



**BASIC COMMUNICATION MANGER (BCM)**



PREPARED BY

MOHAMMED ABDULWAHAB HEGAZY

1. [Project Introduction 2](#_TOC_250039)
   1. [Project Components 2](#_TOC_250038)
      1. Circuit [Schematic 2](#_TOC_250032)
   2. [System Requirements 3](#_TOC_250037)
2. [High Level Design 7](#_TOC_250036)
   1. [System Architecture 7](#_TOC_250035)
      1. [Layered Architecture 7](#_TOC_250033)
      2. [System modules 8](#_TOC_250033)
   2. [Modules Description 9](#_TOC_250031)
      1. [DIO Module 9](#_TOC_250030)
      2. [UART Module 9](#_TOC_250029)
      3. [LED Module 9](#_TOC_250028)
      4. [BCM Module 10](#_TOC_250027)
   3. [Drivers API’s 10](#_TOC_250025)

2.3.1[MCAL APIs 10](#_TOC_250023)

2.3.1.1DIO Driver 10

2.3.1.2UART Driver 11

* + 1. [HAL APIs 12](#_TOC_250022)
       1. LED APIs 12
    2. [SERVICE APIs 13](#_TOC_250021)
       1. BCM APIs 13

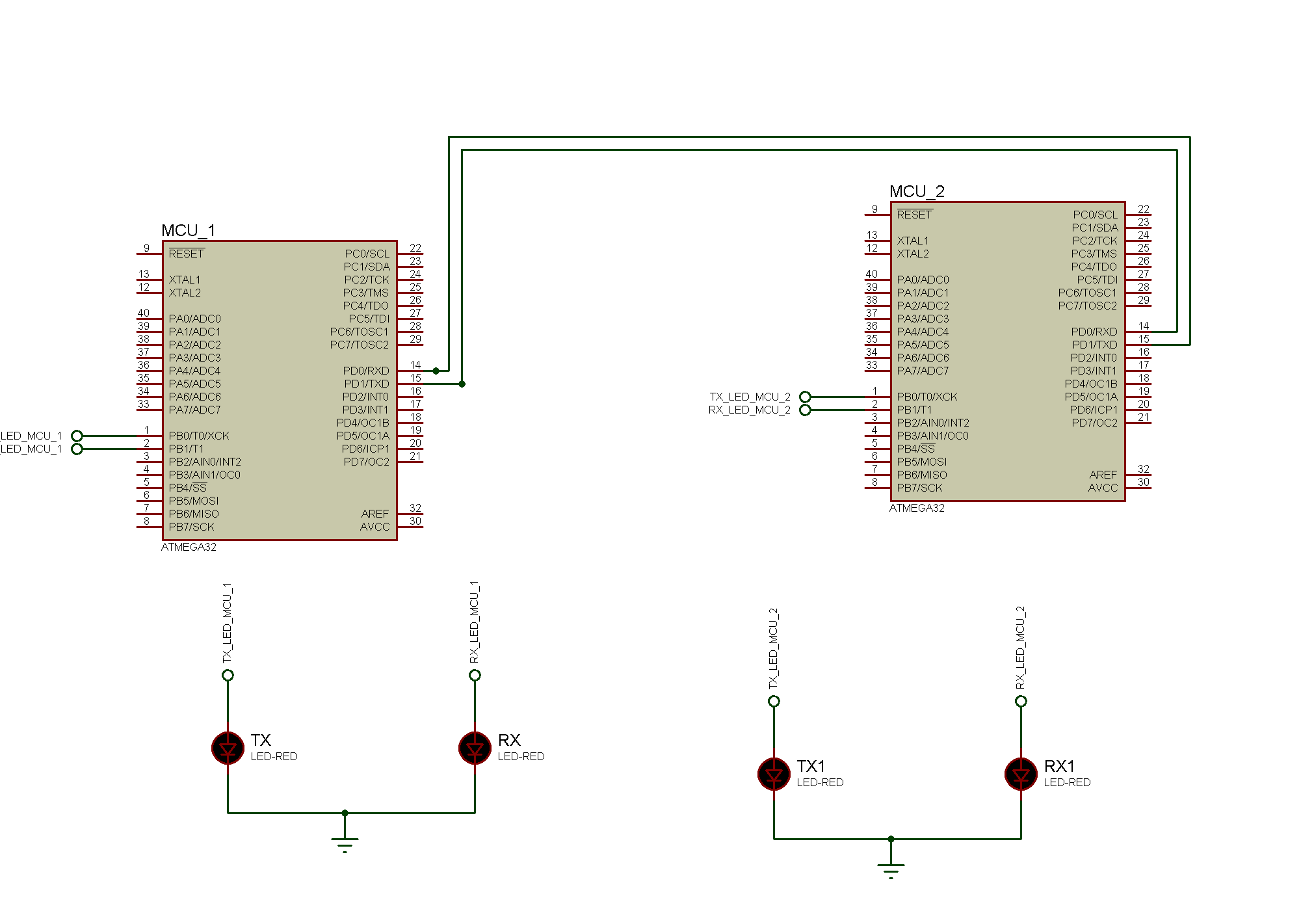
[2.3.4. APP APIs 15](#_TOC_250020)

1. [Low Level Design 15](#_TOC_250019)
   1. [MCAL Layer 15](#_TOC_250018)
      1. [DIO’s Flowcharts 15](#_TOC_250017)
   2. [HAL Layer 19](#_TOC_250016)
      1. [LED’s Flowcharts 19](#_TOC_250015)
   3. [SERV Layer 20](#_TOC_250014)
      1. [BCM’s Flowcharts 20](#_TOC_250013)
2. [Configurations 27](#_TOC_250006)
   1. [UART Configurations 27](#_TOC_250005)
   2. .[LED Configurations 30](#_TOC_250004)

4.[3. BCM Configurations](#_TOC_250004) 31

1. **Project Introduction**

Basic communication manager is responsible for all the communication done by the application layer with different communication protocols but with more abstracted way.

* 1. **Project Components**
* ATMEGA32
* LEDS
  + 1. **Circuit Schematic**
  1. **SYSTEM REQUIRMENTS**

System Requirements:

1. The BCM has the capability to send and receive any data with a maximum length of 65535 bytes (Maximum of unsigned two bytes variable).
2. It can use any communication protocol with the support of Send, Receive or both.
3. Implement **bcm\_Init** using the below table. This function will initialize the corresponding serial communication protocol

|  |  |
| --- | --- |
| **Function Name** | bcm\_init |
| **Syntax** | enu\_system\_status\_t BCM\_init(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance); |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non reentrant |
| **Parameters (in)** | Ptr\_str\_bcm\_instance :the refrence of a bcm structure object |
| **Parameters (out)** | None |
| **Parameters (in, out)** | None |
| **Return** | typedef enum {  BCM\_E\_OK=0,  BCM\_E\_NOK=2,  }enu\_system\_status\_t; |

1. Implement **bcm\_deinit** using the below table. This function will uninitialize the corresponding BCM instance, (instance: is the communication channel)

|  |  |
| --- | --- |
| **Function Name** | **bcm\_deinit** |
| **Syntax** | enu\_system\_status\_t BCM\_deInit(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance); |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non reentrant |
| **Parameters (in)** | Ptr\_str\_bcm\_instance :the refrence of a bcm structure object |
| **Parameters (out)** | None |
| **Parameters (in, out)** | None |
| **Return** | typedef enum {  BCM\_E\_OK=0,  BCM\_E\_NOK=2,  }enu\_system\_status\_t; |

1. Implement **bcm\_send** that will send only 1 byte of data over a specific BCM instance

|  |  |
| --- | --- |
| **Function Name** | **bcm\_send** |
| **Syntax** | enu\_system\_status\_t BCM\_send(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance , uint8\_t u8\_a\_data); |
| **Sync/Async** | Asynchronous(cause using non-blocking) |
| **Reentrancy** | Non reentrant |
| **Parameters (in)** | Ptr\_str\_bcm\_instance :the refrence of a bcm structure object  u8\_a\_data: data to be send |
| **Parameters (out)** | None |
| **Parameters (in, out)** | None |
| **Return** | typedef enum {  BCM\_E\_OK=0,  BCM\_E\_NOK=2,  }enu\_system\_status\_t; |

6. Implement **bcm\_send\_n** will send more than one byte with a length n over a specific BCM instance

|  |  |
| --- | --- |
| **Function Name** | **bcm\_send\_n** |
| **Syntax** | enu\_system\_status\_t BCM\_send\_n(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance , uint8\_t \*u8Arr\_a\_stringData, uint16\_t u16\_a\_stringSize); |
| **Sync/Async** | Asynchronous(cause using non-blocking) |
| **Reentrancy** | Re-entrant |
| **Parameters (in)** | Ptr\_str\_bcm\_instance :the refrence of a bcm structure object  u8Arr\_a\_stringData: string to be send  u16\_a\_stringSize) : string size |
| **Parameters (out)** | None |
| **Parameters (in, out)** | None |
| **Return** | typedef enum {  BCM\_E\_OK=0,  BCM\_E\_NOK=2,  }enu\_system\_status\_t; |

