

**Simple ATM Machine**

**Team Members:**

|  |
| --- |
| Alaa Ibrahim |
| Bassel Yasser |
| Sharpel Malek |
| Sherif Khadr |

Contents

[INTRODUCTION 1](#_Toc134385846)

[High Level Design 1](#_Toc134385847)

[**01)** **Layered Architecture** 1](#_Toc134385848)

[**02)** **Modules Description** 2](#_Toc134385849)

[**03)** **Drivers’ Documentation** 3](#_Toc134385850)

[**MCAL Layer** 3](#_Toc134385851)

[ **DIO** 3](#_Toc134385852)

[ **Timer 0** 4](#_Toc134385853)

[ **EXTINT:** 5](#_Toc134385854)

[ **I2C:** 6](#_Toc134385855)

[ **UART:** 9](#_Toc134385856)

[ **SPI:** 10](#_Toc134385857)

[**HAL Layer:** 11](#_Toc134385858)

[ **Keypad** 11](#_Toc134385859)

[ **HTimer:** 12](#_Toc134385860)

[ **HLCD** 13](#_Toc134385861)

[ **Buzzer** 14](#_Toc134385862)

[ **Button** 14](#_Toc134385863)

[ **EEPROM:** 15](#_Toc134385864)

[**Card\_Database Layer (CARD MCU)** 16](#_Toc134385865)

[**ATM\_MODULE Layer (ATM MCU)** 17](#_Toc134385866)

[ **ATM:** 17](#_Toc134385867)

[ **Database\_check:** 18](#_Toc134385868)

[**Application Layer:** 19](#_Toc134385869)

[Low Level Design 20](#_Toc134385870)

[**MCAL Layer:** 20](#_Toc134385871)

[ **DIO** 20](#_Toc134385872)

[ **Timer:** 23](#_Toc134385873)

[ **EXTINT:** 25](#_Toc134385874)

[ **UART:** 26](#_Toc134385875)

[ **SPI:** 27](#_Toc134385876)

[27](#_Toc134385877)

[ **I2C:** 28](#_Toc134385878)

[**Card\_Database Layer (CARD MCU)** 33](#_Toc134385879)

[ **Database\_check:** 35](#_Toc134385880)

[**HAL Layer** 40](#_Toc134385881)

[ **HTimer0** 40](#_Toc134385882)

[ **HSPI** 41](#_Toc134385883)

[ **LCD** 42](#_Toc134385884)

[ **Keypad** 49](#_Toc134385885)

[ **Buzzer** 52](#_Toc134385886)

[ **HEXTINT:** 52](#_Toc134385887)

[ **Button:** 53](#_Toc134385888)

[**Application Layer:** 54](#_Toc134385889)

[Figure 1: Project Layered Architecture 1](file:///D:\02_workspace\Sprints_ES_L2\Sprints_SimpleATM\Documents\SimpleATMmacine.docx#_Toc134385376)

[Figure 2 DIO\_s8SETPinDir Flow Chart 20](#_Toc134385377)

[Figure 3 DIO\_s8SETPinVal Flow chart 21](#_Toc134385378)

[Figure 4 DIO\_s8GETPinVal Flow Chart 22](#_Toc134385379)

[Figure 5 TIM0\_Init Flow Chart 23](#_Toc134385380)

[Figure 6 TIM0\_Start Flow Chart 23](#_Toc134385381)

[Figure 7 TIM0\_Stop Flow Chart 23](#_Toc134385382)

[Figure 8 TIM0 remaining Flow Charts 24](#_Toc134385383)

[Figure 15 HTIM0\_SyncDelay Flow Chart 40](#_Toc134385384)

[Figure 16 HTIM0\_AsyncDelay and EndDelay 40](#_Toc134385385)

[Figure 17 HLCD\_vidInit Flow Chart 42](file:///D:\02_workspace\Sprints_ES_L2\Sprints_SimpleATM\Documents\SimpleATMmacine.docx#_Toc134385386)

[Figure 18 HLCD\_vidWritecmd Flow Chart 43](file:///D:\02_workspace\Sprints_ES_L2\Sprints_SimpleATM\Documents\SimpleATMmacine.docx#_Toc134385387)

[Figure 19 HLCD\_vidWriteChar Flow Chart 43](file:///D:\02_workspace\Sprints_ES_L2\Sprints_SimpleATM\Documents\SimpleATMmacine.docx#_Toc134385388)

[Figure 20 HLCD\_ClrDisplay Flow Chart 44](file:///D:\02_workspace\Sprints_ES_L2\Sprints_SimpleATM\Documents\SimpleATMmacine.docx#_Toc134385389)

[Figure 21 HLCD\_gotoXY Flow Chart 45](#_Toc134385390)

[Figure 22 HLCD\_WriteString Flow Chart 46](#_Toc134385391)

[Figure 23 HLCD\_WriteInt Flow Chart 47](#_Toc134385392)

[Figure 24 HLCD\_vidCreatCustomChar Flow Chart 48](#_Toc134385393)

[Figure 25 KEYPAD\_Init Flow Chart 49](#_Toc134385394)

[Figure 26 KEYPAD\_CheckRx Flow Chart 50](#_Toc134385395)

[Figure 27 GetButton Flow Chart 51](#_Toc134385396)

[Figure 28 Buzzer Init & SetState Flow Charts 52](file:///D:\02_workspace\Sprints_ES_L2\Sprints_SimpleATM\Documents\SimpleATMmacine.docx#_Toc134385397)

[Figure 30 App. APIs Flow Charts 54](#_Toc134385398)

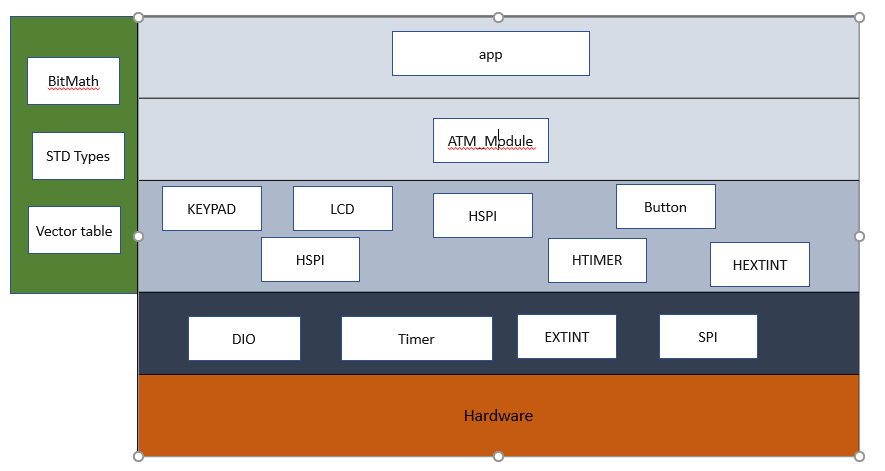
[Figure 31 APP\_Start & App States Flow Charts 55](#_Toc134385399)

[Figure 32 Cont. App States Flow Charts 56](#_Toc134385400)

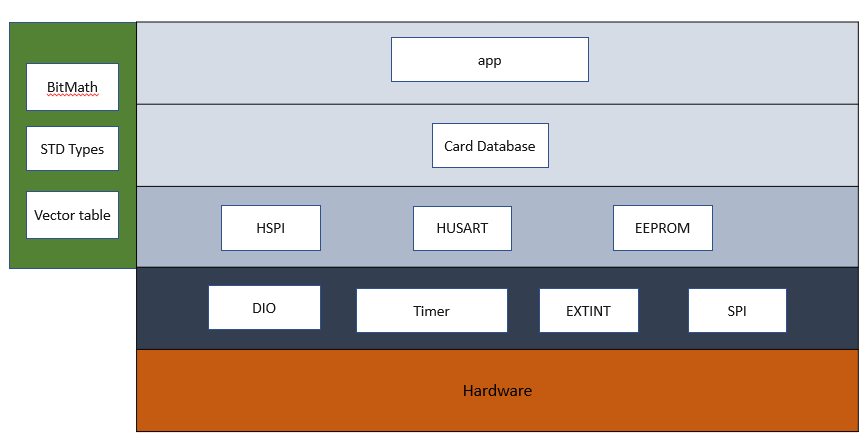
# INTRODUCTION

# High Level Design

## **Layered Architecture**



**ATM MACHINE:**

**CARD MCU**

## **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:**
* **I2C:**

**HAL Layer:**

* **Keypad:** Deal witha 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.
* **EEPROM:**
* **Button:**
* **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.

