BCM Module

DESIGN DOCUMENT

ASSINED BY:

MOHAMED SAYAD

Mohamed Sayed
SPRINTS | MAADI

Sprints

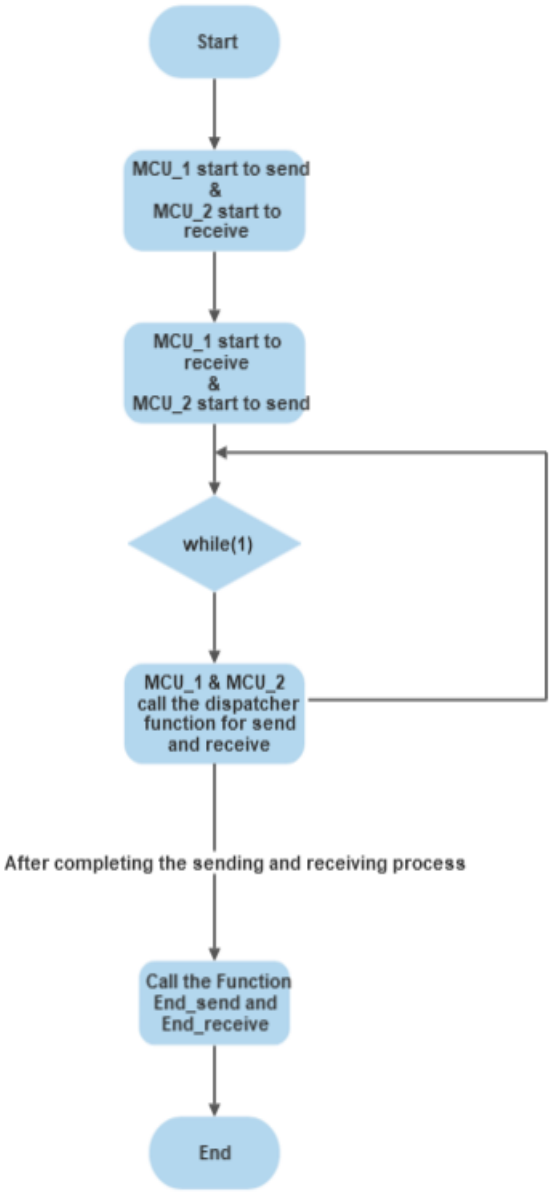
Table of contents

Contents

Firstly: Project Description:

Description

First of all, The MCU_1 start to send and receive as same as MCU_2.
Then, MCU_1 & MCU_2 call the dispatcher function to know if sending and receiving process finish or not. After that the send_end & receive_end functions will be called.

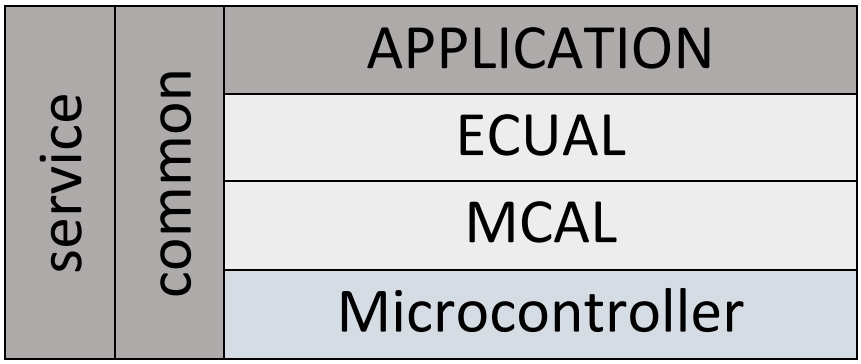


Sprints

Secondly: Layered architecture:

1- Microcontroller

2- MCAL
3- ECUAL
4- SERVICES
5- COMMON
6- Application



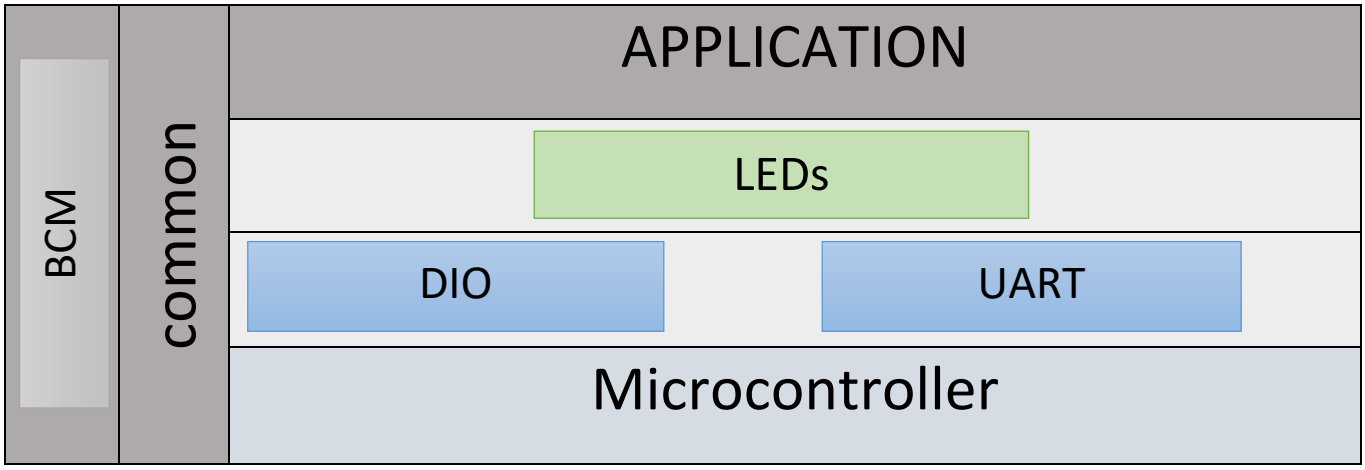
Thirdly : System modules:

1- Specify system modules/drivers:

- DIO, UART
- LEDs
- APPLICATION

2- Assign each module to its related layer:

- By drawing



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
};
```

UART APIs

- enu_uart_status_t uart_init(void);
- void uart_transmit(u8 data);
- void uart_transmitNoBlock(u8 data); void
- uart_transmitComPlete_InterruptEnable(void);
- void uart_transmitComPlete_InterruptDisable(void);
- void uart_transmitComPlete_InterruptSetCallback(void(*fptr)(void));
- u8 uart_reciever(void);
- u8 uart_recieverNoBlock(void);
- void uart_recieveComPlete_InterruptEnable(void);
- void uart_recieveComPlete_InterruptDisable(void);
- void uart_recieveComPlete_InterruptSetCallback(void(*fptr)(void));

Configuration file:

```
#include "uart.h"
#if UART_MODE==ASYNCH
const str_uart_config_t gstr_uart_config_ch[CHANNEL_SIZE]=
{
    {
      Baudrate9600,
      Normal_Speed,
      No_Parity,
      StopBit_1,
      DataBit_8,
      TR
    },
    {
      Baudrate_Total,
      Speed_Total,
      Parity_Total,
      StopBit_Total,
      DataBit_Total,
      Enable_Total
    }
};
#elif UART_MODE==SYNCH
const str_uart_config_t gstr_uart_config_ch[CHANNEL_SIZE]=
{
    {
        Normal_Speed,
        DataBit_8,
        TR
    },
    {
        Speed_Total,
        DataBit_Total,
        Enable_Total
    }
};
#endif
```

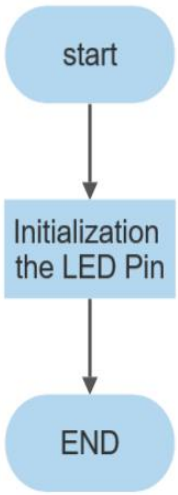
Sprints

ECUAL APIs

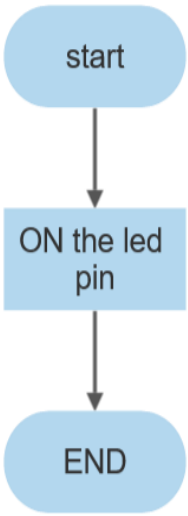
LEDs APIs

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

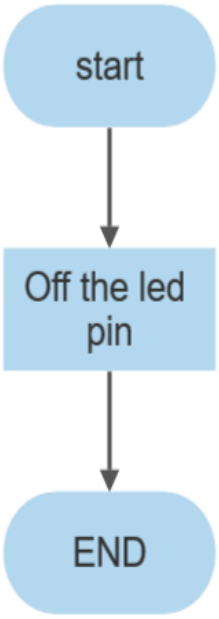
LEDs FLOWCHARTS:



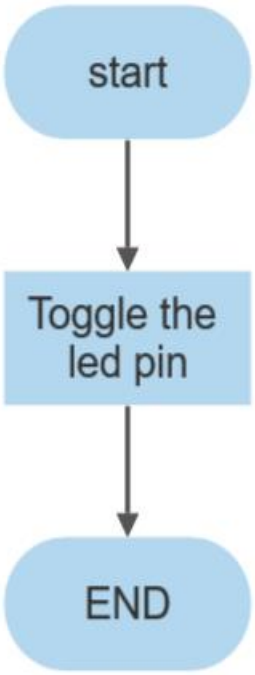
void LED_init(u8 led);



void LED_ON (u8 LED);



void LED_OFF (u8 LED);



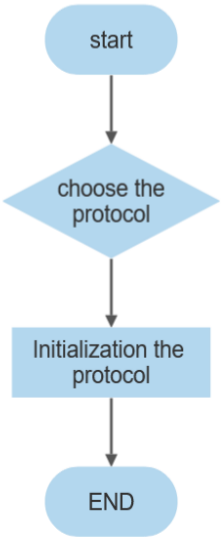
void LED_Toggle (u8 LED);

SECVICE APIs

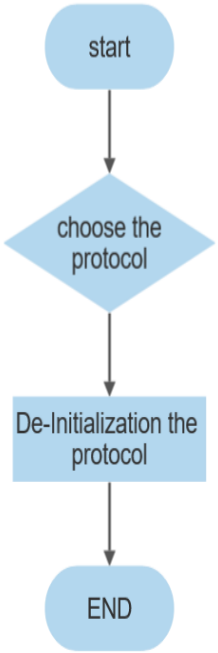
BCM APIs

- enu_system_status_t bcm_init(enu_protocol_t protocol);