1. Implement **bcm\_receive** will receive only 1 byte of data over a specific BCM instance

|  |  |
| --- | --- |
| **Function Name** | **bcm\_receive** |
| **Syntax** | enu\_system\_status\_t BCM\_recieve(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance , uint8\_t \*u8\_a\_data); |
| **Sync/Async** | Asynchronous(cause using non-blocking) |
| **Reentrancy** | Re-entrant |
| **Parameters (in)** | Ptr\_str\_bcm\_instance :the reference of a bcm structure object |
| **Parameters (out)** | None |
| **Parameters (in, out)** | u8Arr\_a\_stringData:refrenc to the data be stored at |
| **Return** | typedef enum {  BCM\_E\_OK=0,  BCM\_E\_NOK=2,  }enu\_system\_status\_t; |

1. Implement **bcm\_ receive\_n** will receive more than one byte with a length n over a specific BCM instance

|  |  |
| --- | --- |
| **Function Name** | **bcm\_send\_n** |
| **Syntax** | enu\_system\_status\_t BCM\_recieve\_n(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance , uint8\_t \*u8Arr\_a\_stringData, uint16\_t u16\_a\_stringSize); |
| **Sync/Async** | Asynchronous(cause using non-blocking) |
| **Reentrancy** | Re-entrant |
| **Parameters (in)** | Ptr\_str\_bcm\_instance :the refrence of a bcm structure object  u16\_a\_stringSize) : string size |
| **Parameters (out)** | None |
| **Parameters (in, out)** | u8Arr\_a\_stringData: reference to the string which will store the data at |
| **Return** | typedef enum {  BCM\_E\_OK=0,  BCM\_E\_NOK=2,  }enu\_system\_status\_t; |

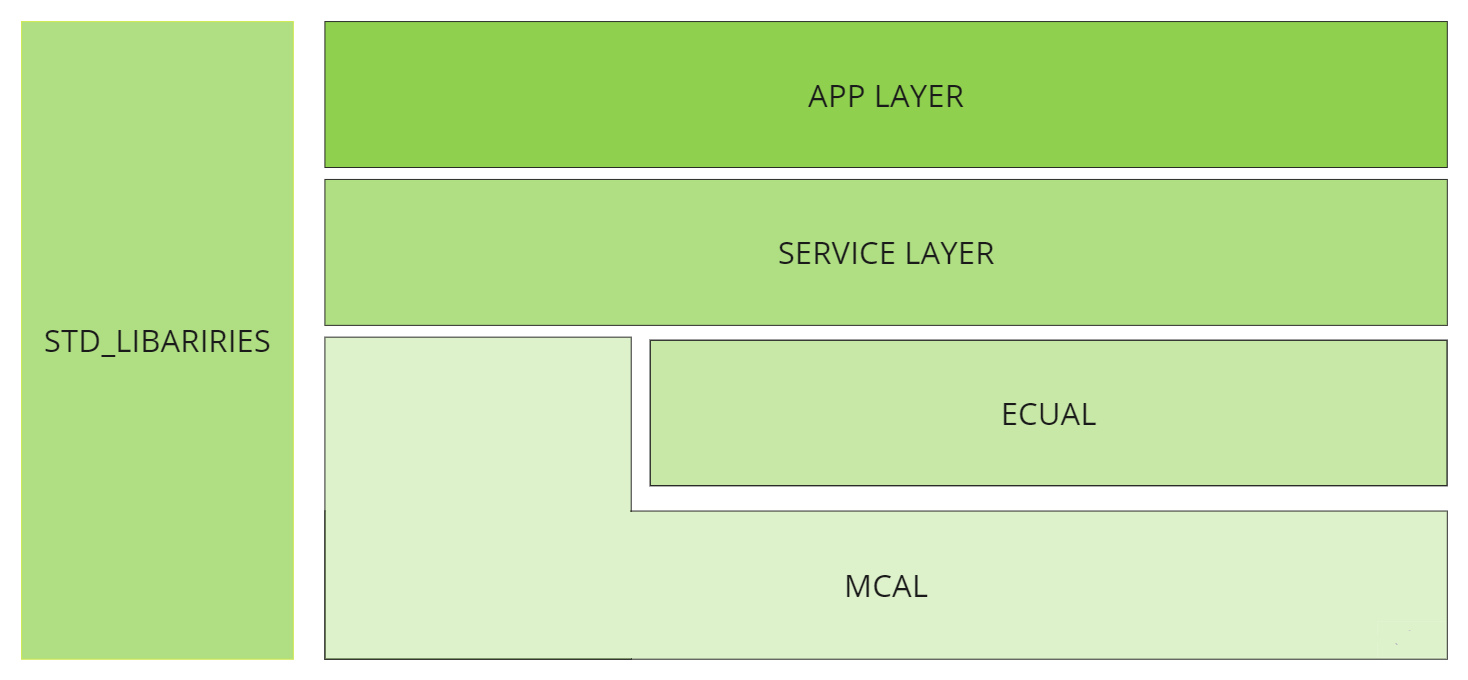
1. Implement **bcm\_dispatcher** will execute the periodic actions and notifies the user with the needed events over a specific BCM instance

|  |  |
| --- | --- |
| **Function Name** | **bcm\_send\_n** |
| **Syntax** | enu\_system\_status\_t BCM\_dispatcher(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance ); |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Re-entrant |
| **Parameters (in)** | Ptr\_str\_bcm\_instance :the refrence of a bcm structure object |
| **Parameters (out)** | None |
| **Parameters (in, out)** | None |
| **Return** | typedef enum {  BCM\_E\_OK=0,  BCM\_E\_NOK=2,  }enu\_system\_status\_t; |

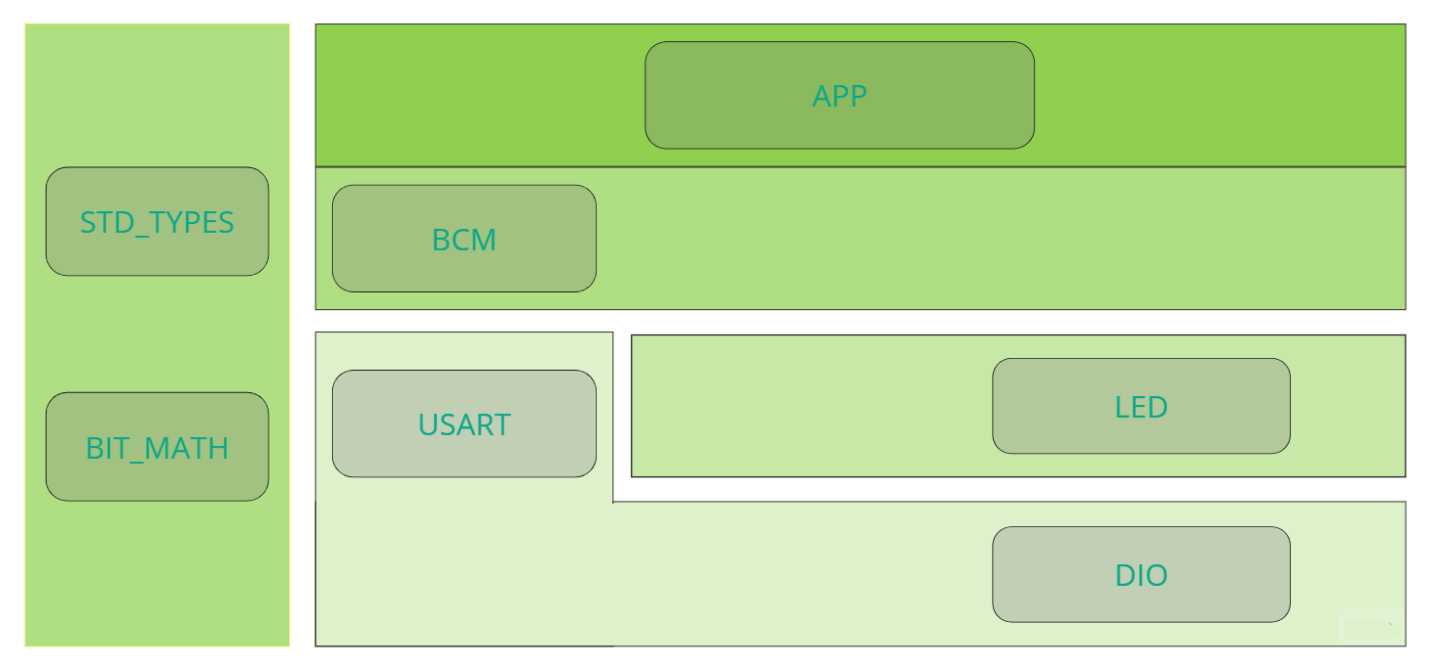
**2. High Level Design**

**2.1. System Architecture**

**2.1.1. Layered Architecture**



**2.1.2. System modules**

****

**2.2. Modules Description**

**2.2.1. DIO Module**

DIO module: Digital Input/output module is used to drive an output digital logic or read digital logic from external devices.

**2.2.2. USART Module**

A USART (universal synchronous/asynchronous receiver/transmitter) is hardware that enables a device to communicate using serial protocols. It can function in a slower asynchronous mode, like a universal asynchronous receiver/transmitter (UART), or in a faster synchronous mode with a clock signal.

**2.2.3. LED Module**

LED is a compact and versatile solution designed to control LEDs in various applications. With its support for ATmega32 microcontroller, it offers seamless integration and efficient LED management. The module provides easy-to-use functions for controlling individual LEDs, allowing for dynamic lighting effects and customization. Its compact design and optimized code ensure minimal resource utilization while delivering reliable and precise LED control.