## **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\_stopvoid**);

#### **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:**

### **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 patterm 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)

#### **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\_EXINT\_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\_EXINT\_NUMBER INTx ,EN\_Sence\_Control INTxSense,void(\*ptrfunc)(void));

#### **EEPROM:**

### **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 : Sharpel

\* 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:**

#### **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:**

/\*\*

\* \brief Initialize all modules and execute welcome routine

\*

\* \param

\*

\* \return void

\*/

void APP\_Init(void);

/\*\*

\* \brief Application main logic

\*

\* \param

\*

\* \return void

\*/

void APP\_Start(void);

/\*\*

\* \brief Initialize temperature adjustment process

\*

\* \param

\*

\* \return void

\*/

static void APP\_adjustInit(void);

/\*\*

\* \brief timeout callback function

\*

\* \return void

\*/

void timeout(void);

# Low Level Design

### **MCAL Layer:**

#### **DIO**

Sint8\_t **DIO\_s8SETPinDir** (enu\_pin enPinCopy, enu\_dir enPortDir)

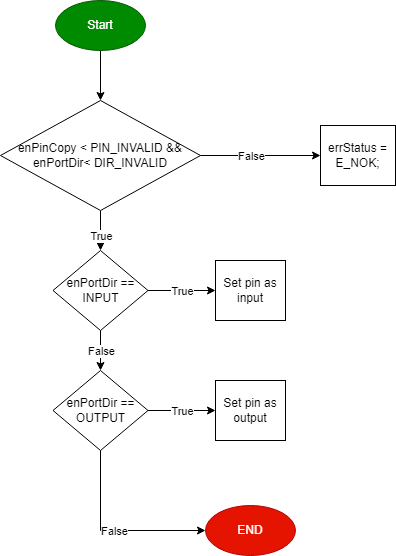
****

Figure 2 DIO\_s8SETPinDir Flow Chart

Sint8\_t **DIO\_s8SETPinVal** (enu\_pin enPinCopy, enu\_val enPortVal)

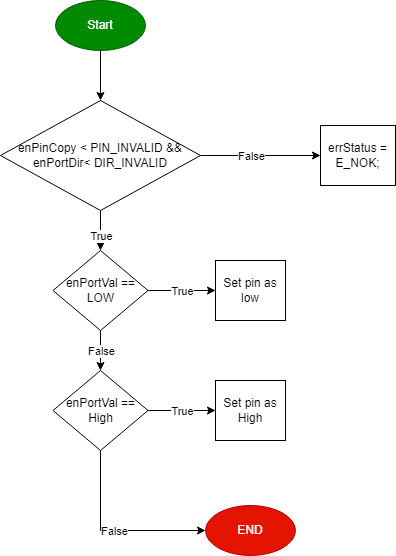
****

Figure 3 DIO\_s8SETPinVal Flow chart

Sint8\_t **DIO\_s8GETPinVal** (enu\_pin enPinCopy, Uint8\_t\* pu8Val)

