



Simple ATM Machine

Team Members:

Alaa Ibrahim
Bassel Yasser
Sharpel Malek
Sherif Khadr

Contents

INTRODUCTION	1
High Level Design	1
01) Layered Architecture	1
02) Modules Description	2
03) Drivers' Documentation	3
MCAL Layer	3
• DIO	3
• Timer 0	4
• EXTINT:	5
• I2C:	6
• UART:	8
• SPI:	9
HAL Layer:	10
• Keypad	10
• HTimer:	11
• HLCD	12
• Buzzer	13
• Button	13
• HEXT_INT	14
• HSPI	14
• EEPROM:	15
Card_Database Layer (CARD MCU)	17
ATM_MODULE Layer (ATM MCU)	18
• ATM:	18
• Database_check:	19
Application Layer:	20
• Card MCU	20
• ATM MCU	20
Low Level Design.....	21
MCAL Layer:	21
• DIO	21

• Timer:	24
• EXTINT:	26
• UART:	29
• SPI:	33
• I2C:	34
HAL Layer	38
• HTimer0	38
• HSPI	39
• LCD	40
• Keypad	47
• Buzzer	50
• HEXTINT:	51
• Button:	52
• EEPROM	53
Card_Database Layer (CARD MCU)	58
• Database_check:	63
• ATM Module	68
Application Layer:	73
• Card App	73
• ATM App	73

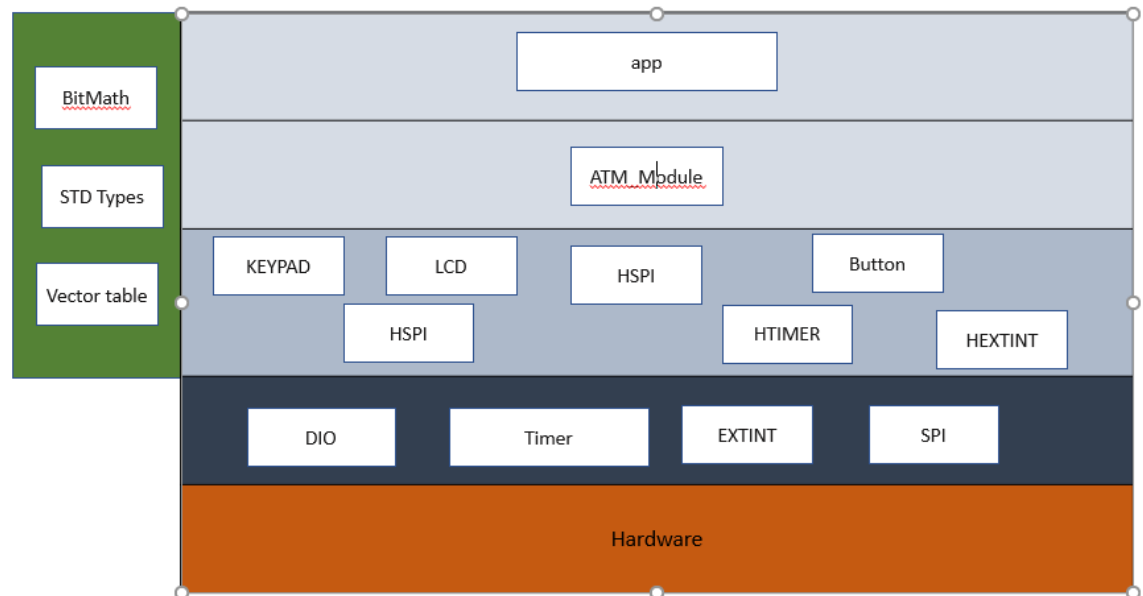
Figure 1: Project Layered Architecture	Error! Bookmark not defined.
Figure 2 DIO_s8SETPinDir Flow Chart.....	21
Figure 3 DIO_s8SETPinVal Flow chart.....	22
Figure 4 DIO_s8GETPinVal Flow Chart.....	23
Figure 5 TIM0_Init Flow Chart	24
Figure 6 TIM0_Start Flow Chart	24
Figure 7 TIM0_Stop Flow Chart	24
Figure 8 TIM0 remaining Flow Charts.....	25
Figure 15 HTIM0_SyncDelay Flow Chart.....	38
Figure 16 HTIM0_AsyncDelay and EndDelay	38
Figure 17 HLCD_vidInit Flow Chart	40
Figure 18 HLCD_vidWritecmd Flow Chart	41
Figure 19 HLCD_vidWriteChar Flow Chart.....	41
Figure 20 HLCD_ClrDisplay Flow Chart	42
Figure 21 HLCD_gotoXY Flow Chart.....	43
Figure 22 HLCD_WriteString Flow Chart.....	44
Figure 23 HLCD_WriteInt Flow Chart.....	45
Figure 24 HLCD_vidCreatCustomChar Flow Chart.....	46
Figure 25 KEYPAD_Init Flow Chart	47
Figure 26 KEYPAD_CheckRx Flow Chart.....	48
Figure 27 GetButton Flow Chart	49
Figure 28 Buzzer Init & SetState Flow Charts	50
Figure 30 App. APIs Flow Charts	73
Figure 31 APP_Start & App States Flow Charts.....	Error! Bookmark not defined.
Figure 32 Cont. App States Flow Charts.....	Error! Bookmark not defined.

INTRODUCTION

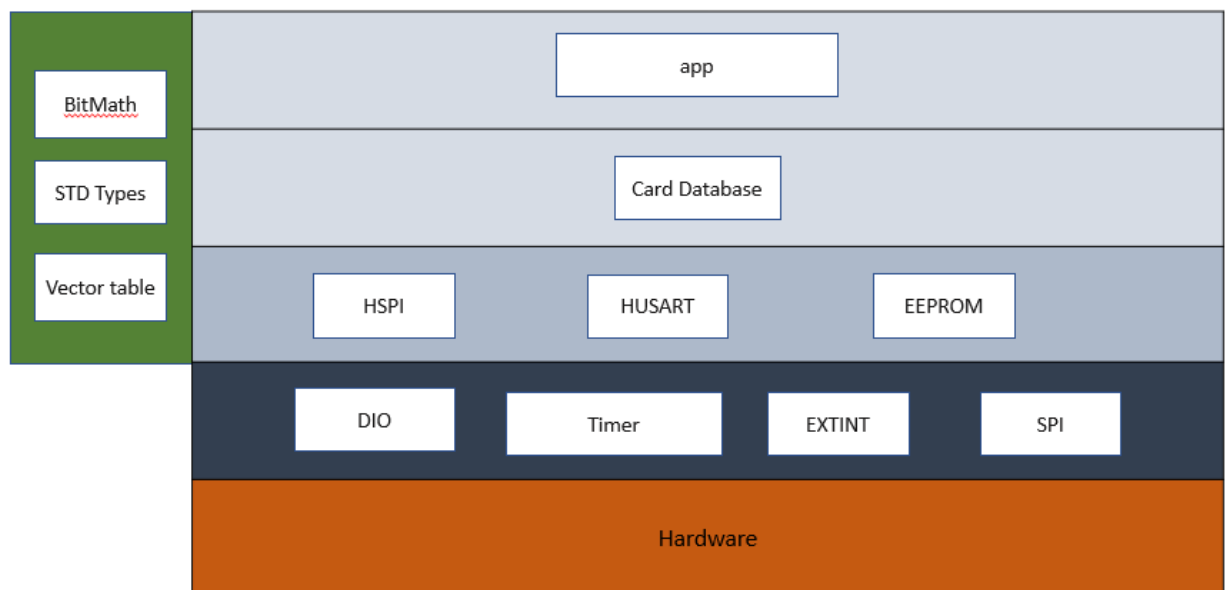
High Level Design

01) Layered Architecture

ATM MACHINE:



CARD MCU



02) Modules Description

MCAL Layer:

- **DIO:** For controlling GPIO pins
- **Timer:** Provides an interface with timer 0 low-level capabilities.
- **EXTINT:** it controls three external hardware interrupts on pins PD2, PD3, and PB2 which are referred to as INT0, INT1, and INT2 respectively
- **USART:** Enables MCU to communicate using serial protocols.
- **SPI:** Provides interface with SPI's low-level functionalities.
- **I2C:** Handles the i2c address oriented, multi-master, multi-slave communication.

HAL Layer:

- **Keypad:** Deal with a set of buttons arranged in a block. The 3 x 3 matrix keypad usually is used as input in a project
- **LCD:** Use for display data
- **Buzzer:** Simple module to control a buzzer.
- **HTimer:** Provides high-level functions using the lower-level timer 0 module capabilities.
- **H_EXT_INT:** Provides middle ware layer between app and external interrupt in MCAL.
- **HUSART:** Provides middle ware layer between app and UART in MCAL.
- **HSPI:** Provides high-level functionalities for using SPI communication.
- **EEPROM:** Enables MCU to interface with external EEPROM using i2c.
- **Button:** Interface with button to get its state (Enter/Zero).
- **Buzzer:** Enables MCU to control any given buzzer.

Service Layer:

- **STD_Types:** Contains all the standard types used by all the layers.
- **BIT_Math:** Provides bit-wise operations.
- **Vect_table:** Contains all interrupt vectors and provides macros for dealing with general interrupt.

Application Layer:

Contains the main logic of the project.

03) Drivers' Documentation

MCAL Layer

• DIO

```
/*
 * AUTHOR      : Bassel Yasser
 * Function    : DIO_s8SETPinDir
 * Description  : Set Pin Direction
 * Arguments   :
 *               - enPinCopy {DIO_PINA_0...., DIO_PIND_7}
 *               - enPortDir {INPUT , OUTPUT}
 * Return      : Sint8_t
 */
```

```
Sint8_t DIO_s8SETPinDir (enu_pin enPinCopy, enu_dir enPortDir)
```

```
/*
 * AUTHOR      : Bassel Yasser
 * Function    : DIO_s8SETPinVal
 * Description  : Set Pin Value
 * Arguments   :
 *               - enPinCopy {DIO_PINA_0...., DIO_PIND_7}
 *               - enPortDir {HIGH , LOW}
 * Return      : Sint8_t
 */
```

```
Sint8_t DIO_s8SETPinVal (enu_pin enPinCopy, enu_val enPortVal)
```

```
/*
 * AUTHOR      : Bassel Yasser
 * Function    : DIO_s8GETPinVal
 * Description  : Set Pin Value
 * Arguments   :
 *               - enPinCopy {DIO_PINA_0...., DIO_PIND_7}
 *               - pu8Val address of variable that u want to save
value on it
 * Return      : Sint8_t
 */
```

```
Sint8_t DIO_s8GETPinVal (enu_pin enPinCopy, Uint8_t* pu8Val)
```

- **Timer 0**

```
/**
 * \brief Initialize the timer with given mode
 * \param u8_a_Mode
 * \return en_TIMErrorState_t
 */
en_TIMErrorState_t TIM0_voidInit(en_TIMMode_t u8_a_Mode);

/**
 * \brief Start the timer clock after prescaling it with given value
 * \param u8_a_prescaler
 * \return en_TIMErrorState_t
 */
en_TIMErrorState_t TIM0_Start(en_TIM_CLK_SELECT_t u8_a_prescaler);

/**
 * \brief Function to stop timer 0
 * \return void
 */
void TIM0_Stop();

/**
 * \brief Set the timer to start from a certain value
 * \param u8_a_FlagValue The value to start the timer from
 * \return void
 */
void TIM0_SetValue(Uchar8_t u8_a_startValue);

/**
 * \brief Function to get the value of the overflow flag of timer 0
 * \param u8_a_FlagValue reference to a variable to store flag value *
 * \return en_TIMErrorState_t
 */
en_TIMErrorState_t TIM0_GetOVF(Uchar8_t* u8_a_FlagValue);

/**
 * \brief Function to clear timer 0 overflow flag
 * \return void
 */
void TIM0_ClearOVF(void);

/**
 * \brief Function to get the timer state (running/stopped)
 * \param u8_a_State reference to a variable to store timer state
 * \return en_TIMErrorState_t
 */
en_TIMErrorState_t TIM0_GetState(en_TIMState_t* u8_a_State);

/**
 * \brief Function to set a function to call when the timer0
 *        Overflow Interrupt is triggered
 * \param pv_a_CallbackFn reference to the function to call
 * \return en_TIMErrorState_t
 */
```


- **EXTINT:**

```
// EXT_INT TYPEDEFS
typedef enum EN_EXTINT_ERROR {
    EXTINT_OK=0,
    EXTINT_NOT_OK
}EN_EXTINT_ERROR;

typedef enum EN_Sence_Control {
    LOW_LEVEL=0,
    FALLING_EDGE,
    RISING_EDGE,
    ANY_LOGICAL_CHANGE
}EN_Sence_Control;

typedef enum EN_EXINT_NUMBER{
    EXTINT0=0,
    EXTINT1,
    EXTINT2,
}EN_EXINT_NUMBER;

typedef enum EN_GLOBAL_INT{
    DISABLE=0,
    ENABLE
}EN_GLOBAL_INT;

// EXT_INT prototypes

/*
Description : This function initializes the GLOBAL_INTERRUPT
ARGS       : takes the state ( ENABLE OR DISABLE )
return      : return EXTINT_OK if the PIN initializes correctly, EXTINT_NOT_OK otherwise
*/
EN_EXTINT_ERROR SET_GLOBAL_INTERRUPT(EN_GLOBAL_INT state);

/*
Description : This function initializes the external interrupt number and it's detecting
type
ARGS       : takes the EXINT_NUMBER( INT0,INT1 OR INT2) and sense control.
return      : return EXTINT_OK if the EXINT_NUMBER initializes correctly, EXTINT_NOT_OK
otherwise
*/
EN_EXTINT_ERROR EXTINT_init(EN_EXINT_NUMBER INTx ,EN_Sence_Control INTxSense);

/*
Description : This function takes the external interrupt number and initialize call back
function.
ARGS       : takes the EXINT_NUMBER( INT0,INT1 OR INT2) and pointer to the function we
want to execute.
return      : return EXTINT_OK if the EXINT_NUMBER initializes correctly, EXTINT_NOT_OK
otherwise
*/
EN_EXTINT_ERROR EXTINT_Callback(EN_EXINT_NUMBER INTx,void(*ptrfunc)(void));
```

- **I2C:**

```

/*****
/*          01- i2c_init_master()
          */
/* -----
* @func    : I2C master Initialization
* @in      : void
* @out     : void
*****
void i2c_init_master(void);

/*****
/*          02- i2c_init_slaver()
          */
/* -----
* @func    : I2C slave Initialization
* @in      : void
* @out     : void
*****
void i2c_init_slave (void);

/*****
/*          03- i2c_start()
          */
/* -----
* @func    : Send start condition
* @in      : void
* @out     : void
*****
void i2c_start(void);

/*****
/*          04- i2c_repeated_start()
          */
/* -----
* @func    : Send repeated start condition
* @in      : void
* @out     : void
*****
void i2c_repeated_start(void);

```

```

/*****
/*      05- i2c_send_slave_address_with_write_req()
/*
/* -----
/* @func      : send write request from master to slave
/* @in[1]     : slave_address
/*             - Slave address that you want to write on it
/* @out       : void
*****/
void i2c_send_slave_address_with_write_req(Uint8_t slave_address);

/*****
/*      06- i2c_send_slave_address_with_read_req()
/*
/* -----
/* @func      : send read request from master to slave
/* @in[1]     : slave_address
/*             - Slave address that you want to read from it
/* @out       : void
*****/
void i2c_send_slave_address_with_read_req(Uint8_t slave_address);

/*****
/*      07- i2c_write_byte()
/*
/* -----
/* @func      : Write data to slave
/* @in[1]     : byte
/*             - write data that u need to send
/* @out       : void
*****/
void i2c_write_byte(Uint8_t byte);

/*****
/*      08- i2c_read_byte()
/*
/* -----
/* @func      : read data from slave
/* @in        : void
/* @out       : returned data
*****/
Uint8_t i2c_read_byte(void);

/*****
/*      09- i2c_stop()
/*
/* -----
/* @func      : Send Stop condition
/* @in        : void
/* @out       : void
*****/
void i2c_stop(void);

```

- **UART:**

```
typedef enum EN_USART_ERROR{
    USART_OK=0,
    USART_NOT_OK
}EN_USART_ERROR;

/*
Name    : USART_init
Description : This function initializes USART Module with selected options in
USART.cfg file.
Args    : Void (all options defined as macros).
return  : Std_ReturnType (E_OK) if Module initializes Correctly, (E_NOT_OK) otherwise.
*/
EN_USART_ERROR USART_init(void);

/*
Name    : USART_sendData
Description : This function Send Data To Receiver (we can change size of data from
USART.cfg file) we select 8 bits data size.
Args    : take one argument (data to be sent) must be same size as the size we select
in Initialization function.
return  : Std_ReturnType (E_OK) if module sent data Correctly, (E_NOT_OK) otherwise.
*/
EN_USART_ERROR USART_sendData(Uchar8_t data);

/*
Name    : USART_receiveData
Description : This function Receive Data from sender (we can change size of data from
USART.cfg file) we select 8 bits data size.
Args    : void
return  : data received. must be same size as the size we select in Initialization
function.
*/
Uchar8_t USART_receiveData(void);

/*
Name    : USART_sendSTRING
Description : This function send array of data To Receiver. (we can change size of
data from USART.cfg file) we select 8 bits data size.
Args    : pointer to the array of data to be sent
return  : void
*/
void USART_sendSTRING(Uchar8_t * str);

/*
Name    : USART_receiveSTRING
Description : This function receive data from sender and store it in array.
Args    : pointer to the array to store data received in it and size of data.
return  : void
*/
void USART_receiveSTRING(Uchar8_t * str,Uchar8_t size);
```

- **SPI:**

```
/**
 * (Author: Alaa Hisham)
 * \brief Initialize the MCU as the Master
 *       in SPI communication
 * \return void
 */
void SPI_MasterInit(void);

/**
 * (Author: Alaa Hisham)
 * \brief Initialize the MCU as a slave
 *       in SPI communication
 *
 * \return void
 */
void SPI_SlaveInit(void);

/**
 * (Author: Alaa Hisham)
 * \brief Sets the value of the SPI data register
 * \param u8_a_data: Desired value
 * \return void
 */
void SPI_SetValue(Uchar8_t u8_a_data);

/**
 * (Author: Alaa Hisham)
 * \brief Exchange a letter with selected slave
 * \param u8_a_character: Character to send
 * \param pu8_a_receivedChar: Pointer to character to store received value
 * \return en_SPI_ErrorState_t
 */
en_SPI_ErrorState_t SPI_TranscieveChar(Uchar8_t u8_a_character, Uchar8_t*
pu8_a_receivedChar);

/**
 * (Author: Alaa Hisham)
 * \brief Exchange a letter with selected slave
 * \param u8_a_character: Character to send
 * \param pu8_a_receivedChar: Pointer to character to store received value
 * \return en_SPI_ErrorState_t
 */
en_SPI_ErrorState_t SPI_SlaveTranscieve(Uchar8_t u8_a_character, Uchar8_t*
pu8_a_receivedChar);

/**
 * (Author: Alaa Hisham)
 * \brief Set a notification function for the SPI Interrupt
 * \param pv_a_CallbackFn: reference to the function to callback
 *                        when the SPI Interrupt is triggered
 * \return en_SPI_ErrorState_t
 */
en_SPI_ErrorState_t SPI_SetCallback(void (*pv_a_CallbackFn)(void));
```

HAL Layer:

- **Keypad**

```
// Macros

#define R1      DIO_PINC_2
#define R2      DIO_PINC_3
#define R3      DIO_PINC_4
#define C1      DIO_PINC_5
#define C2      DIO_PINC_6
#define C3      DIO_PINC_7

// user defined datatypes

typedef enum EN_KEYPAD_BTNS
{
    KEY_1=0,
    KEY_2,
    KEY_3,
    KEY_4T,
    KEY_5,
    KEY_6,
    KEY_7,
    KEY_8,
    KEY_9,
    KEY_NOTHING
}EN_KEYPAD_BTNS;

// functions prototypes

/*****
Name : KEYPAD_init()
Description : This Function Initializes keypad pins (Rows are outputs & Columns are
inputs).
ARGS : void
return : void
*****/

void KEYPAD_init(void);

/*****
*
Name : KEYPAD_GetButton
Description : This Function loops over other three functions (Checks (R1,R2,R3)).
ARGS : void
return : the pressed key or Nothing pressed
*****/

EN_KEYPAD_BTNS KEYPAD_GetButton(void);
/*****
*
*****/
```

Name : KEYPAD_checkR1 , KEYPAD_checkR2, KEYPAD_checkR3
Description : functions are checking the entire row if it pressed or not.

ARGS : void

return : the pressed key or Nothing pressed

```
*****
*****/
```

```
EN_KEYPAD_BTNS KEYPAD_checkR1(void);
EN_KEYPAD_BTNS KEYPAD_checkR2(void);
EN_KEYPAD_BTNS KEYPAD_checkR3(void);
```

- **HTimer:**

```
/**
 * \brief Generate Synchronous delay (busy waiting)*
 * \param Copy_delayTime Desired delay
 * \param Copy_timeUnit Time units (Seconds, mSeconds, uSeconds)
 *
 * \return en_HTIMErrorState_t
 */
en_HTIMErrorState_t TIM0_SyncDelay(Uint32_t u32_a_delay, en_timeUnits_t
u8_a_timeUnit);

/**
 * \brief Generates delay asynchronously
 * \param u32_a_delay desired delay
 * \param u8_a_timeUnit delay time units
 * \param Copy_pvCallbackFn function to call when delay is complete
 *
 * \return en_TIMErrorState_t
 */
en_HTIMErrorState_t TIM0_AsyncDelay(Uint32_t u32_a_delay, en_timeUnits_t
u8_a_timeUnit, void (*Copy_pvCallbackFn)(void));

/**
 * \brief Function to end a delay asynchronously
 * To Stop Async Delay: No Restrictions
 * To Stop Sync Delay: should only be called in an ISR/Callback function
 *
 * \return void
 */
void TIM0_AsyncEndDelay();
```

- **HLCD**

```

/*
 * function          : HLCD_vidInit
 * description       : func to set LCD initialization
 * input param      : void
 * return            : void
 * */
void HLCD_vidInit(void)

/*
 * function          : HLCD_vidWritecmd
 * description       : func to configure some commands on lcd
 * input param      :
 *
 *                                     u8commandCopy --> take lcd cmd instructions from
instruction table
<https://components101.com/sites/default/files/component\_datasheet/16x2%20LCD%20Datasheet.pdf>
 * return            : void
 * */
void HLCD_vidWritecmd(Uint8_t u8commandCopy)

/*
 * function          : HLCD_vidWriteChar
 * description       : func to write char on lcd
 * input param      : u8CharCopy -> take ascii code of char or char address on
CGROM
 * return            : void
 * */
void HLCD_vidWriteChar(Uint8_t u8CharCopy)

/*
 * function          : HLCD_ClrDisplay
 * description       : func to clear anything on lcd
 * input param      : void
 * return            : void
 * */
void HLCD_ClrDisplay(void)

/*
 * function          : HLCD_gotoXY
 * description       : func to determine position which char print at this position on
lcd ### NOTE : (2rows x 16coloms)
 * input param      :
 *
 *                                     row -> take row number 0 or 1
 *                                     pos -> take colom number from 0 ~ 16
 * return            : void
 * */
void HLCD_gotoXY(Uint8_t row, Uint8_t pos)

/*
 * function          : HLCD_WriteString
 * description       : func to write string on lcd
 * input param      : str --> which take string as argument
 * return            : void
 * */
void HLCD_WriteString(Uint8_t* str)
/*

```



```

* function      : HLCD_WriteInt
* description   : func to write integer number on lcd
* input param  : number --> which take number as argument
* return       : void
* */
    void HLCD_WriteInt(Uint32_t number)

/*
* function      : HLCD_vidCreatCustomChar
* description   : func to store new pattern on CGRAM
* input param  :
*               pu8custom -> take pointer to array which having LCD
Custom Character Generated data ### take only 8 characters
*               u8Location -> determine location on CGRAM [0 ~ 8]
* return       : void
* */
    void HLCD_vidCreatCustomChar(Uint8_t* pu8custom, Uint8_t u8Location)

/**
* (Author: Alaa Hisham)
* \brief Display floating point number on LCD
*
* \param f32_a_number: number to display
* \return void
*/
void HLCD_DisplayFloat(float32_t f32_a_number);

```

• Buzzer

```

/**
* \brief Initialize buzzer pin as output
* \param pst_a_buzzer reference to buzzer
* \return void
*/
void BUZ_Init(st_Buzzer_t* pst_a_buzzer);

/**
* \brief Turn the buzzer on/off
* \param pst_a_buzzer reference to buzzer
* \param u16_a_state BUZ_ON (or) BUZ_OFF
* \return en_BuzzerErrorState_t
*/
en_BuzzerErrorState_t BUZ_SetState(st_Buzzer_t* pst_a_buzzer, en_BuzzerState_t
en_a_state);

```

• Button

```

/*
* AUTHOR      : Bassel Yasser Mahmoud
* FUNCTION    : HButton_Init
* DESCRIPTION : Initialize specified pin as input and pull up
* RETURN     : enu_buttonError_t {BUTTON_NOK, BUTTON_OK}
*/
enu_buttonError_t HButton_Init(enu_pin en_pinx);

```

```

/*
* AUTHOR          : Bassel Yasser Mahmoud
* FUNCTION        : HButton_ExtIntInit
* DESCRIPTION     : Initialize specified as pull up for external interrupt
* RETURN         : enu_buttonError_t {BUTTON_NOK, BUTTON_OK}
*/
enu_buttonError_t HButton_ExtIntInit(enu_pin en_pinx);

/*
* AUTHOR          : Bassel Yasser Mahmoud
* FUNCTION        : HButton_getPinVal
* DESCRIPTION     : Get pin status if it is high or low
* RETURN         : enu_buttonError_t {BUTTON_NOK, BUTTON_OK}
*/
enu_buttonError_t HButton_getPinVal(enu_pin en_pinx, Uint8_t* pu8_refVal );

*
Description : This function initializes the external interrupt number and it's detecting
type and initialize call back function.
ARGS       : takes the EXINT_NUMBER( INT0,INT1 OR INT2) and sense control and and
pointer to the function we want to execute when interrupt occurs.
return     : return EXTINT_OK if the EXINT_NUMBER initializes correctly, EXTINT_NOT_OK
otherwise
*/
EN_EXTINT_ERROR H_EXTINT_create(EN_EXTINT_NUMBER INTx ,EN_Sence_Control
INTxSense,void(*ptrfunc)(void));