**2.2.4. BCM Module**

Basic communication manager is responsible for all the communication done by the application layer with different communication protocols but with more abstracted way.

**2.3. Drivers API’s**

**2.3.1 MCAL API’s**

**2.3.1.1 DIO API’s**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Function Declarations \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* @brief Initialize the direction of specific pin @ref direction\_t

\* @param \_pin\_config A Reference of the pin configuration @pin\_config\_t

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType DIO\_pin\_direction\_intialize(const pin\_config\_t \*pin\_config\_ptr,direction\_t a\_direction);

/\*\*

\* @brief Write the logic of specific pin @ref logic\_t

\* @param \_pin\_config A Reference of the pin configuration @pin\_config\_t

\* @param logic

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType DIO\_pin\_write\_logic(const pin\_config\_t \*pin\_config\_ptr,const logic\_t a\_logic);

/\*\*

\* @brief Read the logic of specific pin @ref logic\_t

\* @param \_pin\_config A Reference of the pin configuration @pin\_config\_t

\* @param logic

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType DIO\_pin\_read\_logic(const pin\_config\_t \*pin\_config\_ptr, logic\_t \*logic\_ptr);

/\*\*

\* @brief Toggle the logic of specific pin @ref logic\_t

\* @param \_pin\_config A Reference of the pin configuration @pin\_config\_t

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType DIO\_pin\_toggle\_logic(const pin\_config\_t \*pin\_config\_ptr);

**2.3.1.2 USART API’s**

/\*

\* Description : initialize the USART driver

\* @param A Reference of the USART driver's configuration structure

\* @return Std\_ReturnType: status of the function

\* USART\_E\_OK :the function done successfully

\* USART\_E\_NOK :the function has issues performing the function

\*/

u8\_usartErorrState\_t USART\_init(const st\_usart\_config\_t \*stPtr\_a\_usartConfig);

/\*

\* Description : De-initialize the USART driver

\* @param A Reference of the USART driver's configuration structure

\* @return Std\_ReturnType: status of the function

\* USART\_E\_OK :the function done successfully

\* USART\_E\_NOK :the function has issues performing the function

\*/

u8\_usartErorrState\_t USART\_DeInit(const st\_usart\_config\_t \*stPtr\_a\_usartConfig);

/\*

\* Description : Send one byte via USART bus

\* @param u8\_a\_data : The data to be send

\* stPtr\_a\_usartConfig : A Reference of the USART driver's configuration structure

\* @return Std\_ReturnType: status of the function

\* USART\_E\_OK :the function done successfully

\* USART\_E\_NOK :the function has issues performing the function

\*/

u8\_usartErorrState\_t USART\_sendData(const st\_usart\_config\_t \*stPtr\_a\_usartConfig , uint8\_t u8\_a\_data);

/\*

\* Description : Receive one byte via USART bus

\* @param u8Ptr\_a\_data: A Reference of the container of the received data

\* stPtr\_a\_usartConfig:A Reference of the USART driver's configuration structure

\* @return Std\_ReturnType: status of the function

\* USART\_E\_OK :the function done successfully

\* USART\_E\_NOK :the function has issues performing the function

\*/

u8\_usartErorrState\_t USART\_reciveData(const st\_usart\_config\_t \*stPtr\_a\_usartConfig , uint8\_t \*const u8Ptr\_a\_data);

/\*

\* Description : Send string via USART bus

\* @param The data string -array of characters- to be send

\* stPtr\_a\_usartConfig:A Reference of the USART driver's configuration structure

\* @return Std\_ReturnType: status of the function

\* USART\_E\_OK :the function done successfully

\* USART\_E\_NOK :the function has issues performing the function

\*/

u8\_usartErorrState\_t USART\_sendString(const st\_usart\_config\_t \*stPtr\_a\_usartConfig , uint8\_t \*u8Arr\_a\_stringOfData , uint16\_t u16\_a\_stringSize);

/\*

\* Description : Receive string via USART bus

\* @param The data string -array of characters- to store the received data string

\* stPtr\_a\_usartConfig:A Reference of the USART driver's configuration structure

\* @return Std\_ReturnType: status of the function

\* USART\_E\_OK :the function done successfully

\* USART\_E\_NOK :the function has issues performing the function

\*/

u8\_usartErorrState\_t USART\_reciveString(const st\_usart\_config\_t \*stPtr\_a\_usartConfig ,uint8\_t \*const u8Arr\_a\_stringOfData , uint16\_t u16\_a\_stringSize);

/\*

\* Description : Call the Call Back function in the application after transmissions did its job

\* @param A pointer to function

\* @return status of the function

\* USART\_E\_OK :the function done successfully

\* USART\_E\_NOK :the function has issues performing the function

\*/

u8\_usartErorrState\_t USART\_setCallBackTx( Fptr\_usartCallBack\_t Fptr\_a\_TxCallBack);

/\*

\* Description : Call the Call Back function in the application after Reception did its job

\* @param A pointer to function

\* @return status of the function

\* USART\_E\_OK :the function done successfully

\* USART\_E\_NOK :the function has issues performing the function

\*/

u8\_usartErorrState\_t USART\_setCallBackRx( Fptr\_usartCallBack\_t Fptr\_a\_RxCallBack);

**2.3.2. HAL API’s**

**2.3.1.1. LED API’s**

/\*\*

\* @breif Initialize The led by configuring the pin as output and write low

\* @param Led The reference of the led module configuration

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType LED\_initialize(const led\_t \*led\_ptr);

/\*\*

\* @breif Turn the led on

\* @param led The reference of the led module configuration

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType LED\_turn\_on(const led\_t \*led\_ptr);

/\*\*

\* @breif Turn the led off

\* @param led The reference of the led module configuration

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType LED\_turn\_off (const led\_t \*led\_ptr);

/\*\*

\* @breif Toggle the led

\* @param led The reference of the led module configuration

\* @return status of the function

\* E\_OK :the function done successfully

\* E\_NOT\_OK :the function has issues performing the function

\*/

Std\_ReturnType LED\_turn\_toggle (const led\_t \*led\_ptr);

**2.3.3. SERVICE API’s**

**2.3.3.1 BCM API’s**

/\*

\* Description : initialize the BCM communication unit

\* @param A Reference of the BCM driver's configuration structure

\* @return enu\_system\_status\_t: status of the function

\* BCM\_E\_OK :the function done successfully

\* BCM\_E\_NOK :the function has issues performing the function

\*/

enu\_system\_status\_t BCM\_init(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance);

/\*

\* Description : De-initialize the BCM communication unit

\* @param A Reference of the BCM driver's configuration structure

\* @return enu\_system\_status\_t: status of the function

\* BCM\_E\_OK :the function done successfully

\* BCM\_E\_NOK :the function has issues performing the function

\*/

enu\_system\_status\_t BCM\_deinit(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance);

/\*

\* Description : Send one byte via BCM

\* @param

\* str\_ptr\_a\_bcm\_inctance : A Reference of the BCM driver's configuration structure

\* u8\_a\_data : The data to be send

\* @return enu\_system\_status\_t: status of the function

\* BCM\_E\_OK :the function done successfully

\* BCM\_E\_NOK :the function has issues performing the function

\*/

enu\_system\_status\_t BCM\_send(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance , uint8\_t u8\_a\_data);

/\*

\* Description : Receive one byte via BCM

\* @param

\* str\_ptr\_a\_bcm\_inctance : A Reference of the BCM driver's configuration structure

\* u8\_a\_data : A Reference of The variable which will be store the recived byte at

\* @return enu\_system\_status\_t: status of the function

\* BCM\_E\_OK :the function done successfully

\* BCM\_E\_NOK :the function has issues performing the function

\*/

enu\_system\_status\_t BCM\_recieve(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance , uint8\_t \*u8\_a\_data);

/\*

\* Description : Send Multiple bytes via BCM

\* @param

\* str\_ptr\_a\_bcm\_inctance : A Reference of the BCM driver's configuration structure

\* u8\_a\_data : The string to be send

\* u16\_a\_stringSize : The string size

\* @return enu\_system\_status\_t: status of the function

\* BCM\_E\_OK :the function done successfully

\* BCM\_E\_NOK :the function has issues performing the function

\*/

enu\_system\_status\_t BCM\_send\_n(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance , uint8\_t \*u8Arr\_a\_stringData, uint16\_t u16\_a\_stringSize);

/\*

\* Description : receive Multiple bytes via BCM

\* @param

\* str\_ptr\_a\_bcm\_inctance : A Reference of the BCM driver's configuration structure

\* u8\_a\_data : The string to stored the string at

\* u16\_a\_stringSize : The string size

\* @return enu\_system\_status\_t: status of the function

\* BCM\_E\_OK :the function done successfully

\* BCM\_E\_NOK :the function has issues performing the function

\*/

enu\_system\_status\_t BCM\_recieve\_n(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance , uint8\_t \*u8Arr\_a\_stringData, uint16\_t u16\_a\_stringSize);

/\*

\* Description : Call the Call Back function in the application after transmissions or reception is done

\* @paramstr\_ptr\_a\_bcm\_inctance : A Reference of the BCM driver's configuration structure