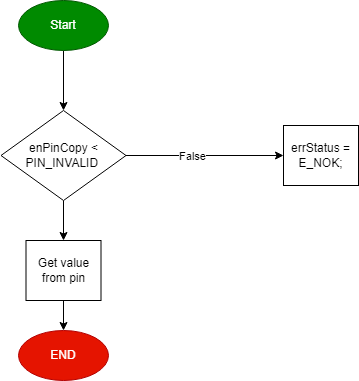
****

Figure 4 DIO\_s8GETPinVal Flow Chart

#### **Timer:**

**TIM0\_Init**

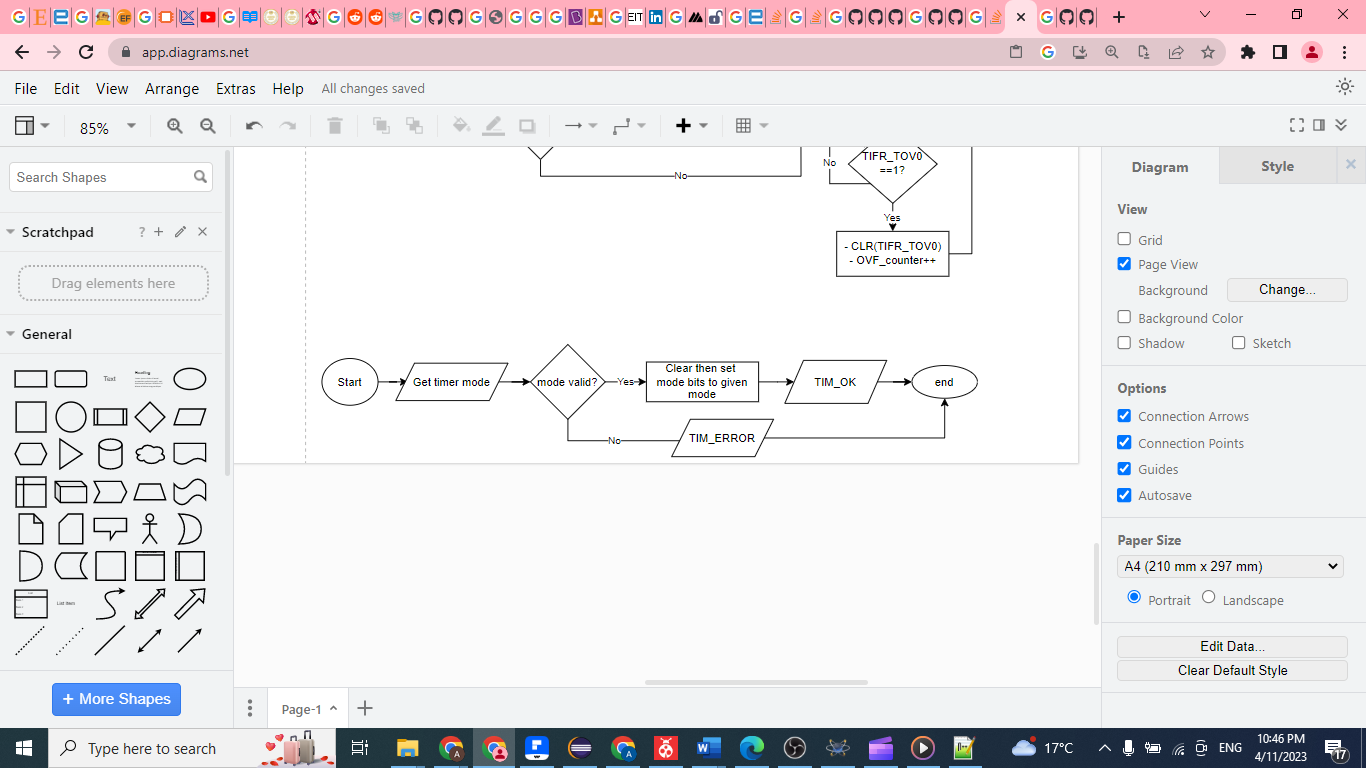


Figure 5 TIM0\_Init Flow Chart

**TIM0\_Start**

****

Figure 6 TIM0\_Start Flow Chart

**TIM0\_Stop**

****

Figure 7 TIM0\_Stop Flow Chart

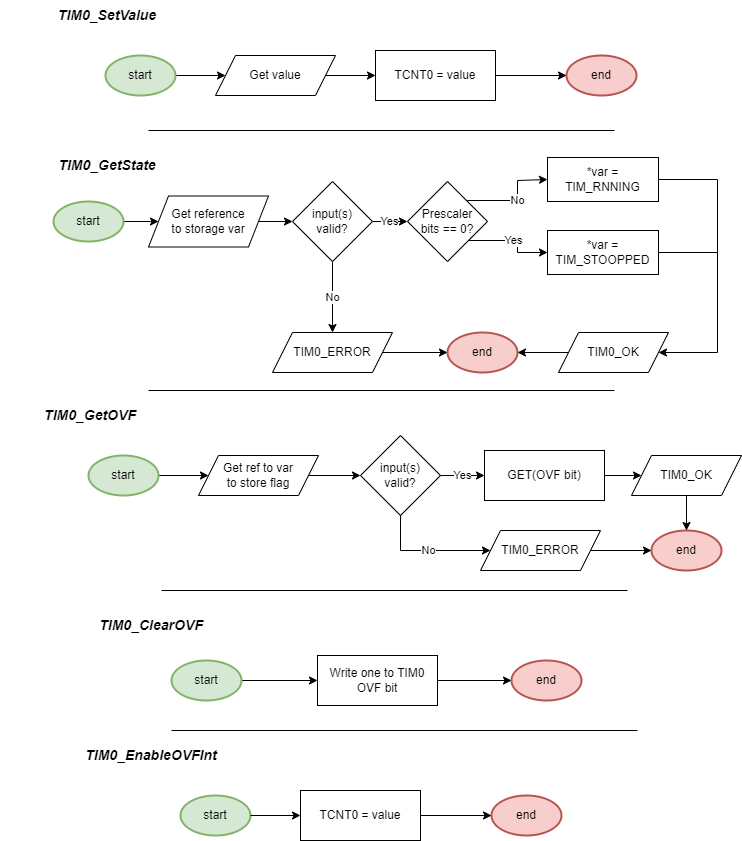
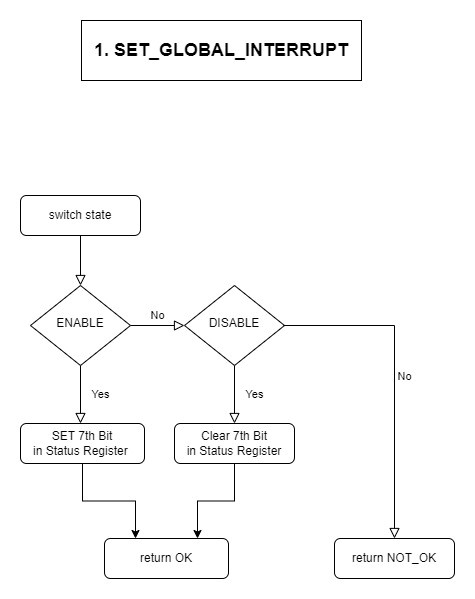
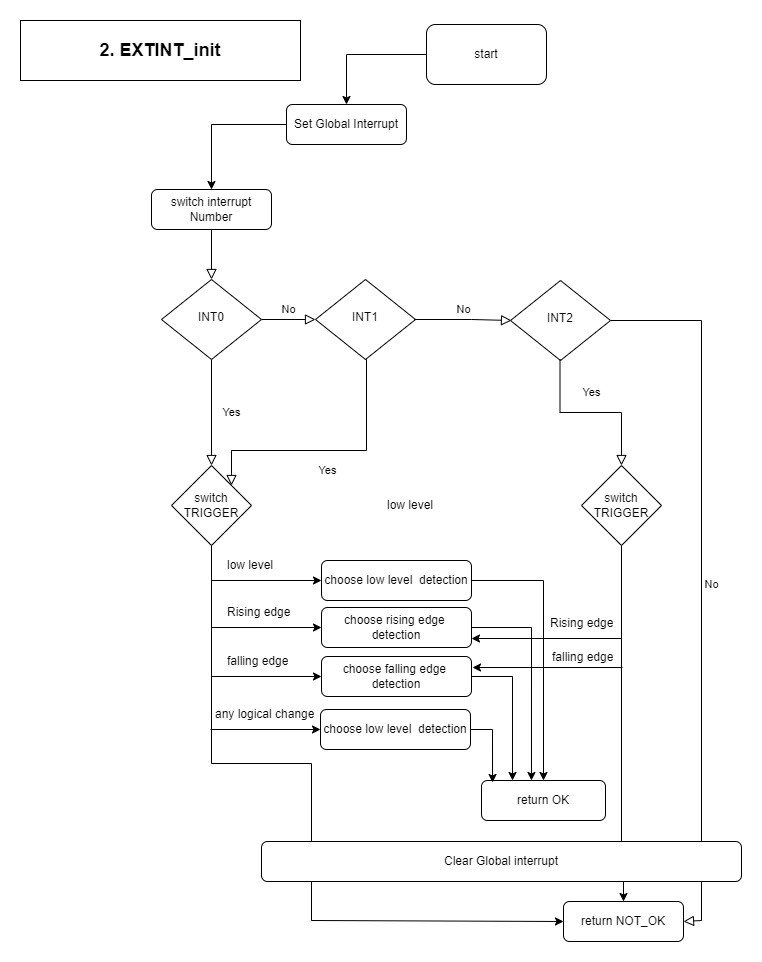
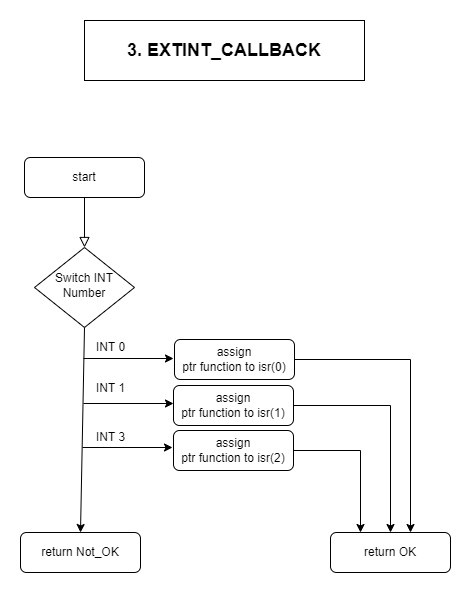


Figure 8 TIM0 remaining Flow Charts

#### **EXTINT:**



****

****

#### **UART:**

#### **C:\Users\Sharpel\Documents\atm_charts\usart_init.drawio.png**

#### **C:\Users\Sharpel\Documents\atm_charts\USART_RECDATA.drawio.png**

#### **C:\Users\Sharpel\Documents\atm_charts\USART_SENData.drawio.png**

#### **C:\Users\Sharpel\Documents\atm_charts\USART_SENDSTRING.drawio.png**

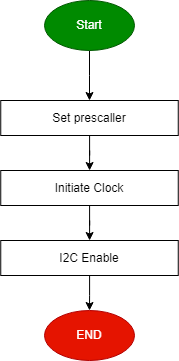
#### **C:\Users\Sharpel\Documents\atm_charts\USART_recSTRING.drawio (1).png**

#### **SPI:**

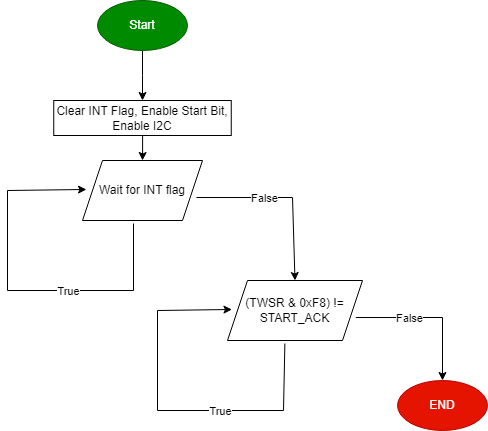
#### 

#### **I2C:**

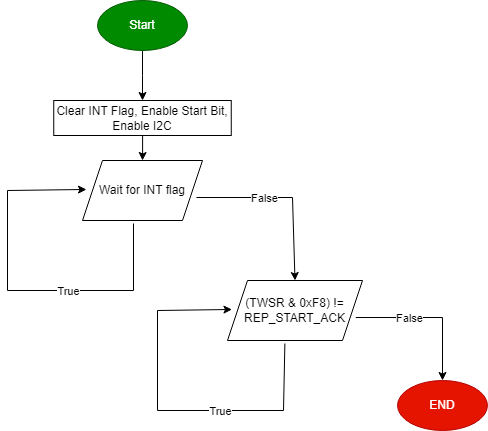
**void** **i2c\_init\_master**(**void**);