```

• HEXT_INT

```

*
Description : This function initializes the external interrupt number and it's
detecting type and initialize call back function.
ARGS       : takes the EXINT_NUMBER( INT0,INT1 OR INT2) and sense control and and
pointer to the function we want to execute when interrupt occurs.
return     : return EXTINT_OK if the EXINT_NUMBER initializes correctly,
EXTINT_NOT_OK otherwise
*/
EN_EXTINT_ERROR H_EXTINT_create(EN_EXTINT_NUMBER INTx ,EN_Sence_Control
INTxSense,void(*ptrfunc)(void));

```

• HSPI

```

/**
* (Author: Alaa Hisham)
* \brief Initialize the MCU as the Master in SPI communication
* \return void
*/
void HSPI_MasterInit(void);

/**
* (Author: Alaa Hisham)
* \brief Initialize the MCU as a slave in SPI communication
* \return void
*/
void HSPI_SlaveInit(void);

```

```
/**
 * (Author: Alaa Hisham)
 * \brief Function to send a single character
 * \param u8_a_character character to send
 * \return void
 */
void HSPI_SendChar(Uchar8_t u8_a_character);

/**
 * \brief Receive data in the given buffer
 *
 * \param pu8_a_data: reference to buffer to store received data
 * \param u8_a_DataSize: Size (length) of data to receive
 *
 * \return void
 */
void HSPI_ReceiveData(Uchar8_t *pu8_a_data, Uchar8_t u8_a_DataSize);

/**
 * \brief Send given data byte by byte to selected slave
 *        and receive data in exchange into given array
 * \param pu8_a_TxDataArr: Reference to array of data to be transmitted
 * \param pu8_a_RxDataArr: Reference to array to store received data
 * \param u8_a_DataLen:    Length of data to exchange (in bytes)
 *                        (Must be less than or equal SPI_BUFFER_SIZE)
 *
 * \return en_HSPI_ErrorState_t
 */
en_HSPI_ErrorState_t HSPI_ExchangeData(Uchar8_t* pu8_a_TxDataArr, Uchar8_t*
pu8_a_RxDataArr, Uchar8_t u8_a_DataLen);

/**
 * (Author: Alaa Hisham)
 * \brief Request to send data to the master
 * \param u8_a_data: data to send
 * \return void
 */
void HSPI_SlaveRequest(Uchar8_t* pu8_a_dataPtr, Uchar8_t u8_a_DataSize);
```

- **EEPROM:**

```
/** AUTHOR : Sherif Ashraf Khadr
 * \brief : This Function Just Call To Initialize I2C as Master
 *
 * \param : Void
 *
 * \return void
 */
void eeprom_init(void)
```

```
/** AUTHOR : Sherif Ashraf Khadr
 * \brief : This Function Call To Make The Sequence Of I2C Frame To Write Byte On A Device
 *
 * \param : Uint16_t address : This Is Device Address
 *          : Uchar8_t data : This Is Data That Will Be Write
 *
 * \return void
 */
void eeprom_write_byte(Uint16_t address, Uchar8_t data)
```

```
/** AUTHOR : Sherif Ashraf Khadr
 * \brief : This Function Call To Make The Sequence Of I2C Frame To Read Byte On A Device
 *
 * \param : Uint16_t address : This Is Device Address
 *
 * \return Uchar8_t : Function Will Return Uchar8_t Contain The Data
 */
Uchar8_t eeprom_read_byte(Uint16_t address)
```

```
/** AUTHOR : Sherif Ashraf Khadr
 * \brief : This Function Call When You Need To Write A String In The EEPROM
 *
 * \param : Uint16_t address : This Is Device Address
 *          : Uchar8_t *str : This Pointer Will Store The Address Of The Array Of The Chars
 *
 * \return Void
 */
void eeprom_write_string(Uint16_t Copy_u8Address, const Uchar8_t* str)
```

```
/** AUTHOR : Sherif Ashraf Khadr
 * \brief : This Function Call When You Need To Read A String From The EEPROM
 *
 * \param : Uint16_t address : This Is Device Address
 *          : Uchar8_t *str : This Pointer Will Store The Address Of The Array Of The Chars That Will
Return String In It
 * \return Void
 */
void eeprom_read_string(Uint16_t Copy_u8Address, Uchar8_t* str)
```

Card Database Layer (CARD MCU)

```
/*
 * AUTHOR           : Bassel Yasser Mahmoud
 * FUNCTION          : APP_terminalPinGet
 * DESCRIPTION       : Get pin from User within terminal and doing some
validation
 * RETURN           : en_terminalPinGetStatus_t {PINGET_NOK or PINGET_OK}
 */
en_terminalPinGetStatus_t APP_terminalPinGet(Uchar8_t* arr);

/*
 * AUTHOR           : Sharpe1
 * FUNCTION          : APP_terminalPanGet
 * DESCRIPTION       : Get pan from User within terminal and doing some
validation
 * RETURN           : en_terminalPanGetStatus_t {PANGET_NOK or PANGET_OK}
 */
en_terminalPanGetStatus_t APP_terminalPanGet(Uchar8_t* arr);

/*
 * FUNCTION          : SaveCardData
 * DESCRIPTION       : Saving PAN and PIN in EEPROM
 * RETURN           : EN_TerminalDataState {DATA_SAVED, DATA_NSAVED,
DATA_READ, DATA_NREAD}
 */
EN_TerminalDataState SaveCardData(Uchar8_t *CardPan,Uchar8_t *CardPin);

/*
 * FUNCTION          : ReadCardData
 * DESCRIPTION       : Reading PAN and PIN from EEPROM
 * RETURN           : EN_TerminalDataState {DATA_SAVED, DATA_NSAVED,
DATA_READ, DATA_NREAD}
 */
EN_TerminalDataState ReadCardData(Uchar8_t *CardPan,Uchar8_t *CardPin);

/*
 * FUNCTION          : CARD_MatchPINS
 * DESCRIPTION       : Validate if PIN no and Confirmed PIN no is Matched or
not
 * RETURN           : en_CardPinMatchError_t {PIN_Match_NOK, PIN_Match_OK}
 */
en_CardPinMatchError_t CARD_MatchPINS();
```

ATM MODULE Layer (ATM MCU)

- **ATM:**

```
/**
 * \brief Displays welcome routine
 *
 * \return void
 */
void Welcome(void);

/**
 * (Author: Sherif Ashraf)
 * \brief Check if the pin user enters is the same as the cardholder's pin
 * \param pinFromAtm : reference to pin entered by user
 * \param pinFromServer: reference to pin received from card
 * \return EN_PinState
 */
EN_PinState PIN_checkPinMatching(Uchar8_t *pinFromAtm,Uchar8_t *pinFromServer);

/*
 * AUTHOR : Sharpel
 * FUNCTION : Get_pin
 * DESCRIPTION : get pin from user (on the atm )
 * ARGS : pointer to array (the size of array must be 5 or more) to
store entered pin by user
 * RETURN : PIN_OK if user enters 4 numbers , PIN_NOT_OK otherwise
 */
EN_PinState Get_pin(Uchar8_t *enteredpin);

/**
 * (Author: Sherif Ashraf)
 * \brief Locks the system and sound the buzzer
 * \param pst_a_buzzer: reference to the buzzer
 * \return en_BuzzerErrorState_t
 */
en_BuzzerErrorState_t deinitAtm(st_Buzzer_t* pst_a_buzzer);

/**
 * (Author: Alaa Hisham)
 * \brief Get the card pan and pin
 * \param pu8_a_pan: reference to buffer to receive pan from card
 * \param pu8_a_pin: reference to buffer to receive pin from card
 * \return EN_PinState
 */
EN_PinState ATM_GetCardData(Uchar8_t *pu8_a_pan, Uchar8_t *pu8_a_pin);

/*
 * AUTHOR : Sharpel
 * FUNCTION : get_amount_left
 * DESCRIPTION : get amount from user ( on the atm )
 * ARGS : pointer to array (the size of array must be 8 or more and
equal "0000.00" initial value) to store entered pin by user
 * RETURN : void
 */
void get_amount_left (Uchar8_t * amount);
```

```

/*
 * AUTHOR           : Bassel Yasser
 * FUNCTION          : EXTINT_FUNC
 * DESCRIPTION       : when timer 2 ISR is fire it changes the state of (Enter or
Zero)
 * ARGS             : void
 * RETURN            : void
 */
void EXTINT_FUNC(void);

/**
 * (Author: Alaa Hisham)
 * \brief Carries out the routine for approved card
 * \param f32_a_NewBalance: the balance to display after transaction
 * \return void
 */
void ATM_ApprovedCard(float32_t f32_a_NewBalance);

```

• Database_check:

```

/*
 * FUNCTION          : isValidPanAccount
 * DESCRIPTION       : Check If PAN No. Valid or not
 * RETURN            : EN_dataError_t {APPROVED
,FRAUD_CARD,CARD_STOLEN,EXCEED_MAX_DAILY_AMOUNT,INSUFFICIENT_FUND, DATA_ERROR}
 */
EN_dataError_t isValidPanAccount(Uchar8_t * pan);

/*
 * FUNCTION          : isRunningAccount
 * DESCRIPTION       : Checking if card stolen or not
 * RETURN            : EN_dataError_t {APPROVED
,FRAUD_CARD,CARD_STOLEN,EXCEED_MAX_DAILY_AMOUNT,INSUFFICIENT_FUND, DATA_ERROR}
 */
EN_dataError_t isRunningAccount(Uchar8_t * pan);

/*
 * FUNCTION          : isValidAccountAmount
 * DESCRIPTION       : Checking if there is INSUFFICIENT_FUND or not
 * RETURN            : EN_dataError_t {APPROVED
,FRAUD_CARD,CARD_STOLEN,EXCEED_MAX_DAILY_AMOUNT,INSUFFICIENT_FUND, DATA_ERROR}
 */
EN_dataError_t isValidAccountAmount(Uchar8_t * pan,Uchar8_t * amount,float32_t
*newAmount);

/*
 * FUNCTION          : isBelowMaxDailyAmount
 * DESCRIPTION       : Checking if transfered money is below limited daily
amount or not
 * RETURN            : EN_dataError_t {APPROVED
,FRAUD_CARD,CARD_STOLEN,EXCEED_MAX_DAILY_AMOUNT,INSUFFICIENT_FUND, DATA_ERROR}
 */
EN_dataError_t isBelowMaxDailyAmount(Uchar8_t * amount);

```

```
/*
 * FUNCTION          : DATABASE_checking
 * DESCRIPTION       : Card Database checking
 * RETURN            : EN_dataError_t {APPROVED
,FRAUD_CARD,CARD_STOLEN,EXCEED_MAX_DAILY_AMOUNT,INSUFFICIENT_FUND, DATA_ERROR}
 */
EN_dataError_t DATABASE_checking (Uchar8_t * pan,Uchar8_t * amount,float32_t
*newAmount);
```

Application Layer:

- **Card MCU**

```
/**
 * \brief Initializations of all used peripherals
 * \return void
 */
void APP_Init(void);

/**
 * \brief The main logic of the Card
 * \return void
 */
void APP_Start(void);
```

- **ATM MCU**

```
/**
 * \brief Initializations of all used peripherals
 * \return void
 */
void APP_Init(void);

/**
 * \brief The main logic of the ATM
 * \return void
 */
void APP_Start(void);
```


Low Level Design

MCAL Layer:

- DIO

```
Sint8_t DIO_s8SETPinDir (enu_pin enPinCopy, enu_dir enPortDir)
```

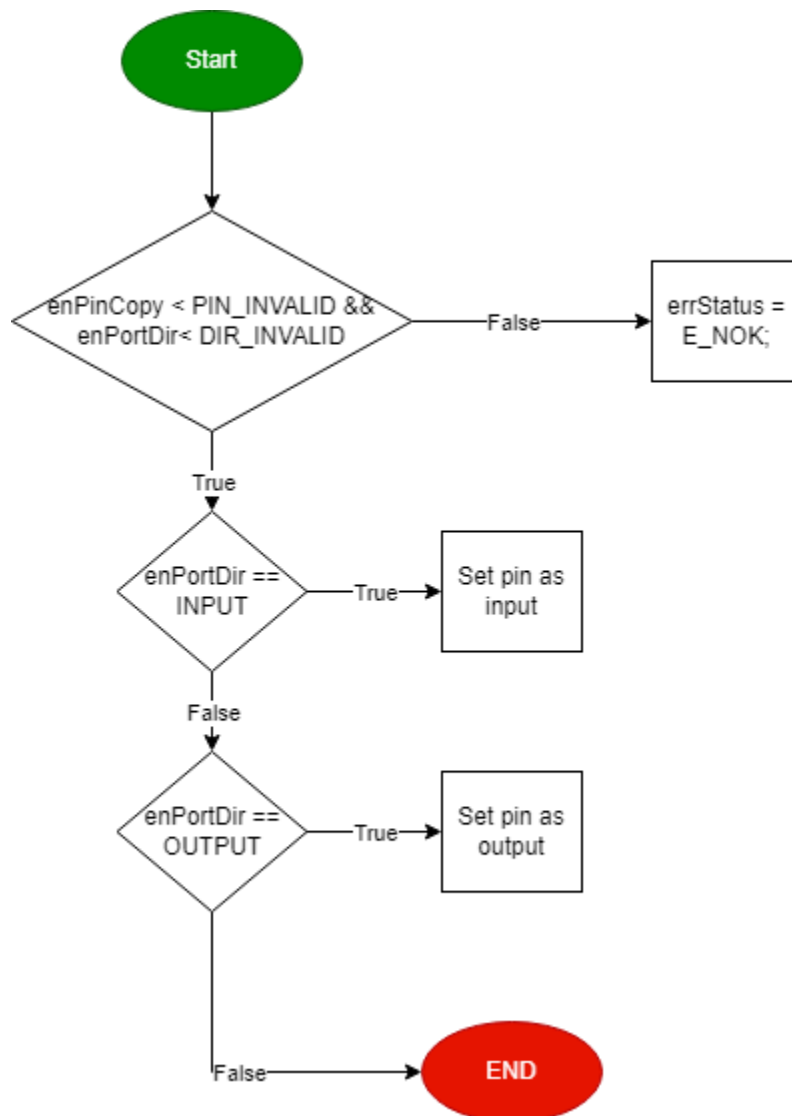


Figure 1 DIO_s8SETPinDir Flow Chart

Sint8_t DIO_s8SETPinVal (enu_pin enPinCopy, enu_val enPortVal)

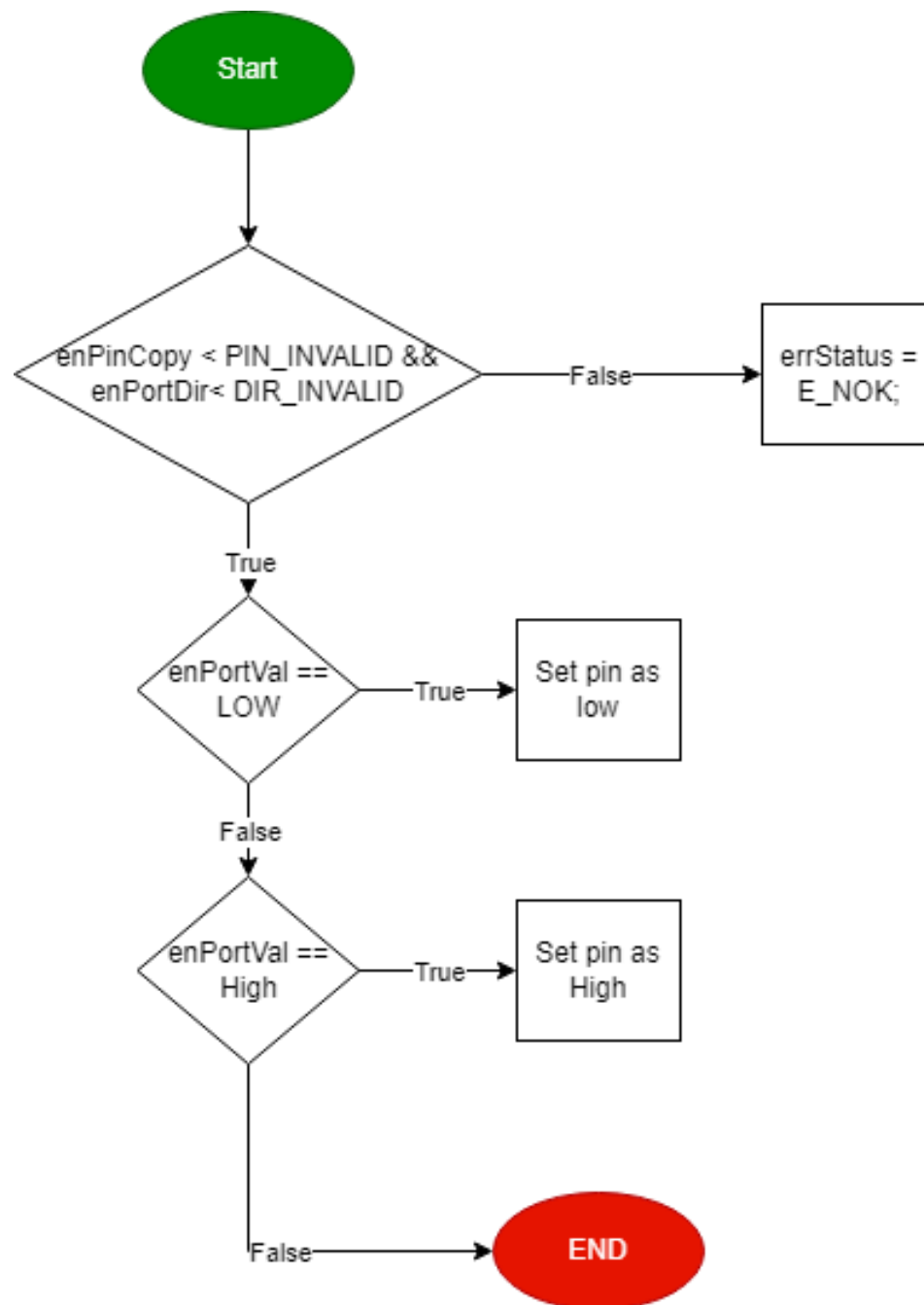


Figure 2 DIO_s8SETPinVal Flow chart

```
Sint8_t DIO_s8GETPinVal (enu_pin enPinCopy, Uint8_t* pu8Val)
```

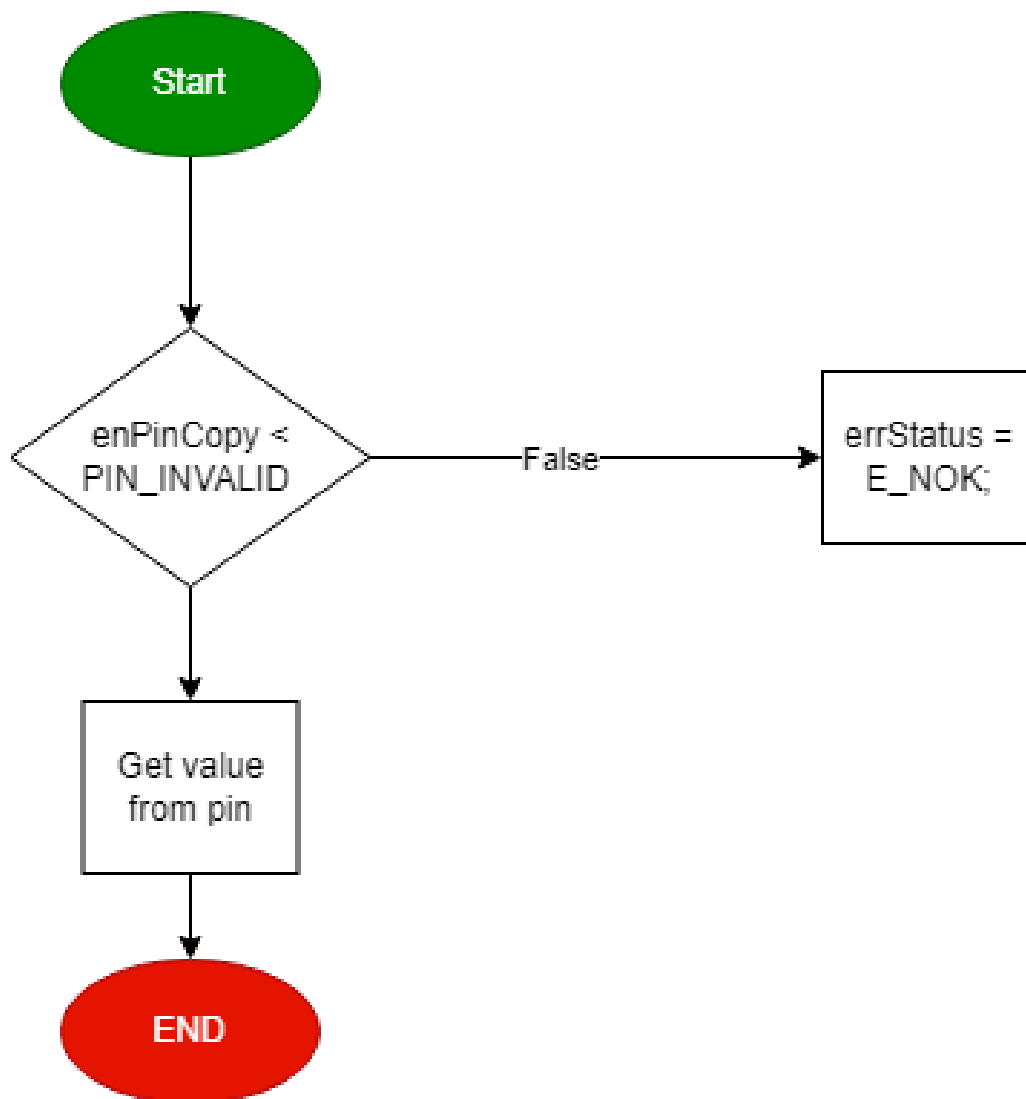


Figure 3 DIO_s8GETPinVal Flow Chart

- **Timer:**
TIM0_Init

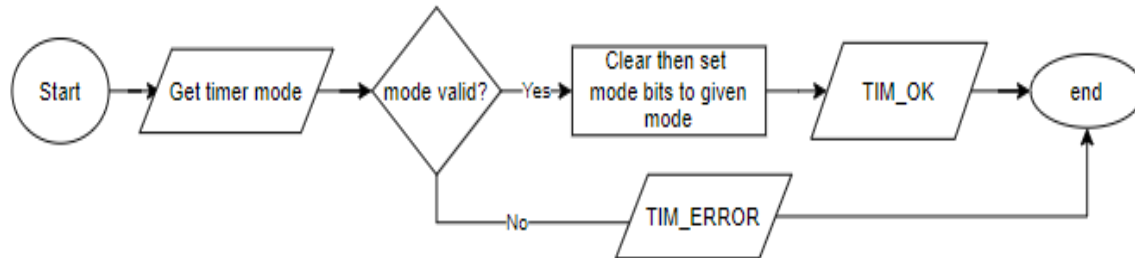


Figure 4 TIM0_Init Flow Chart

TIM0_Start

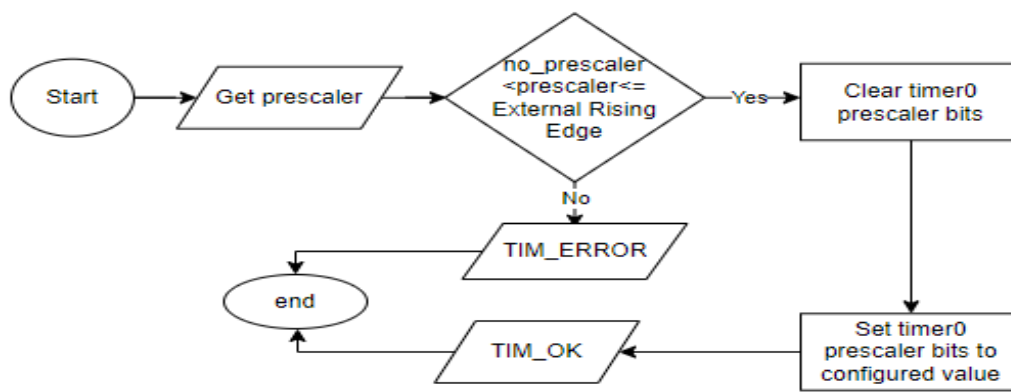


Figure 5 TIM0_Start Flow Chart

TIM0_Stop

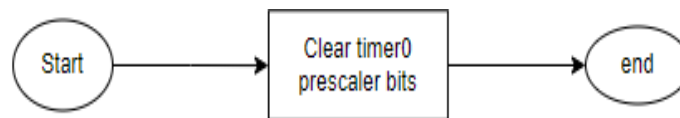
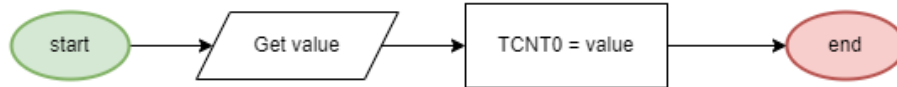
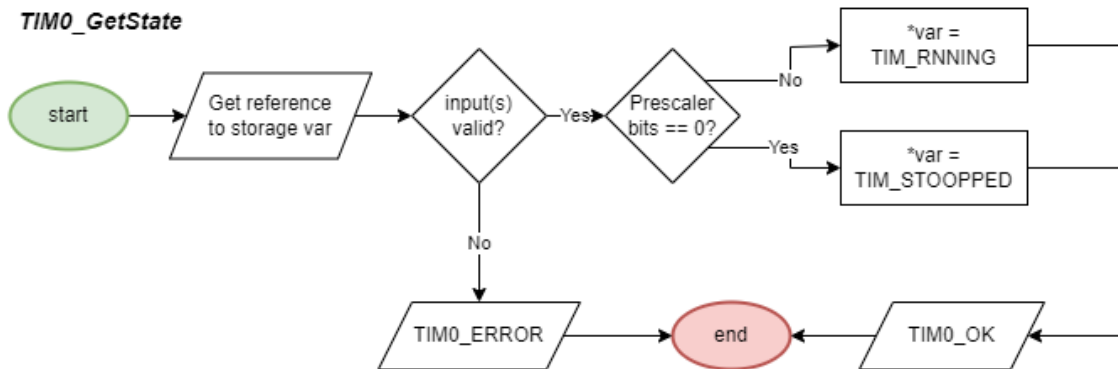