\* @return enu\_system\_status\_t: status of the function

\* BCM\_E\_OK :the function done successfully

\* BCM\_E\_NOK :the function has issues performing the function

\*/

enu\_system\_status\_t BCM\_setCallBack(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance );

/\*

\* Description :Will execute the periodic actions and notifies the user with the needed events over a specific BCM instance

\* @paramstr\_ptr\_a\_bcm\_inctance : A Reference of the BCM driver's configuration structure

\* @return enu\_system\_status\_t: status of the function

\* BCM\_E\_OK :the function done successfully

\* BCM\_E\_NOK :the function has issues performing the function

\*/

enu\_system\_status\_t BCM\_dispatcher(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance );

**2.3.3. APP API’s**

void APP\_init(void);

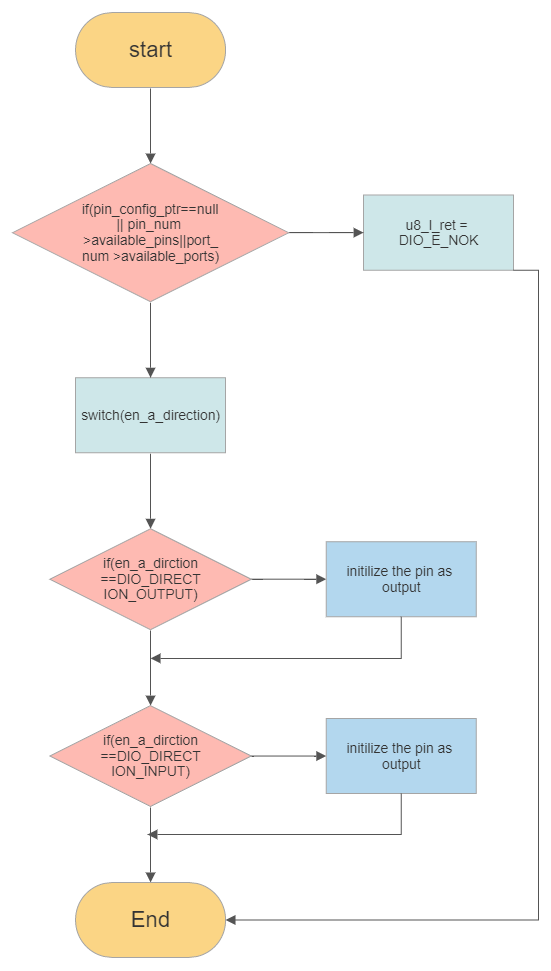
void APP\_start(void);

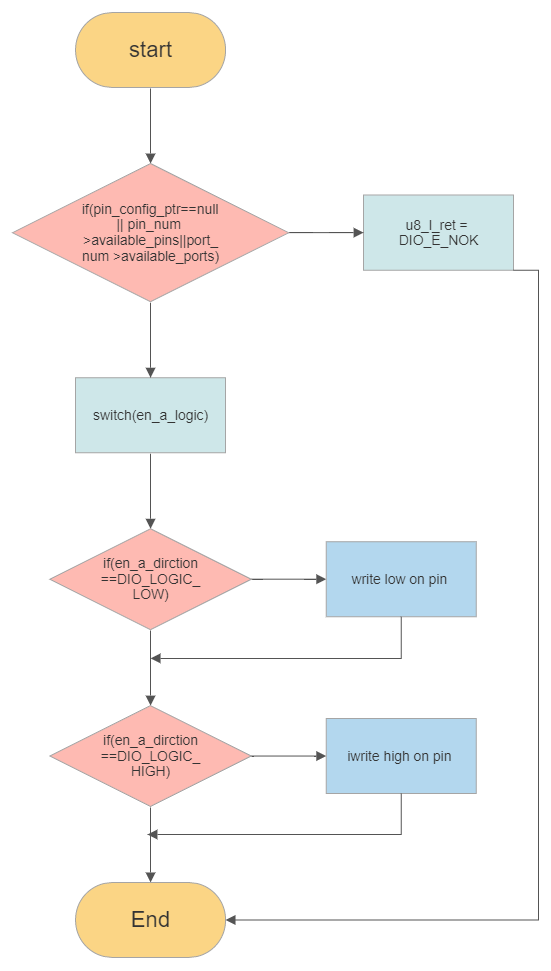
**3. Low Level Design**

**3.1. MCAL**

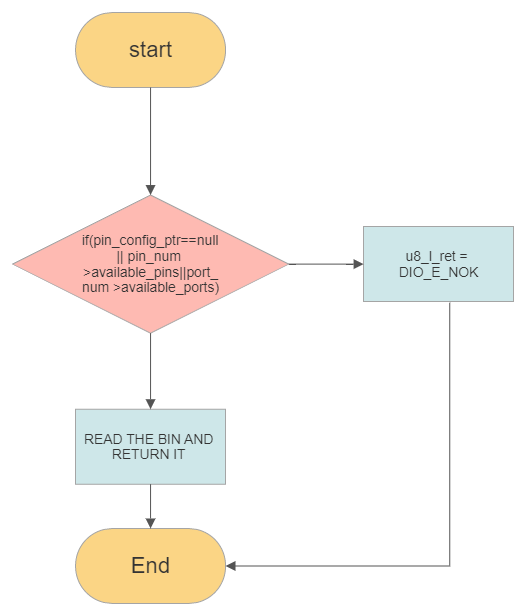
**3.1.1 DIO’S FLOWCHARTS**

Std\_ReturnType DIO\_pin\_direction\_intialize(const pin\_config\_t \*pin\_config\_ptr,direction\_t a\_direction);

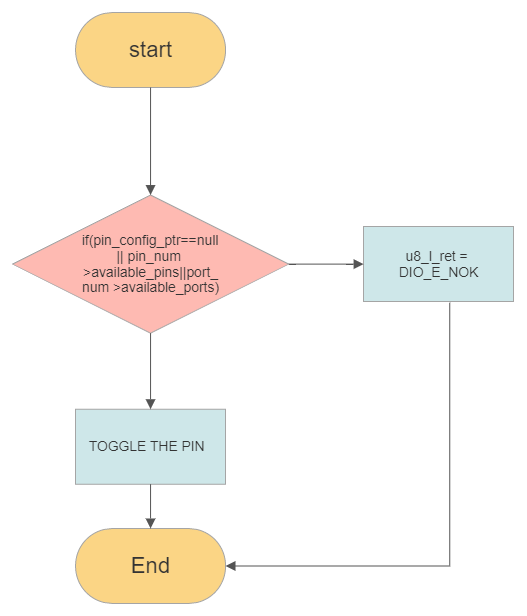


Std\_ReturnType DIO\_pin\_write\_logic(const pin\_config\_t \*pin\_config\_ptr,const logic\_t a\_logic);

Std\_ReturnType DIO\_pin\_read\_logic(const pin\_config\_t \*pin\_config\_ptr, logic\_t \*logic\_ptr);



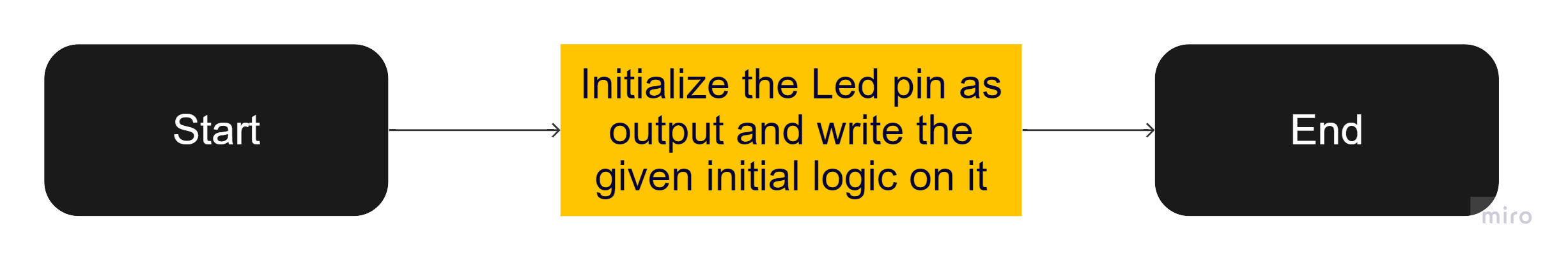
Std\_ReturnType DIO\_pin\_toggle\_logic(const pin\_config\_t \*pin\_config\_ptr);



**3.2. HAL**

**3.2.1. LED’S FLOWCHARTS**

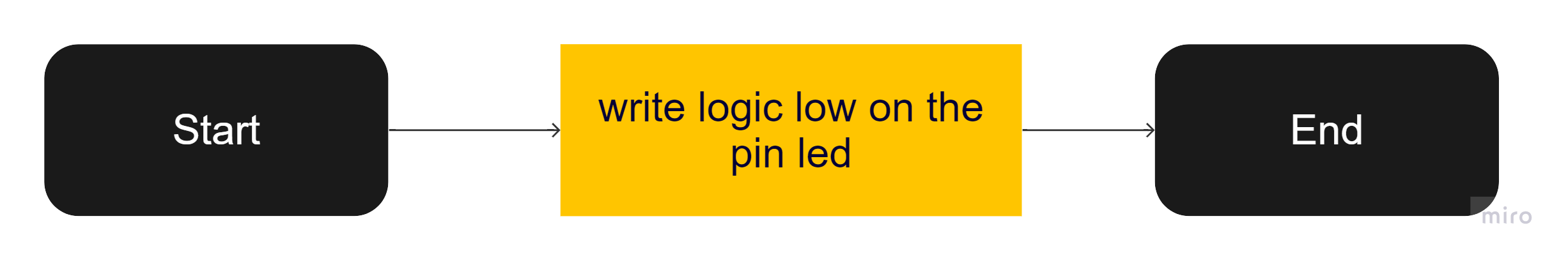
Std\_ReturnType LED\_initialize(const led\_t \*led\_ptr);

