

**Simple ATM Machine**

**Team Members:**

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

Contents

[INTRODUCTION 1](#_Toc132883074)

[High Level Design 1](#_Toc132883075)

[**01)** **Layered Architecture** 1](#_Toc132883076)

[**02)** **Modules Description** 2](#_Toc132883077)

[**03)** **Drivers’ Documentation** 3](#_Toc132883078)

[**MCAL Layer** 3](#_Toc132883079)

[ **DIO** 3](#_Toc132883080)

[ **Timer 0** 4](#_Toc132883081)

[ **ADC** 5](#_Toc132883082)

[**HAL Layer:** 7](#_Toc132883083)

[ **Keypad** 7](#_Toc132883084)

[ **HTimer:** 8](#_Toc132883085)

[ **HLCD** 9](#_Toc132883086)

[ **Buzzer** 10](#_Toc132883087)

[ **Temperature Sensor** 10](#_Toc132883088)

[**Application Layer:** 11](#_Toc132883089)

[Low Level Design 12](#_Toc132883090)

[**MCAL Layer:** 12](#_Toc132883091)

[ **DIO** 12](#_Toc132883092)

[ **Timer:** 15](#_Toc132883093)

[ **ADC** 17](#_Toc132883094)

[**HAL Layer** 21](#_Toc132883095)

[ **HTimer0** 21](#_Toc132883096)

[ **LCD** 22](#_Toc132883097)

[ **Keypad** 29](#_Toc132883098)

[ **Buzzer** 32](#_Toc132883099)

[ **Temperature Sensor** 32](#_Toc132883100)

[**Application Layer:** 33](#_Toc132883101)

[Figure 1: Project Layered Architecture 1](file:///E:\Automotive_Bootcamp\Sprints_AirConditioningSystem\Documents\AirConditioning.docx#_Toc132883134)

[Figure 2 DIO\_s8SETPinDir Flow Chart 12](#_Toc132883135)

[Figure 3 DIO\_s8SETPinVal Flow chart 13](#_Toc132883136)

[Figure 4 DIO\_s8GETPinVal Flow Chart 14](#_Toc132883137)

[Figure 5 TIM0\_Init Flow Chart 15](#_Toc132883138)

[Figure 6 TIM0\_Start Flow Chart 15](#_Toc132883139)

[Figure 7 TIM0\_Stop Flow Chart 15](#_Toc132883140)

[Figure 8 TIM0 remaining Flow Charts 16](#_Toc132883141)

[Figure 9 ADC\_Init Flow Chart 17](#_Toc132883142)

[Figure 10 ADC\_SetChannel Flow Chart 18](file:///E:\Automotive_Bootcamp\Sprints_AirConditioningSystem\Documents\AirConditioning.docx#_Toc132883143)

[Figure 11 ADC\_Deinit Flow Chart 18](#_Toc132883144)

[Figure 12 ADC\_StartConversion Flow Chart 18](#_Toc132883145)

[Figure 13 ADC\_GetConversionResult Flow Chart 19](#_Toc132883146)

[Figure 14 ADC\_Conversion Flow Chart 20](#_Toc132883147)

[Figure 15 HTIM0\_SyncDelay Flow Chart 21](#_Toc132883148)

[Figure 16 HTIM0\_AsyncDelay and EndDelay 21](#_Toc132883149)

[Figure 17 HLCD\_vidInit Flow Chart 22](file:///E:\Automotive_Bootcamp\Sprints_AirConditioningSystem\Documents\AirConditioning.docx#_Toc132883150)

[Figure 18 HLCD\_vidWritecmd Flow Chart 23](file:///E:\Automotive_Bootcamp\Sprints_AirConditioningSystem\Documents\AirConditioning.docx#_Toc132883151)

[Figure 19 HLCD\_vidWriteChar Flow Chart 23](file:///E:\Automotive_Bootcamp\Sprints_AirConditioningSystem\Documents\AirConditioning.docx#_Toc132883152)

[Figure 20 HLCD\_ClrDisplay Flow Chart 24](file:///E:\Automotive_Bootcamp\Sprints_AirConditioningSystem\Documents\AirConditioning.docx#_Toc132883153)

[Figure 21 HLCD\_gotoXY Flow Chart 25](#_Toc132883154)

[Figure 22 HLCD\_WriteString Flow Chart 26](#_Toc132883155)

[Figure 23 HLCD\_WriteInt Flow Chart 27](#_Toc132883156)

[Figure 24 HLCD\_vidCreatCustomChar Flow Chart 28](#_Toc132883157)

[Figure 25 KEYPAD\_Init Flow Chart 29](#_Toc132883158)