Figure 6 TIM0_Stop Flow Chart

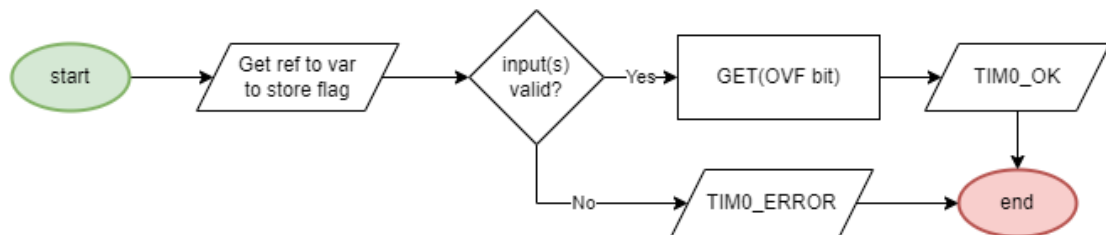
TIM0_SetValue



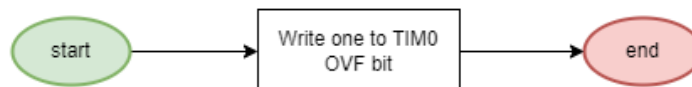
TIM0_GetState



TIM0_GetOVF



TIM0_ClearOVF



TIM0_EnableOVFInt

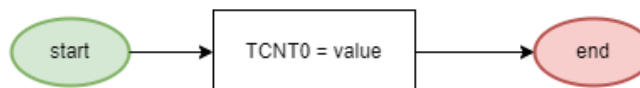
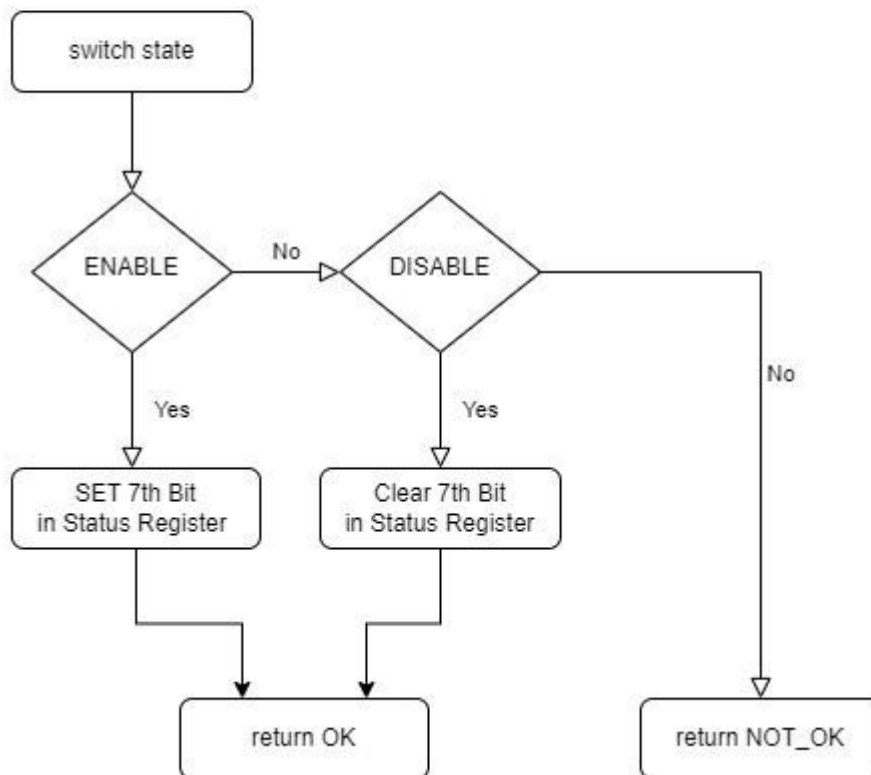
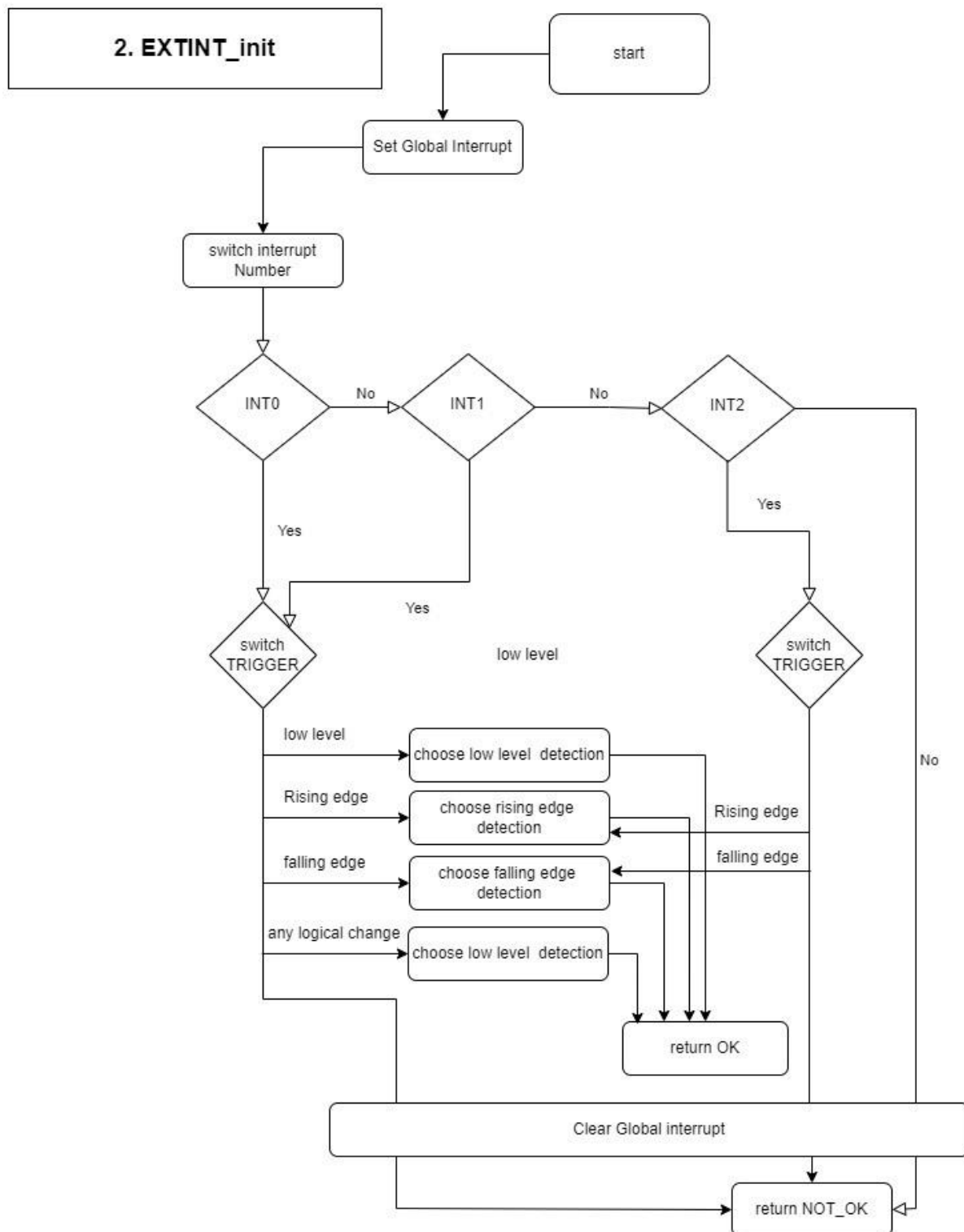


Figure 7 TIM0 remaining Flow Charts

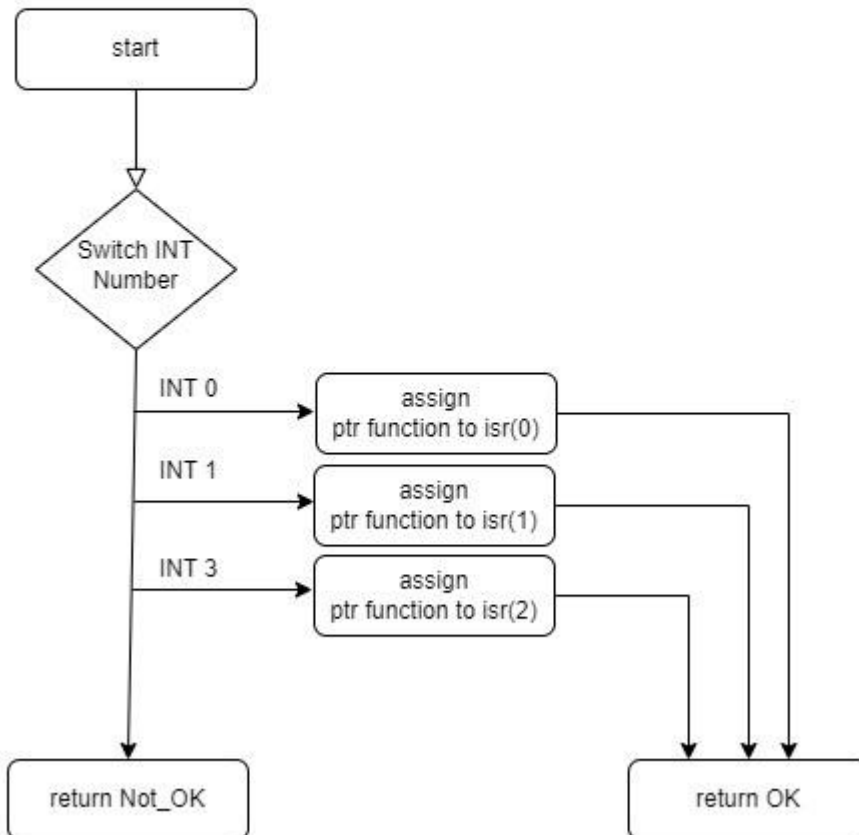
- **EXTINT:**

1. SET_GLOBAL_INTERRUPT



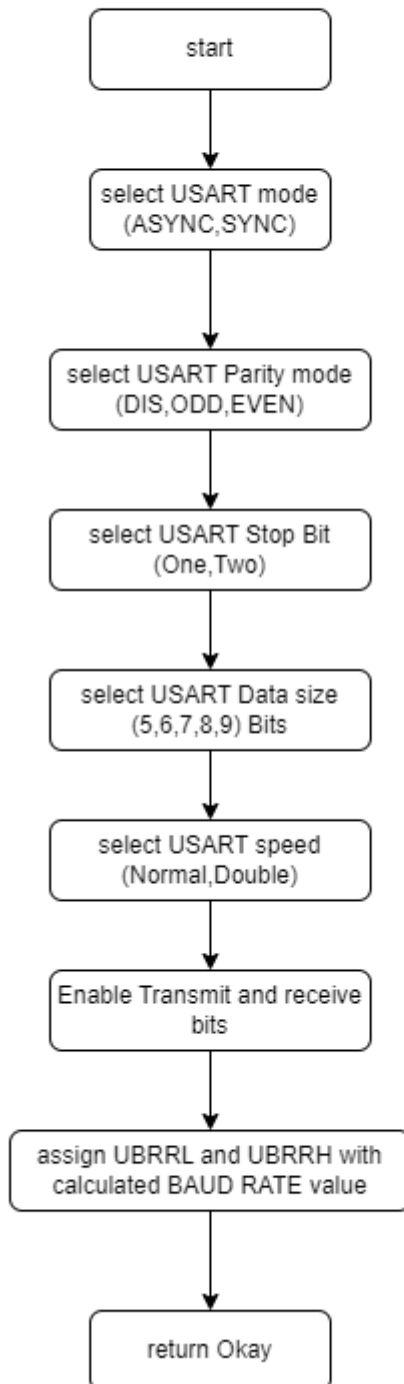


3. EXTINT_CALLBACK

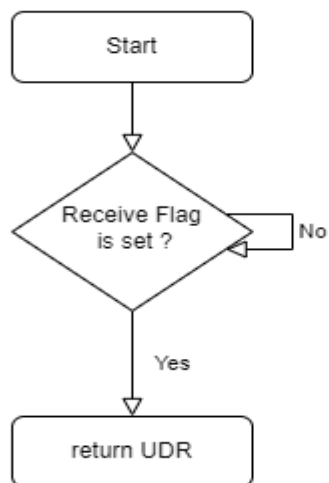


- **UART:**

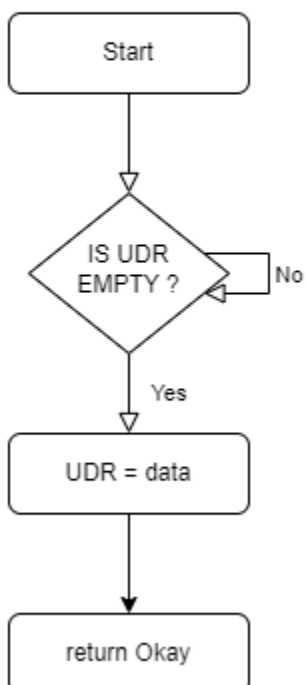
USART_init()



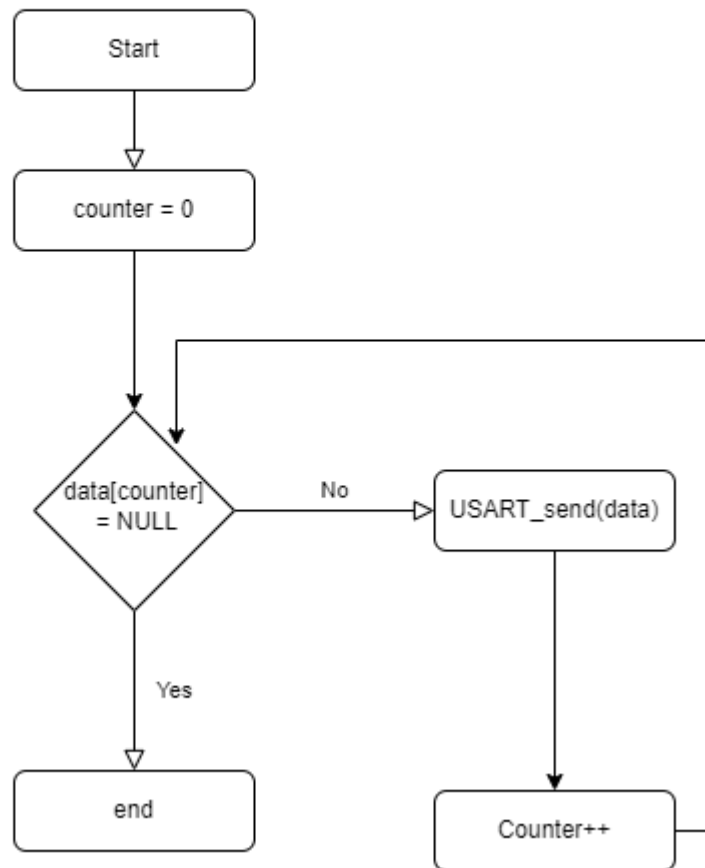
USART_receiveDATA()



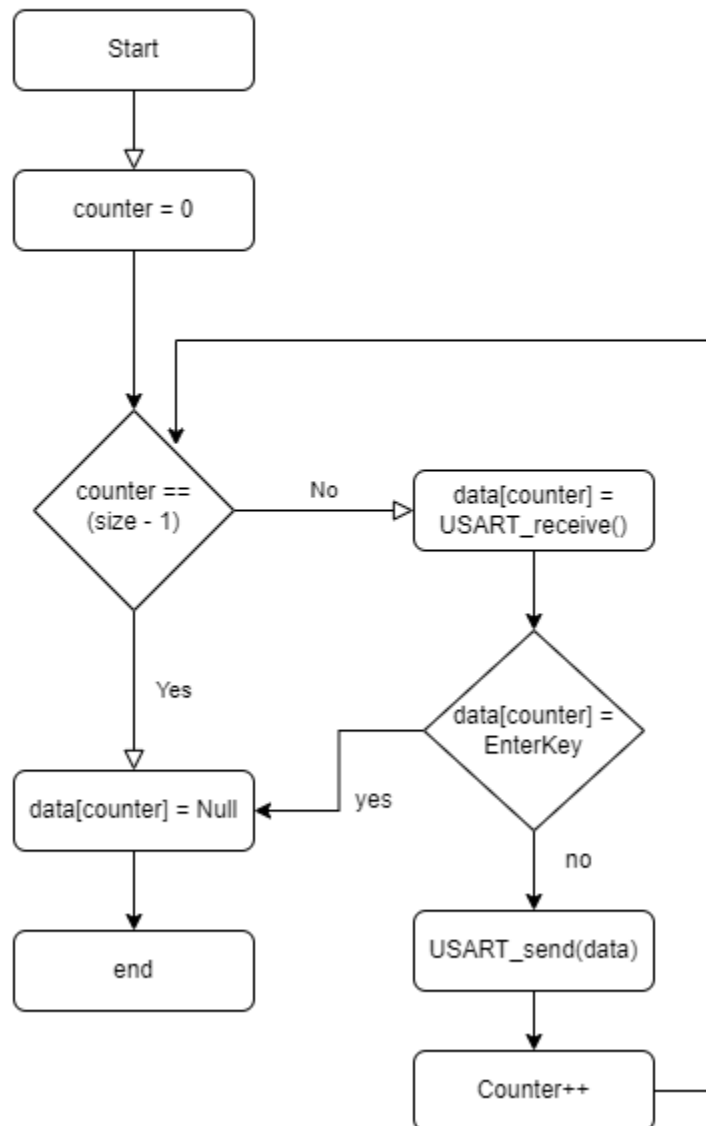
USART_sendData(data)



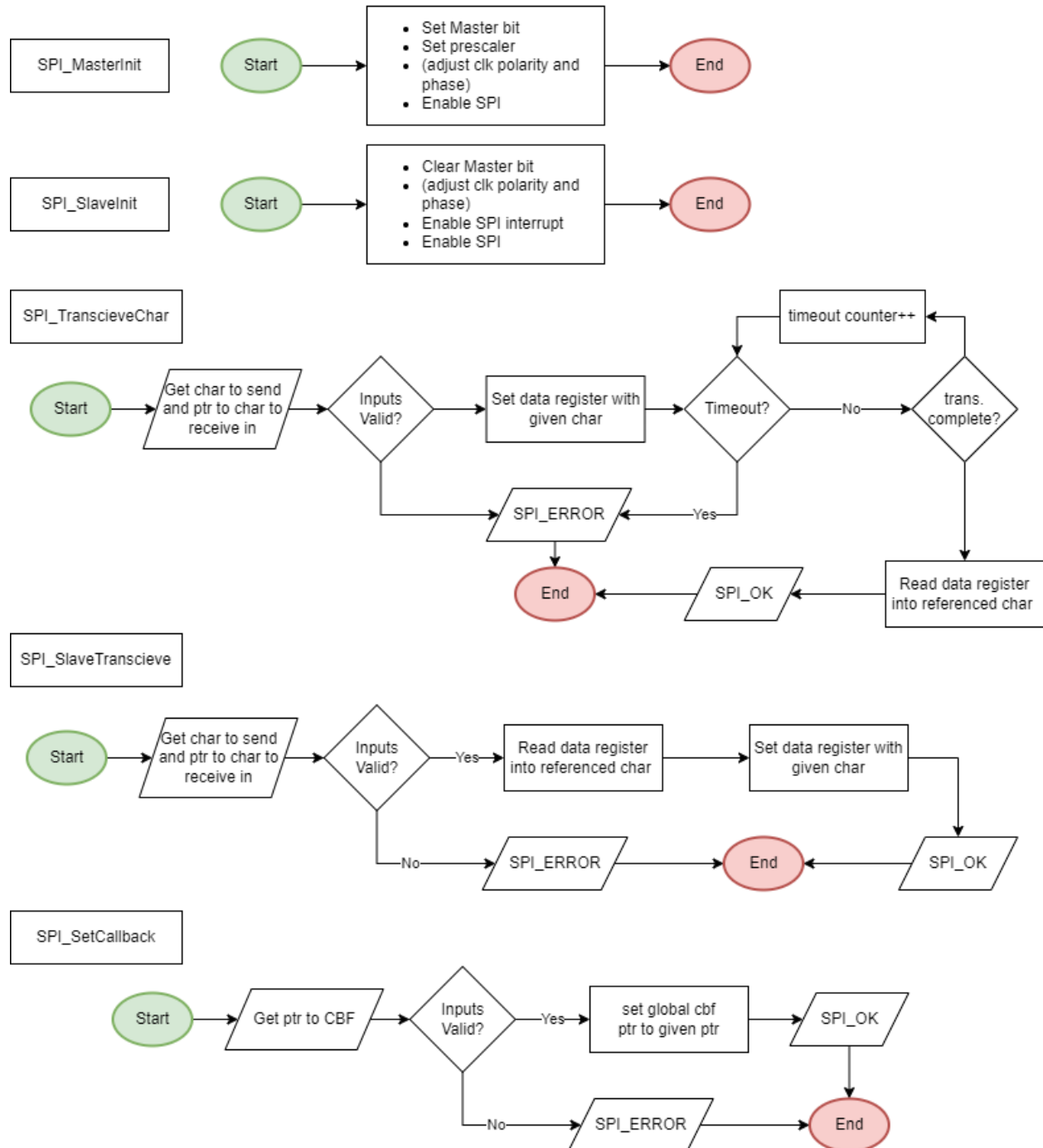
USART_sendString(*data)



USART_receiveString(*data,Size)

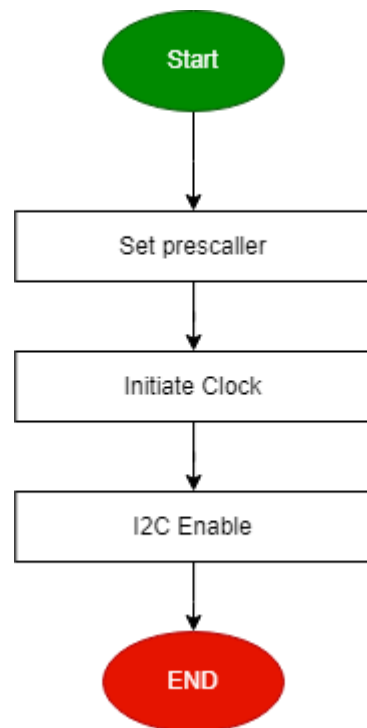


• SPI:

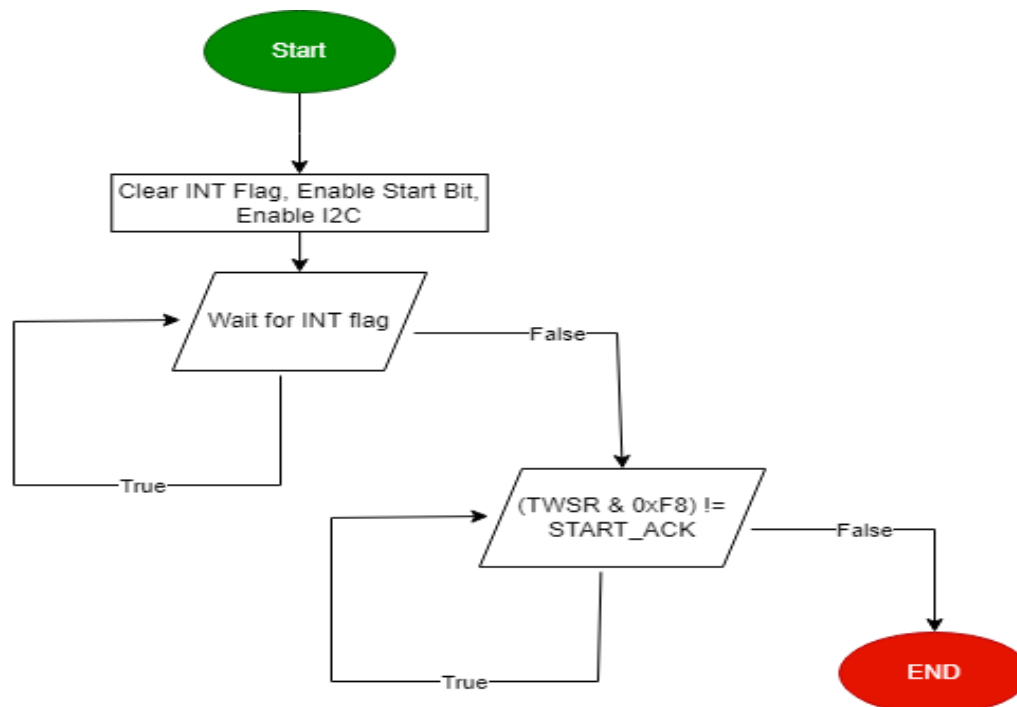


- **I2C:**

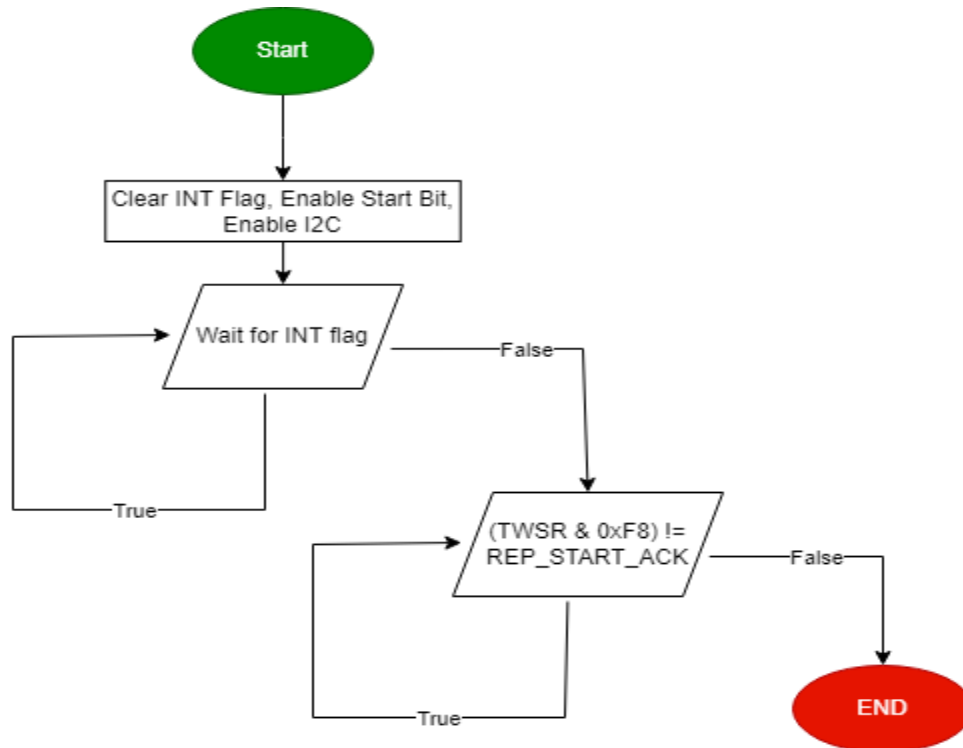
```
void i2c_init_master(void);
```



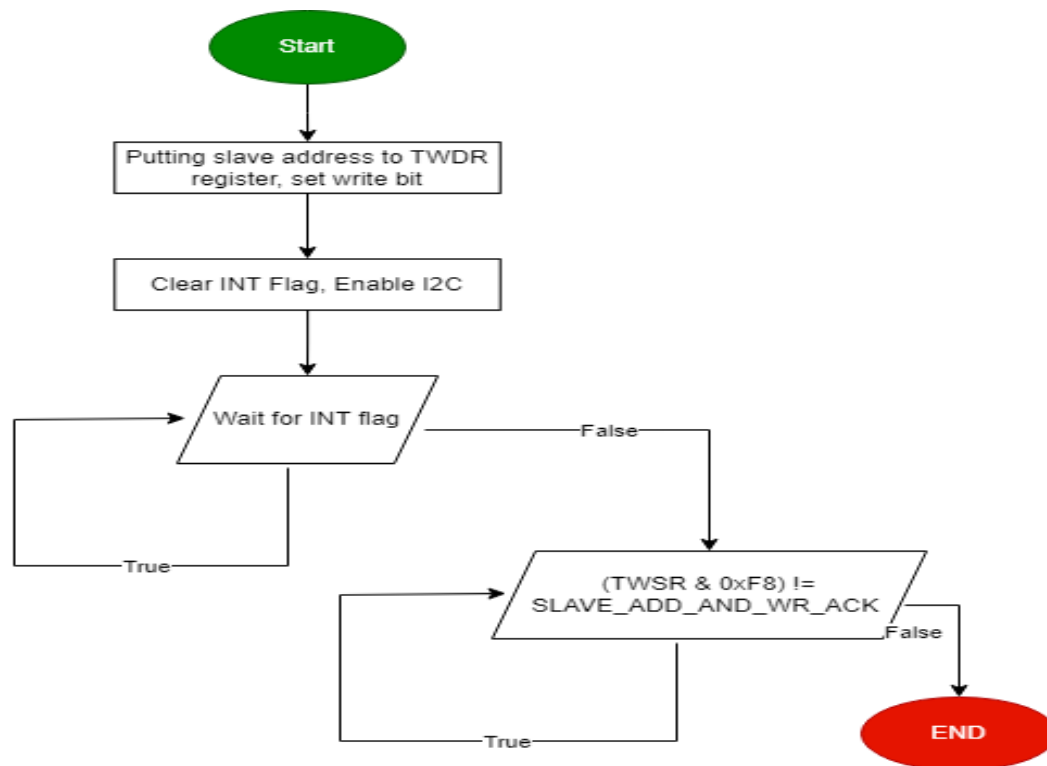
```
void i2c_start(void);
```



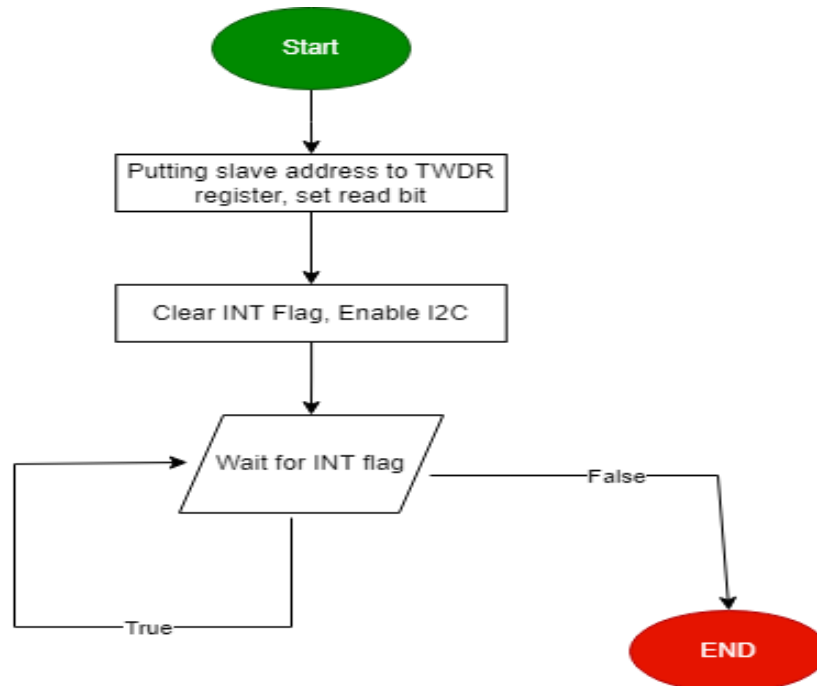
```
void i2c_repeated_start(void);
```



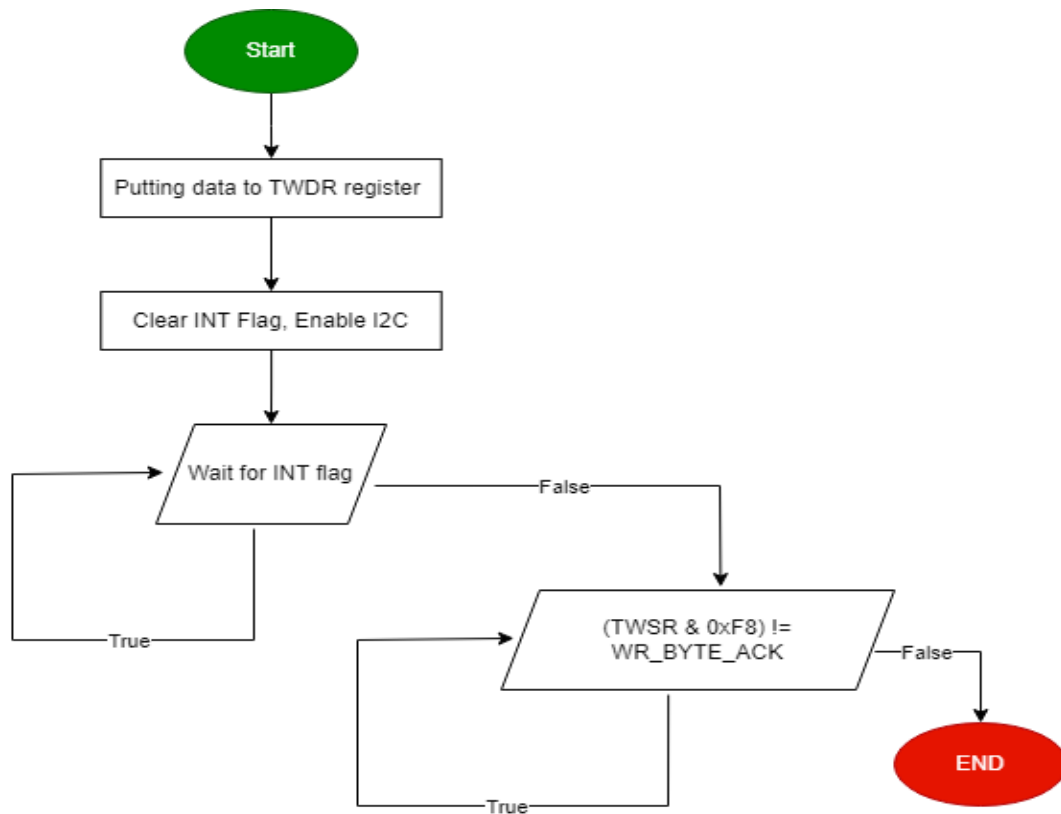
```
void i2c_send_slave_address_with_write_req(Uint8_t slave_address);
```



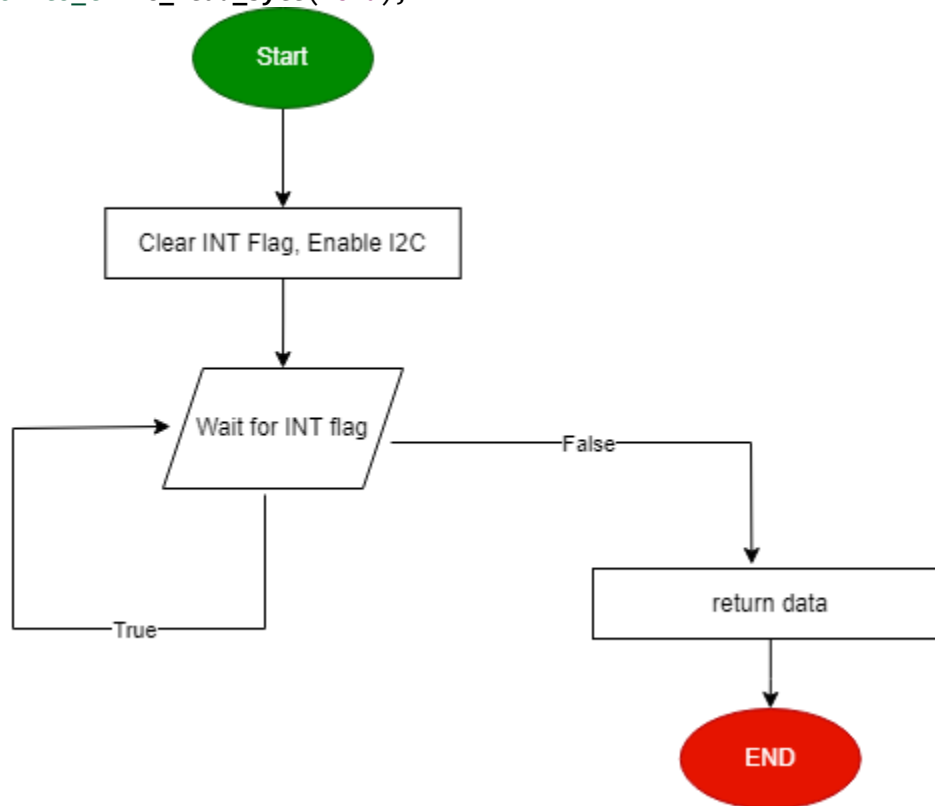
```
void i2c_send_slave_address_with_read_req(Uint8_t slave_address);
```



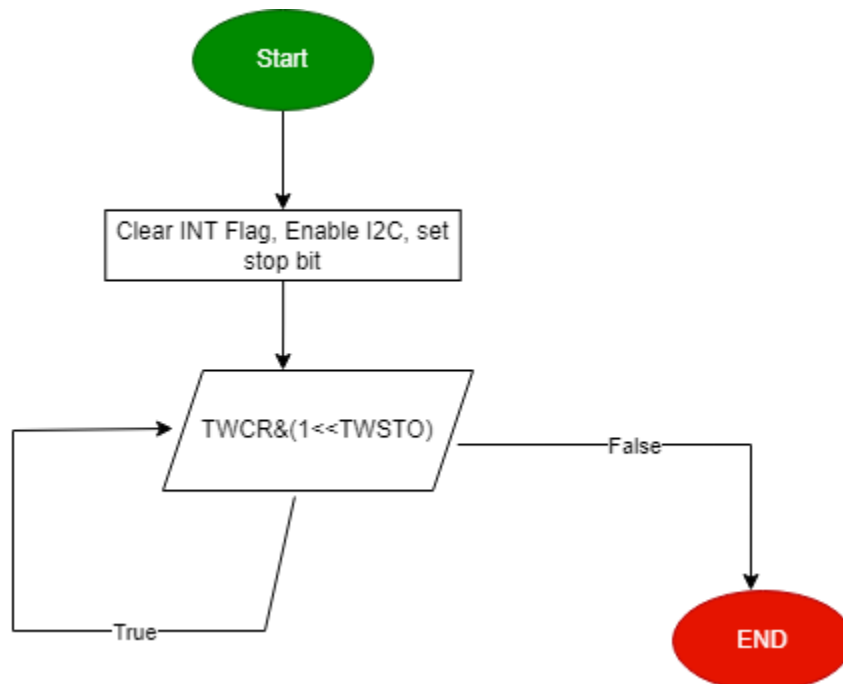
```
void i2c_write_byte(Uint8_t byte);
```




```
Uint8_t i2c_read_byte(void);
```



```
void i2c_stop(void);
```



HAL Layer

- HTimer0
HTIM0_SyncDelay

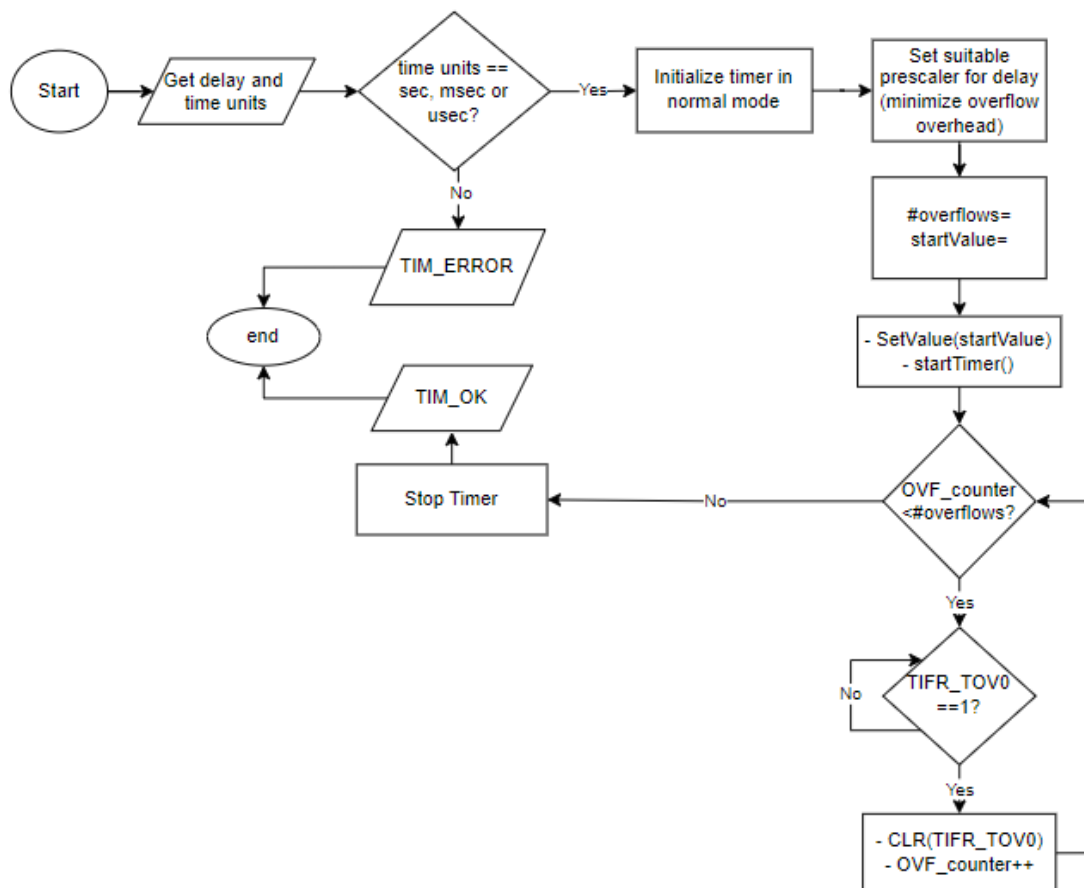


Figure 8 HTIM0_SyncDelay Flow Chart

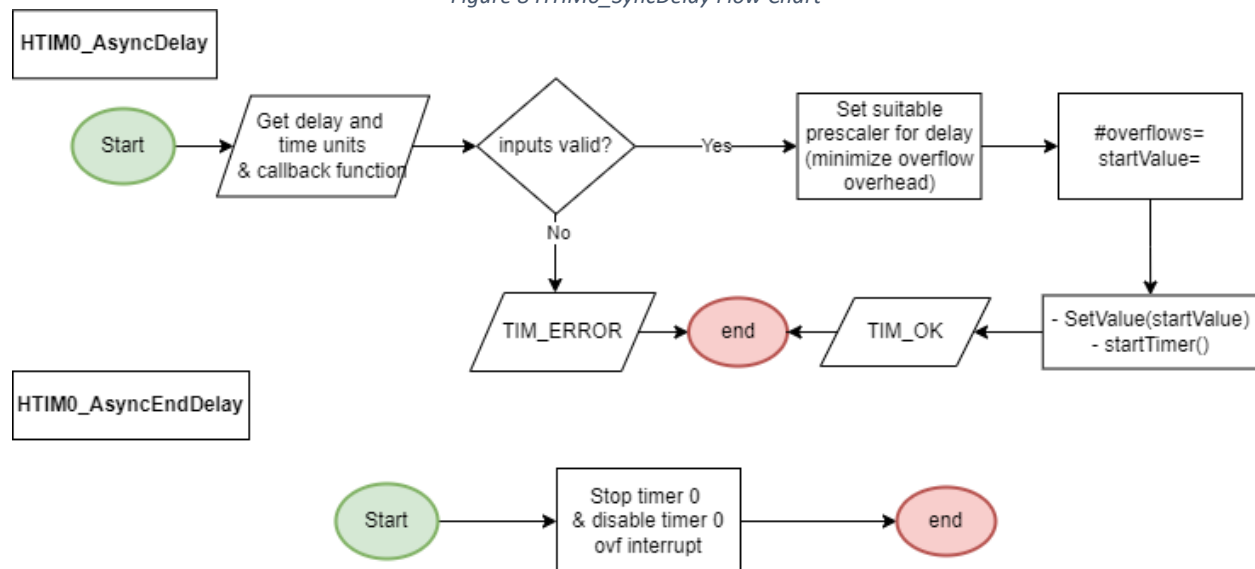
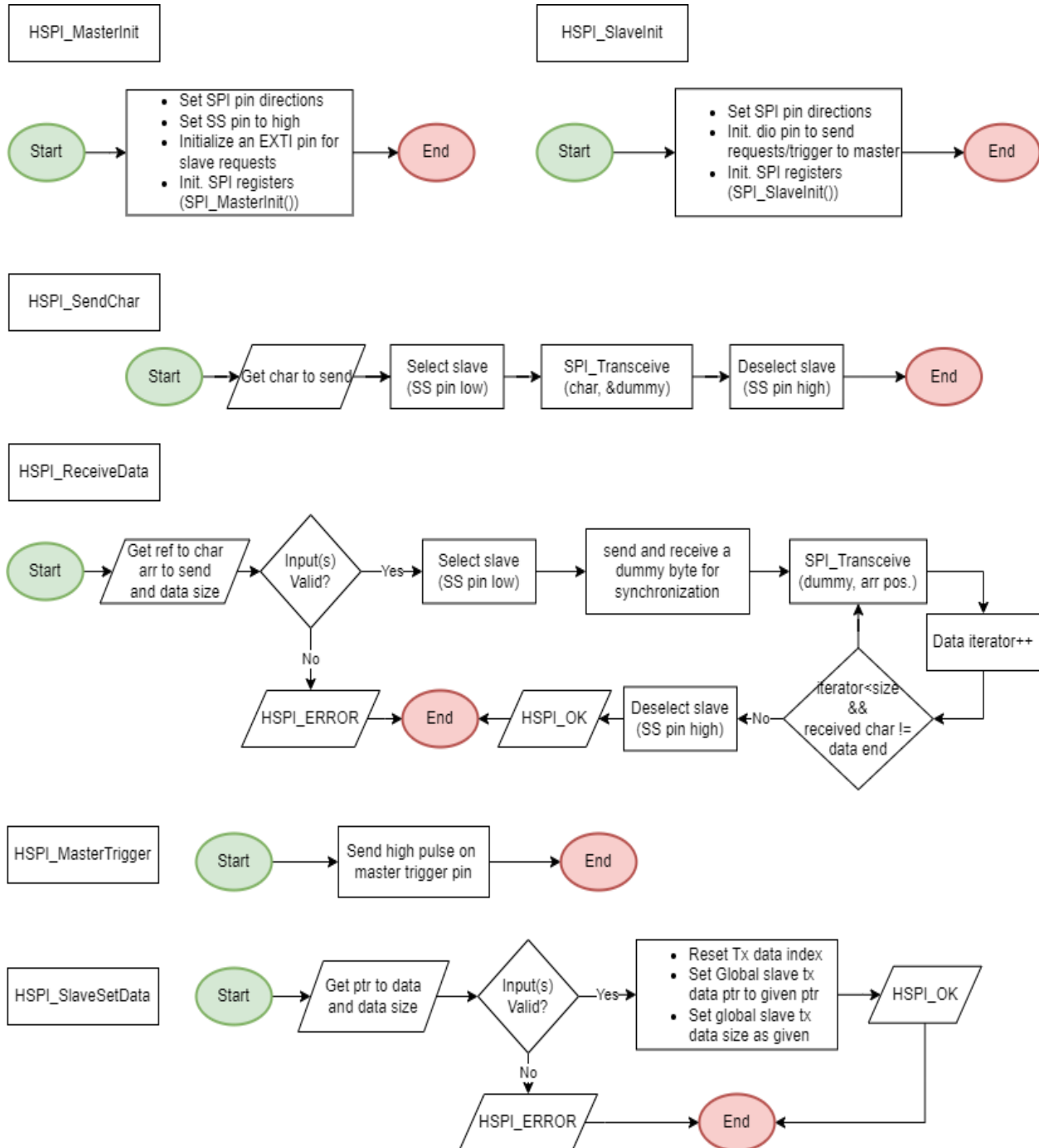


Figure 9 HTIM0_AsyncDelay and EndDelay

• HSPI



• LCD

`void HLCD_vidInit(void)`

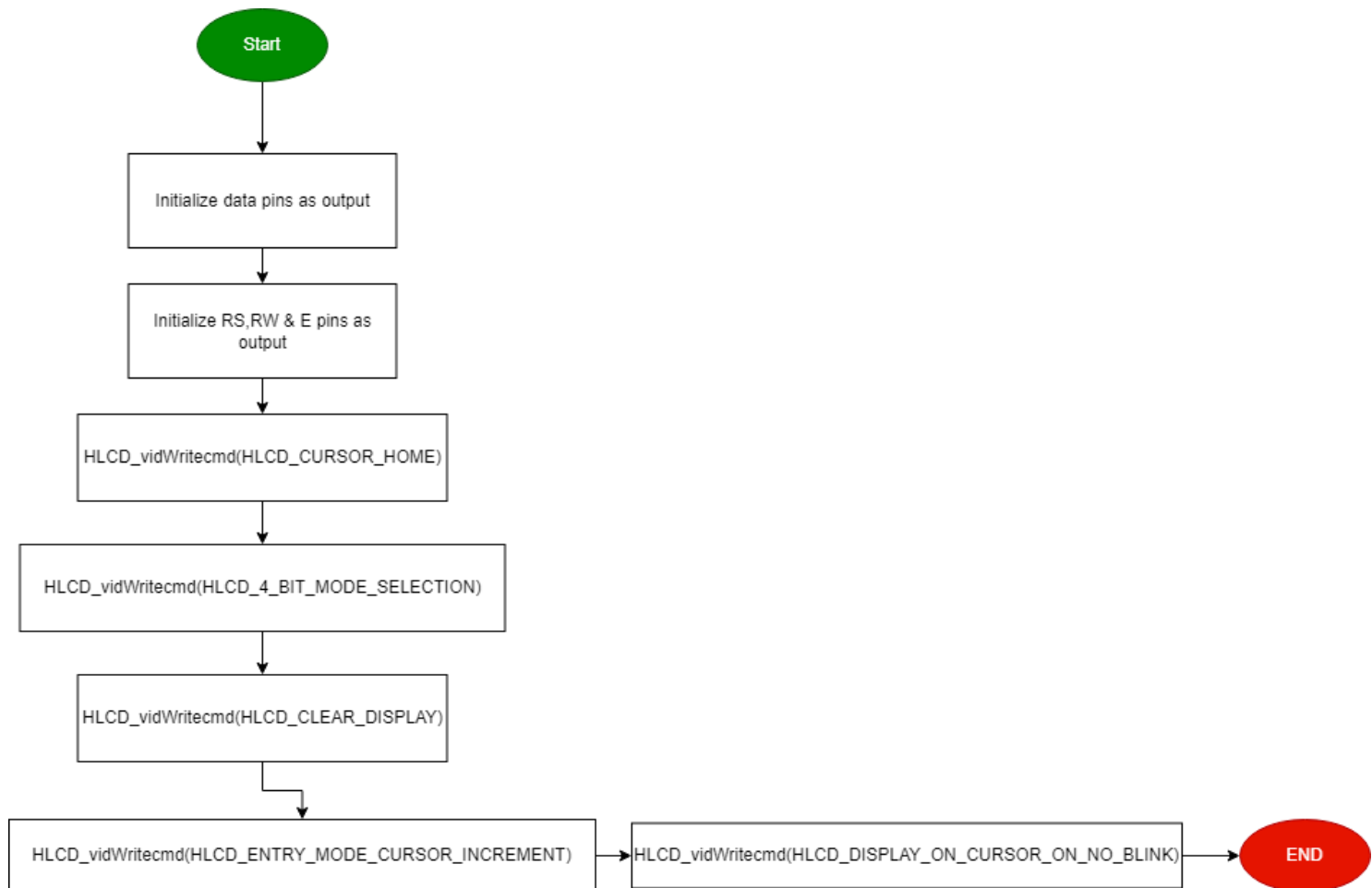


Figure 10 HLCD_vidInit Flow Chart

```
void HLCD_vidWritecmd(Uint8_t u8commandCopy)
```

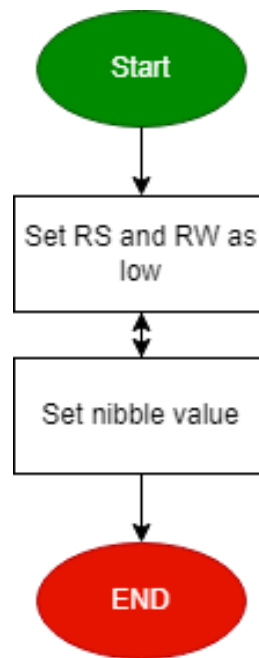


Figure 11 HLCD_vidWritecmd Flow Chart

```
void HLCD_vidWriteChar(Uint8_t u8CharCopy)
```

