Basic Communication Manager

(BCM) Design

Prepared by:

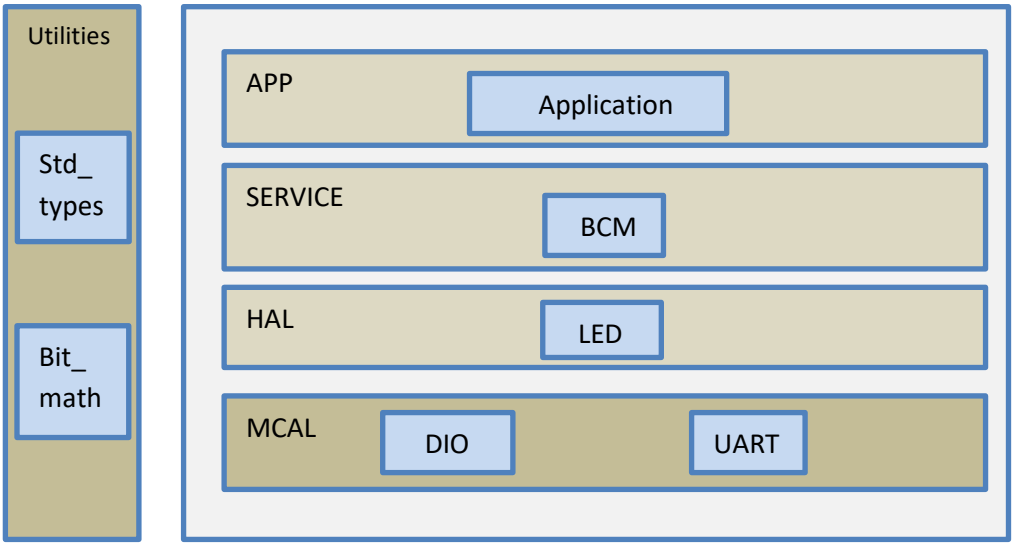
Mustafa Mohammed Abdou

Table of content:

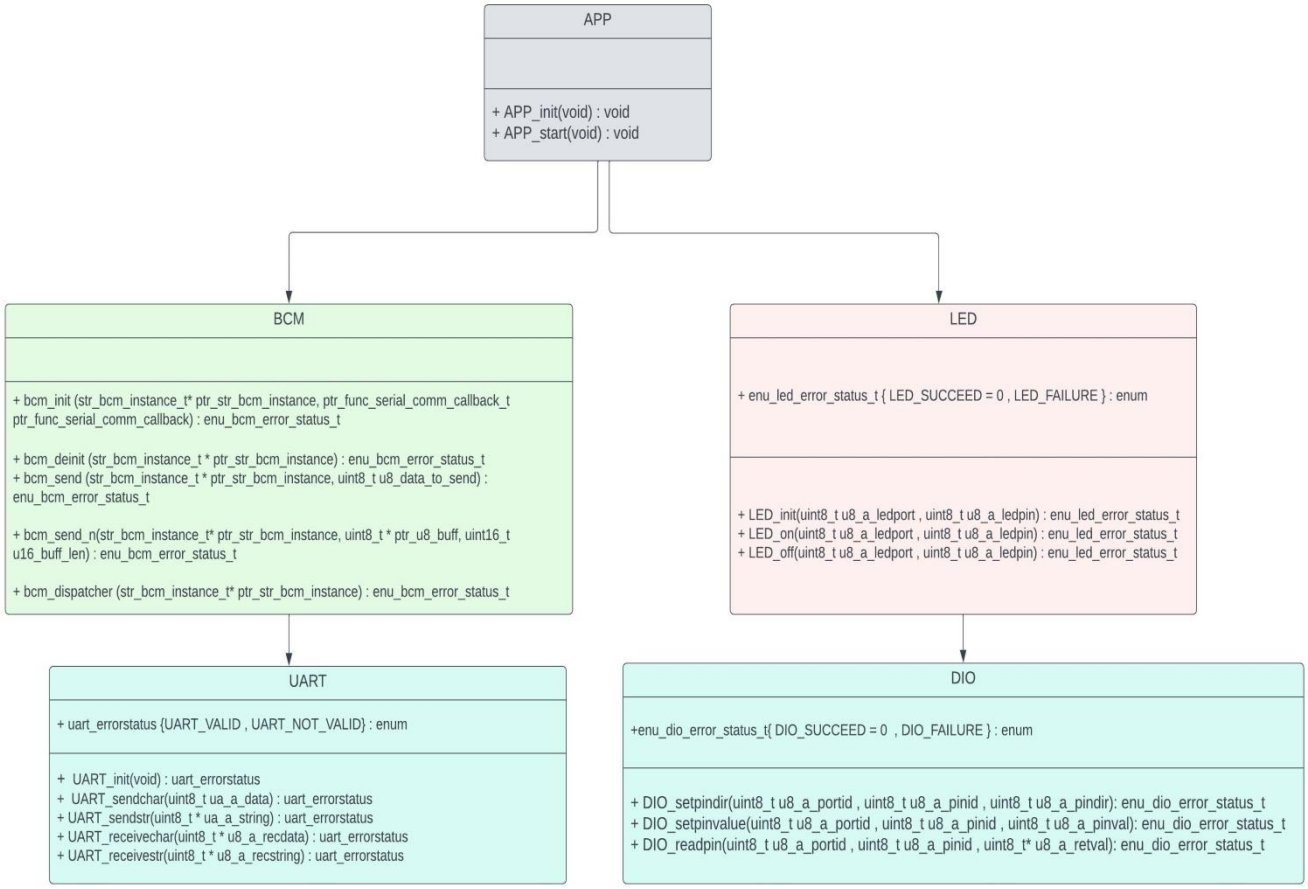
1 – Description

BCM is mainly a component in the service layer which is developed to abstract the APP layer from any communication protocol so it's used to handle any transmission and receiving operation based on serial protocol.