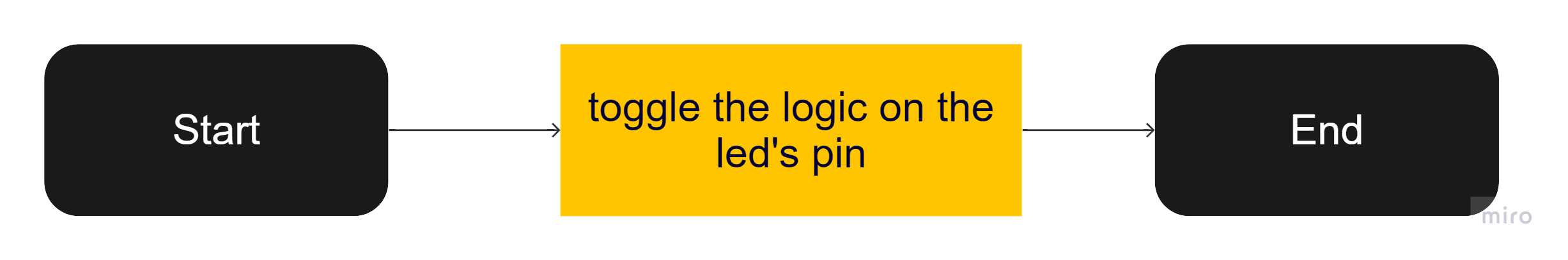
****

Std\_ReturnType LED\_turn\_on(const led\_t \*led\_ptr);

****

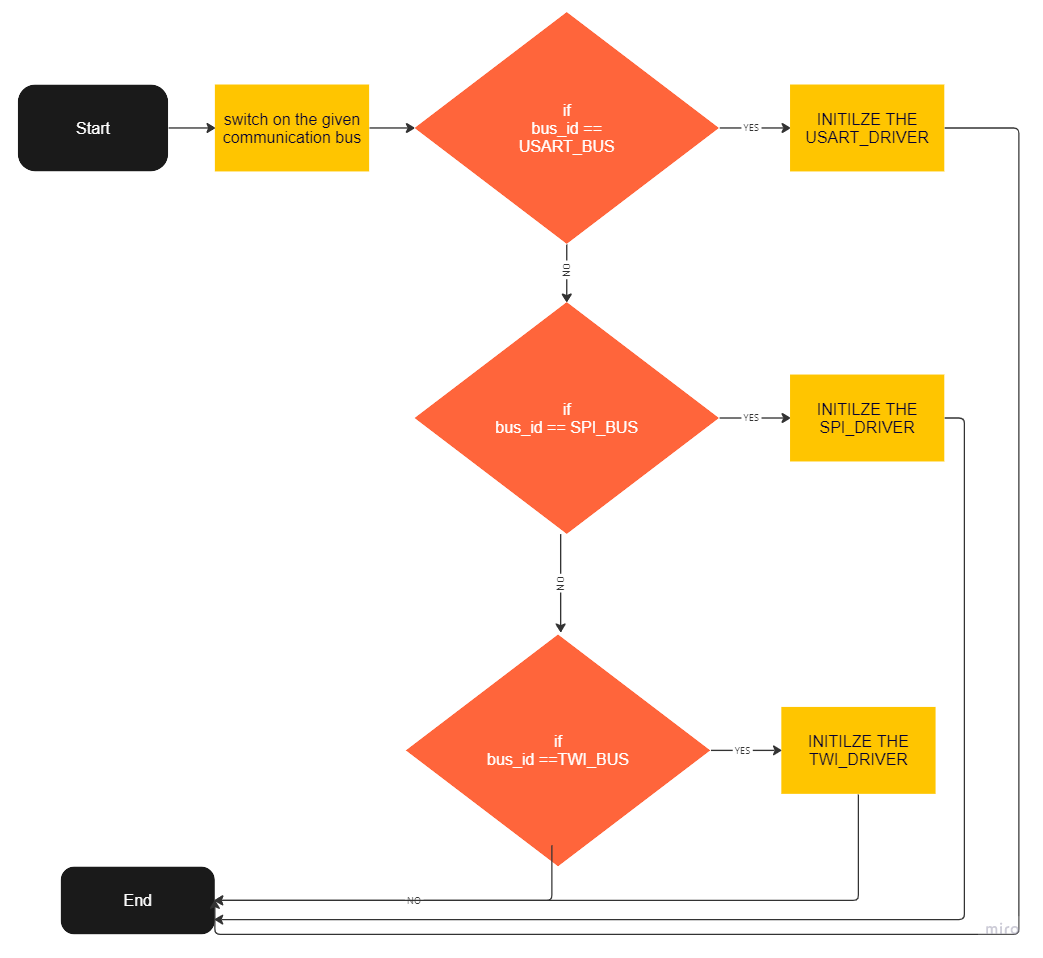
Std\_ReturnType LED\_turn\_off (const led\_t \*led\_ptr);