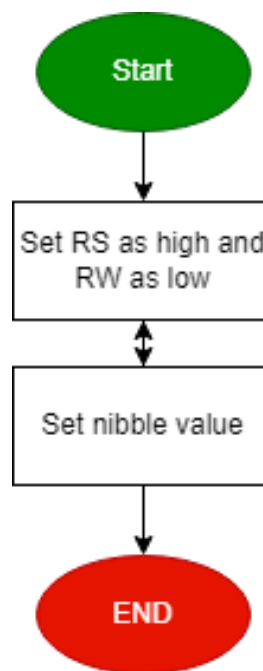


Figure 12 HLCD_vidWriteChar Flow Chart

```
void HLCD_ClrDisplay(void)
```

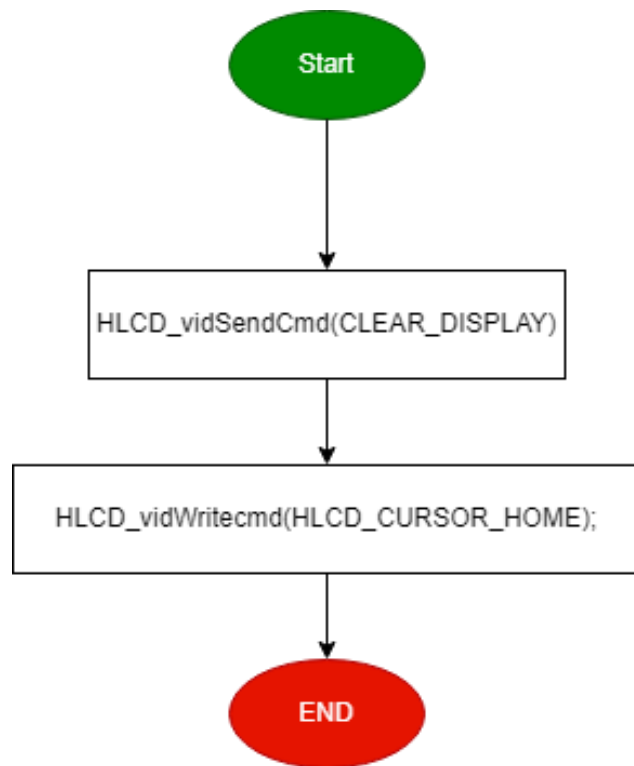


Figure 13 HLCD_ClrDisplay Flow Chart

```
void HLCD_gotoXY(Uint8_t row, Uint8_t pos)
```

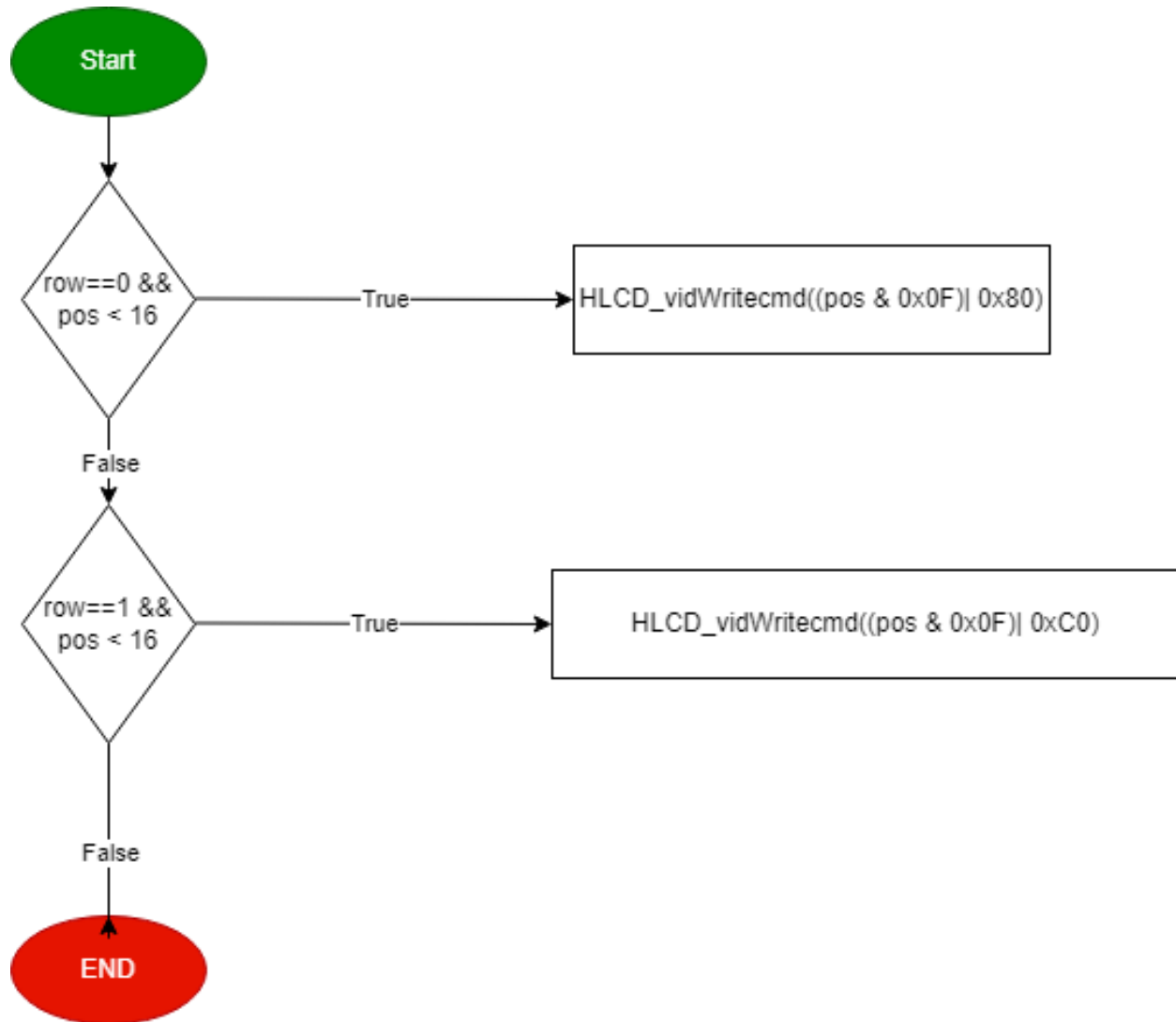


Figure 14 HLCD_gotoXY Flow Chart

```
void HLCD_WriteString(Uint8_t* str)
```

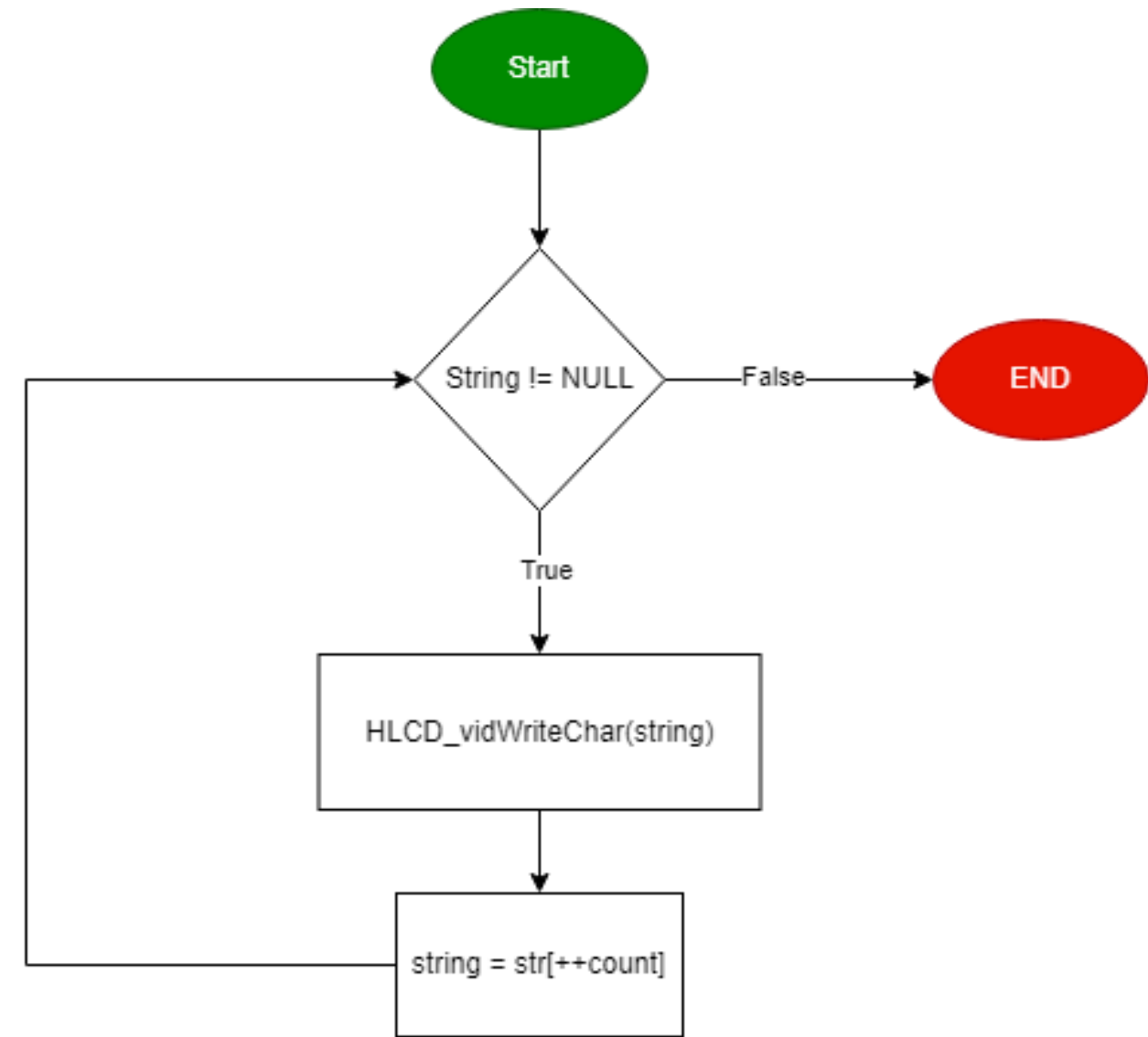


Figure 15 HLCD_WriteString Flow Chart


```
void HLCD_WriteInt(Uint32_t number)
```

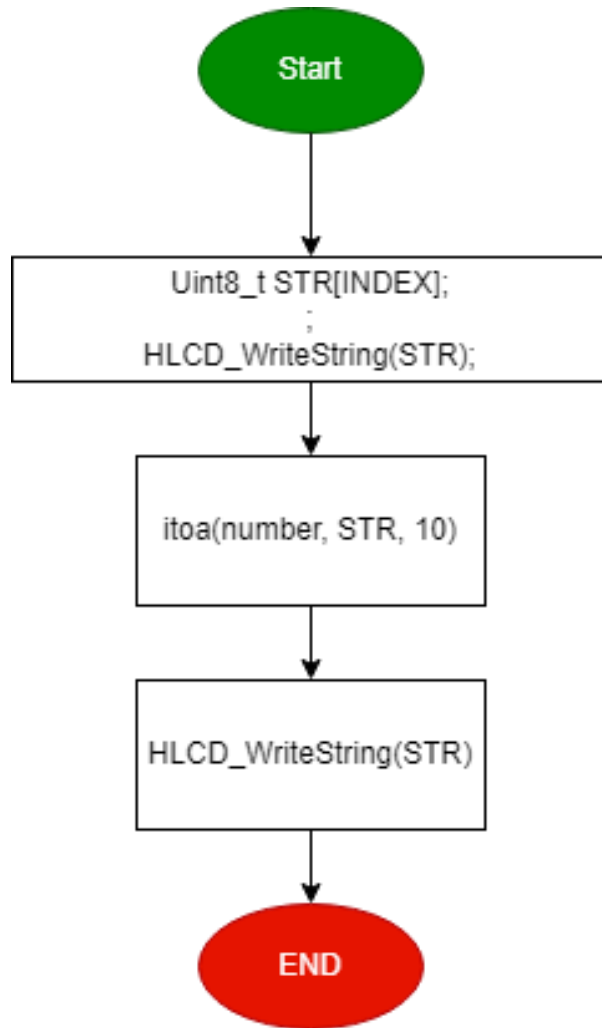


Figure 16 HLCD_WriteInt Flow Chart

```
void HLCD_vidCreatCustomChar(Uint8_t* pu8custom, Uint8_t u8Location)
```

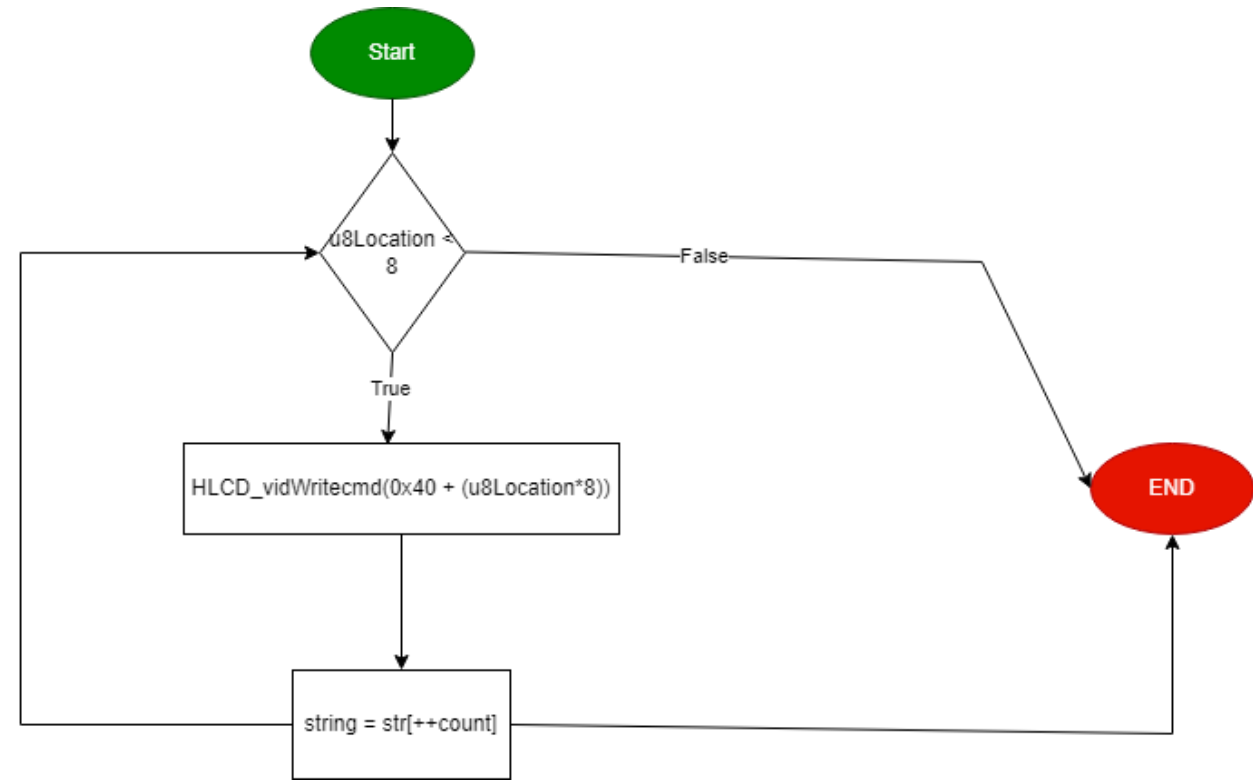
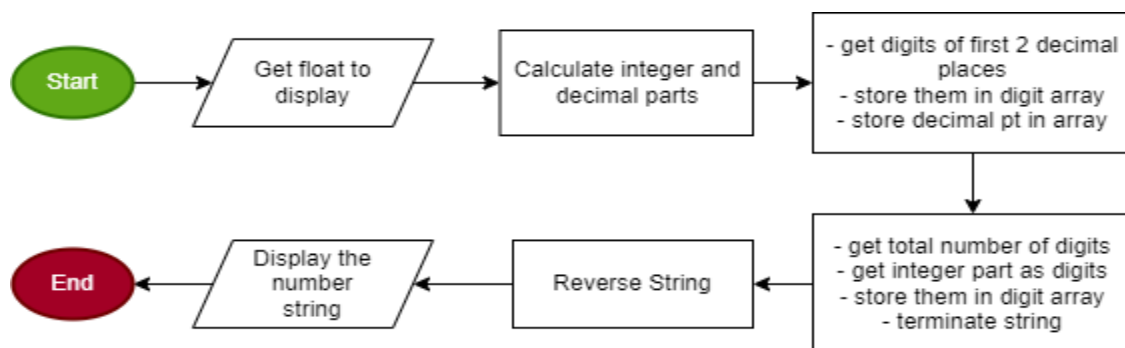


Figure 17 HLCD_vidCreatCustomChar Flow Chart

```
void HLCD_DisplayFloat(float32_t f32_a_number);
```



- Keypad

KEYPAD_init(void)

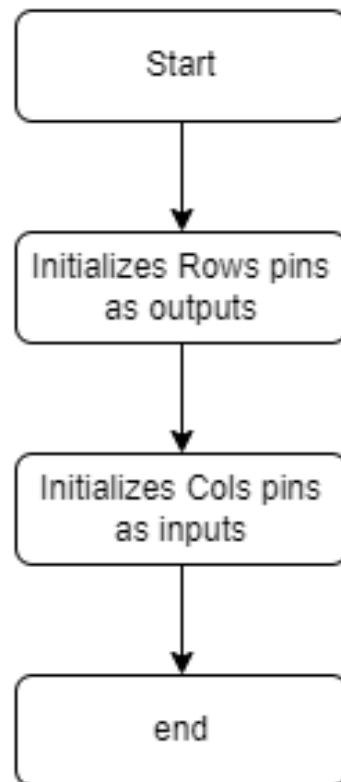


Figure 18 KEYPAD_Init Flow Chart

KEYPAD_CheckRx(void) **x here (1.2.3)**

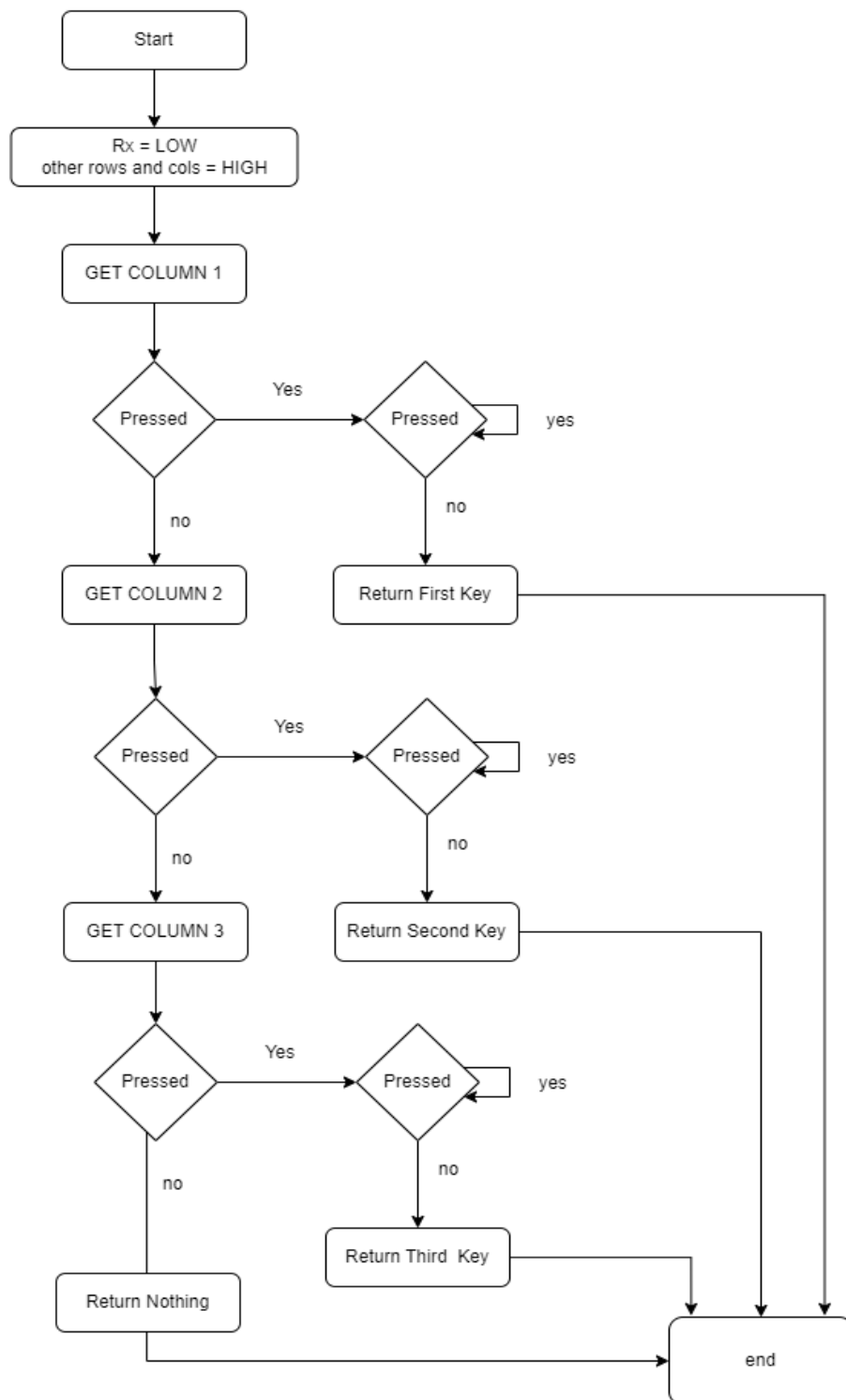


Figure 19 KEYPAD_CheckRx Flow Chart

GetButton(void)

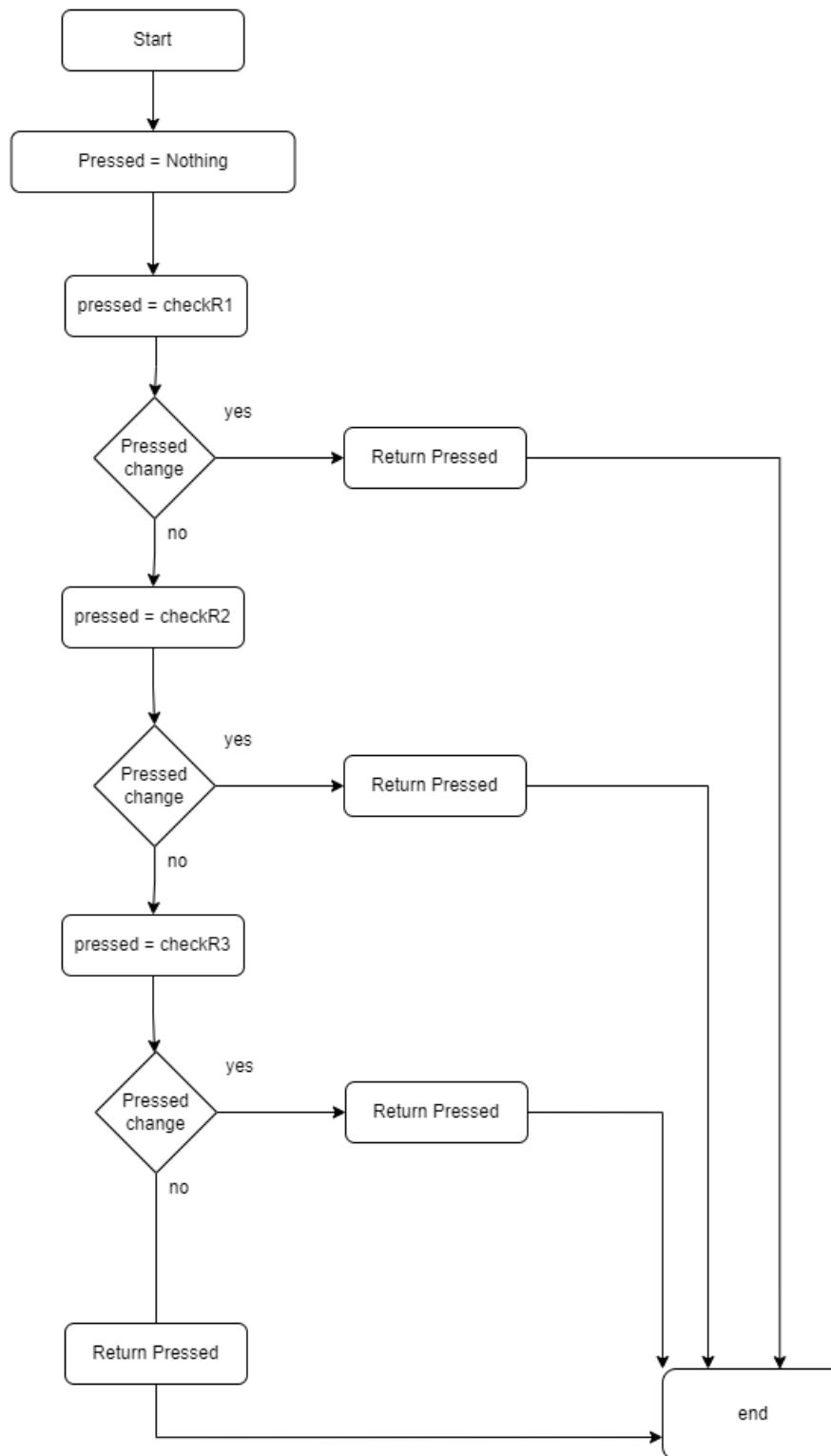
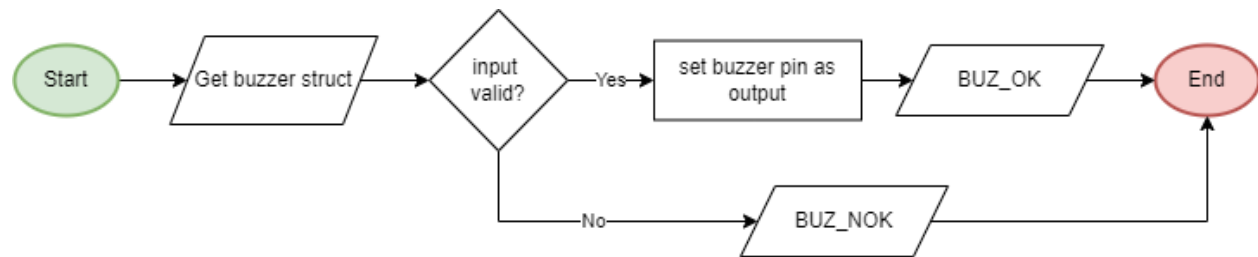