- enu_system_status_t bcm_deinit(enu_protocol_t protocol);
- void bcm_send(u8 data); //Blocking Function
- void bcm_send_Blocking(u8*str);// Blocking function
- void bcm_send_Non_Blocking(u8*str,u8 length, u8 start); //Non Blocking function
- void uart_bcm_send_dispatcher(void);
- void bcm_send_End_Setcallback(void (*fptr)(void));
- void bcm_receive(u8*data); //Blocking Function
- void bcm_receive_Blocking(u8*str); //Blocking Function
- void bcm_receive_Non_Blocking(u8*str); //Non Blocking function
- void bcm_receive_End_Setcallback(void (*fptr)(void));
- void uart_bcm_recieve_dispatcher(void);

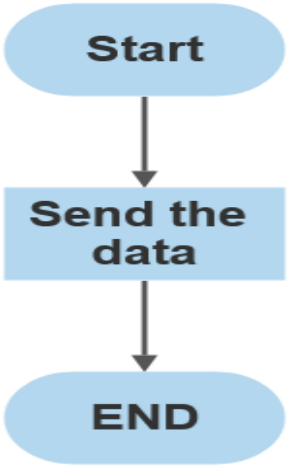
BCM FLOWCHARTS:



enu_system_status_t bcm_init(enu_protocol_t protocol);

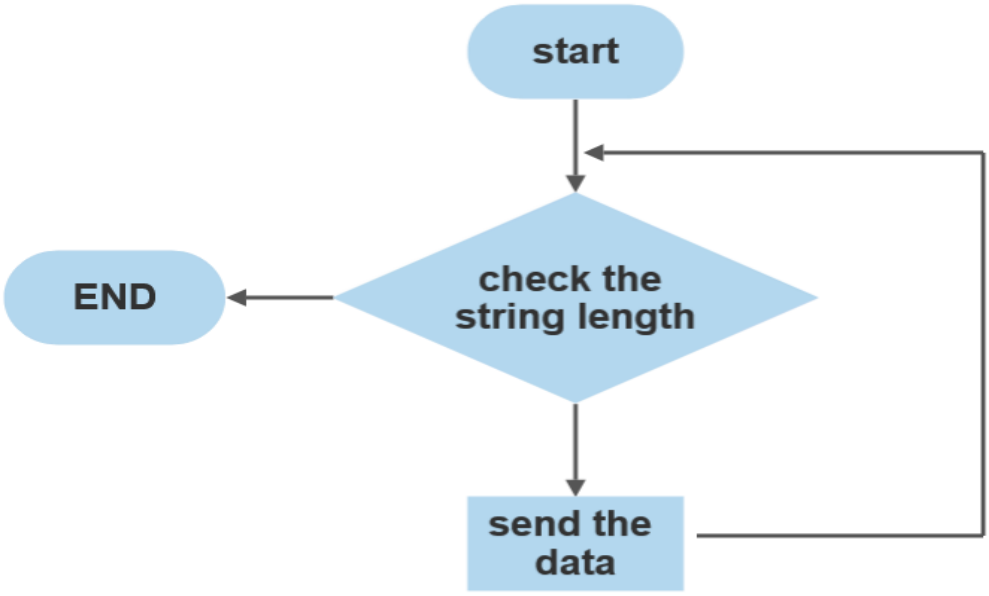


enu_system_status_t bcm_deinit(enu_protocol_t protocol);

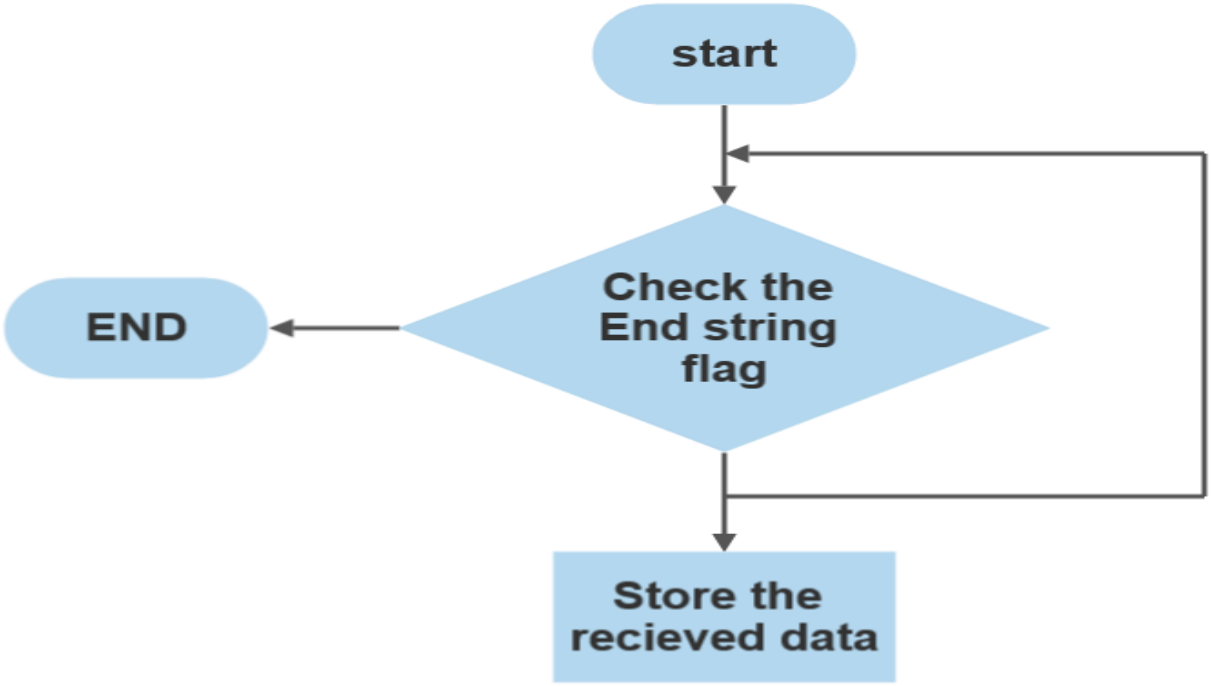


void bcm_send(u8 data);

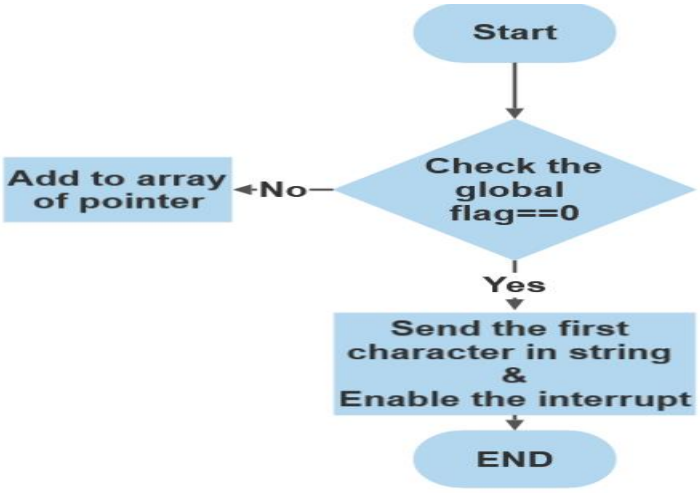
void bcm_recieve(u8*data);



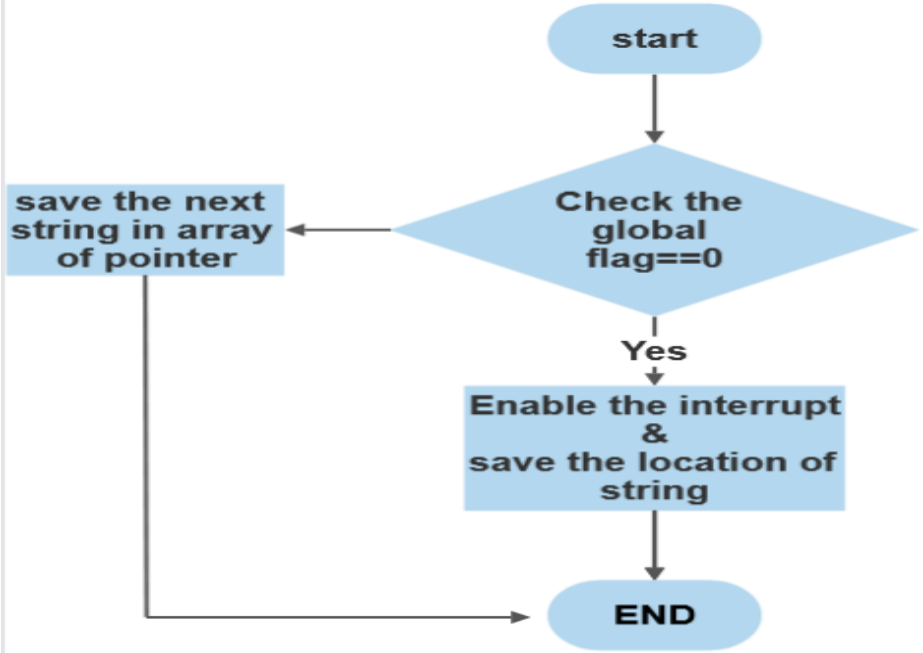
void bcm_send_Blocking(u8*str);



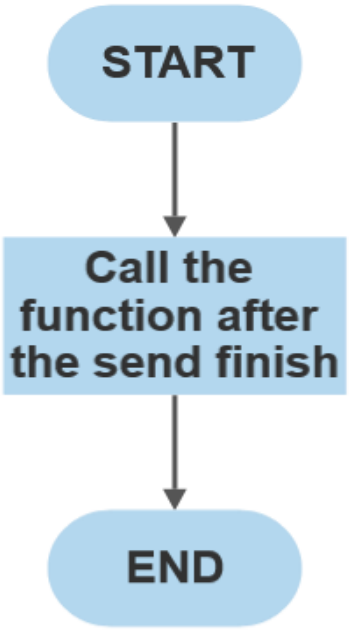
void bcm_recieve_Blocking(u8*str);



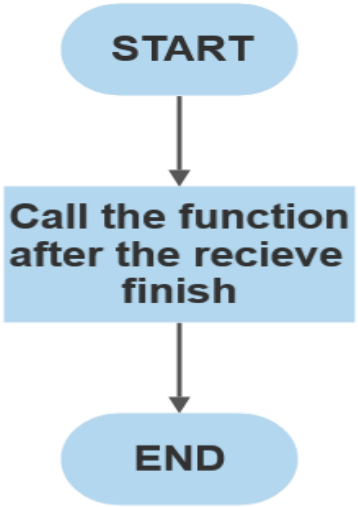