****Std\_ReturnType LED\_turn\_toggle (const led\_t \*led\_ptr);

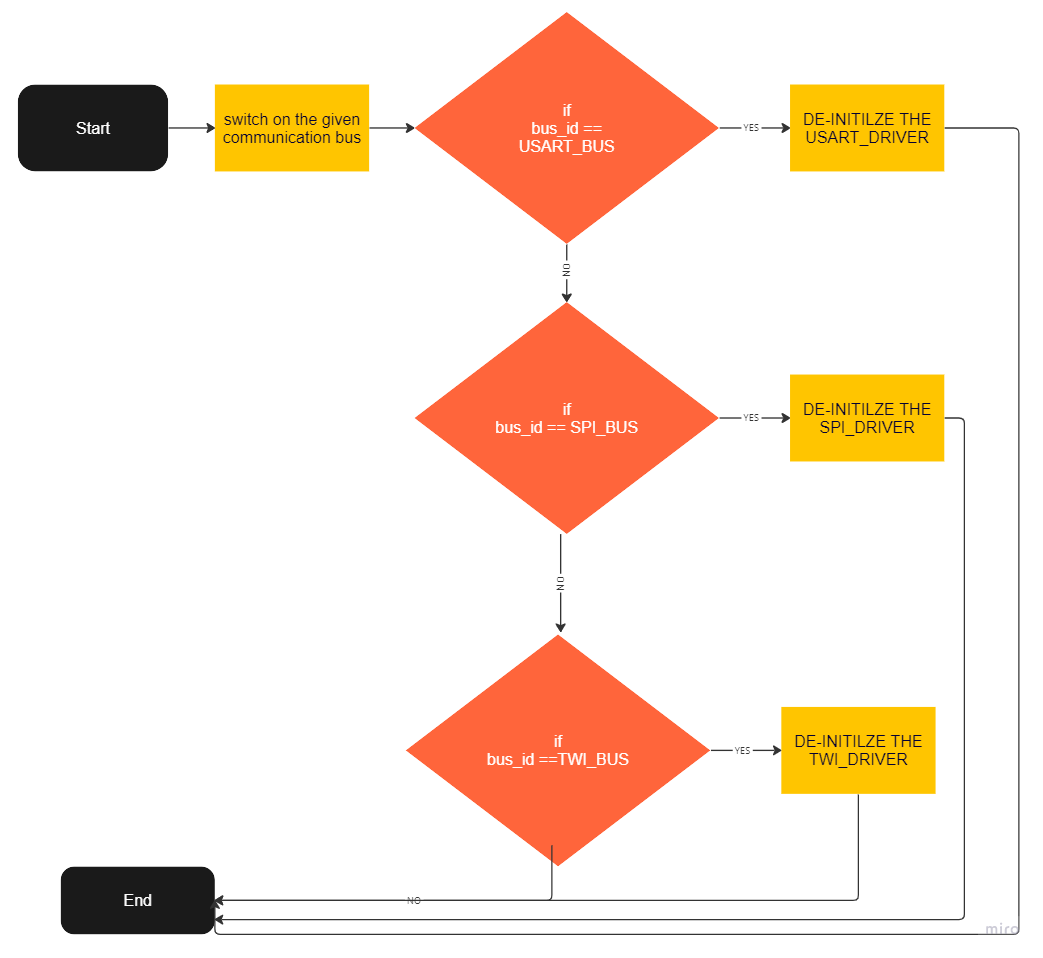


**3.2. SERVICE LAYER**

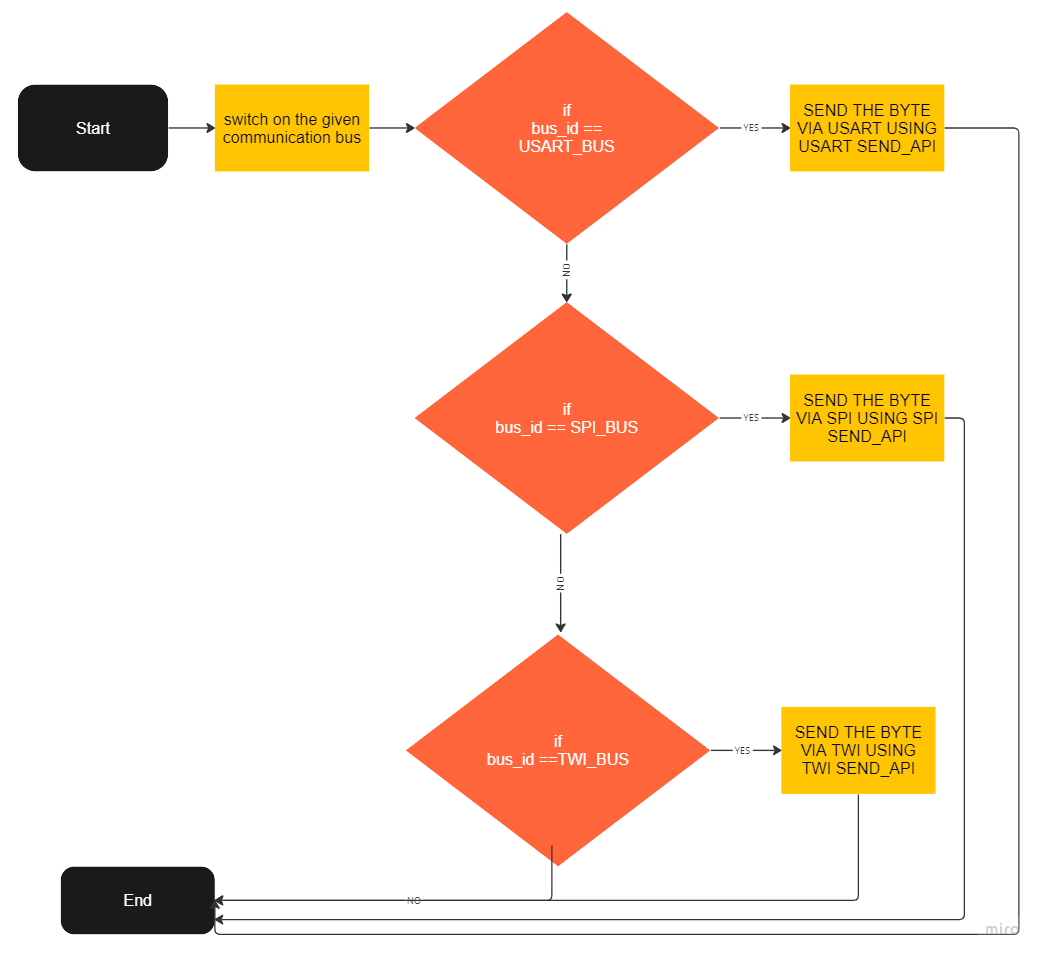
**3.2.1. BCM’S FLOWCHARTS**

****enu\_system\_status\_t BCM\_init(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance);

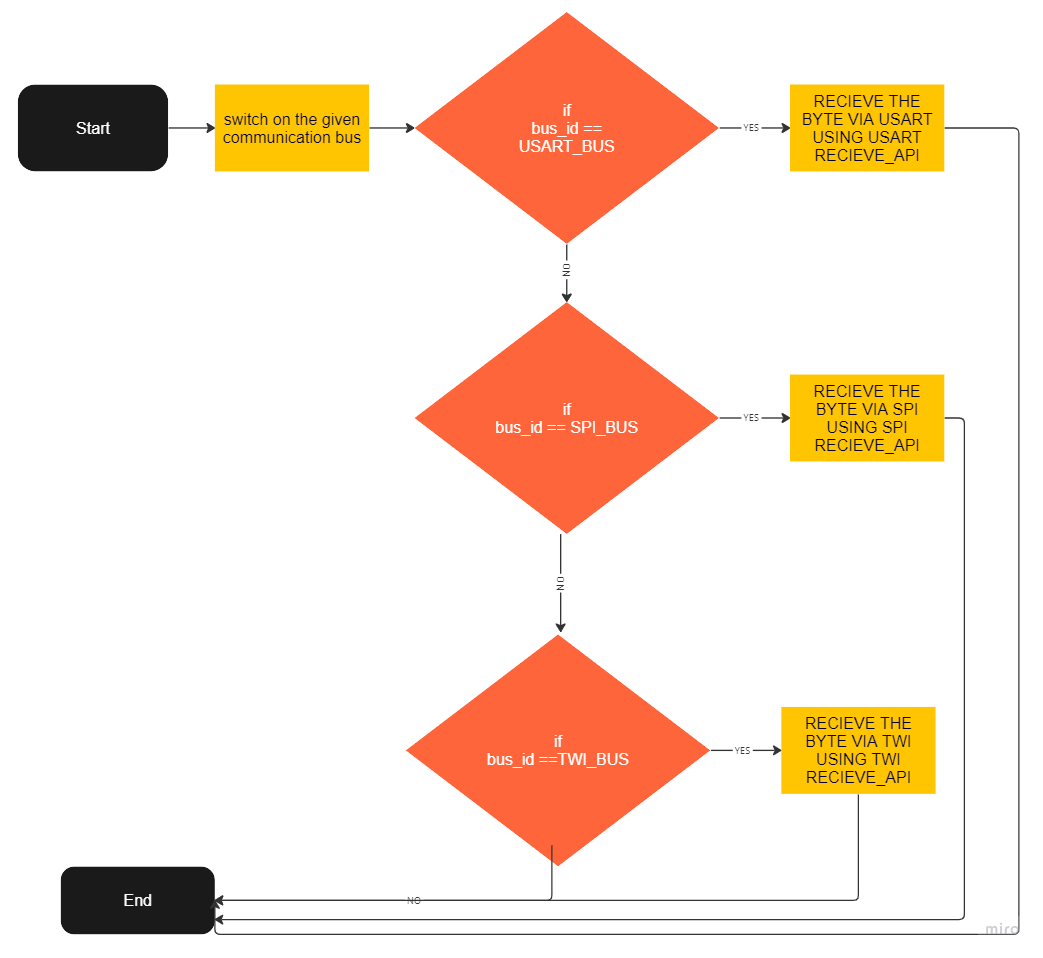
enu\_system\_status\_t BCM\_deinit(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance);



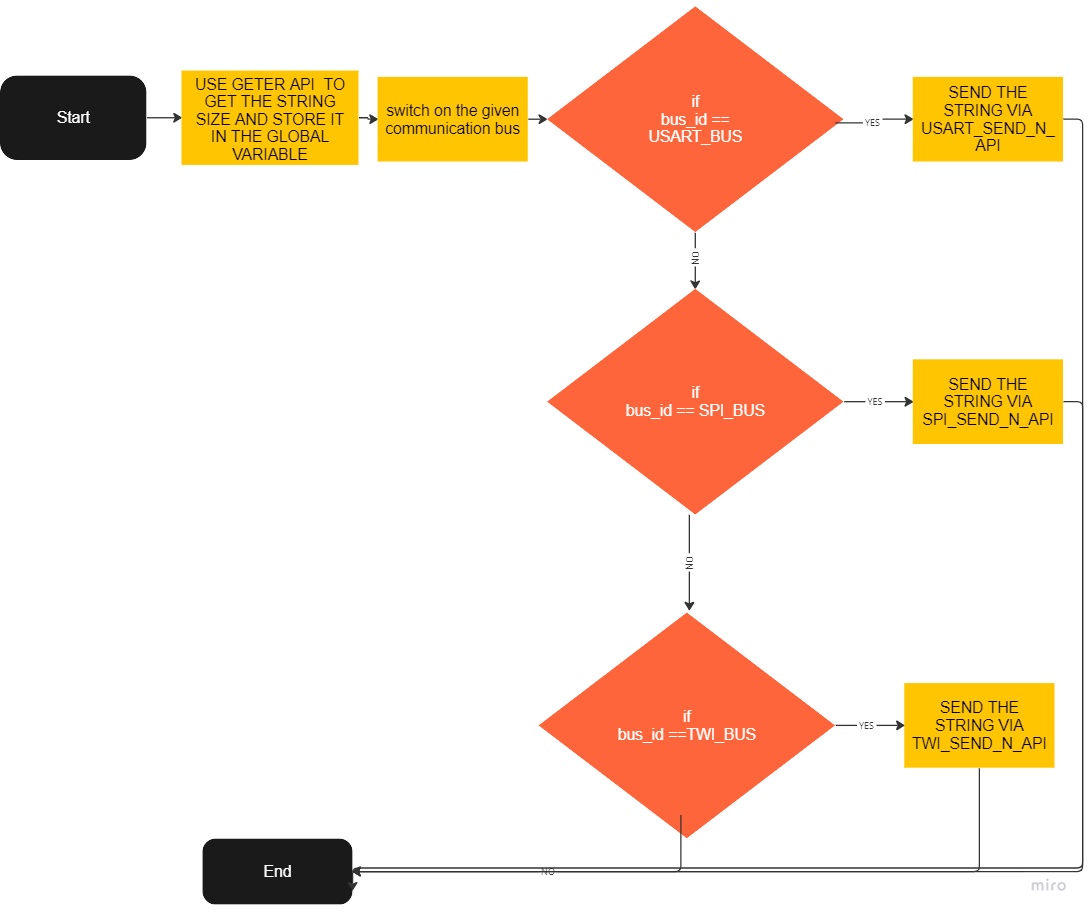
enu\_system\_status\_t BCM\_send(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance , uint8\_t u8\_a\_data);