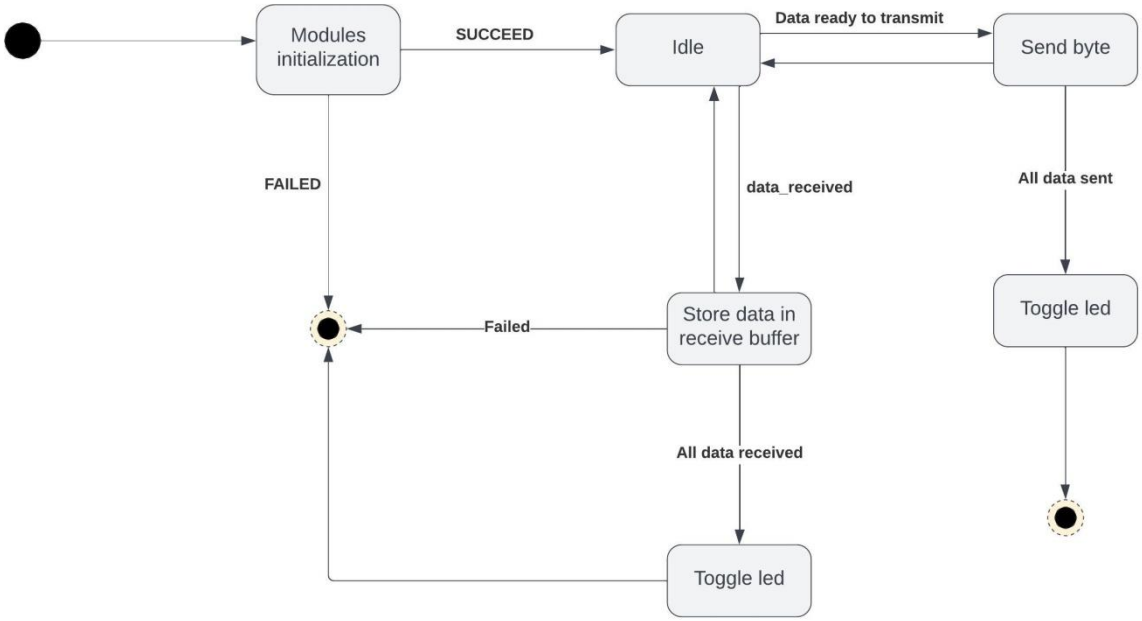
2 – Layered Architecture



3 – BCM class diagram



4 – Application state machine



5 – Module's description

5.1 DIO Driver

Configuration: Consist of 4 API's

Location: MCAL

Function: used to set pin direction (input or output), pin value (high or low) or read a value from a pin or toggle a pin

5.2 UART Driver

Configuration: Consist of 5 API's

Location: MCAL

Function: used to configure the communication protocol (UART) to operate with specific configurations also send and receive data via RX, TX pins (UART pins).

5.3 LED Driver

Configuration: Consist of 3 API's

Location: HAL

Function: used to initialize LEDs using DIO driver also contain LED functionalities such as illuminate LED and toggle LEDs and also turns LEDs off.

5.4 BCM Driver

Configuration: Consist of 5 API's

Location: SERVICE layer

Function: used to initialize a basic communication manager to abstract application layer from any communication protocol also includes APIs to handle data transmission and reception using any serial protocol (UART, SPI or TWI).

6 – Module's APIs

6.1 DIO APIs

```
/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO SET THE DIRECTION OF SPECIFIC PIN   */
/* INPUT   : PORT , PINID , DIRECTION                             */
/* RETURNS : PinDirection_t                                       */
/*****************************************************************/
PinDirection_t DIO_setpindir(uint8_t u8_a_portid , uint8_t u8_a_pinid , uint8_t u8_a_pindir);

/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO SET THE DIRECTION OF SPECIFIC PORT   */
/* INPUT   : PORT , DIRECTION                                      */
/* RETURNS : PinDirection_t                                        */
/*****************************************************************/
PinDirection_t DIO_setportdir(uint8_t u8_a_portid , uint8_t u8_a_portdir);


/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO SET THE VALUE OF SPECIFIC PIN       */
/* INPUT   : PORT , PINID , DIRECTION                             */
/* RETURNS : PinValue_t                                           */
/*****************************************************************/
PinValue_t DIO_setpinvalue(uint8_t u8_a_portid , uint8_t u8_a_pinid , uint8_t u8_a_pinval);

/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO SET THE VALUE OF SPECIFIC PORT       */
/* INPUT   : PORT , PINID , DIRECTION                              */
/* RETURNS : PinValue_t                                            */
/*****************************************************************/
PinValue_t DIO_setportvalue(uint8_t u8_a_portid , uint8_t u8_a_portval);


/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO GET THE VALUE OF SPECIFIC PIN       */
/* INPUT   : PORTID , PINID , POINTER TO SET THE VALUE IN IT      */
/* RETURNS : PinRead_t                                            */
/*****************************************************************/
PinRead_t DIO_readpin(uint8_t u8_a_portid , uint8_t u8_a_pinid , uint8_t* u8_a_val);


/*****************************************************************/
/* DESCRIBTION  : FUNCTION TO TOGGLE SPECIFIC PIN                 */
/* INPUT   : PORTID , PINID                                       */
/* RETURNS : PinRead_t                                            */
/*****************************************************************/
PinRead_t DIO_togglepin(uint8_t u8_a_portid , uint8_t u8_a_pinid );
```

