BCM

Kareem Magdy Albolaqi

SPRINT
10

Table of Contents

LAYERED ARCHTICTURE 2

PROJECT FLOWCHART 5

DRIVERS 7

MCAL 7

DIO DRIVER *7*

USART DRIVER *10*

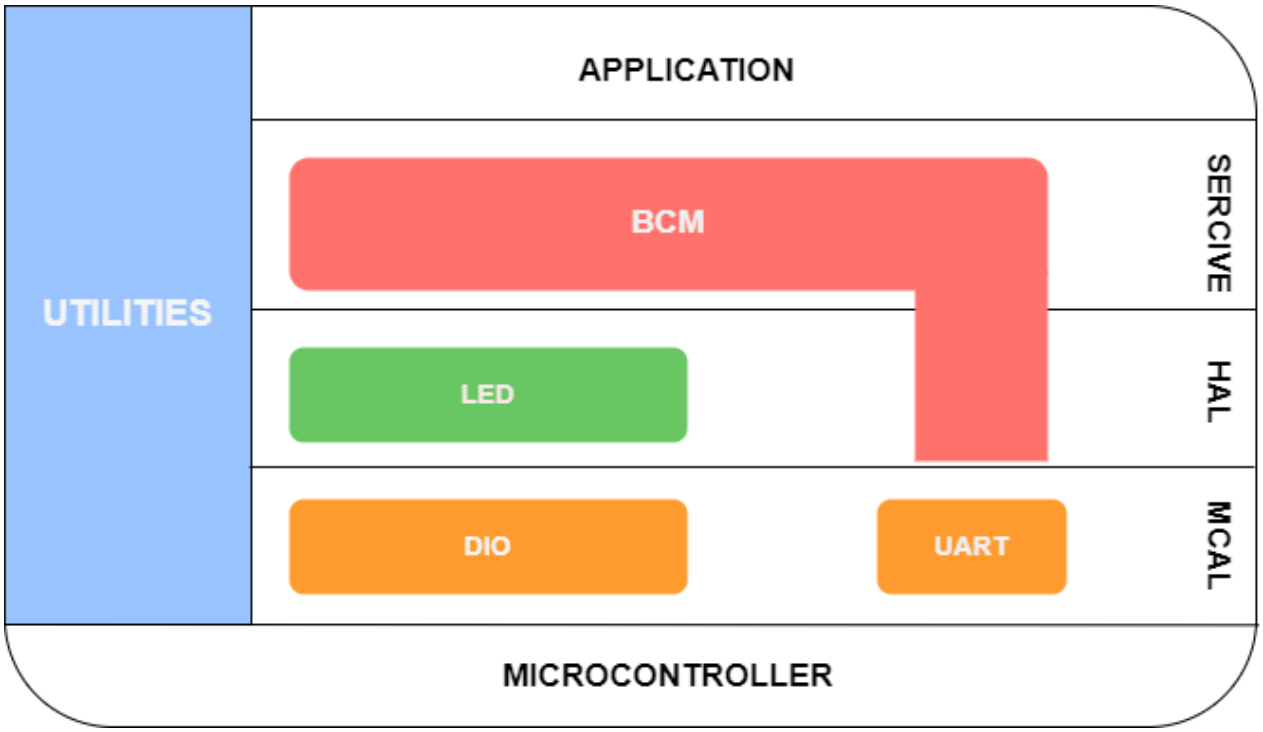
HAL 13

LED DRIVER *13*

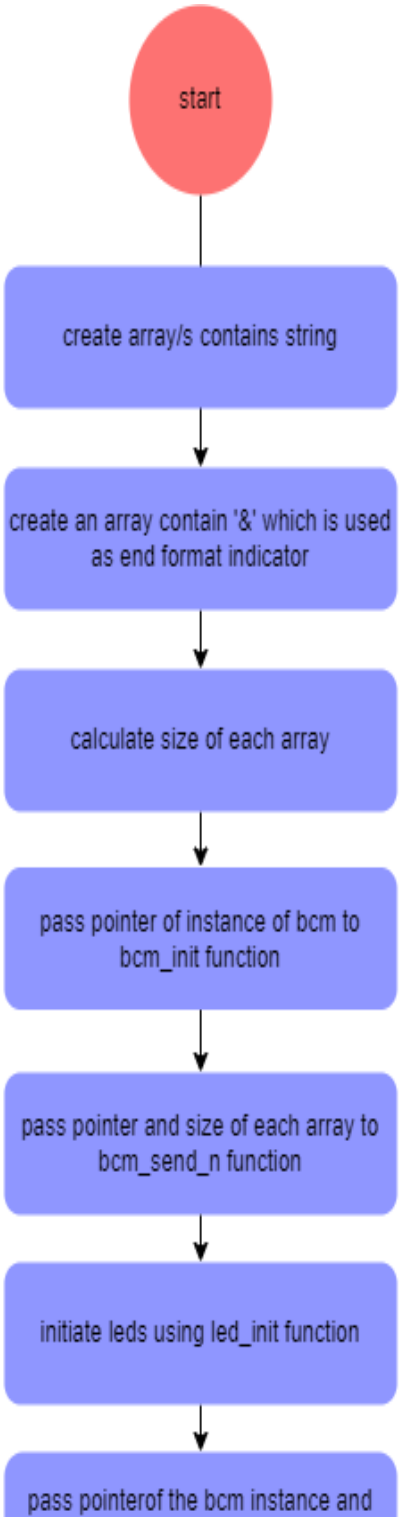