[Figure 26 KEYPAD\_CheckRx Flow Chart 30](#_Toc132883159)

[Figure 27 GetButton Flow Chart 31](#_Toc132883160)

[Figure 28 Buzzer Init & SetState Flow Charts 32](file:///E:\Automotive_Bootcamp\Sprints_AirConditioningSystem\Documents\AirConditioning.docx#_Toc132883161)

[Figure 29 Temp Sensor Init & ReadValue Flow Charts 32](file:///E:\Automotive_Bootcamp\Sprints_AirConditioningSystem\Documents\AirConditioning.docx#_Toc132883162)

[Figure 30 App. APIs Flow Charts 33](#_Toc132883163)

[Figure 31 APP\_Start & App States Flow Charts 34](#_Toc132883164)

[Figure 32 Cont. App States Flow Charts 35](#_Toc132883165)

# INTRODUCTION

# High Level Design

## **Layered Architecture**

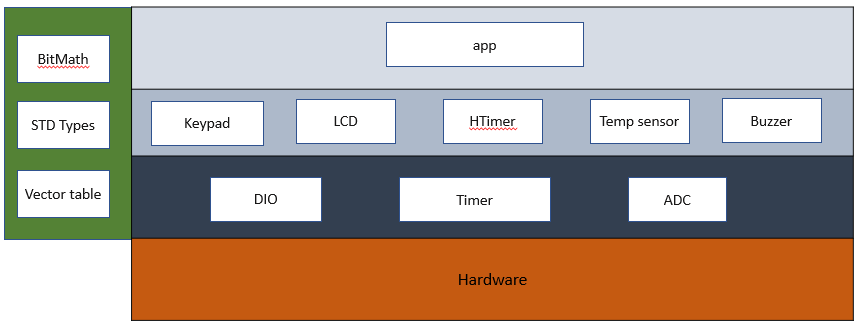


Figure : Project Layered Architecture

## **Modules Description**

**MCAL Layer:**

* **DIO:** For controlling GPIO pins
* **Timer:** Provides an interface with timer 0 low-level capabilities.
* **EXTINT:**
* **USART:**
* **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:**

#### **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:**

#### **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\_INCREAMENT=0,

KEY\_DECREAMENT,

KEY\_SET,

KEY\_ADJUST,

KEY\_RESET,

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 );

