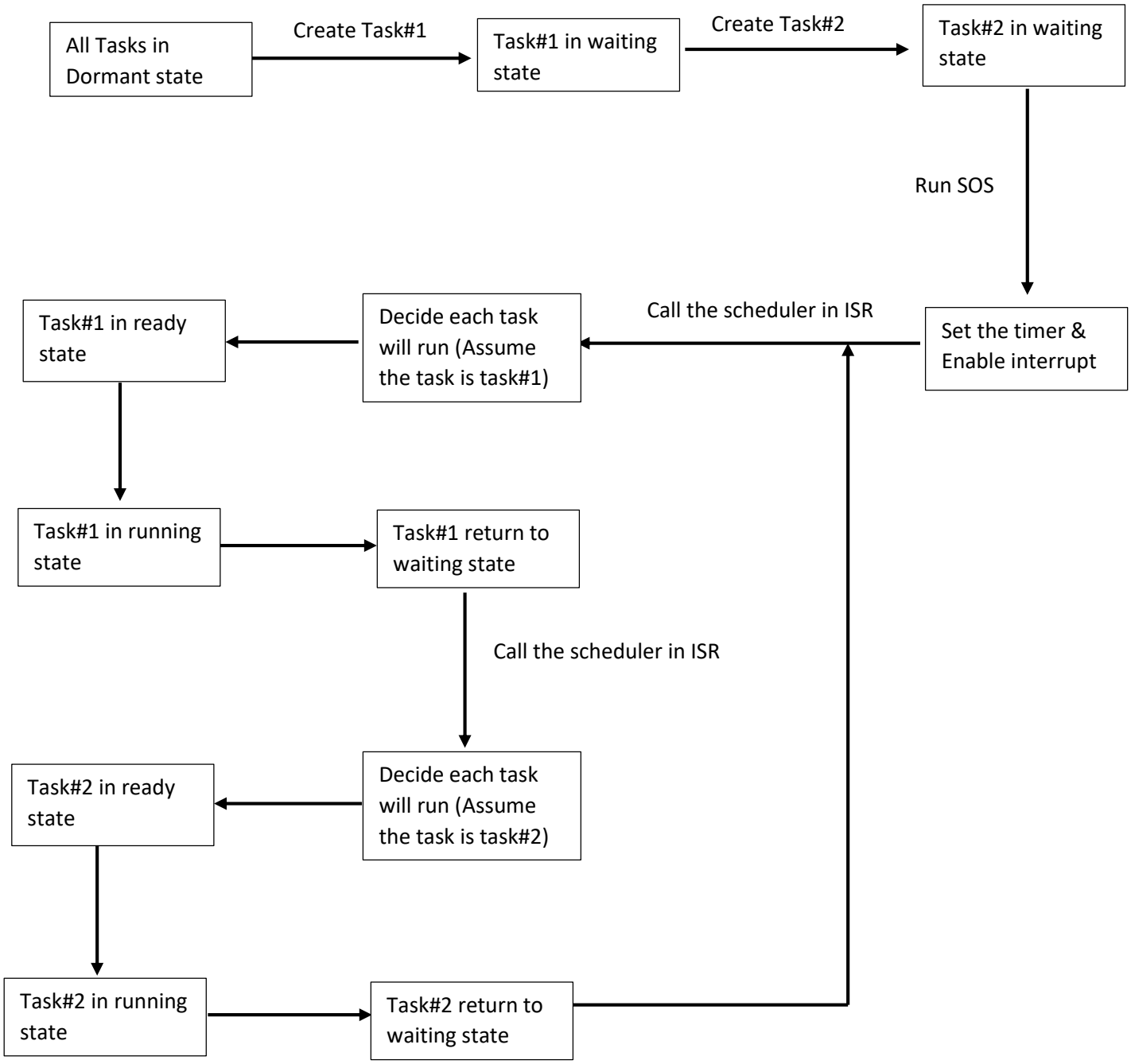
- **enu_system_status_t sos_init (void);**
- **enu_system_status_t sos_deinit (void);**
- **enu_creat_task_status_t sos_create_task (u8 Copy_u8Priority , void(*fptr)(void) , u16 Copy_u16Periodicity);**
- **enu_delete_task_status_t sos_delete_task (u8 Copy_u8Priority);**
- **enu_modify_task_status_t sos_modify_task (void(*fptr)(void) , u16 Copy_u16Periodicity);**
- **enu_sos_operation_status_t sos_run (void);**
- **enu_sos_operation_status_t sos_disable (void);**

Configuration file:

```
#define TASK_NUM             3
#define INVALID_TASK         'I'
typedef struct
{
    u8 Copy_u8periodicity;
    void(*TaskFunc)(void);
    u8 Copy_u8Priority;
}Task_t;


Task_t SystemTasks[TASK_NUM]={
    /*Task#1*/
    {
        .periodicity=300,
        .TaskFunc=Func1,
        .Copy_u8Priority=0;
    },
    /*Task#2*/
    {
        .periodicity=500,
        .TaskFunc=Func2,
        .Copy_u8Priority=1;
    },
    /*Task#3*/
    {
        .periodicity=INVALID_TASK,
        .TaskFunc=INVALID_TASK,
        .Copy_u8Priority=INVALID_TASK;
    },
    /*Idle task*/
    {
        .periodicity=50,
        .TaskFunc=Idle_Func,
        .Copy_u8Priority=TASK_NUM-1;
    }
};
```

State machine

Assume that task#1 is high priority

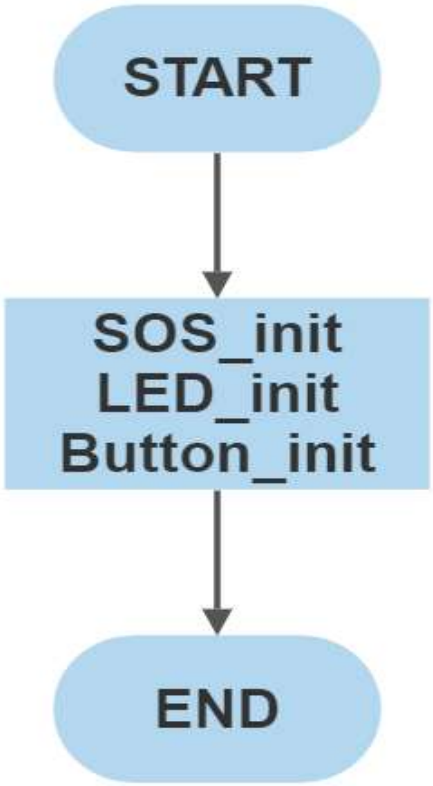


Class diagram:

APPLICATION APIs:

- void APP_Init(void);
- void APP_Start(void);

APPLICATION FLOWCHARTS:



void APP_Init(void);