void bcm_send_Non_Blocking(u8*str,u8 length, u8 start);



void bcm_recieve_Non_Blocking(u8*str);



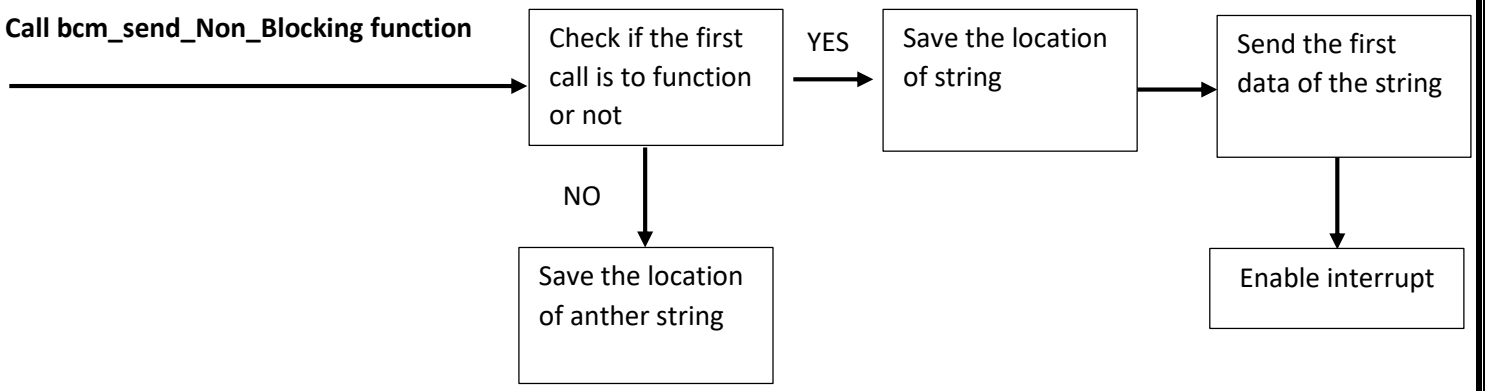
void bcm_send_End_Setcallback(void (*fptr)(void));

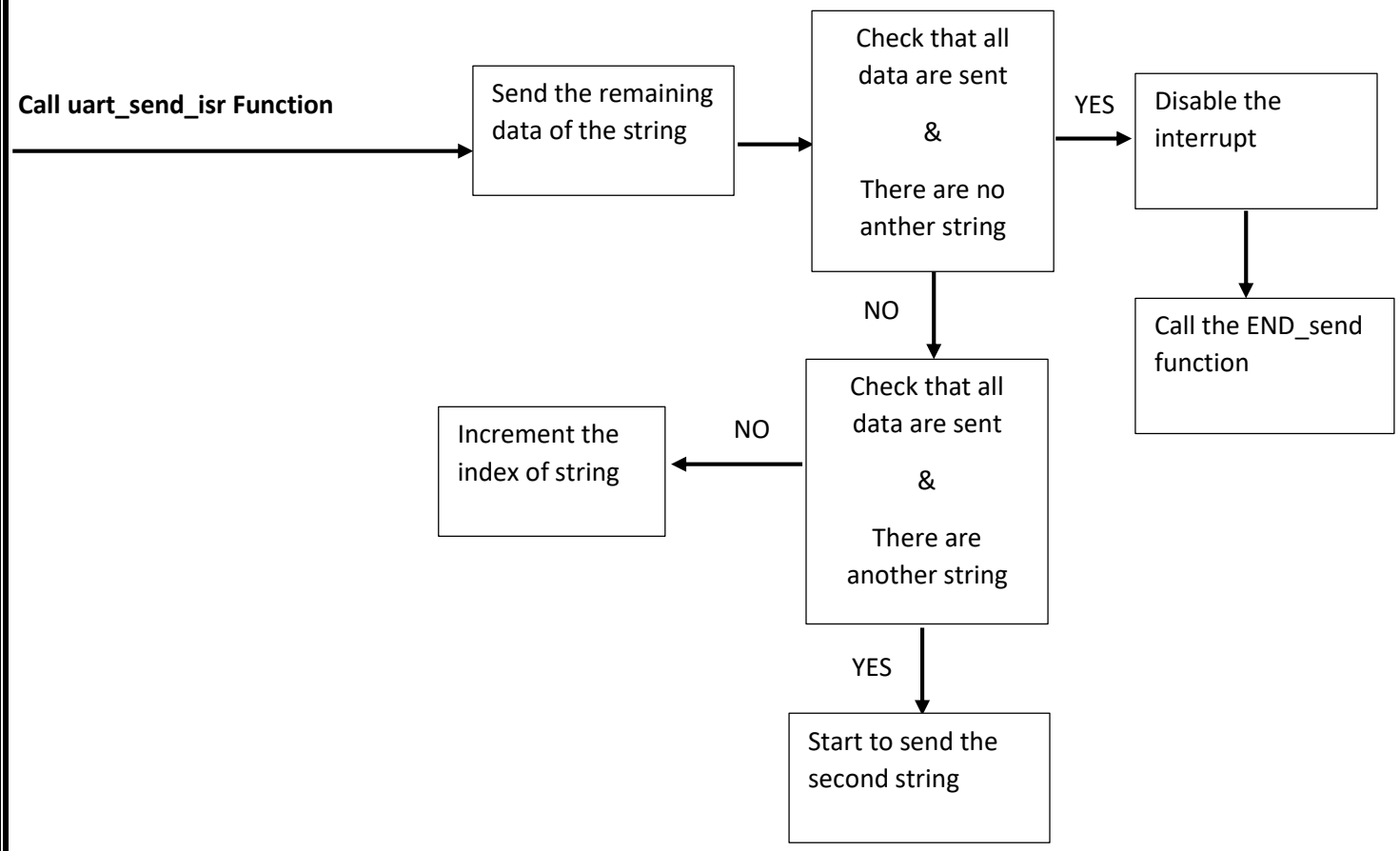


void bcm_recieve_End_Setcallback(void (*fptr)(void));

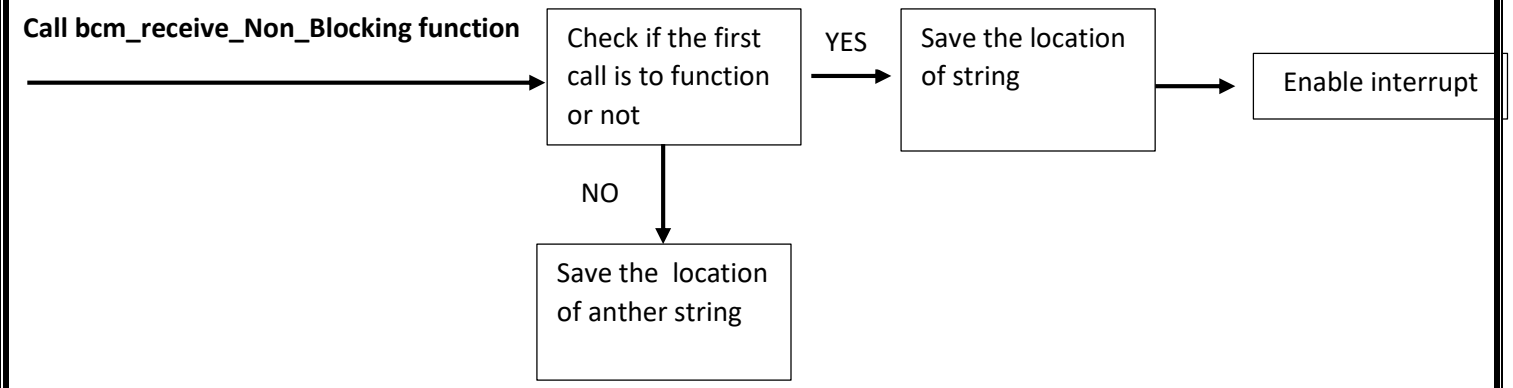
statmachine

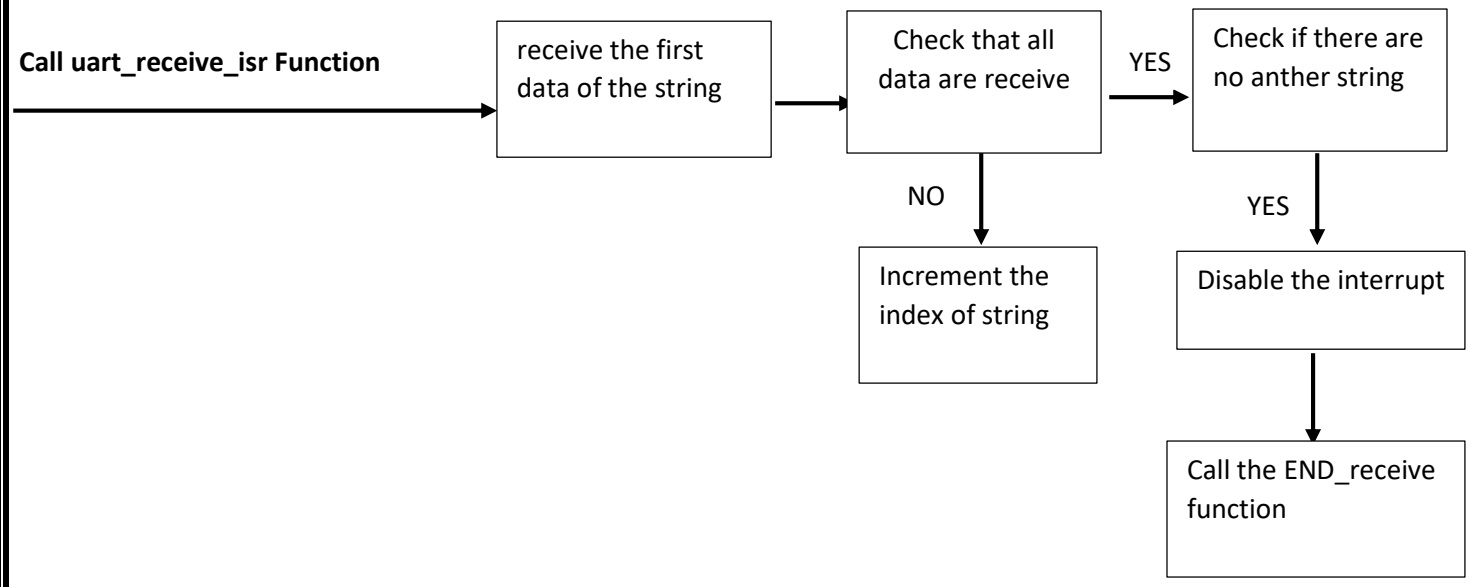
- **BCM Send**





BCM Receive

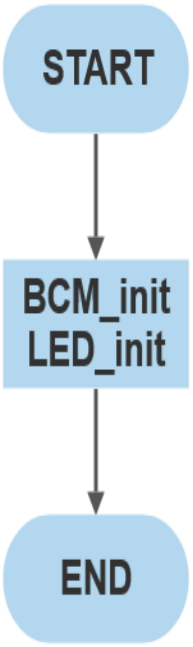

Call bcm_receive_Non_Blocking function
Check if the first call is to function or not
YES