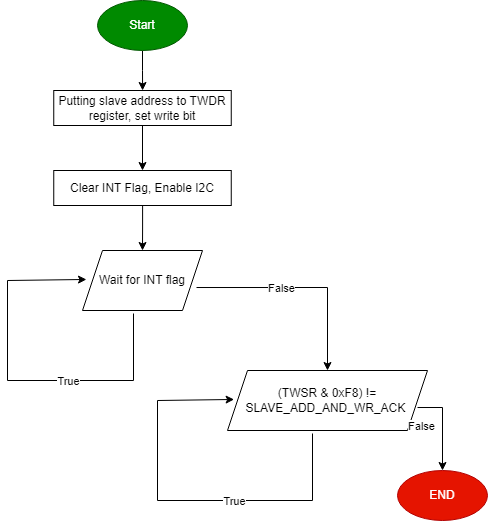
**void** **i2c\_start**(**void**);



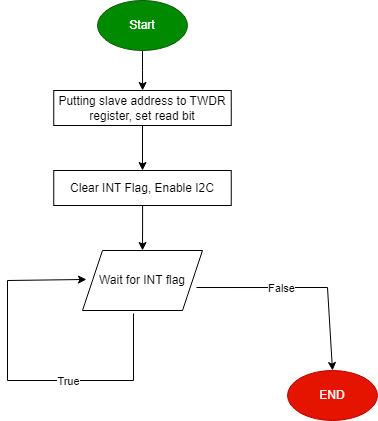
**void** **i2c\_repeated\_start**(**void**);



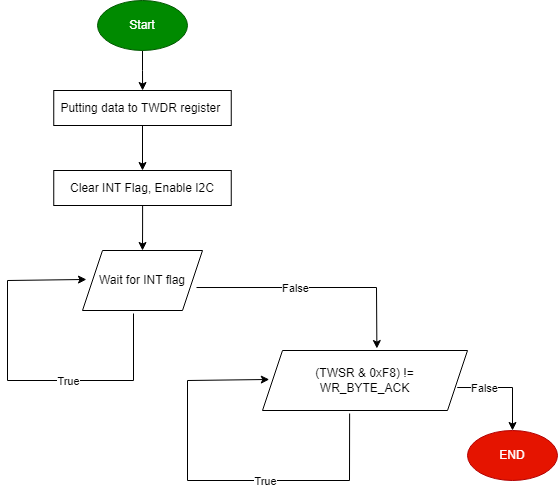
**void** **i2c\_send\_slave\_address\_with\_write\_req**(Uint8\_t slave\_address);



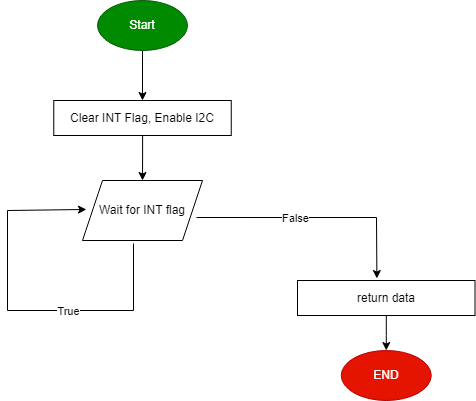
**void** **i2c\_send\_slave\_address\_with\_read\_req**(Uint8\_t slave\_address);



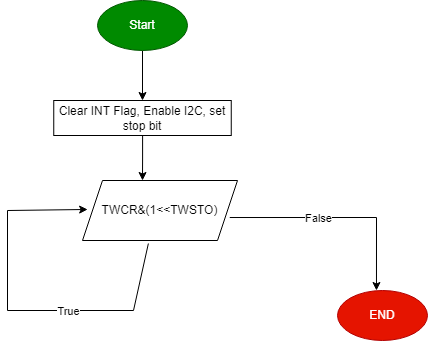
**void** **i2c\_write\_byte**(Uint8\_t byte);



Uint8\_t **i2c\_read\_byte**(**void**);