#### **EEPROM**

### **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)

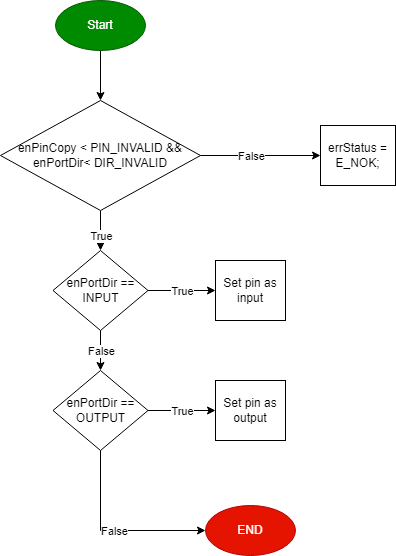
****

Figure 2 DIO\_s8SETPinDir Flow Chart

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

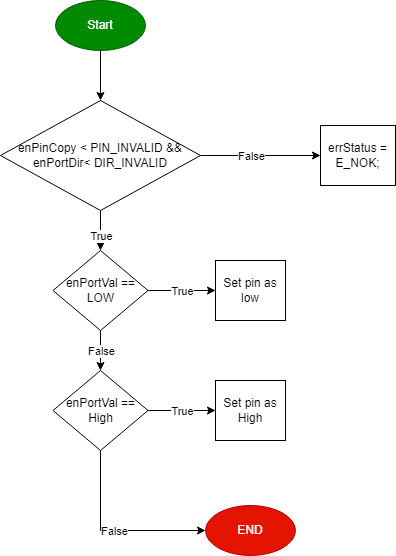
****

Figure 3 DIO\_s8SETPinVal Flow chart

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

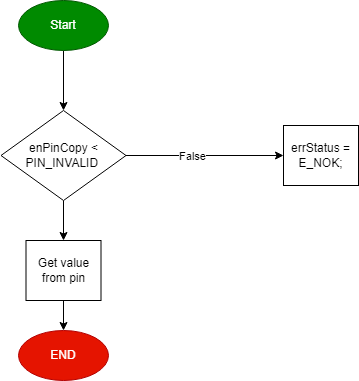
****

Figure 4 DIO\_s8GETPinVal Flow Chart

#### **Timer:**

**TIM0\_Init**

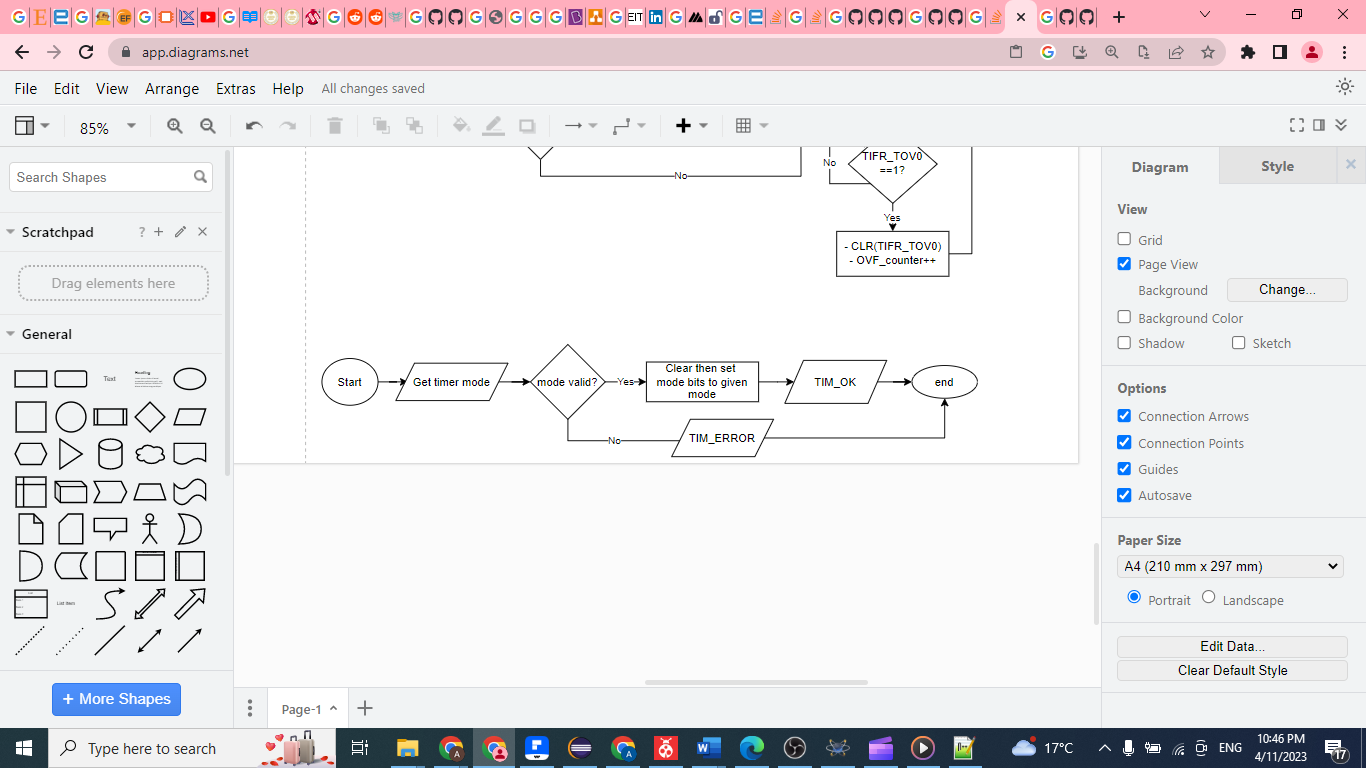


Figure 5 TIM0\_Init Flow Chart