Save the location of string
Enable interrupt
NO
Save the location of anther string


Call uart_receive_isr Function
receive the first data of the string
Check that all data are receive
YES
Check if there are no anther string
NO
YES
Increment the index of string
Disable the interrupt
Call the END_receive function

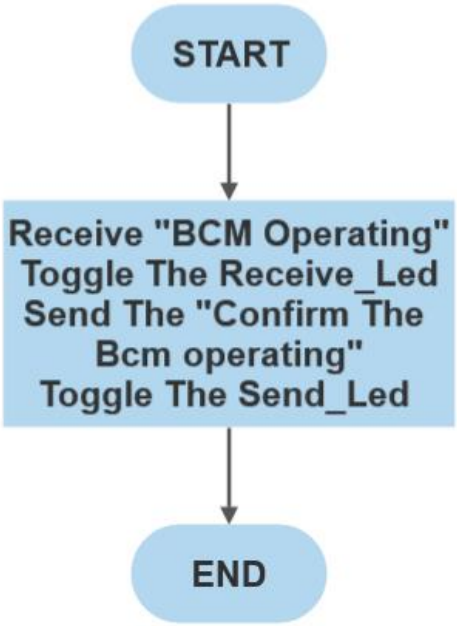
APPLICATION APIs:

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

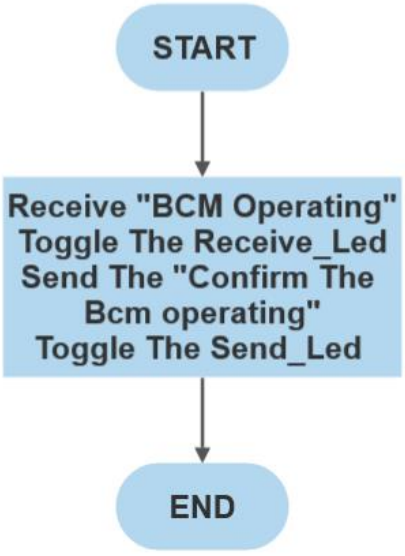
APPLICATION FLOWCHARTS:



void APP_Init(void);



void APP_start(void); (MUC1)



void APP_start(void); (MUC2)