SERVICE 14

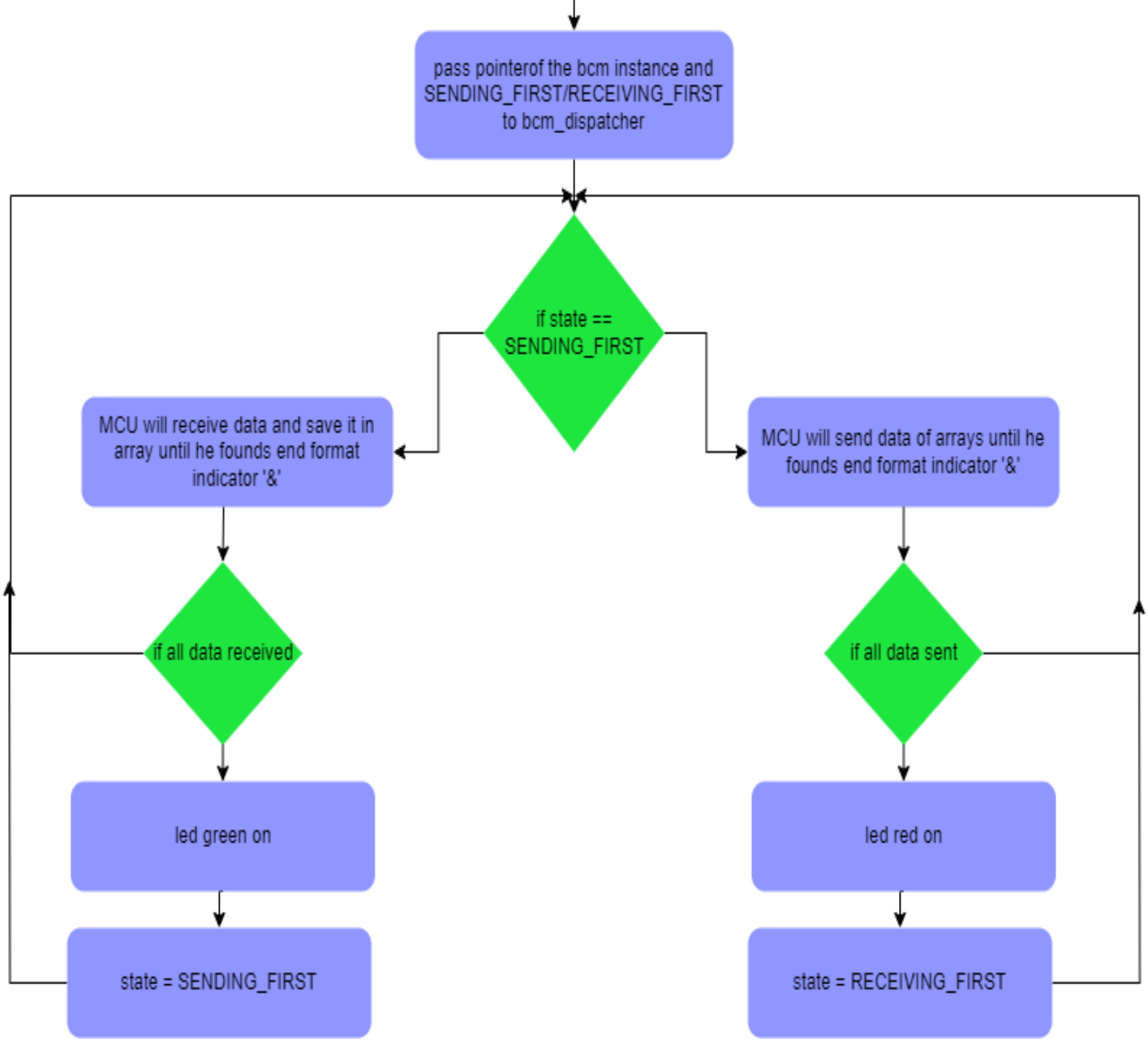
BCM *14*

LAYERED ARCHTICTURE



PROJECT FLOWCHART



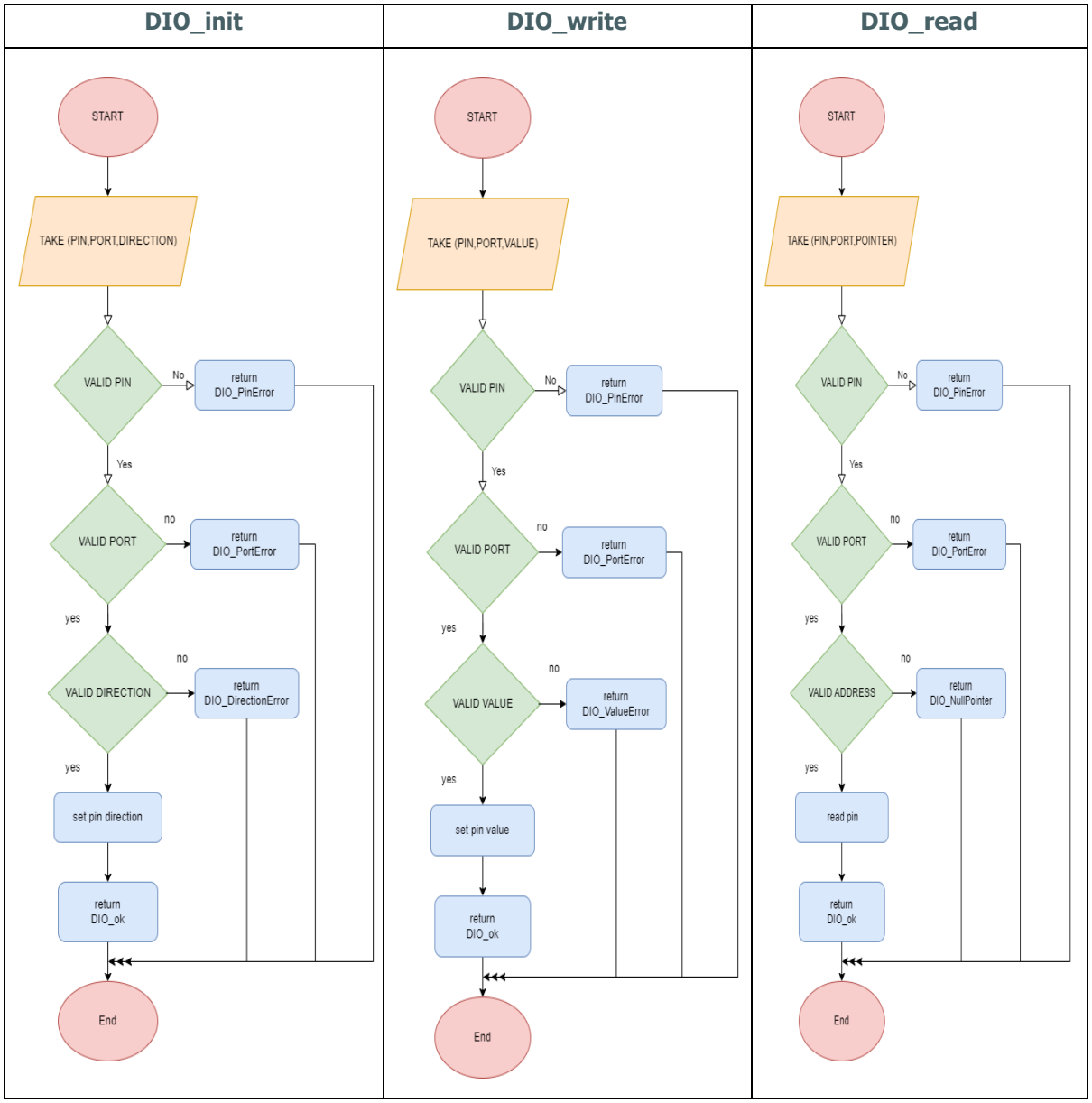

pass pointerof the bcm instance and SENDING_FIRST/RECEIVING_FIRST to bcm_dispatcher
if state == SENDING_FIRST
MCU will receive data and save it in array until he founds end format indicator '&'
MCU will send data of arrays until he founds end format indicator '&'
if all data received
if all data sent
led green on
led red on
state = SENDING_FIRST
state = RECEIVING_FIRST