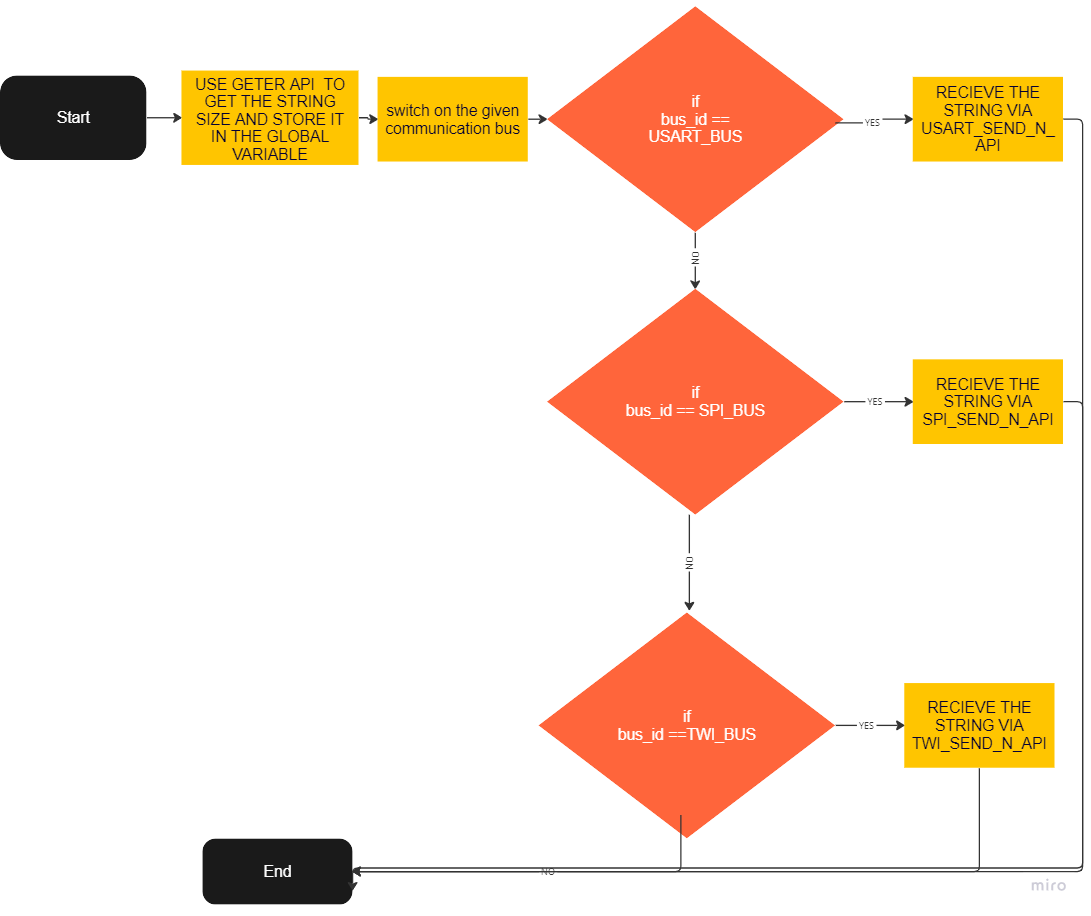


enu\_system\_status\_t BCM\_recieve(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance , uint8\_t \*u8\_a\_data);

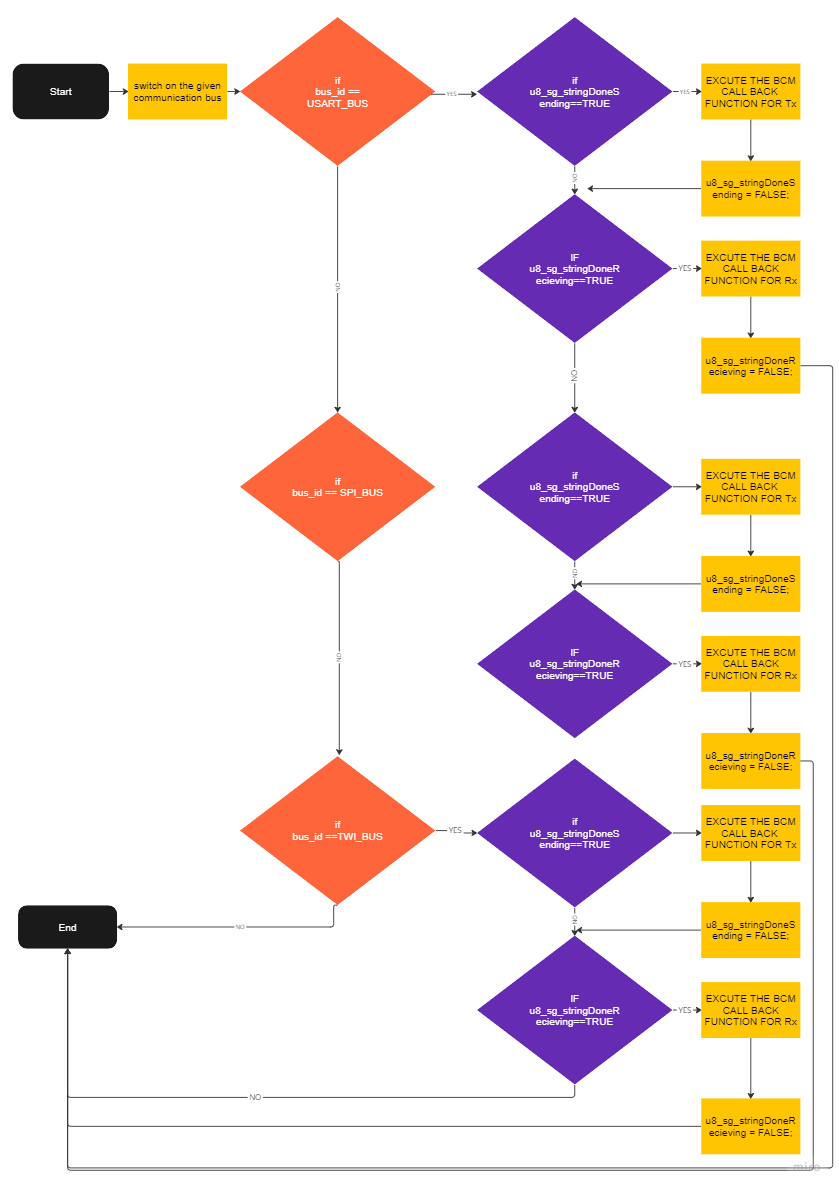


enu\_system\_status\_t BCM\_send\_n(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance , uint8\_t \*u8Arr\_a\_stringData, uint16\_t u16\_a\_stringSize);



enu\_system\_status\_t BCM\_recieve\_n(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance , uint8\_t \*u8Arr\_a\_stringData, uint16\_t u16\_a\_stringSize);

enu\_system\_status\_t BCM\_dispatcher(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance );

enu\_system\_status\_t BCM\_dispatcher(str\_bcm\_inctance\_t \*str\_ptr\_a\_bcm\_inctance );

**4. CONFIGURATION**

**4.1. USART CONFIGURATION**

/\* ------------------------------------------------------Data Type Declarations ----------------------------------------------------------\*/

typedef void (\*Fptr\_usartCallBack\_t) (void);

typedef uint8\_t u8\_usartMode\_t;

typedef uint8\_t u8\_usartTx\_enable\_t;

typedef uint8\_t u8\_usartRx\_enable\_t;

typedef uint8\_t u8\_usartTx\_interruptMask\_t;

typedef uint8\_t u8\_usartRx\_interruptMask\_t;

typedef uint8\_t u8\_usartStopBit\_t;

typedef uint8\_t u8\_usartParityBit\_t;

typedef uint8\_t u8\_usartDataSize\_t;

typedef uint8\_t u8\_usartTxClkPolarity\_t;

typedef uint8\_t u8\_usartRxClkPolarity\_t;

typedef uint8\_t u8\_usartErorrState\_t;

typedef struct{

u8\_usartMode\_t usartMode; /\*@ref u8\_usartMode\_t\*/

u8\_usartTx\_enable\_t usartTxEnable; /\*@ref u8\_usartTx\_enable\_t\*/

u8\_usartTx\_interruptMask\_t usartTxInterrupt; /\*@ref u8\_usartTx\_interruptMask\_t\*/

u8\_usartRx\_enable\_t usartRxEnable; /\*@ref u8\_usartRx\_enable\_t\*/

u8\_usartRx\_interruptMask\_t usartRxInterrupt; /\*@ref u8\_usartRx\_interruptMask\_t\*/

u8\_usartStopBit\_t usartStopBitNum; /\*@ref u8\_usartStopBit\_t\*/

u8\_usartParityBit\_t usartParityBit; /\*@ref u8\_usartParityBit\_t\*/

u8\_usartDataSize\_t usartDataSize; /\*@ref u8\_usartDataSize\_t\*/

u8\_usartTxClkPolarity\_t usartTxClkPolarity; /\*@ref u8\_usartTxClkPolarity\_t\*/

u8\_usartRxClkPolarity\_t usartRxClkPolarity; /\*@ref u8\_usartRxClkPolarity\_t\*/

uint32\_t usartBaudRate; /\*define the Buadrate value\*/

}st\_usart\_config\_t;