Figure 20 GetButton Flow Chart

• Buzzer

BUZ_Init



BUZ_SetState

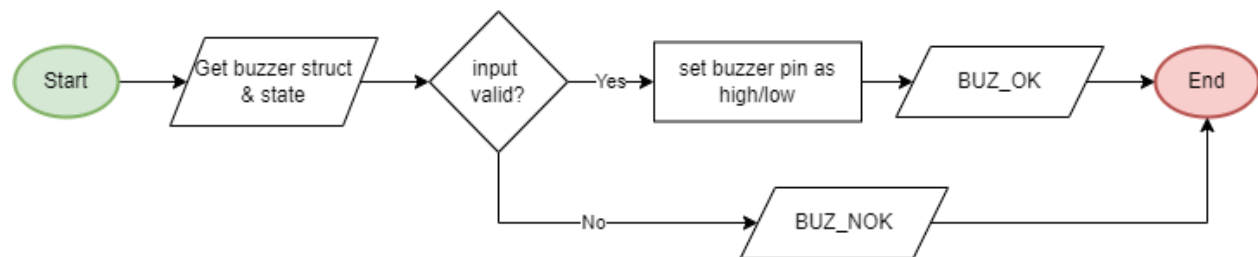
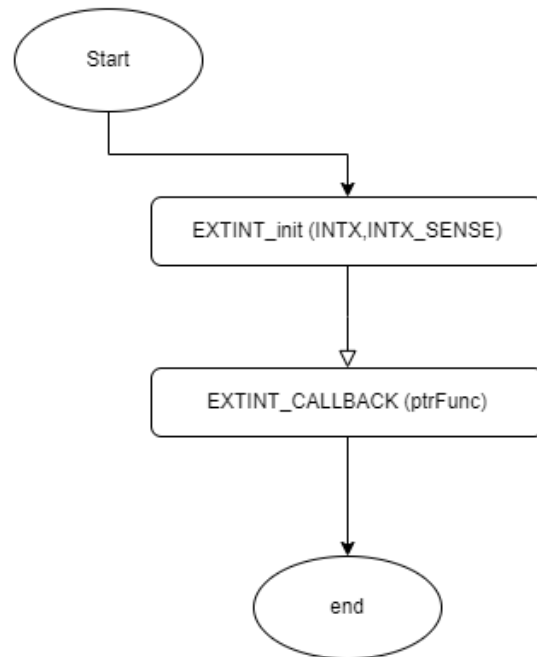


Figure 21 Buzzer Init & SetState Flow Charts

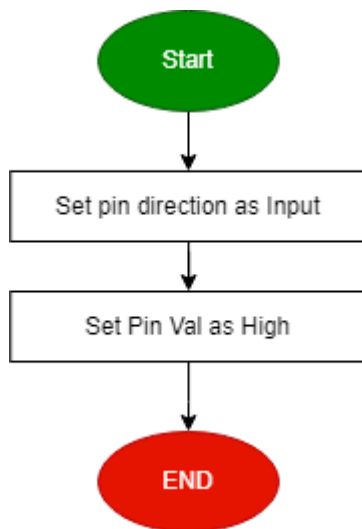
- **HEXTINT:**

H_EXTINT_create(INTx ,INTxSense,*ptrfunc)

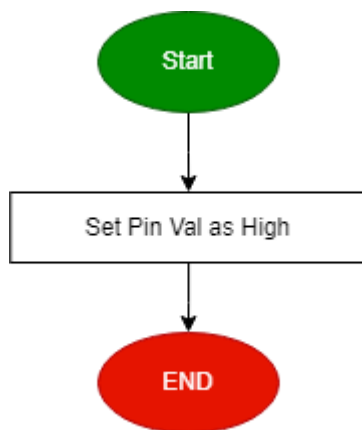


- **Button:**

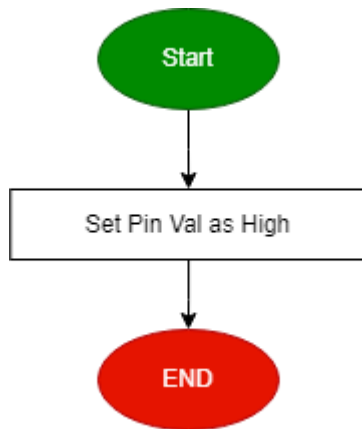
```
enu_buttonError_t HButton_Init(enu_pin en_pinx);
```



```
enu_buttonError_t HButton_ExtIntInit(enu_pin en_pinx);
```

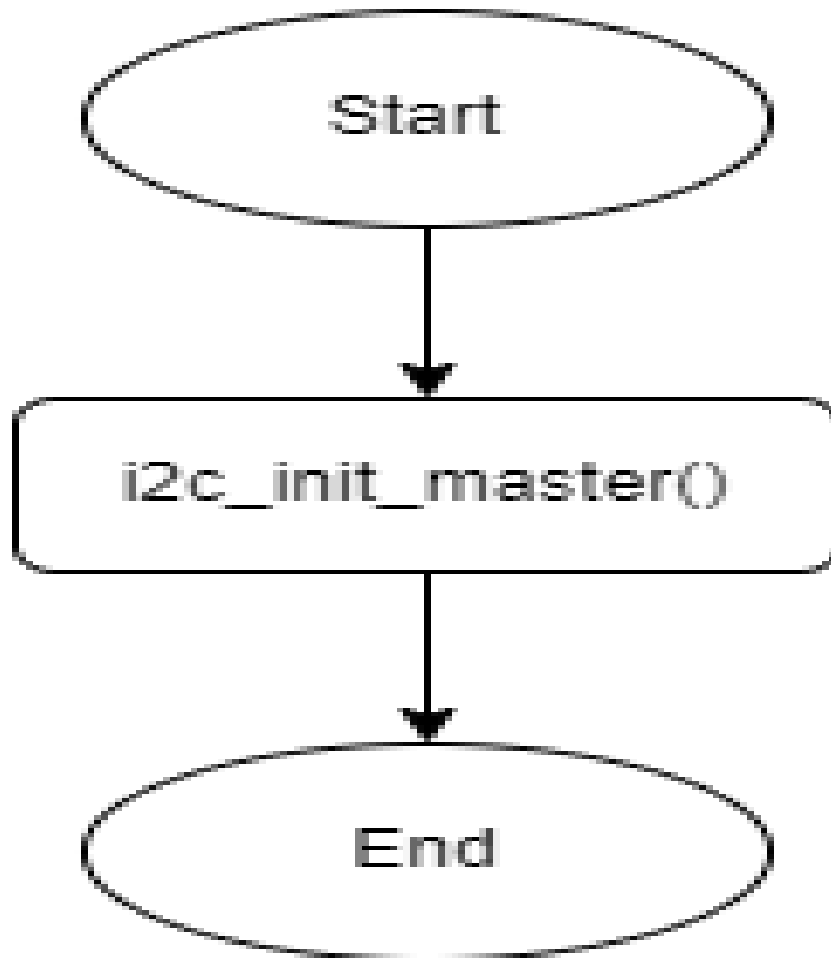


```
enu_buttonError_t HButton_getPinVal(enu_pin en_pinx, Uint8_t* pu8_refVal );
```

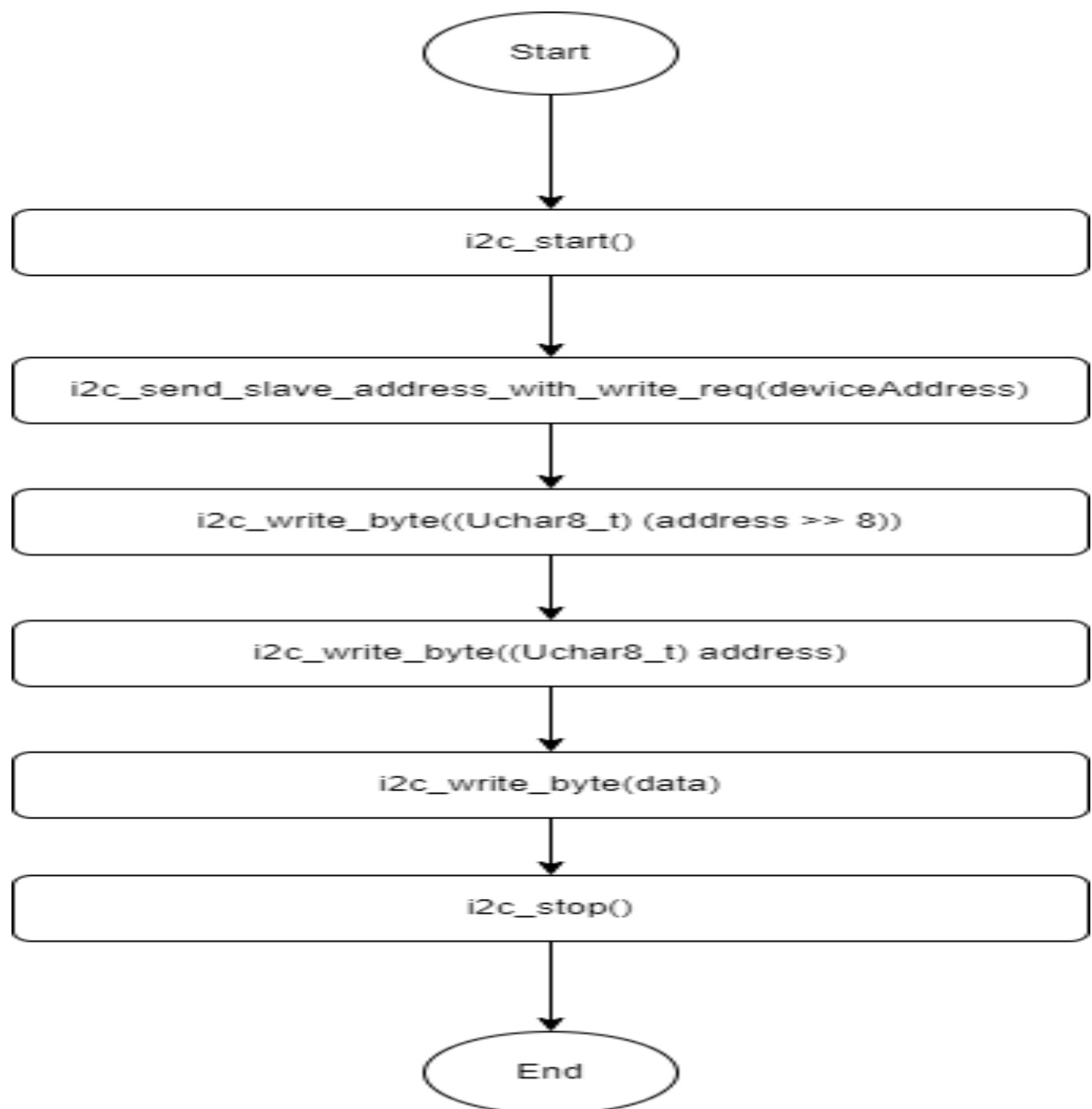


- EEPROM

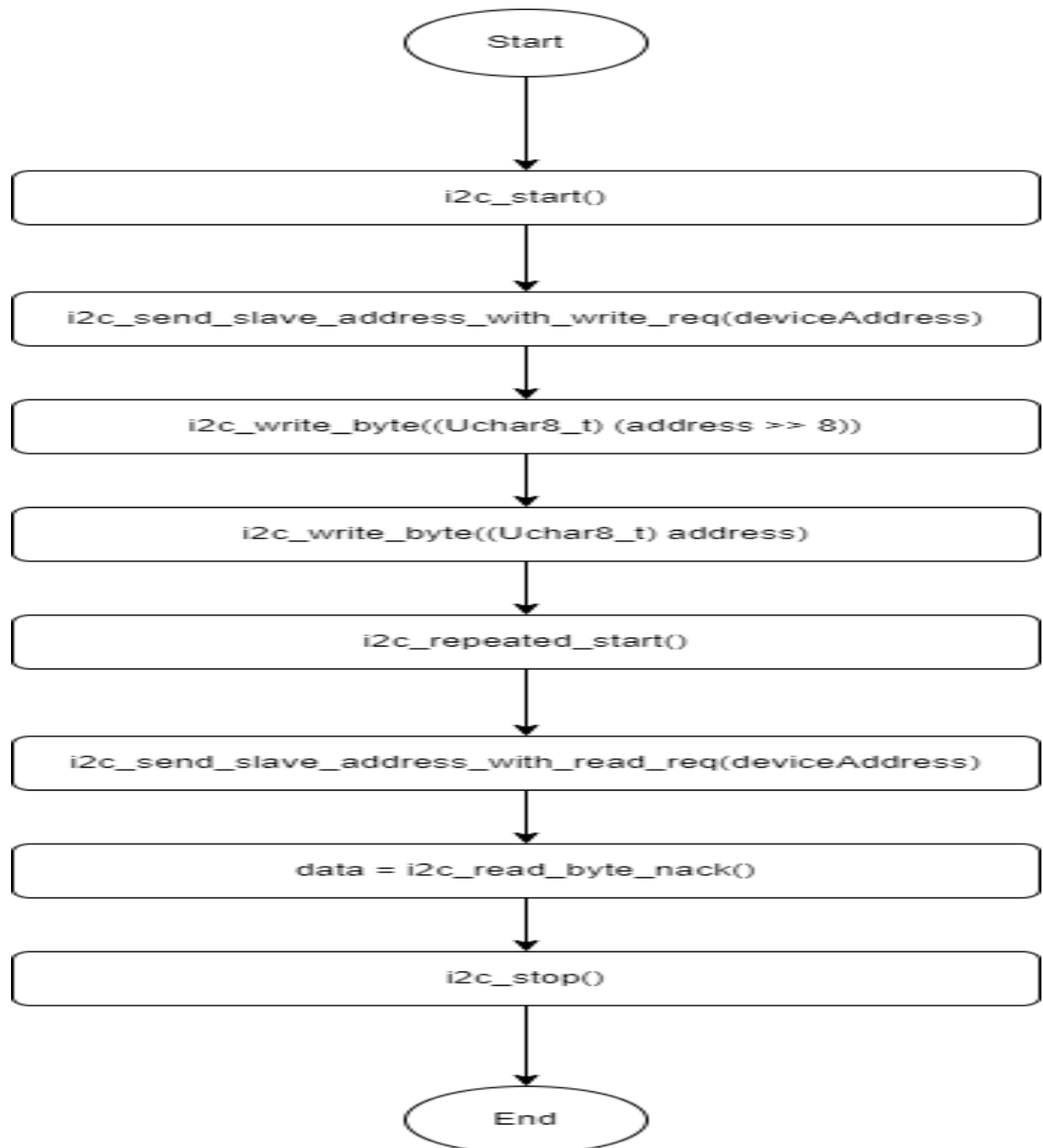
```
void eeprom_init(void)
```



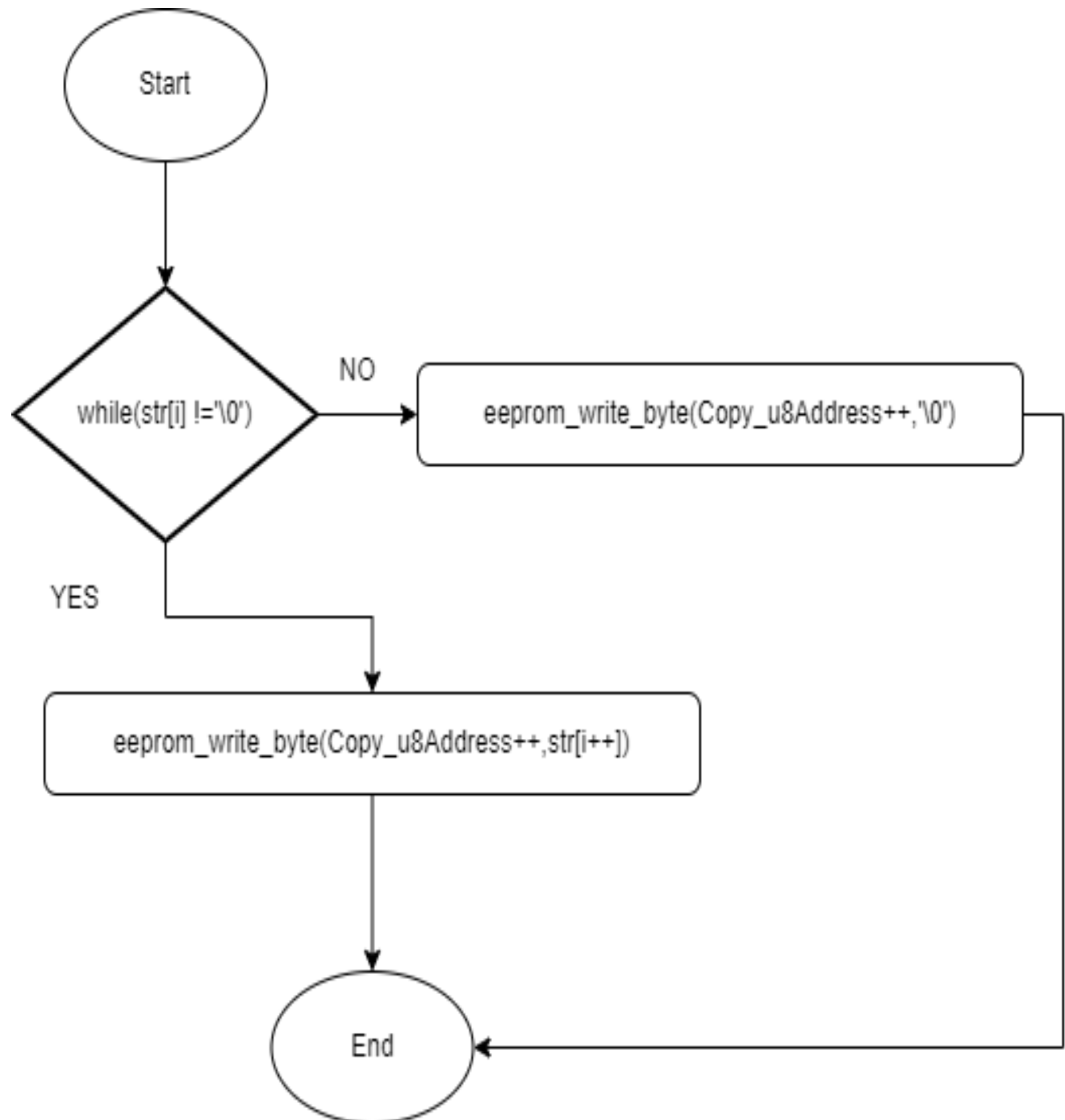
```
void eeprom_write_byte(uint16_t address, uint8_t data)
```



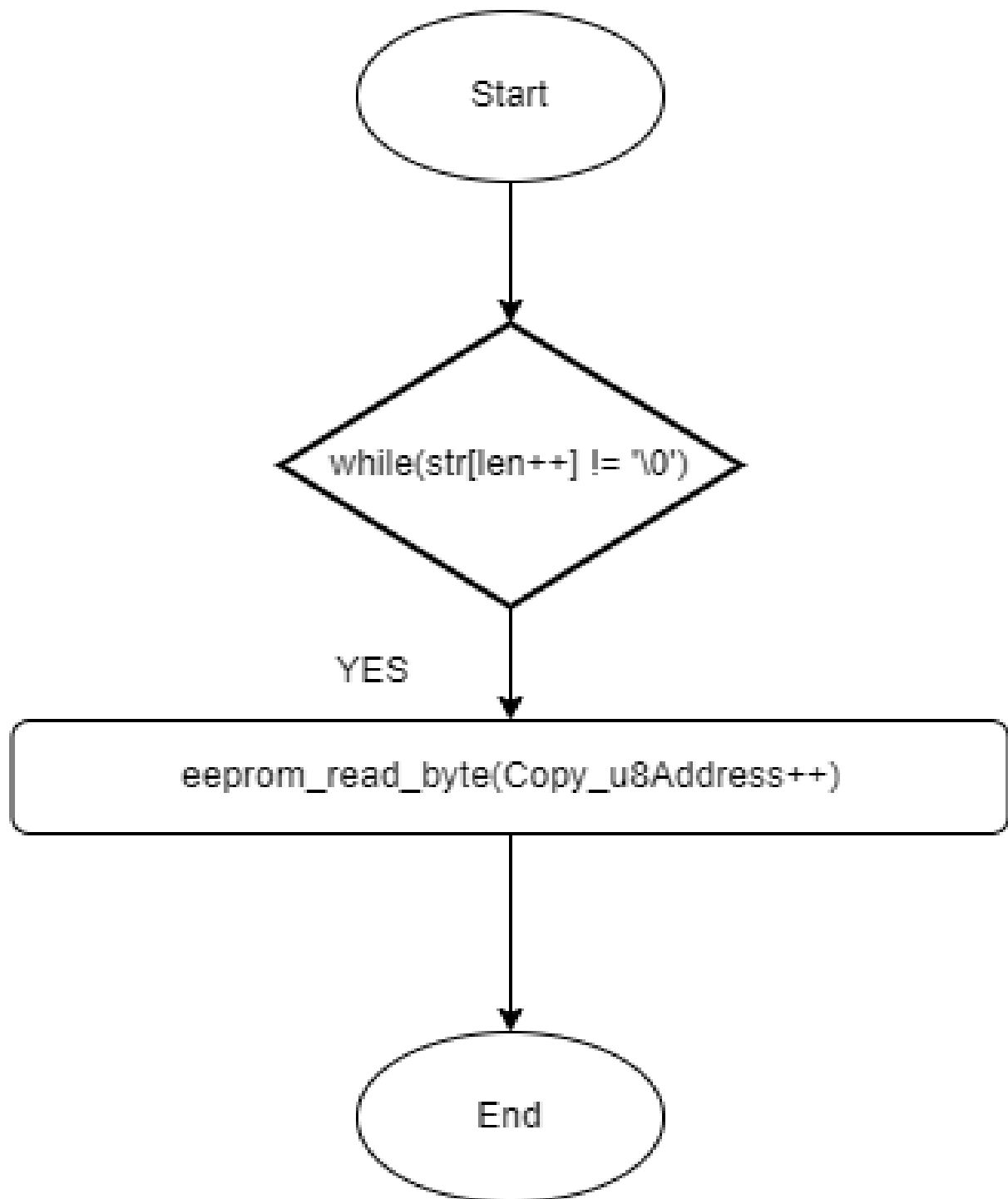
```
Uchar8_t eeprom_read_byte(Uint16_t address)
```



```
void eeprom_write_string(Uint16_t Copy_u8Address, const Uchar8_t* str)
```

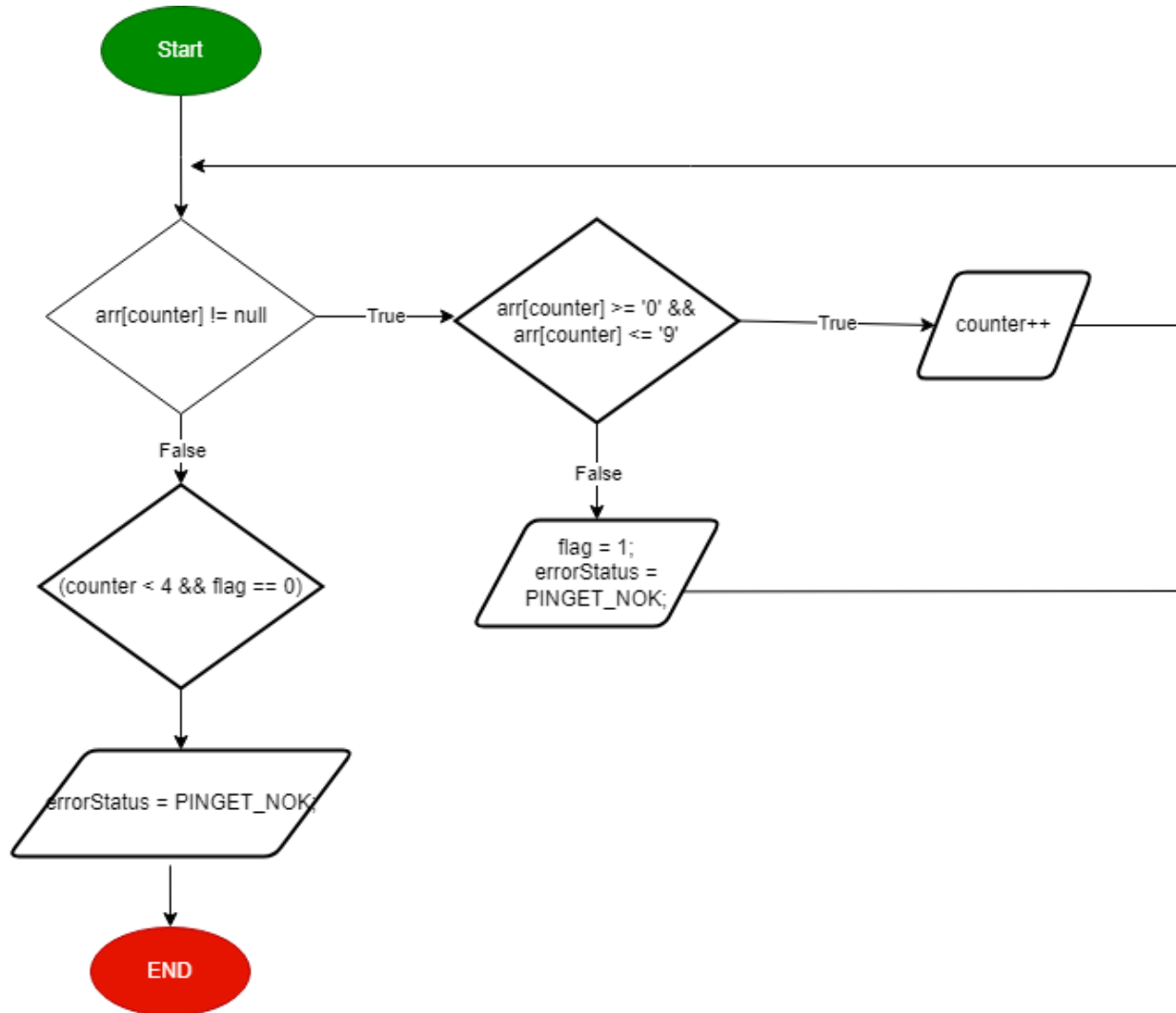


```
void eeprom_read_string(Uint16_t Copy_u8Address, Uchar8_t* str)
```