DRIVERS

MCAL

DIO DRIVER

```
Dio_ErrorStatus Dio_init(void);
Dio_ErrorStatus Dio_WriteChannel(Dio_Channel channel , Dio_status state);
Dio_ErrorStatus Dio_ReadChannel(Dio_Channel channel);
```



Configurations :

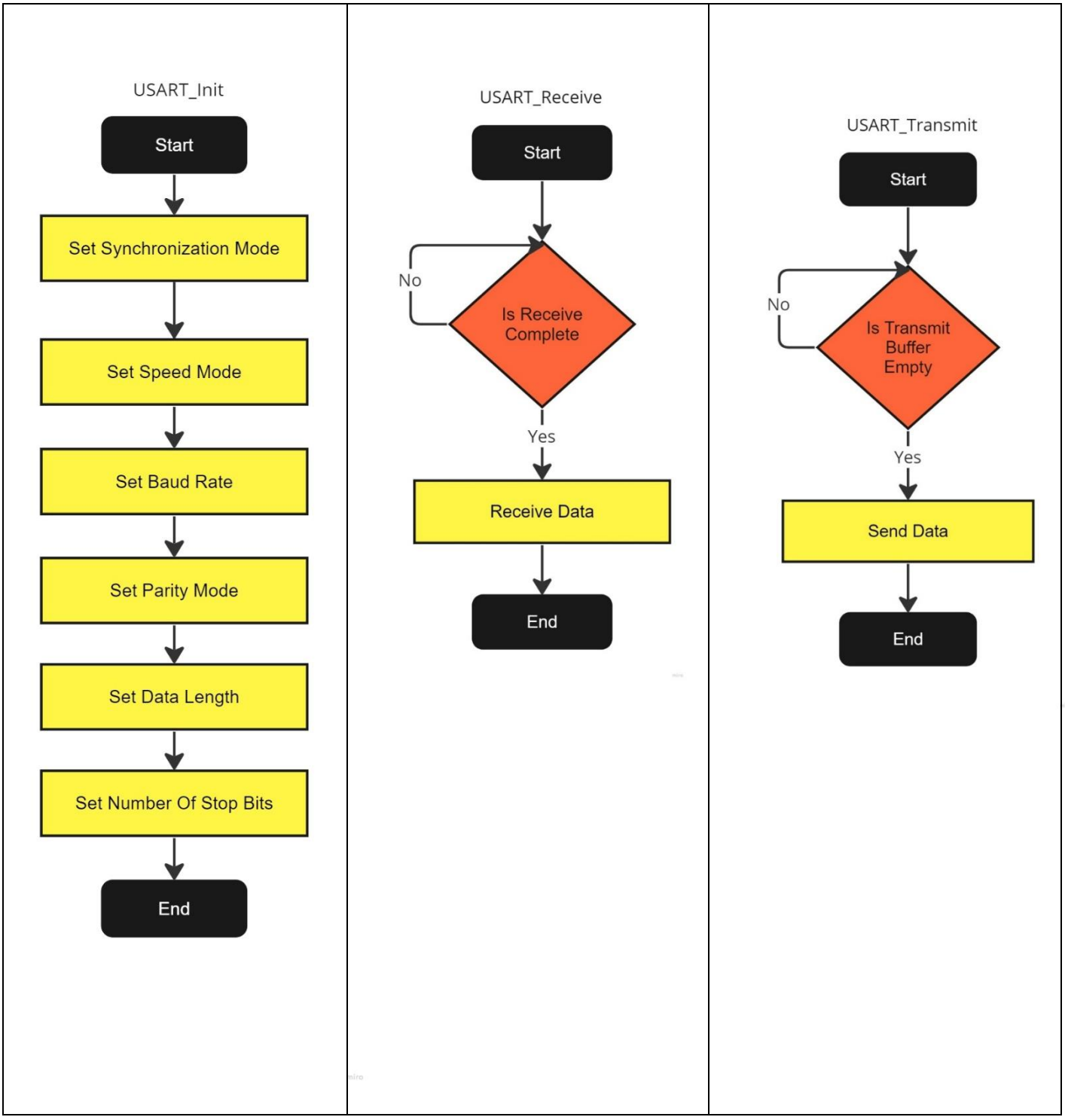
```c
typedef enum {
    portA_0,
    portA_1,
    portA_2,
    portA_3,
    portA_4,
    portA_5,
    portA_6,
    portA_7,

    portB_0,
    portB_1,
    portB_2,
    portB_3,
    portB_4,
    portB_5,
    portB_6,
    portB_7,

    portC_0,
    portC_1,
    portC_2,
    portC_3,
    portC_4,
    portC_5,
    portC_6,
    portC_7,

    portD_0,
    portD_1,
    portD_2,
    portD_3,
    portD_4,
    portD_5,
    portD_6,
    portD_7,

}Dio_Channel;
```

```c
typedef enum {
    LOW,
    HIGH

}Dio_status;

typedef enum {
    PIN_0,
    PIN_1,
    PIN_2,
    PIN_3,
    PIN_4,
    PIN_5,
    PIN_6,
    PIN_7,

}Dio_PIN;

typedef enum {
    PORT_A,
    PORT_B,
    PORT_C,
    PORT_D

}Dio_PORT;


typedef enum {
    INPUT,
    OUTPUT

}Dio_DIR;

typedef enum {
    PULLUP_OFF,
    PULLUP_ON

}Dio_PULLUP;
```

```
Dio_PinCfg Dio_PINS_Cfg[PIN_COUNT]= {

    //LCD
    {   PORT_A,PIN_4,OUTPUT,PULLUP_OFF},
    {   PORT_A,PIN_5,OUTPUT,PULLUP_OFF},
    {   PORT_A,PIN_6,OUTPUT,PULLUP_OFF},
    {   PORT_A,PIN_7,OUTPUT,PULLUP_OFF},
    {   PORT_B,PIN_0,OUTPUT,PULLUP_OFF},
    {   PORT_B,PIN_1;OUTPUT,PULLUP_OFF},

    //KEYPAD
    {   PORT_C,PIN_0,OUTPUT,PULLUP_OFF},
    {   PORT_C,PIN_1,OUTPUT,PULLUP_OFF},
    {   PORT_C,PIN_2,OUTPUT,PULLUP_OFF},
    {   PORT_C,PIN_3,OUTPUT,PULLUP_OFF},
    {   PORT_C,PIN_4,INPUT,PULLUP_ON},
    {   PORT_C,PIN_5,INPUT,PULLUP_ON},
    {   PORT_C,PIN_6,INPUT,PULLUP_ON},
    {   PORT_C,PIN_7,INPUT,PULLUP_ON},


    // BUTTON
    {   PORT_D,PIN_2,INPUT,PULLUP_ON},

};
```