**void** **i2c\_stop**(**void**);

****

### **HAL Layer**

#### **HTimer0**

**HTIM0\_SyncDelay**

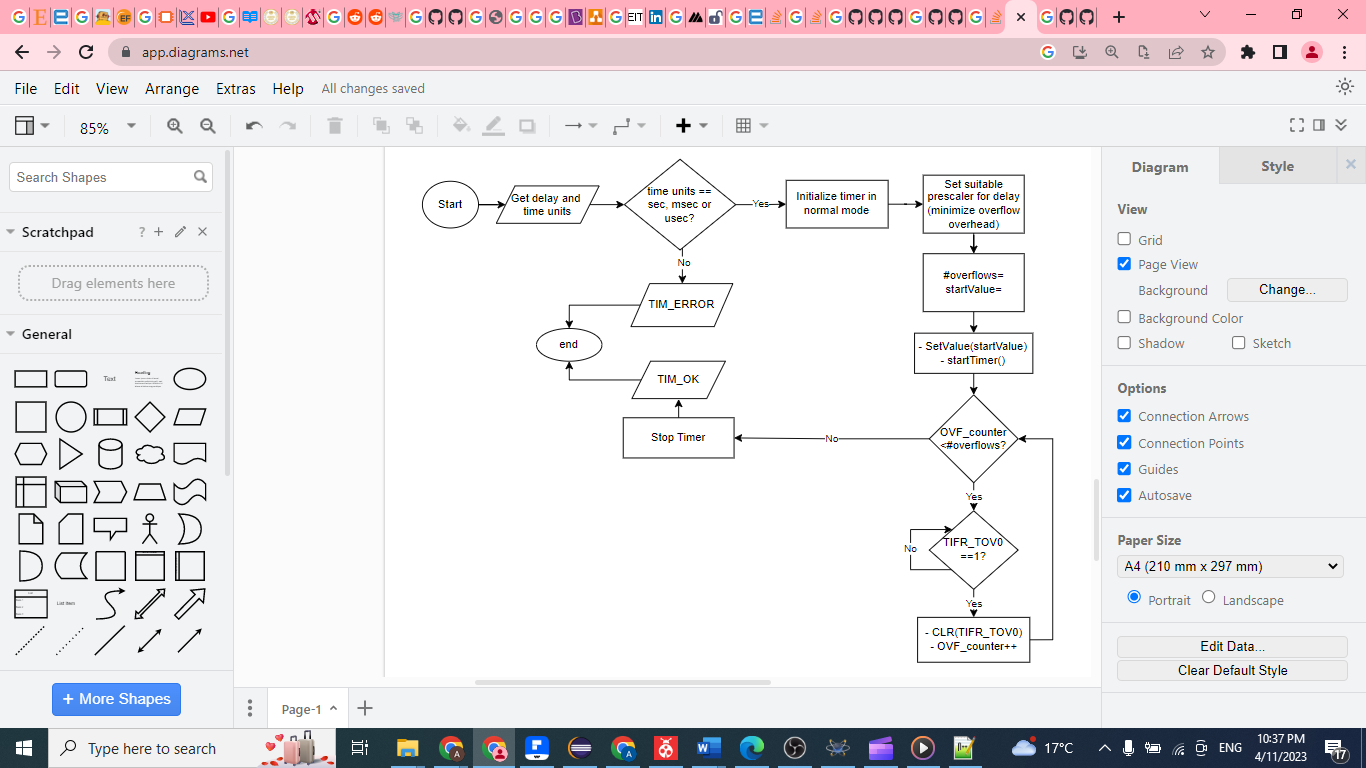


Figure 15 HTIM0\_SyncDelay Flow Chart

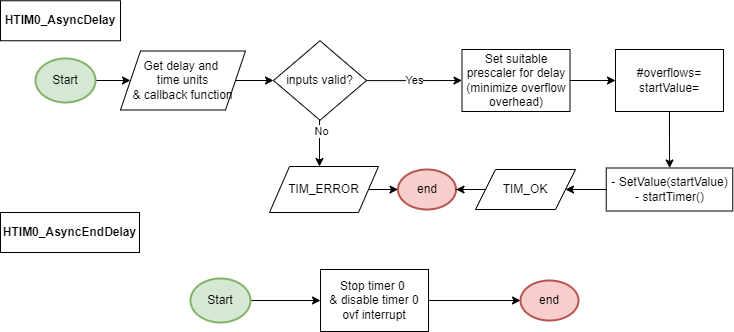
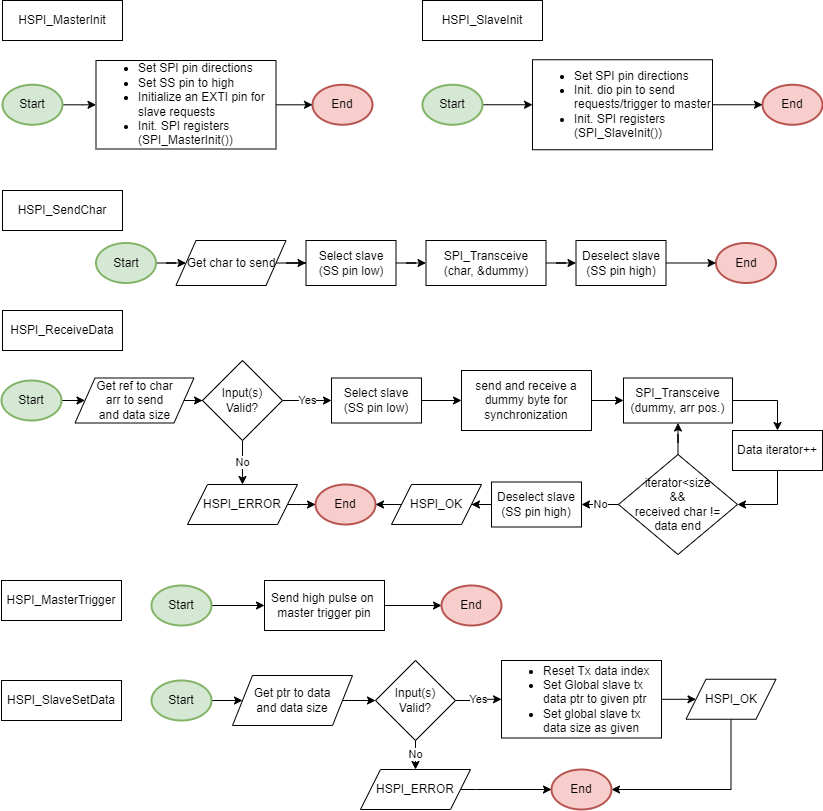


Figure 16 HTIM0\_AsyncDelay and EndDelay

#### **HSPI**



#### **LCD**

**void** **HLCD\_vidInit**(**void**)

Figure 17 HLCD\_vidInit Flow Chart

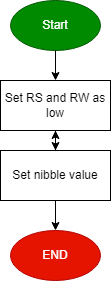
**void** **HLCD\_vidWritecmd**(Uint8\_t u8commandCopy)

Figure 18 HLCD\_vidWritecmd Flow Chart

**void** **HLCD\_vidWriteChar**(Uint8\_t u8CharCopy)

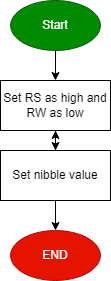
****

Figure 19 HLCD\_vidWriteChar Flow Chart

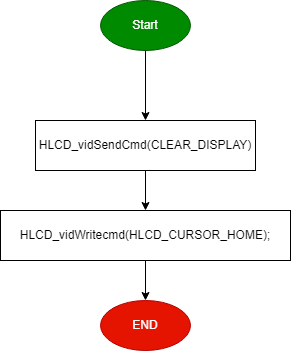
**void** **HLCD\_ClrDisplay**(**void**)