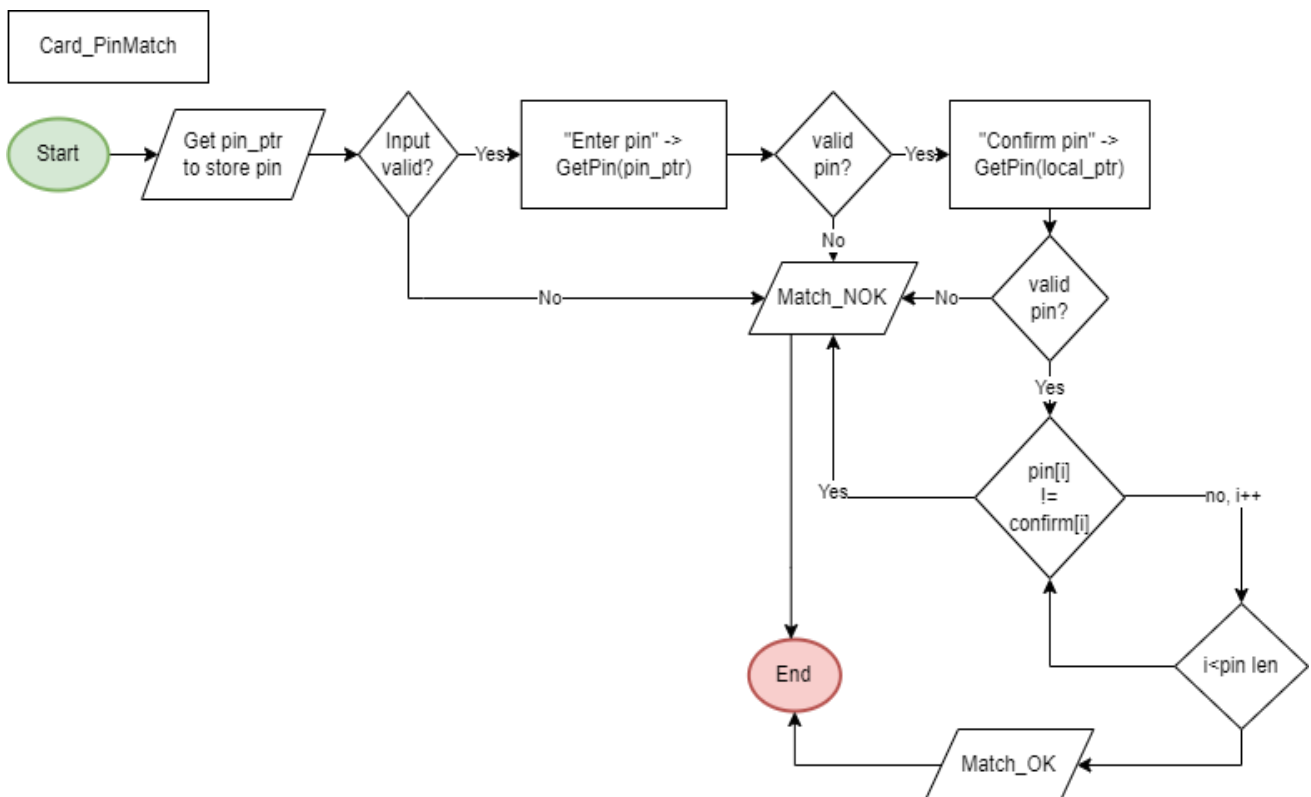
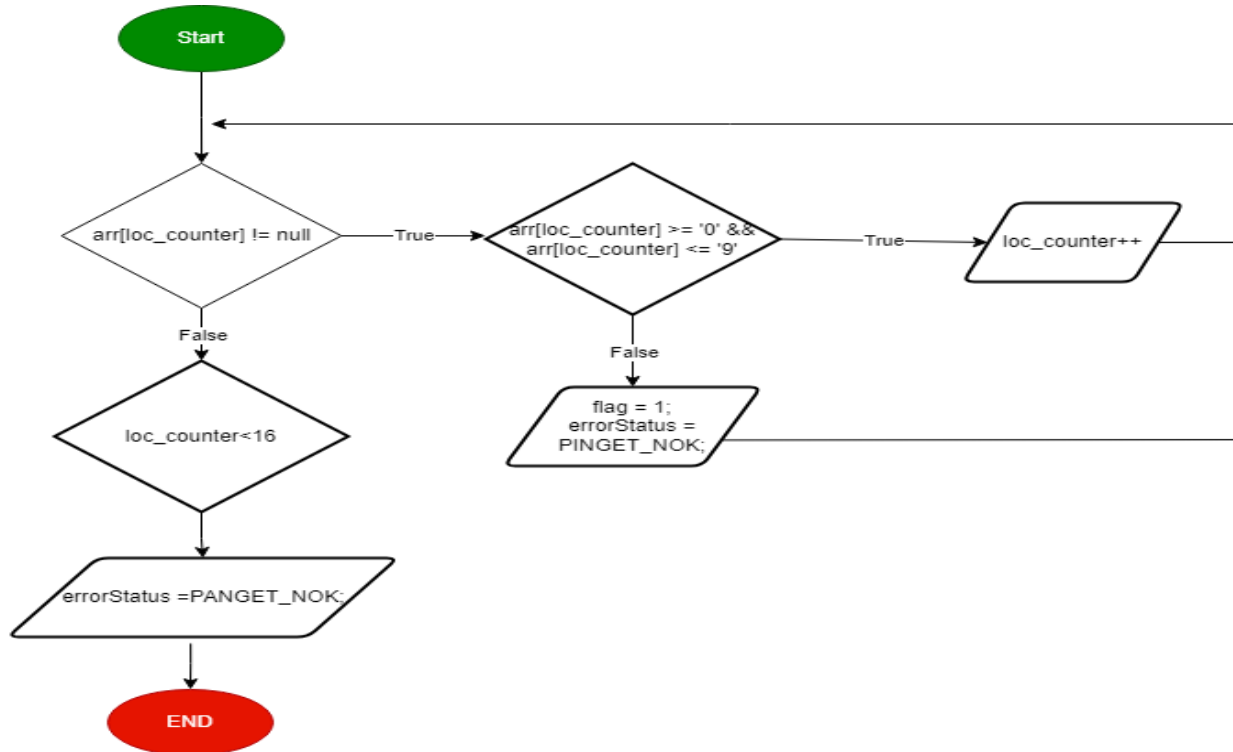


Card Database Layer (CARD MCU)

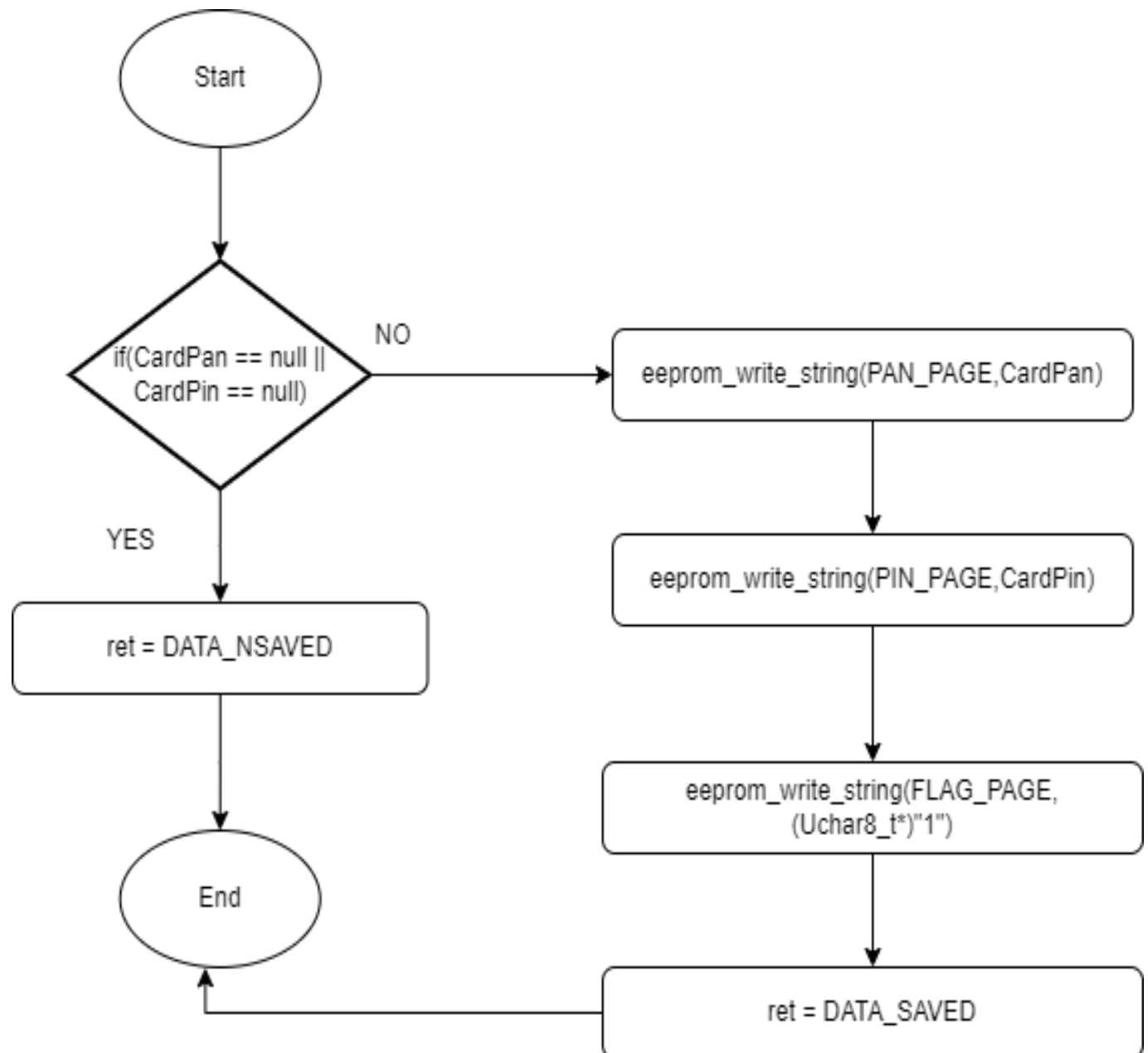
```
en_terminalPinGetStatus t APP_terminalPinGet(Uchar8 t* arr);
```



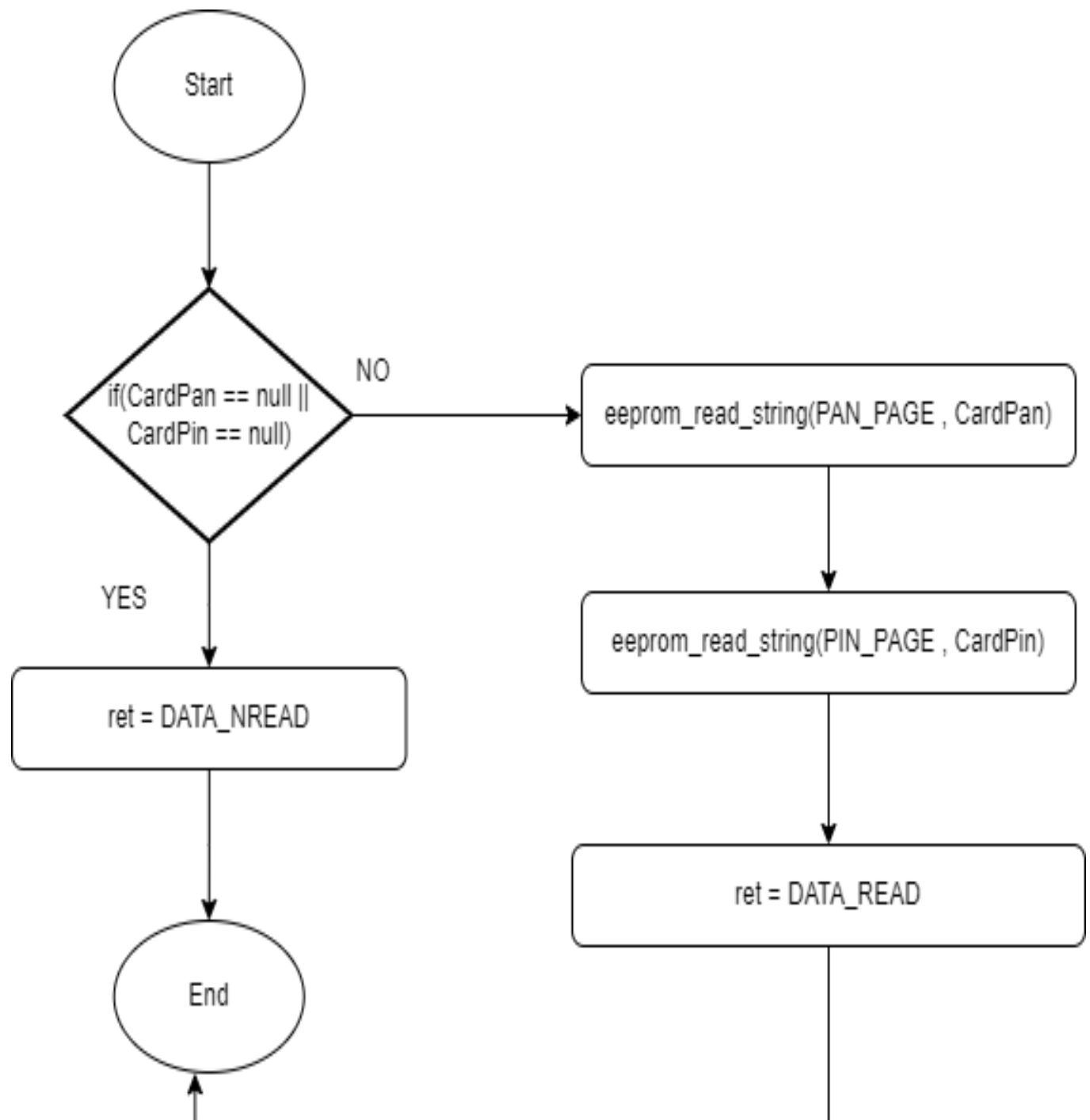
```
en_terminalPanGetStatus t APP_terminalPanGet(Uchar8 t* arr);
```



SaveCardData

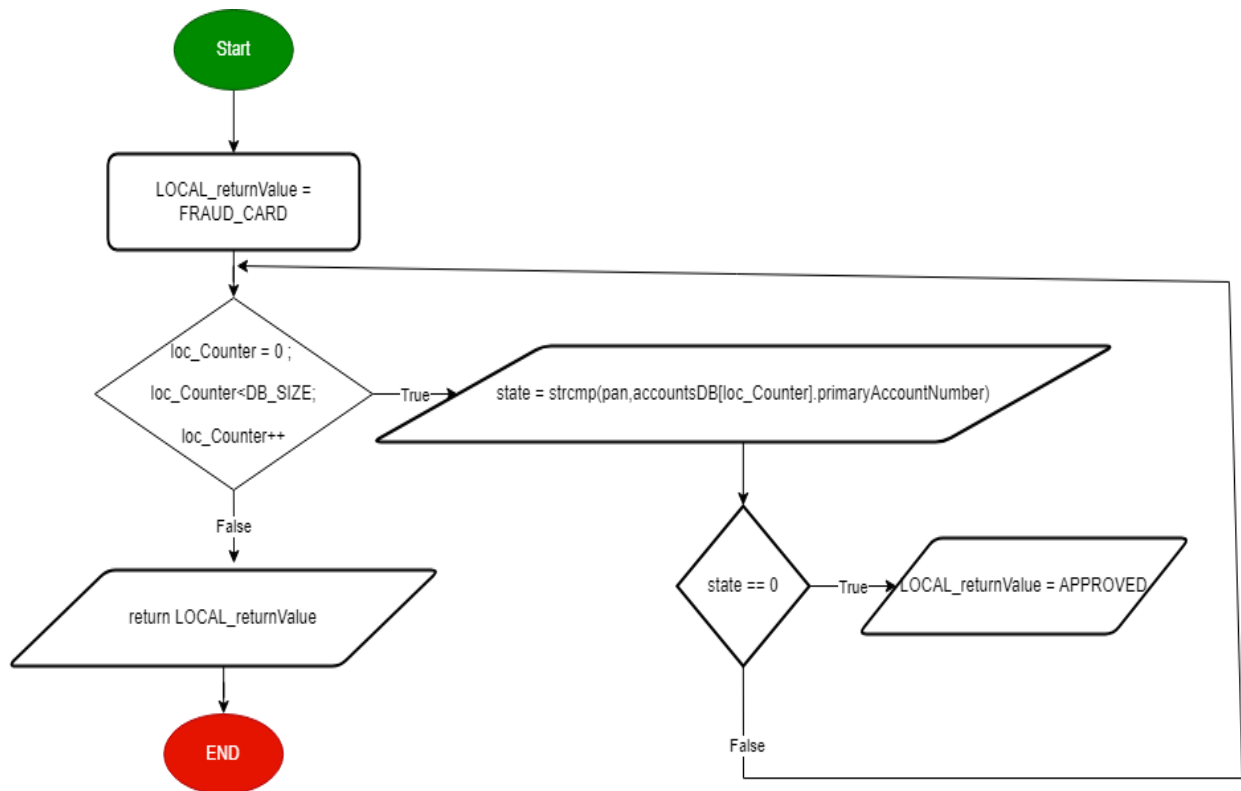


ReadCardData

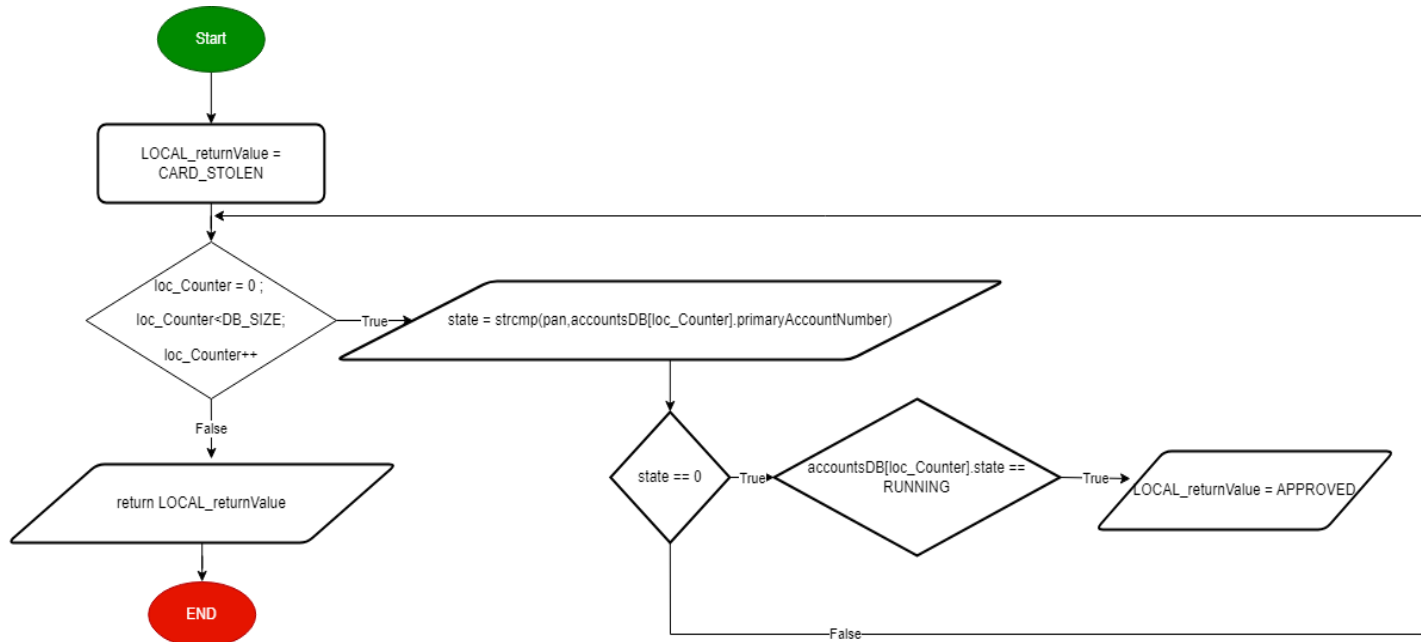


- **Database_check:**

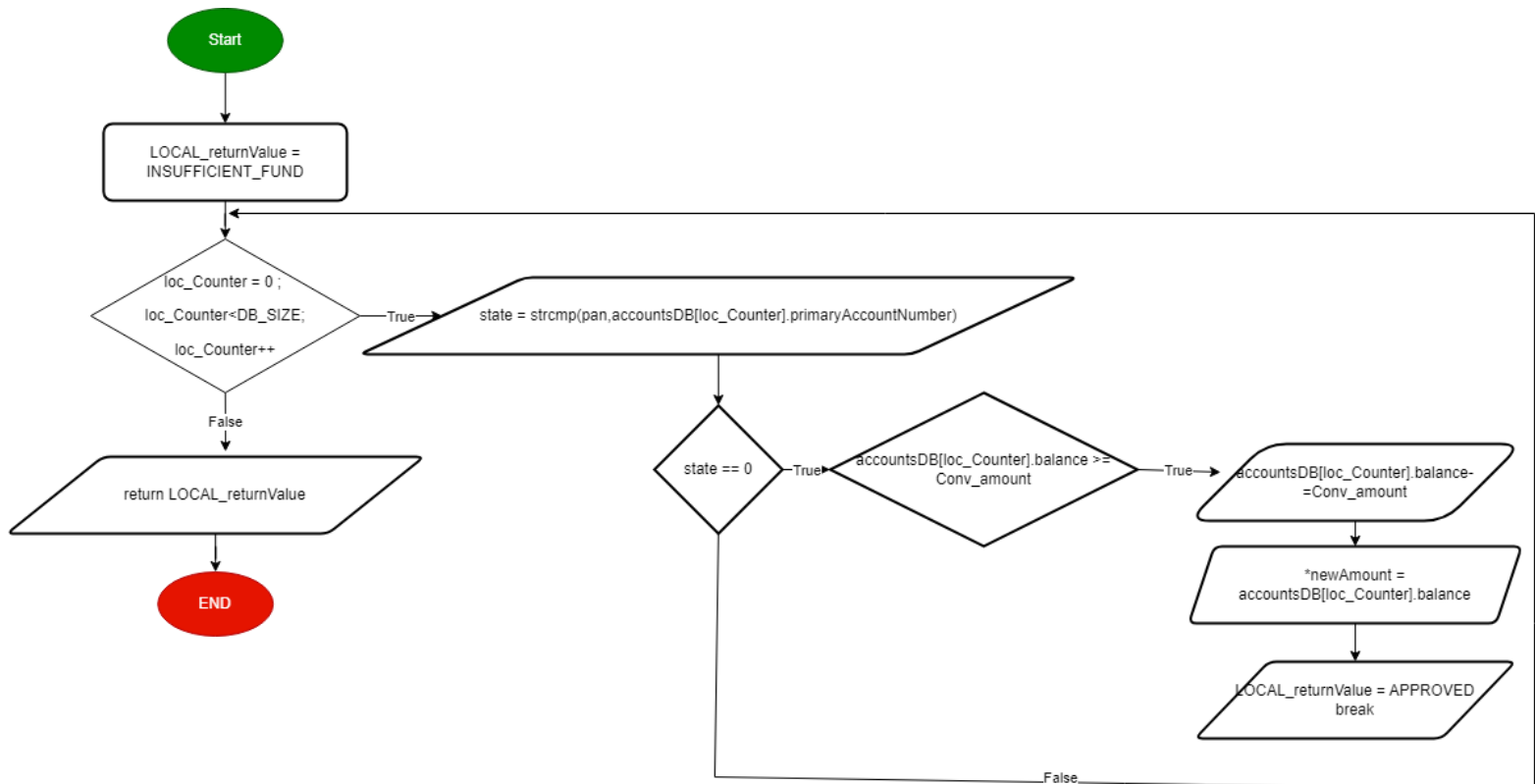
```
EN_dataError_t isValidPanAccount(Uchar8_t * pan);
```