USART DRIVER

```
uart_error_state USART_init(void);
uart_error_state USART_transmit(uint8_t data);
uart_error_state USART_receive(uint8_t *data);
```



Configurations :

```c
#define UART_CHANEELS  3

typedef enum{
    CHANNEL_0,
    CHANNEL_1,
    CHANNEL_2
}enu_uart_channels_t;


typedef enum{
    normal_speed =0,
    double_speed,
    total_speed
}enu_speed_mode_selector_t;



typedef enum{
    transmit_enable =0,
    receive_enable,
    transmit_receive_enable,
    total_enable
}enu_role_selector_t;

typedef enum{
    no_parity =0,
    even_parity,
    odd_parity,
    total_parity
}enu_parity_mode_selector_t;
```

```c
typedef enum{
    synchronous =0,
    asynchronous,
    total_synch
}enu_sync_mode_selector_t;


typedef enum{
    one_stop_bit =0,
    two_stop_bit,
    total_stop
}enu_number_stopBits_selector_t;


typedef enum{
    _5_data_bits =0,
    _6_data_bits,
    _7_data_bits,
    _8_data_bits,
    _9_data_bits,
    total_bits
}enu_number_dataBits_selector_t;
```

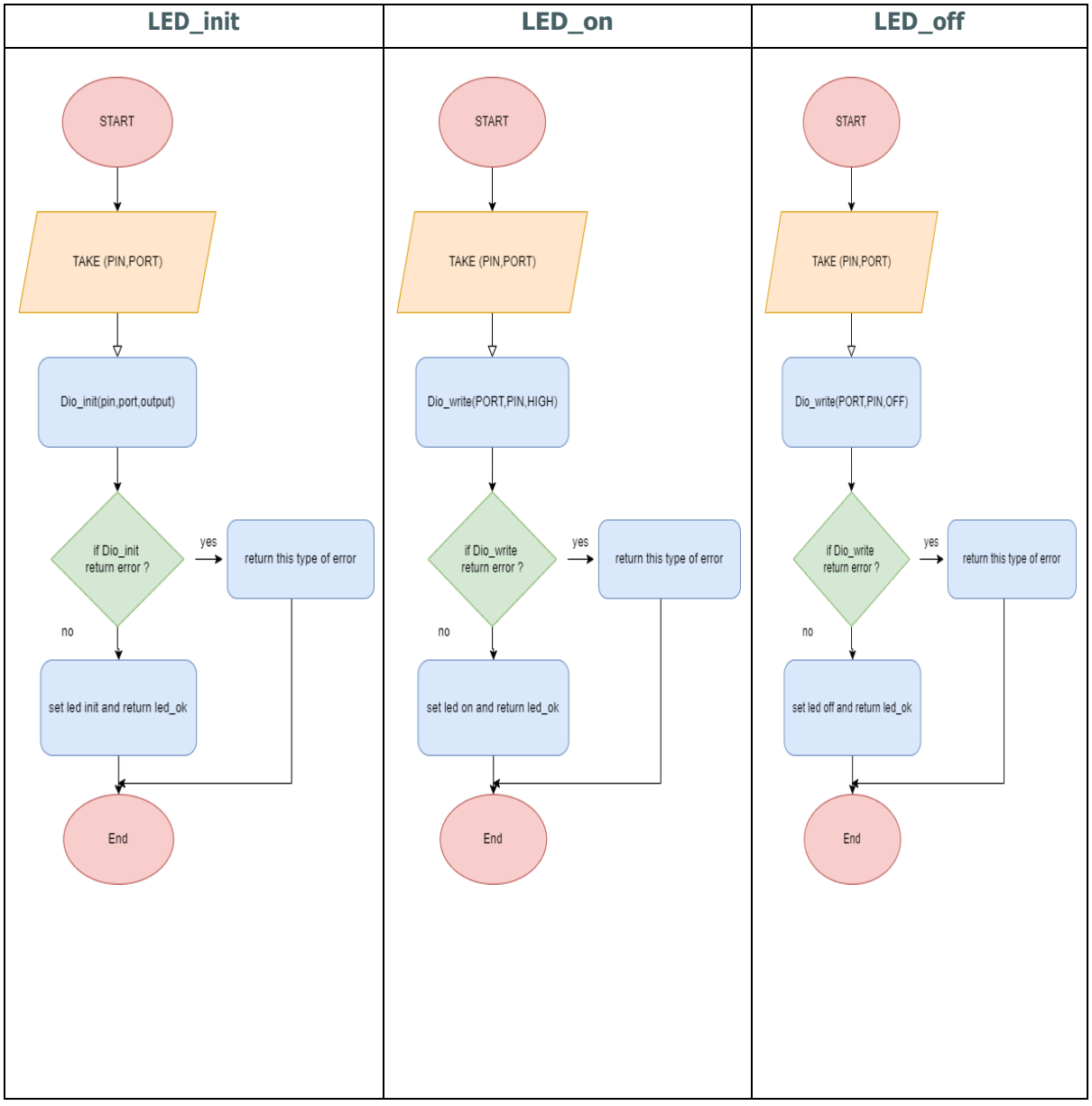
```c
typedef struct
{
    enu_speed_mode_selector_t               enu_speed_config;
    enu_role_selector_t                     enu_role_config;
    enu_parity_mode_selector_t              enu_parity_config;
    enu_sync_mode_selector_t                enu_mode_config;
    enu_number_stopBits_selector_t          enu_stop_bits_config;
    enu_number_dataBits_selector_t          enu_data_size_config;
}str_uart_config_t;


extern str_uart_config_t g_str_uart_config [UART_CHANEELS];
```

```c
str_uart_config_t g_str_uart_config [UART_CHANEELS]=
{
    {
        normal_speed,
        transmit_receive_enable,
        no_parity,
        asynchronous,
        one_stop_bit,
        _8_data_bits
    }

};
```