/\* ------------------------------------------------------ Macro Declarations ------------------------------------------------------------\*/

/\* USART Working Mode \*/

#define USART\_ASYNCHRONOUS\_NORMAL\_SPEED\_MODE ((u8\_usartMode\_t)0x00)

#define USART\_ASYNCHRONOUS\_DOUBLE\_SPEED\_MODE ((u8\_usartMode\_t)0x01)

#define USART\_SYNCHRONOUS\_MODE ((u8\_usartMode\_t)0x02)

#define USART\_INVALID\_MODE ((u8\_usartMode\_t)0x03)

/\* USART Transmit Enable \*/

#define USART\_TX\_DISABLE ((u8\_usartTx\_enable\_t)0x00)

#define USART\_TX\_ENABLE ((u8\_usartTx\_enable\_t)0x01)

/\* USART Receiver Enable \*/

#define USART\_RX\_DISABLE ((u8\_usartRx\_enable\_t)0x00)

#define USART\_RX\_ENABLE ((u8\_usartRx\_enable\_t)0x01)

/\* USART Transmit Interrupt Enable Feature \*/

#define USART\_TX\_INTERRUPT\_DISABLE ((u8\_usartTx\_interruptMask\_t)0x00)

#define USART\_TX\_INTERRUPT\_ENABLE ((u8\_usartTx\_interruptMask\_t)0x01)

/\* EUSART Receiver Interrupt Enable Feature\*/

#define USART\_RX\_INTERRUPT\_DISABLE ((u8\_usartRx\_interruptMask\_t)0x00)

#define USART\_RX\_INTERRUPT\_ENABLE ((u8\_usartRx\_interruptMask\_t)0x01)

/\*Select Number of stop-bit either one or two \*/

#define USART\_ONE\_STOP\_BIT ((u8\_usartStopBit\_t)0x00)

#define USART\_TWO\_STOP\_BITS ((u8\_usartStopBit\_t)0x01)

#define USART\_INVALID\_STOP\_BITS ((u8\_usartStopBit\_t)0x02)

/\*Select Parity mode or disabled parity\*/

#define USART\_DISABLED\_PARITY\_BIT ((u8\_usartParityBit\_t)0x00)

#define USART\_EVEN\_PARITY\_BIT ((u8\_usartParityBit\_t)0x01)

#define USART\_ODD\_PARITY\_BIT ((u8\_usartParityBit\_t)0x02)

#define USART\_INVALID\_PARITY\_BIT ((u8\_usartParityBit\_t)0x03)

/\*Select the data-bit number\*/

#define USART\_FIVE\_BIT\_DATA ((u8\_usartDataSize\_t)0x00)

#define USART\_SIX\_BIT\_DATA ((u8\_usartDataSize\_t)0x01)

#define USART\_SEVEN\_BIT\_DATA ((u8\_usartDataSize\_t)0x02)

#define USART\_EIGHT\_BIT\_DATA ((u8\_usartDataSize\_t)0x03)

#define USART\_NINE\_BIT\_DATA ((u8\_usartDataSize\_t)0x04)

#define USART\_INVALID\_BIT\_DATA ((u8\_usartDataSize\_t)0x05)

/\*SELCET THE CLOCK POLARITY IN CASE OF SYNCHRONOUS MODE ONLY \*/

#define USART\_SYNCHRONOUS\_TX\_RISING\_XCK\_EDGE ((u8\_usartTxClkPolarity\_t)0x00)

#define USART\_SYNCHRONOUS\_TX\_FALLING\_XCK\_EDGE ((u8\_usartTxClkPolarity\_t)0x01)

#define USART\_SYNCHRONOUS\_RX\_RISING\_XCK\_EDGE ((u8\_usartRxClkPolarity\_t)0x00)

#define USART\_SYNCHRONOUS\_RX\_FALLING\_XCK\_EDGE ((u8\_usartRxClkPolarity\_t)0x01)

/\*The Error state of The USART\*/

#define USART\_E\_OK ((u8\_usartErorrState\_t)0x00)

#define USART\_E\_NOK ((u8\_usartErorrState\_t)0x01)

/\*AN INDICATION TO TERMINATE RECIEVING BYTES AND STORE IT IN THE CHARCTER ARRAY (ASCII OF ENTER)\*/

#define END\_OF\_STRING\_SYMPOL ((uint8\_t)0x0D)

/\* ------------------------------------------------------ Macro Like Functions Declarations ---------------------------------------------\*/

#define ENABLE\_TX\_INTERRUPT() (SET\_BIT(UCSRB,TXCIE))

#define DISABLE\_TX\_INTERRUPT() (CLEAR\_BIT(UCSRB,TXCIE))

#define ENABLE\_EMPTY\_DATA\_REG\_INTERRUPT() (SET\_BIT(UCSRB,UDRIE))

#define DISABLE\_EMPTY\_DATA\_REG\_INTERRUPT() (CLEAR\_BIT(UCSRB,UDRIE))

#define ENABLE\_RX\_INTERRUPT() (SET\_BIT(UCSRB,RXCIE))

#define DISABLE\_RX\_INTERRUPT() (CLEAR\_BIT(UCSRB,RXCIE))

/\* ------------------------------------------------- Software Interfaces Declarations ---------------------------------------------------\*/

st\_usart\_config\_t st\_g\_usartObjForBcm = {

.usartBaudRate=USART\_CONFIG\_BAUDRATE,

.usartDataSize = USART\_EIGHT\_BIT\_DATA,

.usartMode = USART\_ASYNCHRONOUS\_NORMAL\_SPEED\_MODE,

.usartParityBit = USART\_DISABLED\_PARITY\_BIT,

.usartRxEnable = USART\_RX\_ENABLE,

.usartTxEnable = USART\_TX\_ENABLE,

.usartRxInterrupt = USART\_RX\_INTERRUPT\_ENABLE,

.usartTxInterrupt = USART\_TX\_INTERRUPT\_ENABLE,

.usartStopBitNum = USART\_ONE\_STOP\_BIT,

};

#define USART\_PRE\_COMPILE\_CONFIG\_H\_

#ifndef F\_CPU

# define F\_CPU 8000000UL

#endif

#ifndef USART\_CONFIG\_BAUDRATE

# define USART\_CONFIG\_BAUDRATE 9600UL

#endif

#ifndef DATA\_TO\_SEND\_MAX\_BUFFER

# define DATA\_TO\_SEND\_MAX\_BUFFER 250U

#endif

#ifndef DATA\_TO\_RECIVE\_MAX\_BUFFER

# define DATA\_TO\_RECIVE\_MAX\_BUFFER 250U

#endif

#endif /\* USART\_PRE\_COMPILE\_CONFIG\_H\_ \*/

**4.2. LED CONFIGURATION**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* section 4: Data Type Declarations \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef enum{

LED\_OFF=0,

LED\_ON

}led\_status\_t;

typedef struct{

pin\_config\_t led\_pin;

led\_status\_t led\_status;

}led\_t;

led\_t st\_g\_led0\_instance = {

.led\_pin.pin=PIN0,

.led\_pin.port=PORTB\_INDEX,

.led\_status = LED\_OFF

};

led\_t st\_g\_led1\_instance = {

.led\_pin.pin=PIN1,

.led\_pin.port=PORTB\_INDEX,

.led\_status = LED\_OFF

};

**4.3. BCM CONFIGURATION**

// typedef void (\*Fptr\_bcmtCallBack\_t) (uint16\_t u16\_a\_sizeOfData);

typedef void (\*Fptr\_bcmtCallBack\_t) (void);

typedef enum {

BCM\_E\_OK=0,

BCM\_E\_NOK=2,

}enu\_system\_status\_t;

typedef enum {

BCM\_USART\_BUS=0,

BCM\_SPI\_BUS,

BCM\_TWI\_BUS,

BCM\_INVALID\_BUS\_ID

}enu\_bcm\_busID\_t;

typedef enum {

BCM\_TRANSIMTER=0,

BCM\_RECIEVER,

BCM\_TRANSIMTER\_RECIEVER,

BCM\_INVALID\_OBERATION

}enu\_bcm\_operation\_t;

typedef struct{

enu\_bcm\_busID\_t bcm\_busID; //@ref enu\_bcm\_busID\_t

enu\_bcm\_operation\_t bcm\_operation; //@ref enu\_bcm\_operation\_t

Fptr\_bcmtCallBack\_t Fptr\_bcmtTxCallBack; //@ref Fptr\_bcmtCallBack\_t

Fptr\_bcmtCallBack\_t Fptr\_bcmtRxCallBack; //@ref Fptr\_bcmtCallBack\_t

}str\_bcm\_inctance\_t;

str\_bcm\_inctance\_t str\_g\_bcm\_inctance = {

.bcm\_busID = BCM\_USART\_BUS,

.bcm\_operation = BCM\_TRANSIMTER\_RECIEVER

};