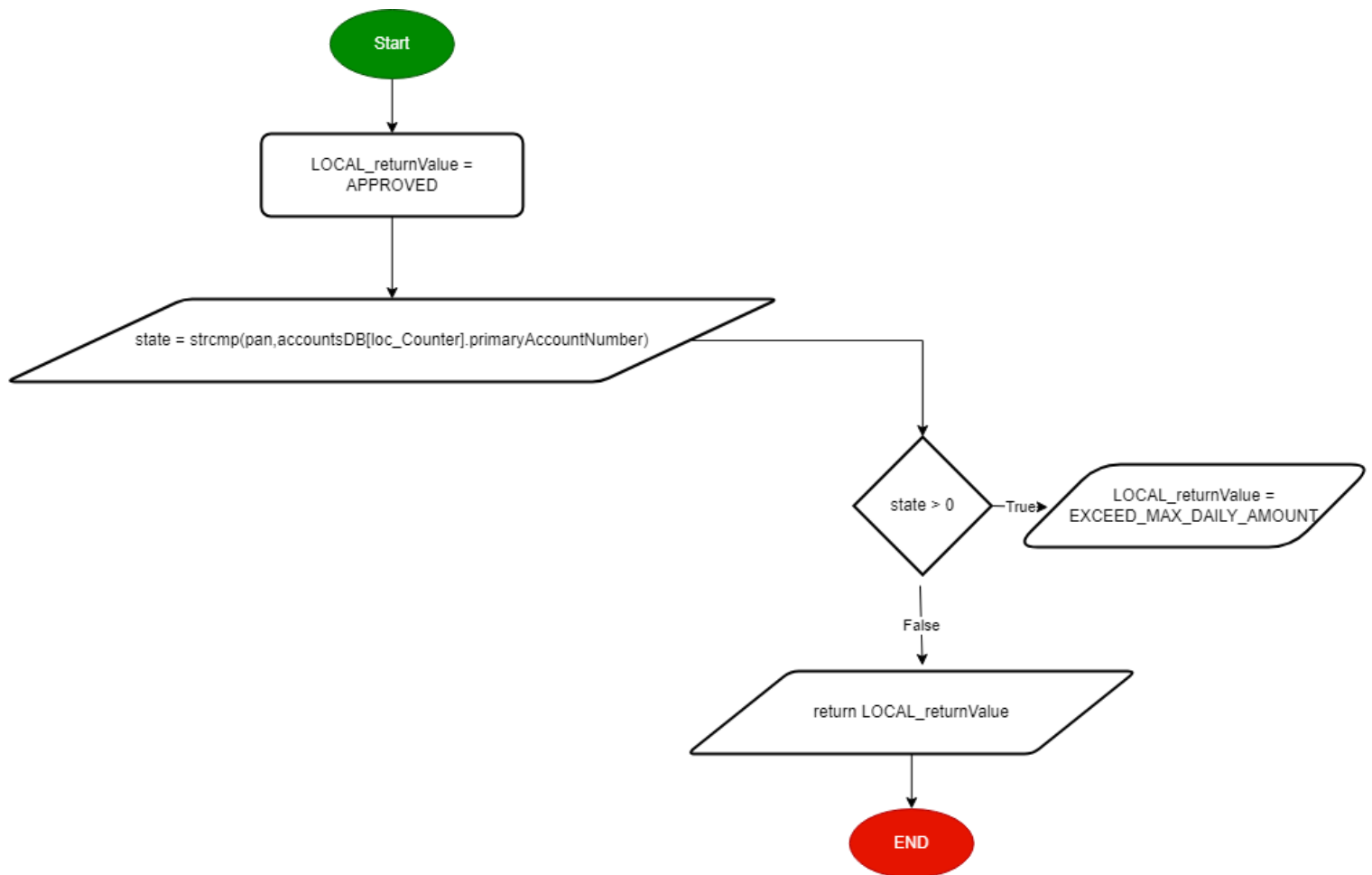
EN_dataError t isRunningAccount(Uchar8 t * pan);



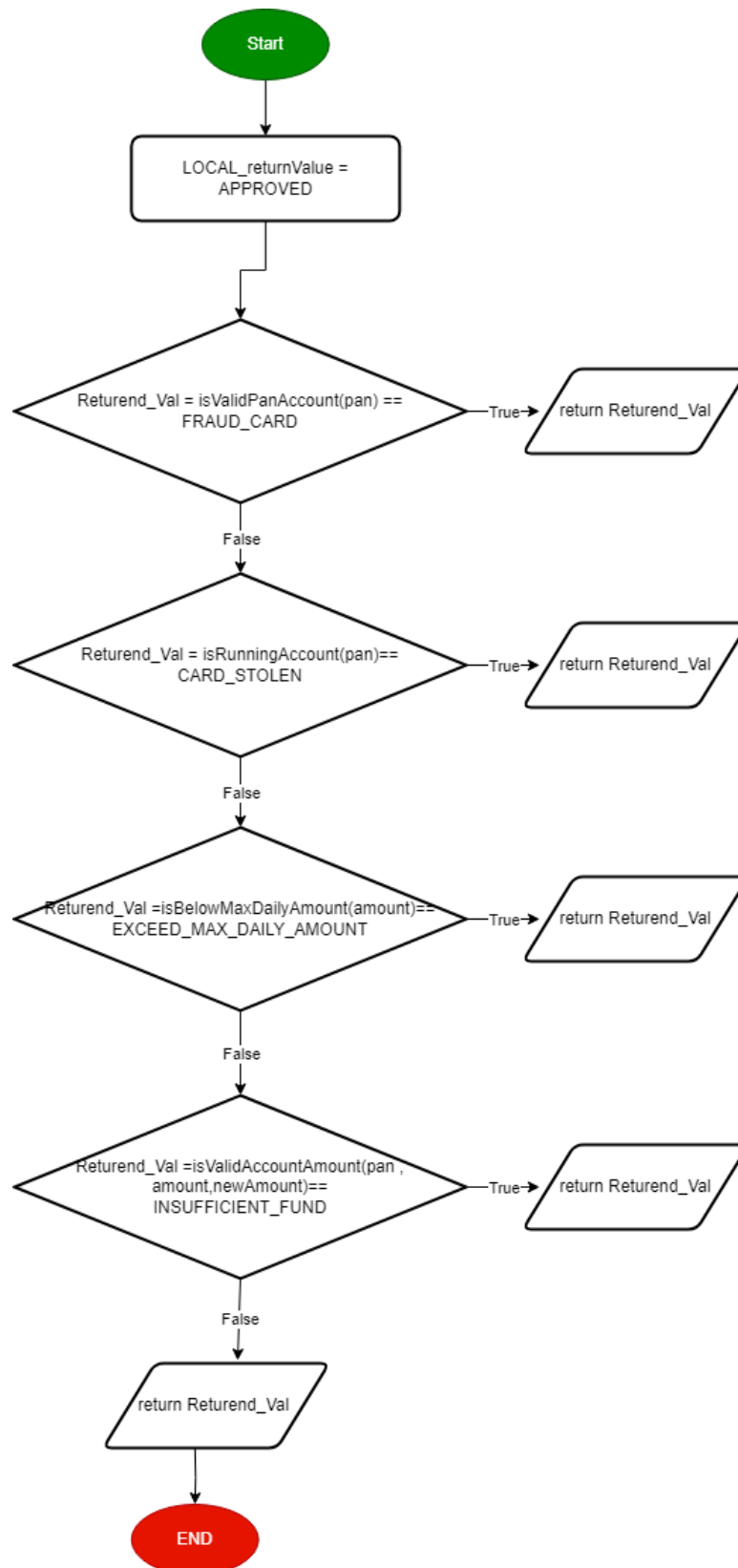
EN_dataError t isValidAccountAmount(Uchar8 t * pan, Uchar8 t * amount, float32 t *newAmount);



EN_dataError_t isBelowMaxDailyAmount(Uchar8_t * amount);

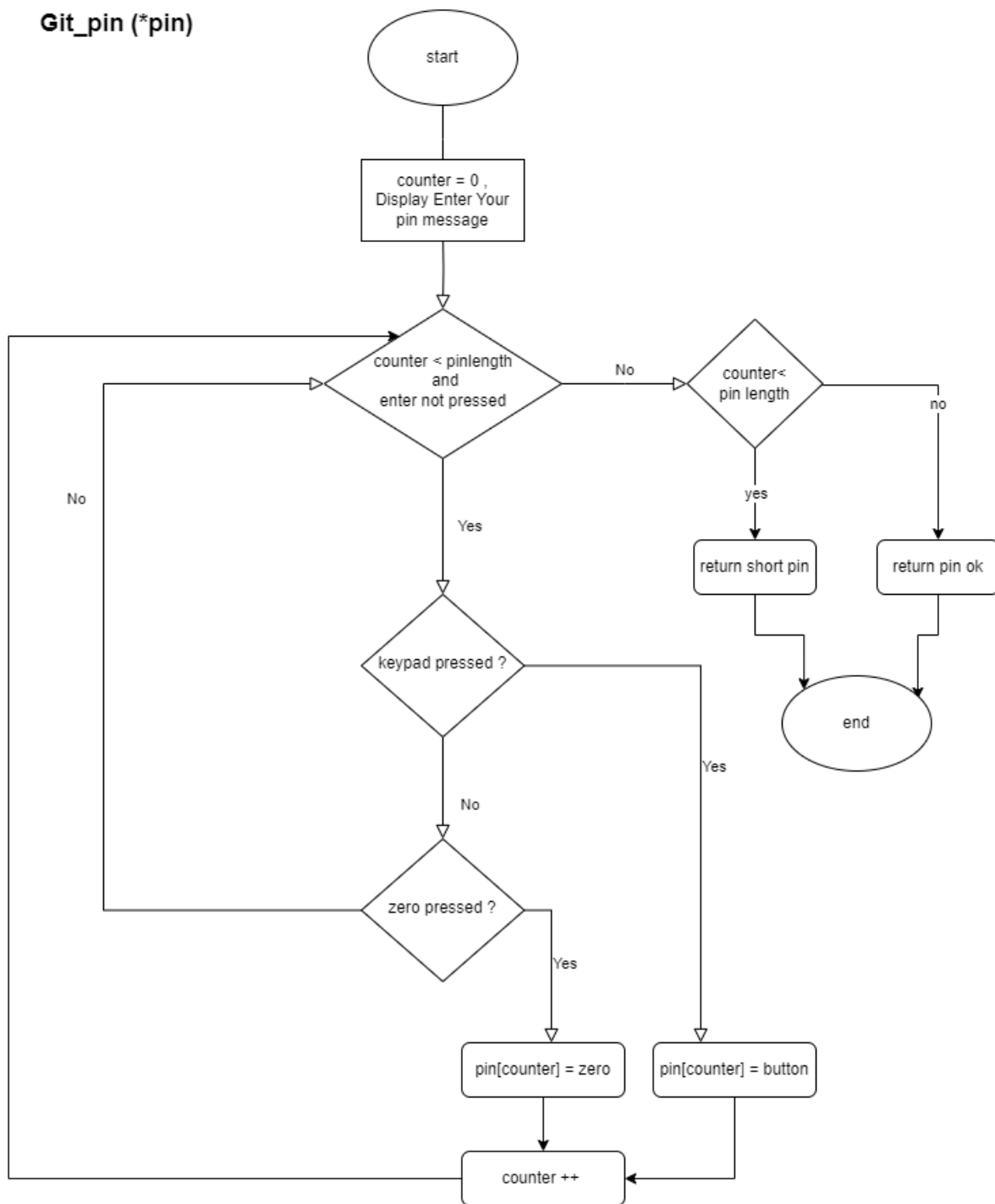


```
EN_dataError t DATABASE_checking (Uchar8_t * pan,Uchar8_t *  
amount,float32_t *newAmount);
```

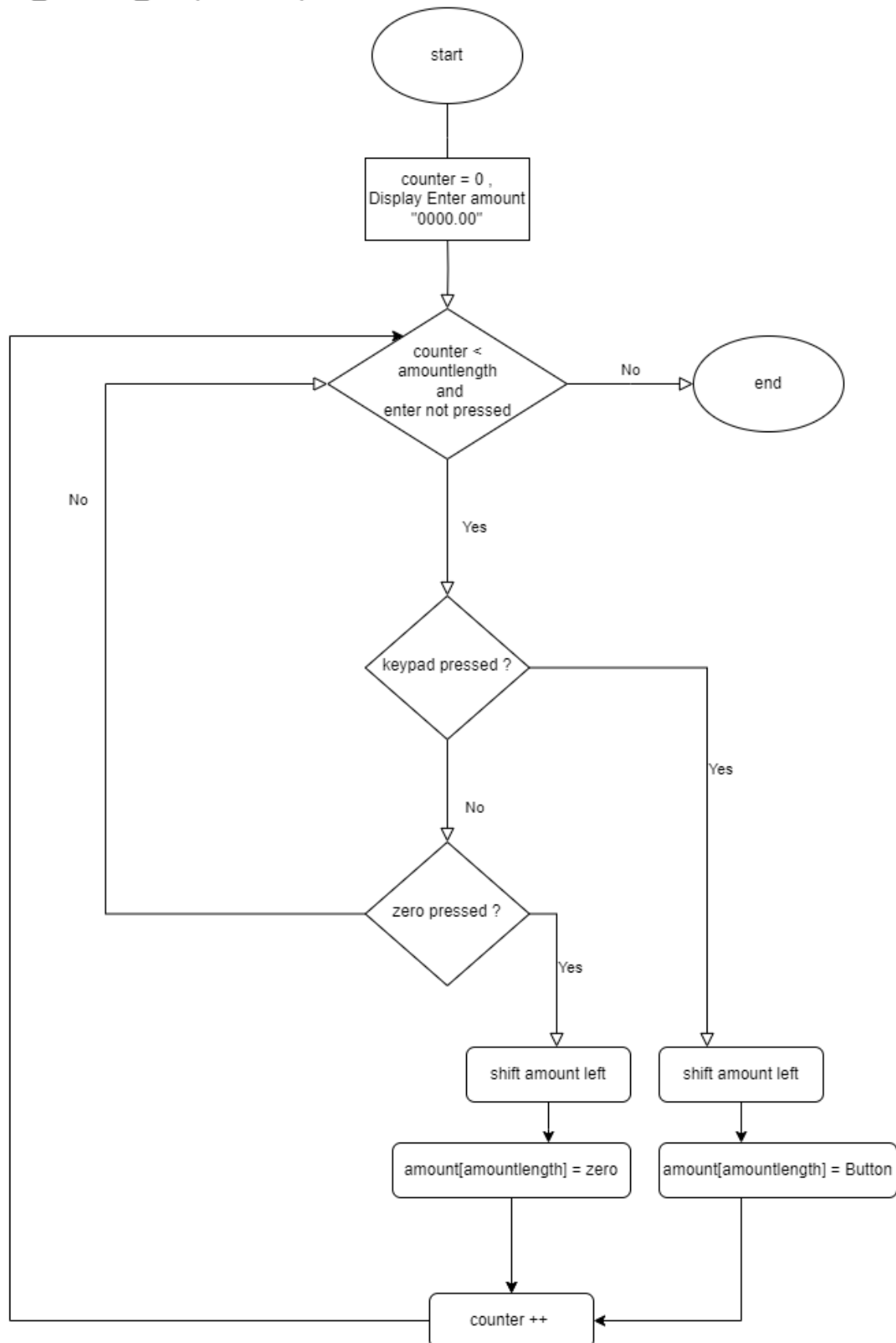


- ATM Module

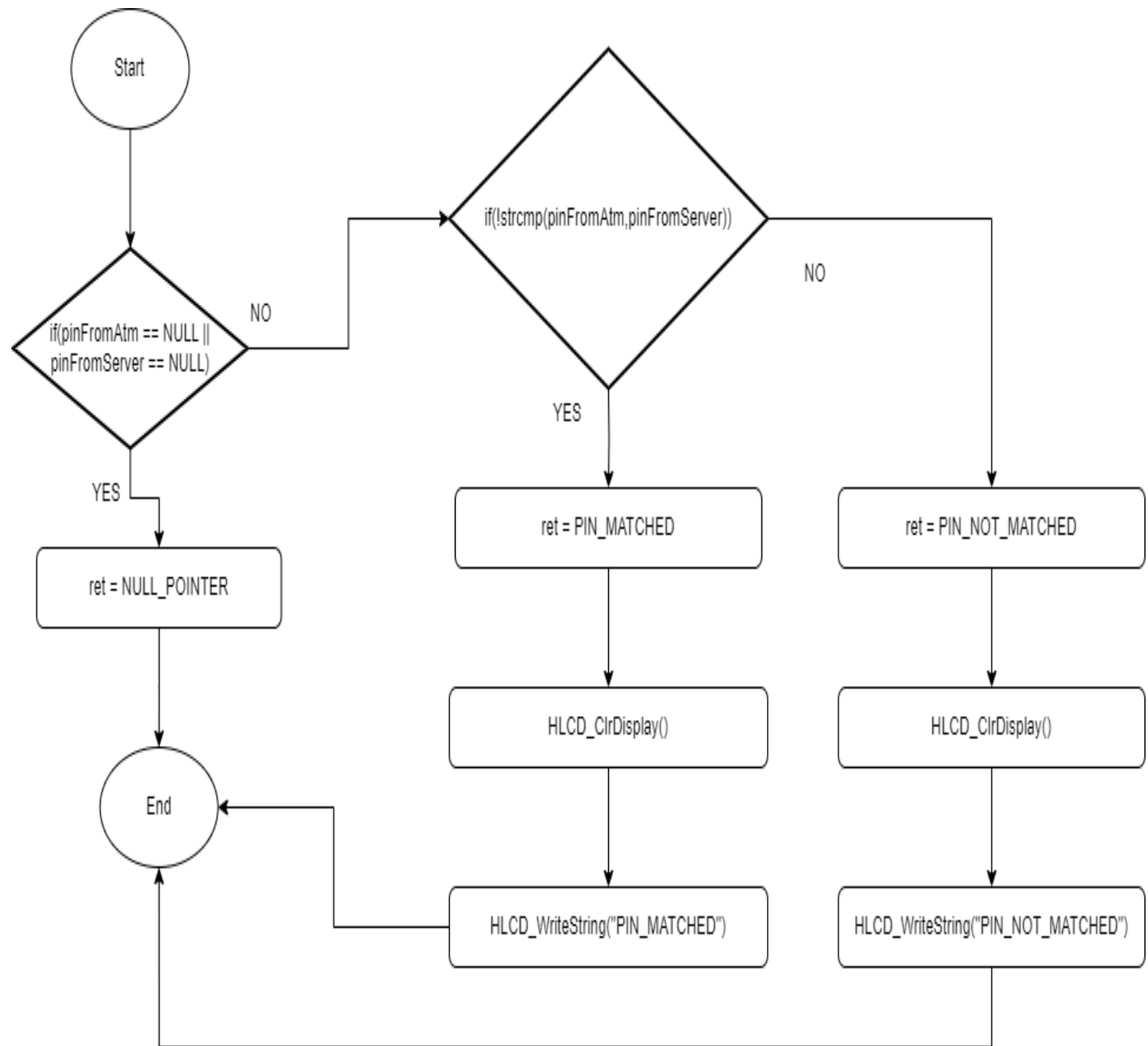
Git_pin (*pin)



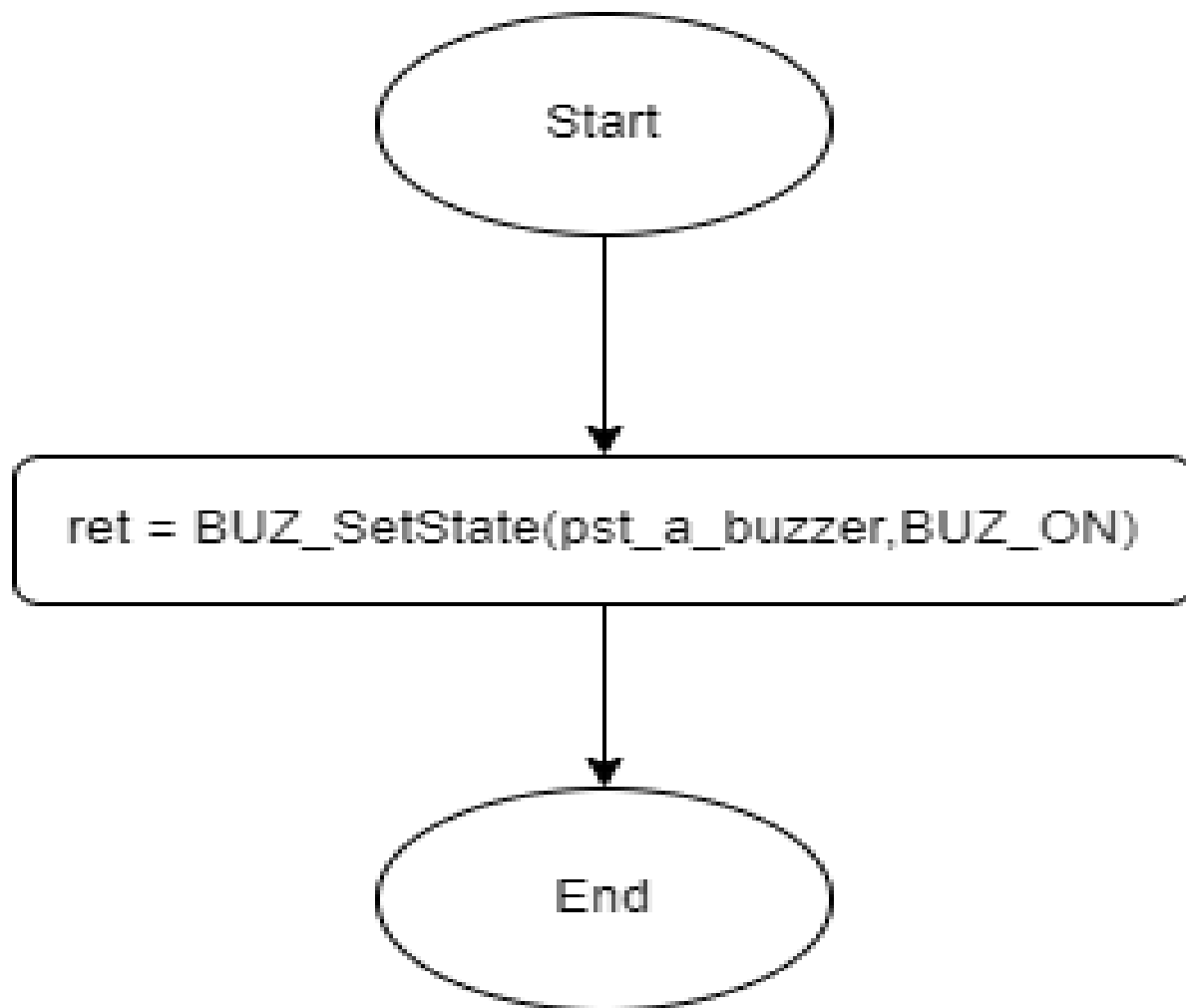
Git_amount_left (*amount)

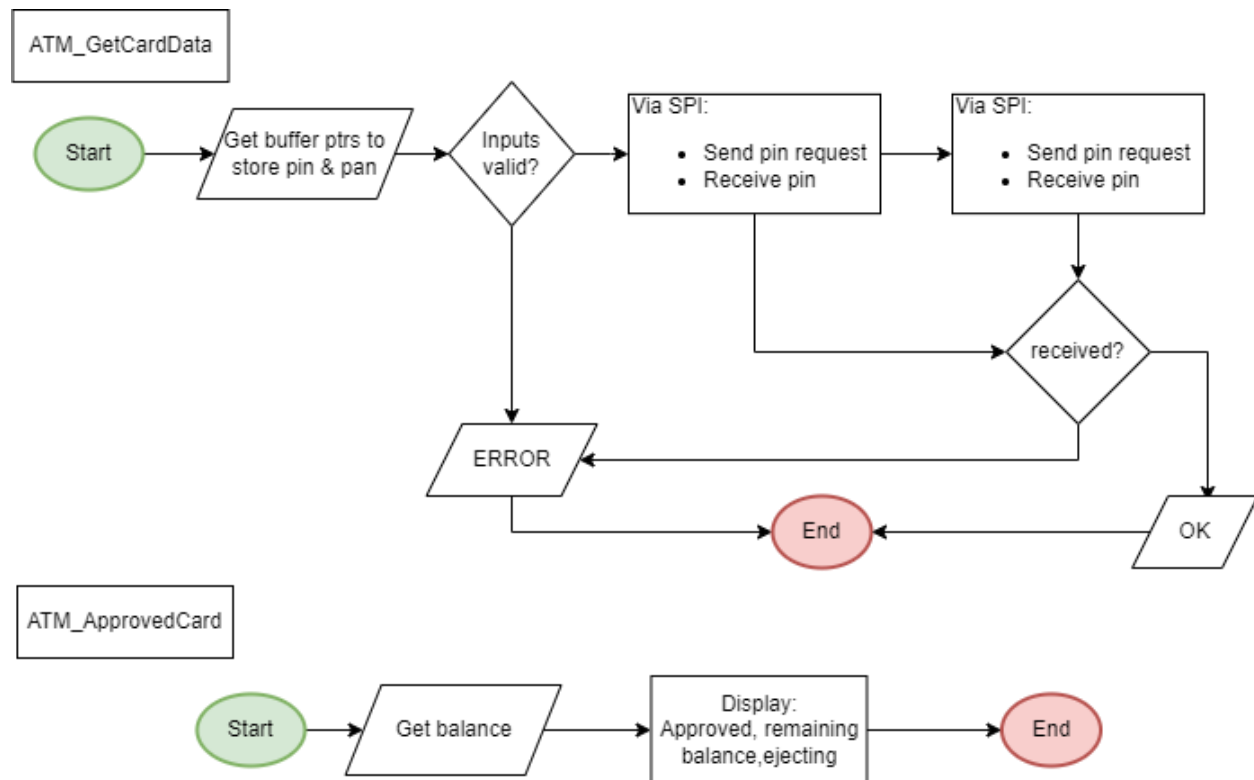


PIN_checkPinMatching



deinitAtm





Application Layer:

- **Card App**

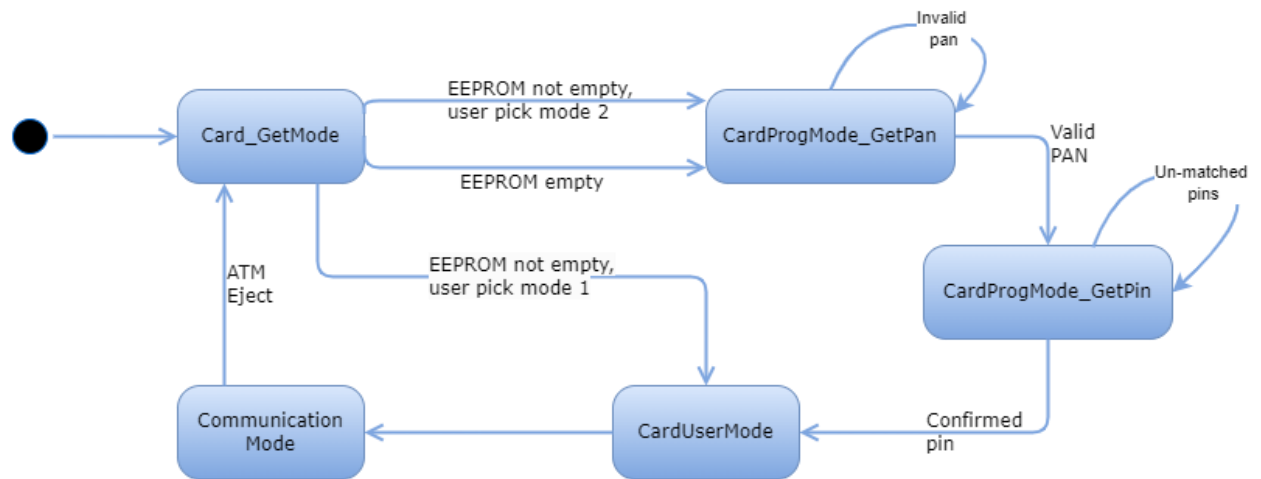


Figure 22 Card App. APIs Flow Charts

- **ATM App**