HAL

LED DRIVER

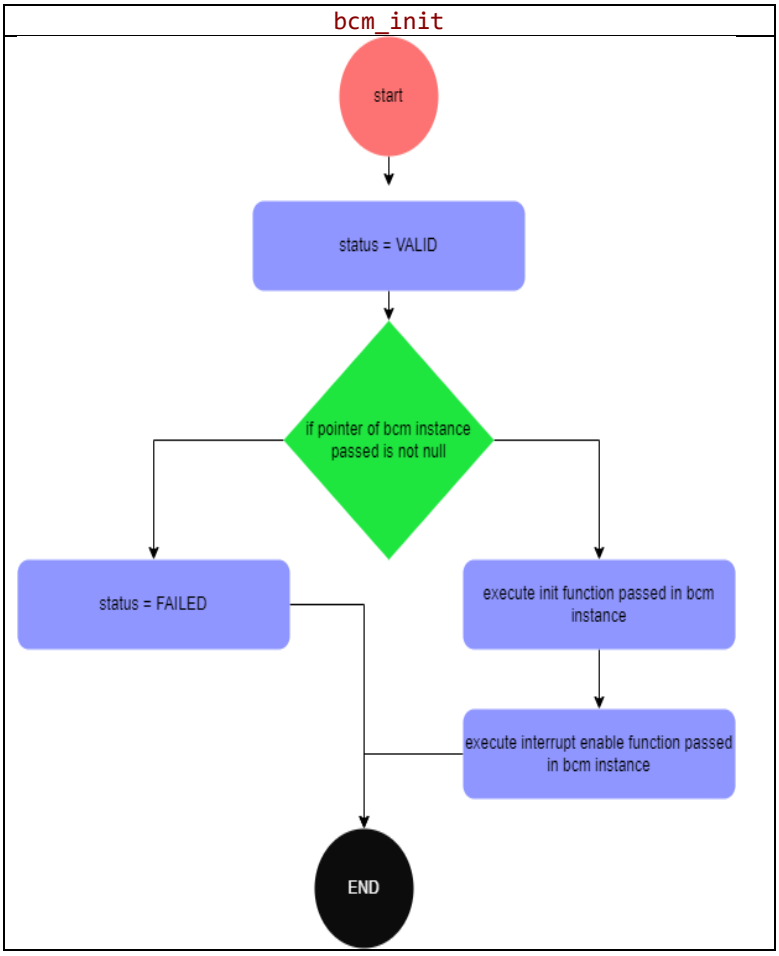
```
Dio_ErrorStatus LED_init(PORT_NUM portNum , PIN_NUM pinNum);
Dio_ErrorStatus LED_on (PORT_NUM portNum , PIN_NUM pinNum);
Dio_ErrorStatus LED_off(PORT_NUM portNum , PIN_NUM pinNum);
```