Figure 20 HLCD\_ClrDisplay Flow Chart

**void** **HLCD\_gotoXY**(Uint8\_t row, Uint8\_t pos)

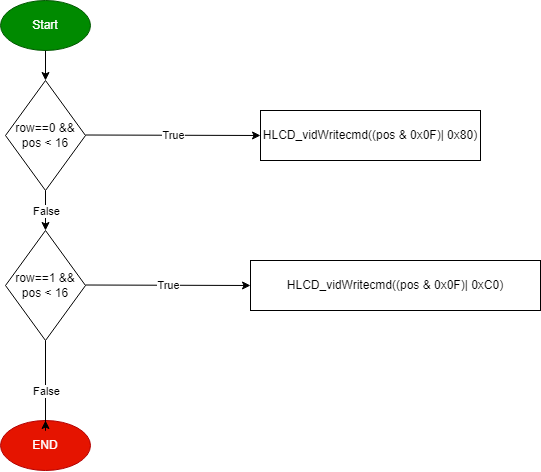
****

Figure 21 HLCD\_gotoXY Flow Chart

**void** **HLCD\_WriteString**(Uint8\_t\* str)

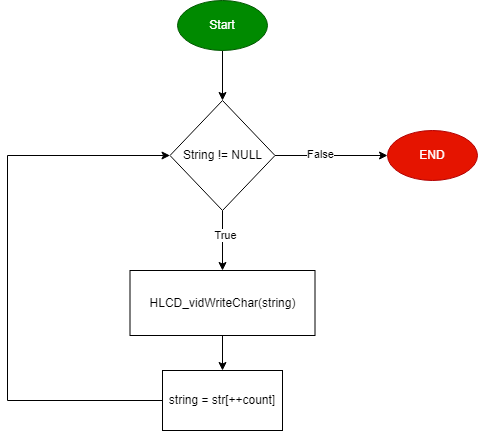
****

Figure 22 HLCD\_WriteString Flow Chart

**void** **HLCD\_WriteInt**(Uint32\_t number)

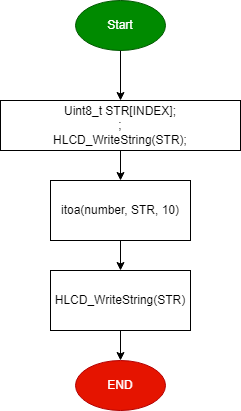
****

Figure 23 HLCD\_WriteInt Flow Chart

**void** **HLCD\_vidCreatCustomChar**(Uint8\_t\* pu8custom, Uint8\_t u8Location)

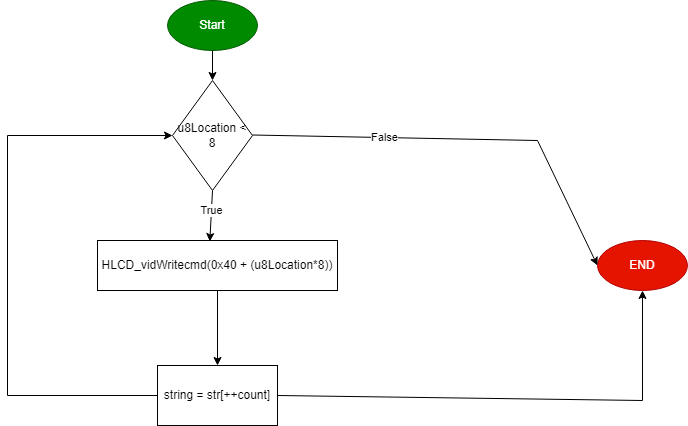
****

Figure 24 HLCD\_vidCreatCustomChar Flow Chart

#### **Keypad**

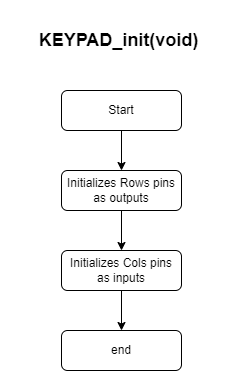


Figure 25 KEYPAD\_Init Flow Chart

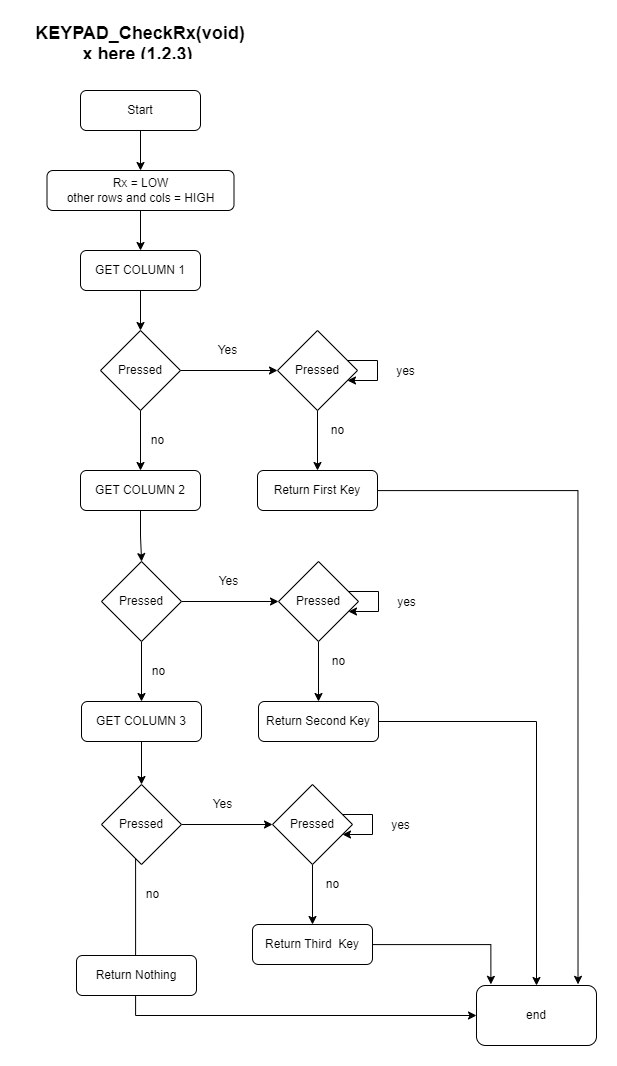


Figure 26 KEYPAD\_CheckRx Flow Chart

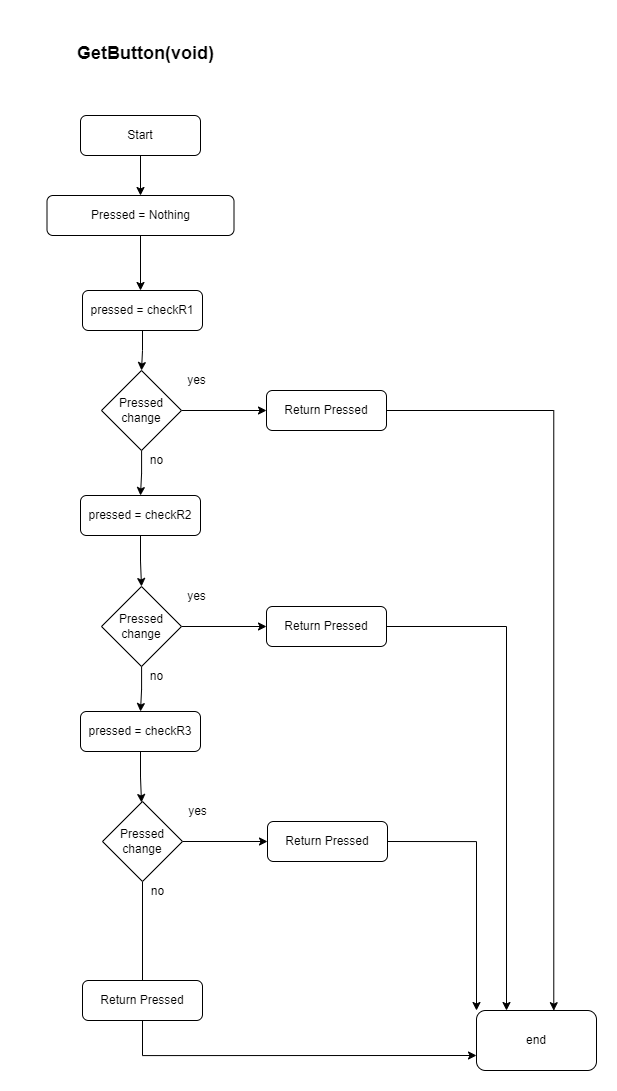
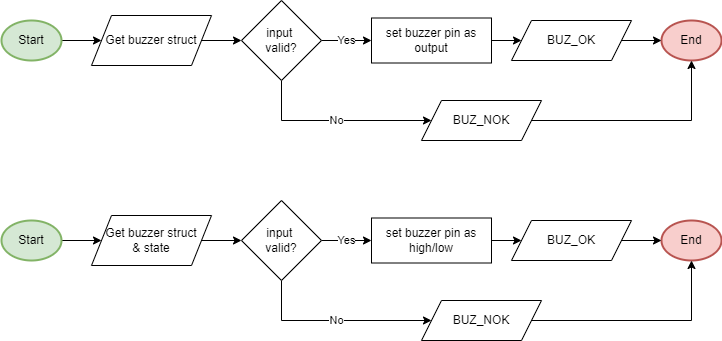


Figure 27 GetButton Flow Chart

#### **Buzzer**

****

**BUZ\_SetState**

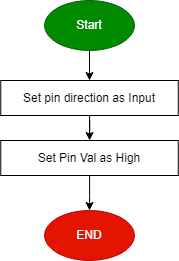
Figure 28 Buzzer Init & SetState Flow Charts

**BUZ\_Init**

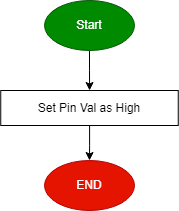
#### **HEXTINT:**

#### **Button:**

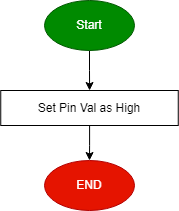
enu\_buttonError\_t **HButton\_Init**(enu\_pin en\_pinx);



enu\_buttonError\_t **HButton\_ExtIntInit**(enu\_pin en\_pinx);

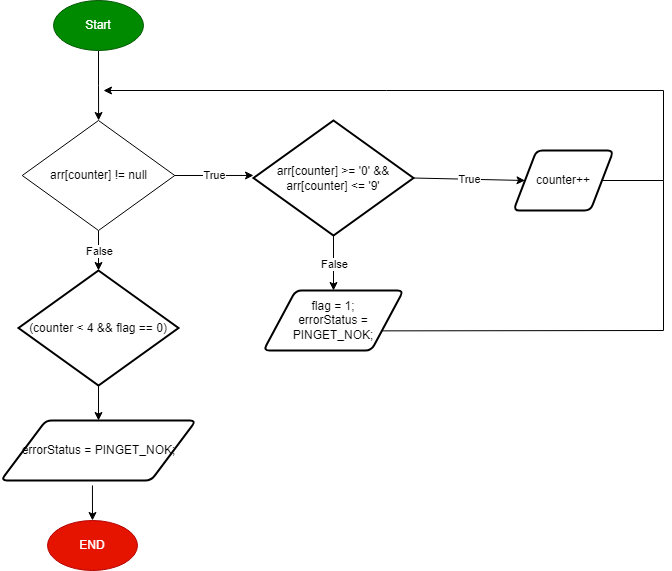


enu\_buttonError\_t **HButton\_getPinVal**(enu\_pin en\_pinx, Uint8\_t\* pu8\_refVal );

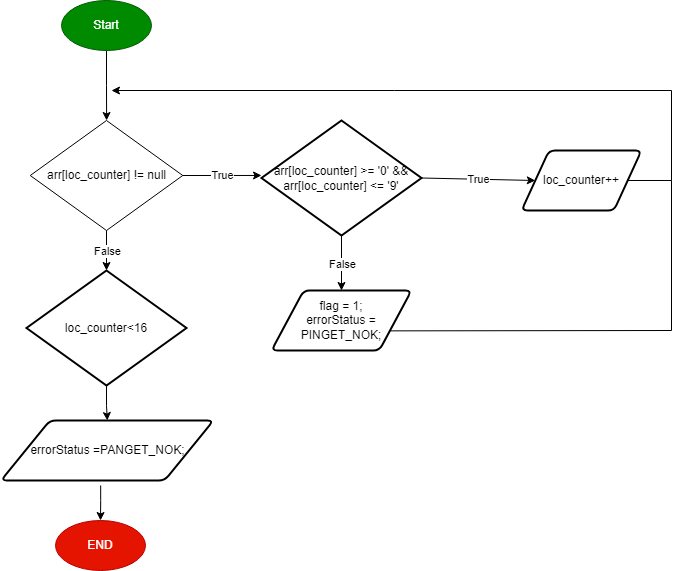
****

### **Card\_Database Layer (CARD MCU)**

en\_terminalPinGetStatus\_t **APP\_terminalPinGet**(Uchar8\_t\* arr);