6.2 UART APIs

```
/************************************************************/
/** FUNCTION TO INITIALIZE THE UART                        **/
/** ARGUMENTS : VOID                                       **/
/** RETURNS   : uart_errorstatus                           **/
/************************************************************/
uart_errorstatus UART_init(void);

/************************************************************/
/** FUNCTION TO SEND BYTE VIA UART                         **/
/** ARGUMENTS : ua_a_data                                  **/
/** RETURNS   : uart_errorstatus                           **/
/************************************************************/
uart_errorstatus UART_sendchar(uint8_t ua_a_data);

/************************************************************/
/** FUNCTION TO SEND STRING VIA UART                       **/
/** ARGUMENTS : ua_a_string                                **/
/** RETURNS   : uart_errorstatus                           **/
/************************************************************/
uart_errorstatus UART_sendstr(uint8_t * ua_a_string);

/*******************************************************************/
/** FUNCTION TO RECEIVE BYTE VIA UART                             **/
/** ARGUMENTS : u8_recdata (POINTER TO STORE THE RECEIVED DATA)   **/
/** RETURNS   : uart_errorstatus                                  **/
/*******************************************************************/
uart_errorstatus UART_receivechar(uint8_t * u8_a_recdata);


/*******************************************************************/
/** FUNCTION TO RECEIVE STRING VIA UART                           **/
/** ARGUMENTS : u8_a_recstring (POINTER TO STORE THE RECEIVED DATA)**/
/** RETURNS   : uart_errorstatus                                  **/
/*******************************************************************/
uart_errorstatus UART_receivestr(uint8_t * u8_a_recstring);
```

6.3 LED APIs

```
/************************************************/
/** FUNCTION TO INITIALIZE A PIN                **/
/** INPUT : LED PORT , LED PIN                  **/
/** RETURNS : enu_led_error_status_t            **/
/************************************************/
enu_led_error_status_t LED_init(uint8_t u8_a_ledport , uint8_t u8_a_ledpin);


/************************************************/
/** FUNCTION TO SET A LED AS ON                 **/
/** INPUT : LED PORT , LED PIN                  **/
/** RETURNS : enu_led_error_status_t            **/
/************************************************/
enu_led_error_status_t LED_on(uint8_t u8_a_ledport , uint8_t u8_a_ledpin);

/************************************************/
/** FUNCTION TO SET A LED AS OFF                **/
/** INPUT : LED PORT , LED PIN                  **/
/** RETURNS : enu_led_error_status_t            **/
/************************************************/
enu_led_error_status_t LED_off(uint8_t u8_a_ledport , uint8_t u8_a_ledpin);
```

6.4 BCM APIs

```
/*****************************************************************/
/** FUNCTION TO INITIALIZE THE COMMUNICATION MEDIUM              **/
/** ARGUMENTS  : enu_serialprotocol (MEDUIM TO SEND DATA OVER IT **/
/** RETURNS    : enu_bcm_error_status_t                          **/
/*****************************************************************/
enu_bcm_error_status_t BCM_init(enu_serialprotocol_t enu_serialprotocol );

/*************************************************/
/** FUNCTION TO INITIALIZE THE COMMUNICATION MEDIUM  **/
/** ARGUMENTS  :                                     **/
/** RETURNS    : enu_bcm_error_status_t              **/
/*************************************************/
enu_bcm_error_status_t BCM_deinit();

/*************************************************************************************/
/** FUNCTION TO SEND A DATA BYTE                                                    **/
/** ARGUMENTS  : enu_serialprotocol (COMMUNICATION MEDIUM) ,  u8_a_databyte (DATA TO BE SENT)  **/
/** RETURNS    : enu_bcm_error_status_t                                             **/
/*************************************************************************************/
enu_bcm_error_status_t BCM_send(enu_serialprotocol_t enu_serialprotocol , uint8_t u8_a_databyte );

/*****************************************************************************************************************/
/** FUNCTION TO SEND ARRAY OF DATA                                                                              **/
/** ARGUMENTS  : enu_serialprotocol (COMMUNICATION MEDIUM) , (* u8_a_data) POINTER TO DATA , u8_a_datalength (DATA LENGTH) **/
/** RETURNS    : enu_bcm_error_status_t                                                                         **/
/*****************************************************************************************************************/
enu_bcm_error_status_t BCM_send_n(enu_serialprotocol_t enu_serialprotocol , uint8_t * u8_a_data , uint8_t u8_a_datalength);

void BCM_dispatcher(enu_serialprotocol_t enu_serialprotocol);
```