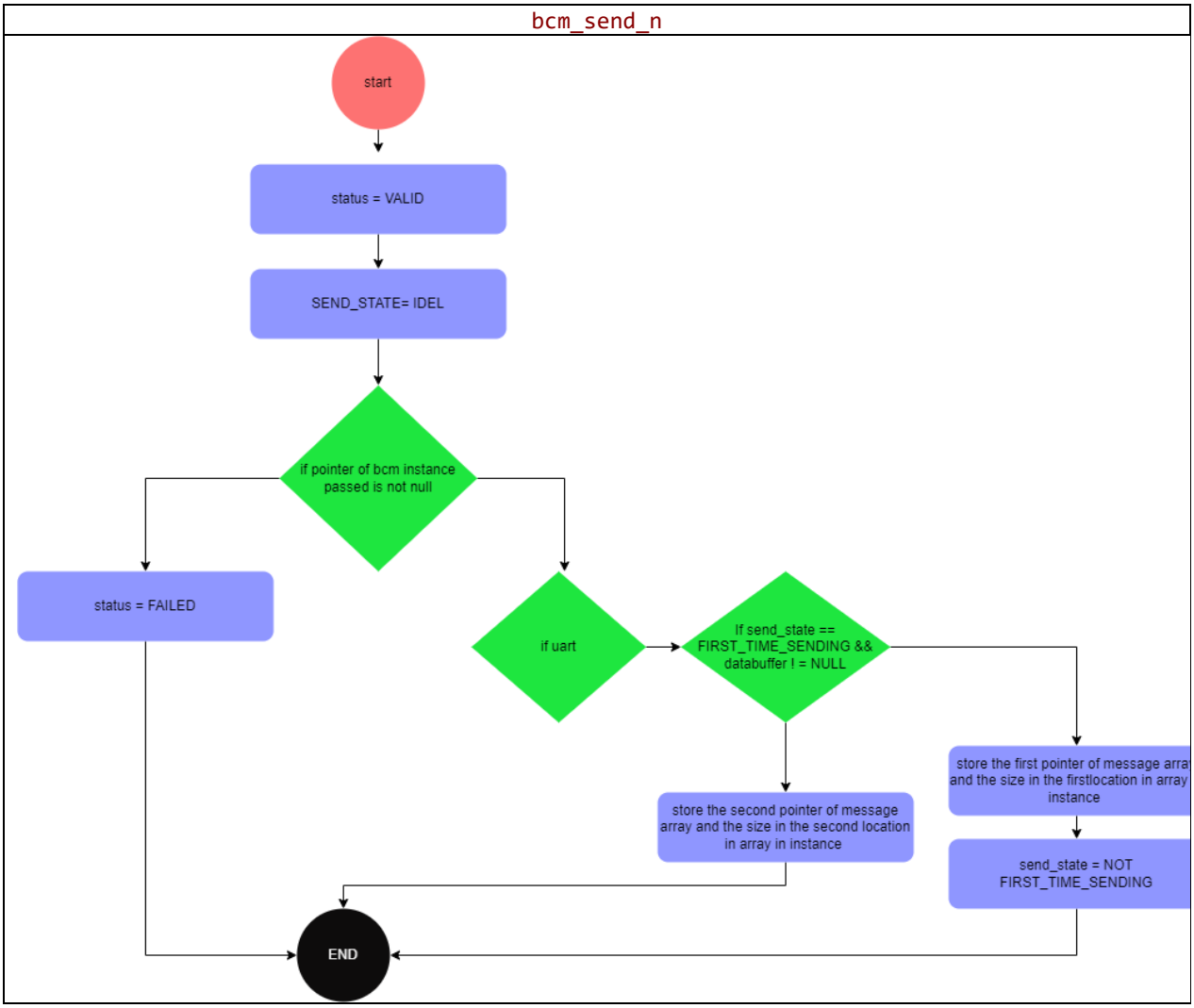


SERVICE

BCM

```
enu_sysem_status_t bcm_init(str_bcm_instance_t *ptr_bcm_instance_t);
enu_sysem_status_t bcm_denit(str_bcm_instance_t *ptr_bcm_instance_t);
enu_bcm_send_state_t bcm_send( str_bcm_instance_t *ptr_bcm_instance_t ,uint8_t *
ptr_buffer);
enu_bcm_send_state_t bcm_send_n( str_bcm_instance_t *ptr_bcm_instance_t , uint8_t *
ptr_buffer,uint8_t a_buffer_length);
enu_bcm_send_state_t bcm_disbatcher(str_bcm_instance_t *ptr_bcm_instance_t ,
enu_bcm_first_use a_first_use);
```




bcm_send_n
start
status = VALID
SEND_STATE= IDEL
if pointer of bcm instance passed is not null
status = FAILED
if uart
If send_state == FIRST_TIME_SENDING && databuffer ! = NULL
store the second pointer of message array and the size in the second location in array in instance
store the first pointer of message arra and the size in the firstlocation in array instance
send_state = NOT FIRST_TIME_SENDING
END

Configurations :

```c
typedef struct {

    void (*ptr_func_init)();
    void (*ptr_func_interrupt_enable)();
    void (*ptr_func_denit)();
    void (*ptr_func_interrupt_disable)();

}str_bcm_instance_t;

typedef enum {
    READY,
    TX_DONE,
    RX_DONE,
}enu_uart_interrupt_flag;



extern str_bcm_instance_t str_bcm_init_t;
extern enu_uart_interrupt_flag gl_uart_state ;