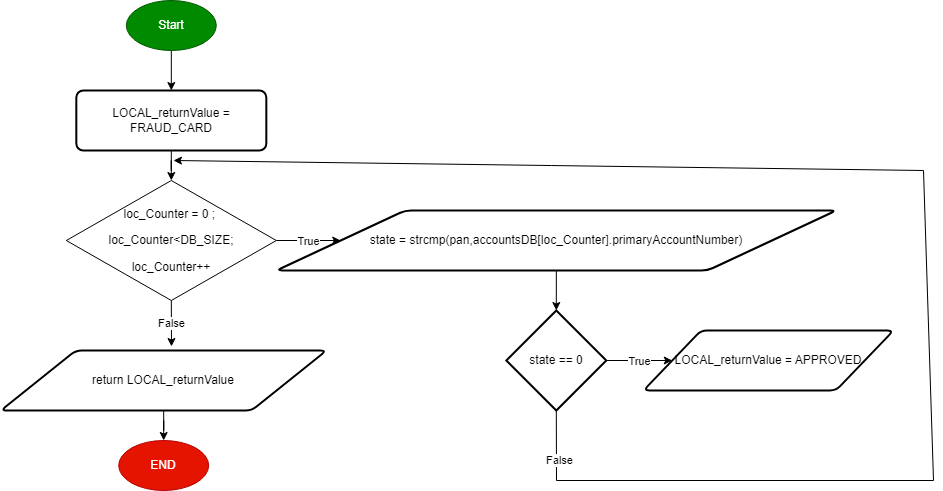


en\_terminalPanGetStatus\_t **APP\_terminalPanGet**(Uchar8\_t\* arr);

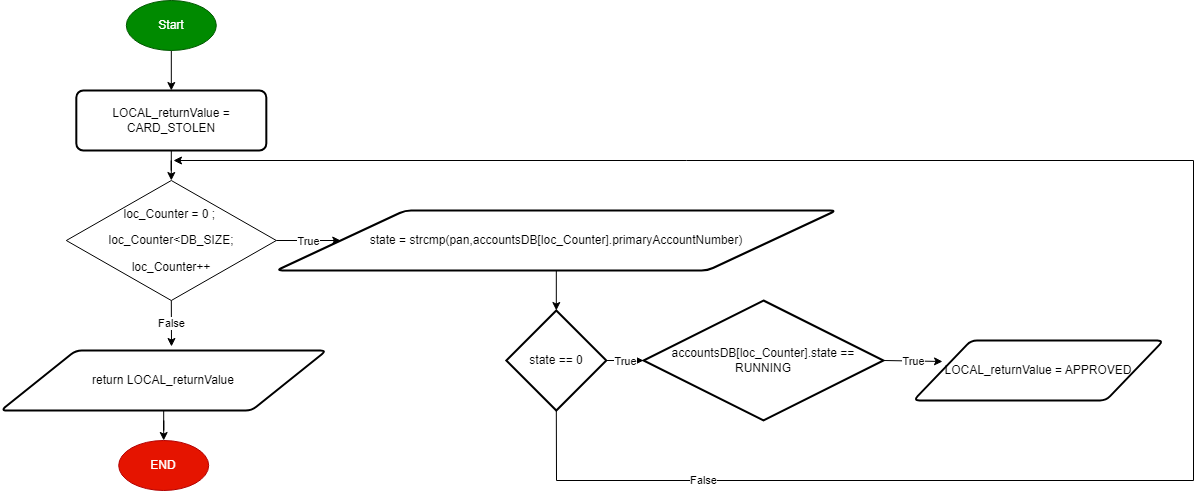
****

#### **Database\_check:**

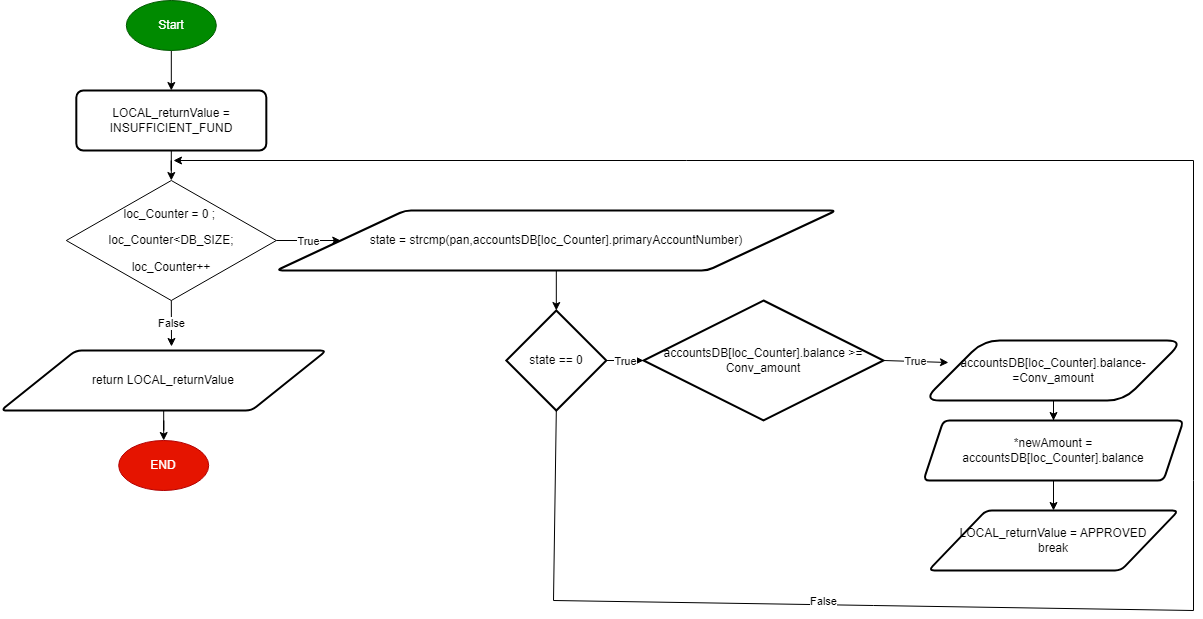
EN\_dataError\_t **isValidPanAccount**(Uchar8\_t \* pan);



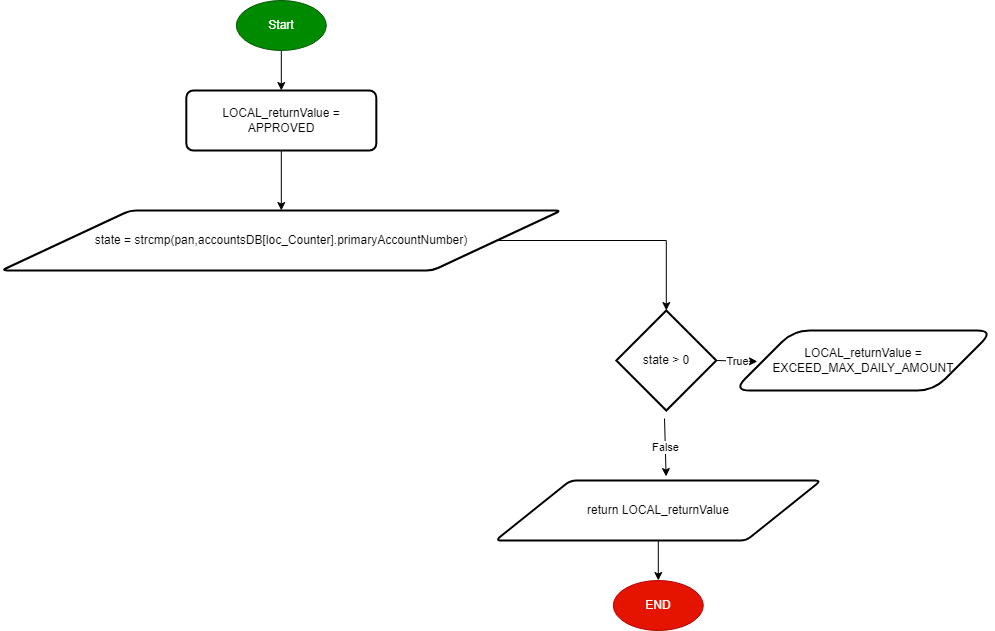
EN\_dataError\_t **isRunningAccount**(Uchar8\_t \* pan);