extern enu_uart_interrupt_flag gl_uart_state2 ;

void uart_isr_send_func (void);

void uart_isr_recieve_func (void);
```

```c
typedef enum{

    _INVALID_PROTOCOL_,
    BCM_VALID,
    BCM_FAILD
}enu_sysem_status_t;

typedef enum{
    _UART_,
    _SPI_,
    _I2C_,
    _NOT_SUPPORTED_
}enu_bcm_communciation_protocol;


typedef enum{


    IDLE,
    INVALID_STATE,
    INTERRUPT_FAILED,
    SENDING,
    RECEVING,
    DATA_SENT_OK,
    DATA_SENT_FAILD,
    FIRST_TIME_SENDING,
    NOT_FIRST_TIME_SENDING,
    SEND_FAILD_NULL_PTR,
    BUFFER_SENT_OK,
    BUFFER_RECEIVED_OK,
}enu_bcm_send_state_t;
```

```c
typedef enum{
    SENDING_FIRST = 1,
    RECEVING_FIRST

}enu_bcm_first_use;

#define  FIRST_BUFFER   0
#define  FIRST_ELEMENT  0

typedef struct {

    uint8_t  *ptr_data_buffer[200];
    uint16  u8_dataSize[200];
    uint8_t ptr_dataBASE[20];
}str_sending_receving_queue_t;
```

```c
enu_uart_interrupt_flag  gl_uart_state = READY ;
enu_uart_interrupt_flag  gl_uart_state2 = READY ;
str_bcm_instance_t str_bcm_init_t = {

    uart_init,
    uart_TX_RX_interrupt_enable,
};


//--------------------------- UART INTERRUPT HANDLING -------------------- //

void uart_isr_send_func (void){

    gl_uart_state = TX_DONE ;

}

void uart_isr_recieve_func (void){

    gl_uart_state2 = RX_DONE ;


}
```