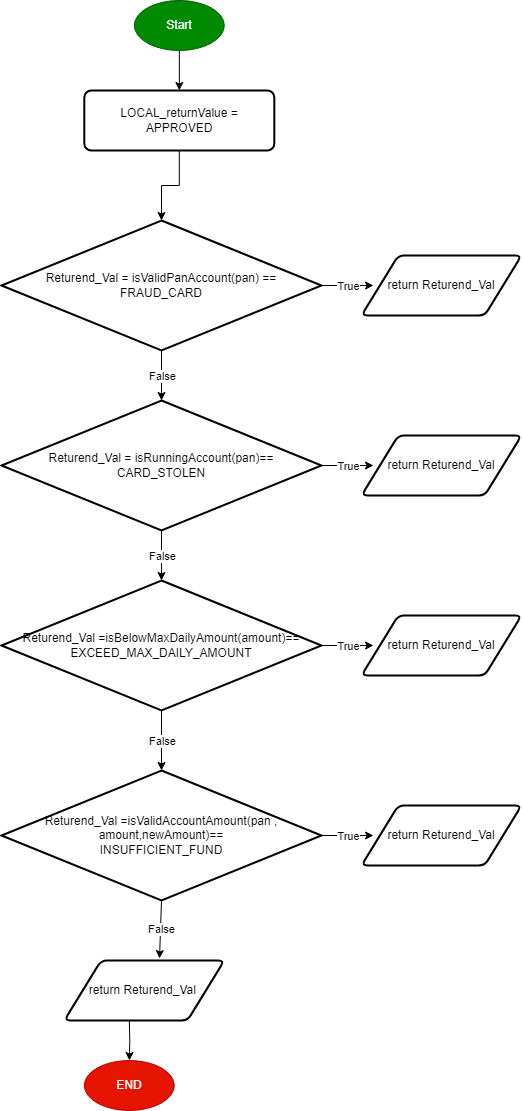


EN\_dataError\_t **isValidAccountAmount**(Uchar8\_t \* pan,Uchar8\_t \* amount,float32\_t \*newAmount);

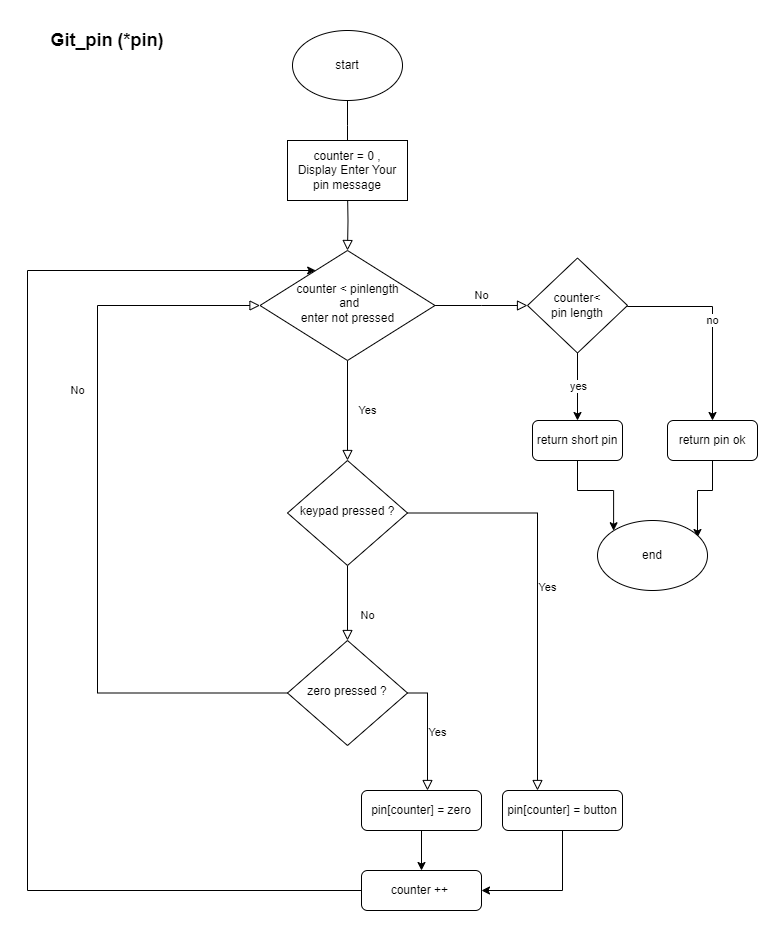


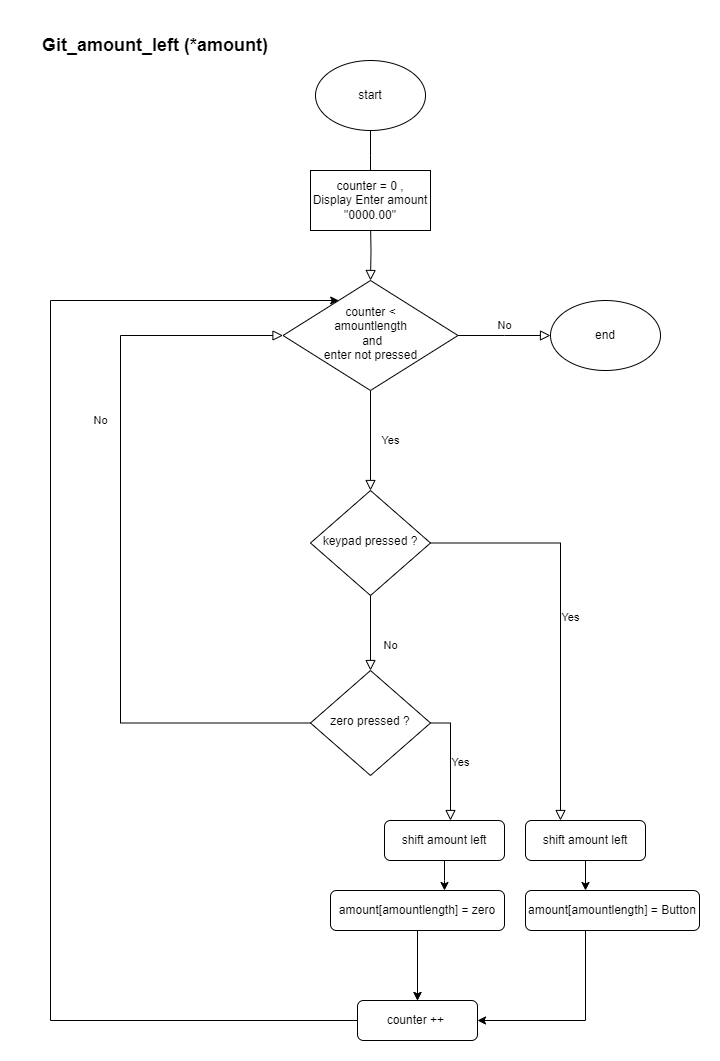
EN\_dataError\_t **isBelowMaxDailyAmount**(Uchar8\_t \* amount);



EN\_dataError\_t **DATABASE\_checking** (Uchar8\_t \* pan,Uchar8\_t \* amount,float32\_t \*newAmount);

#### **ATM Module**





### **Application Layer:**

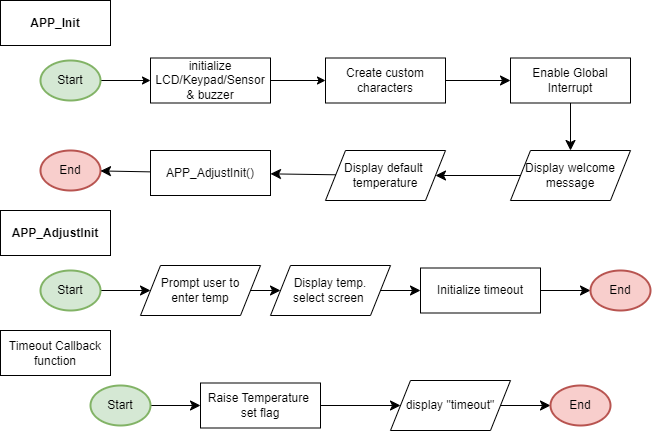


Figure 30 App. APIs Flow Charts



Figure 31 APP\_Start & App States Flow Charts



Figure 32 Cont. App States Flow Charts