**TIM0\_Start**

****

Figure 6 TIM0\_Start Flow Chart

**TIM0\_Stop**

****

Figure 7 TIM0\_Stop Flow Chart

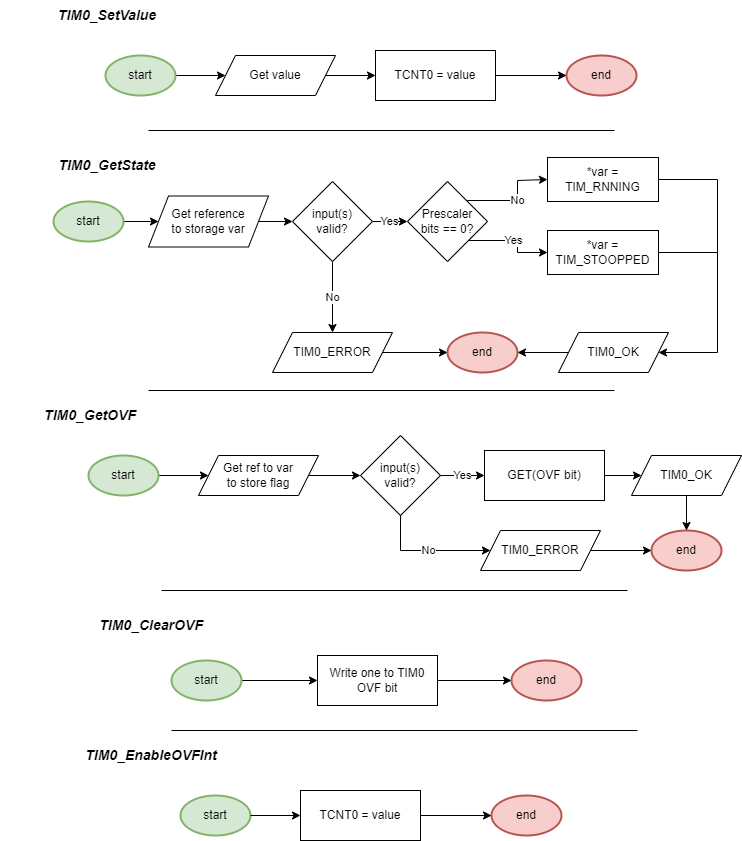


Figure 8 TIM0 remaining Flow Charts

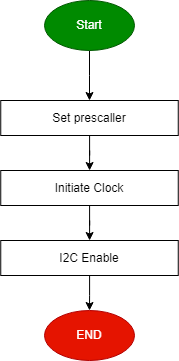
#### **EXTINT:**

#### **UART:**

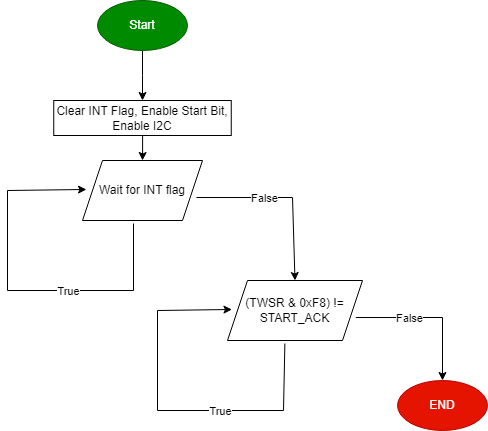
#### **SPI:**

#### **I2C:**

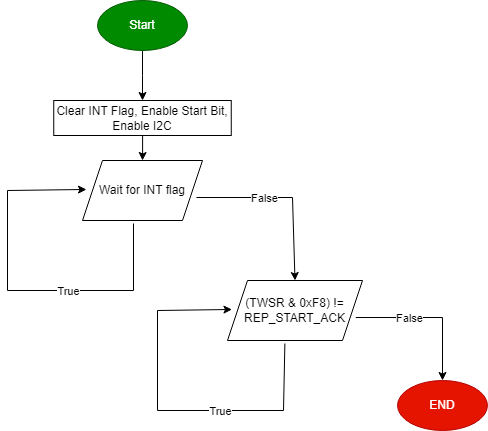
**void** **i2c\_init\_master**(**void**);



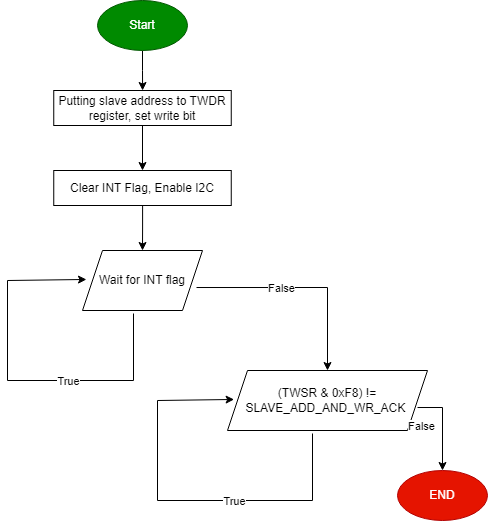
**void** **i2c\_start**(**void**);



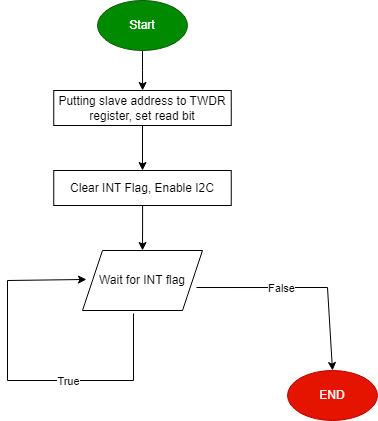
**void** **i2c\_repeated\_start**(**void**);



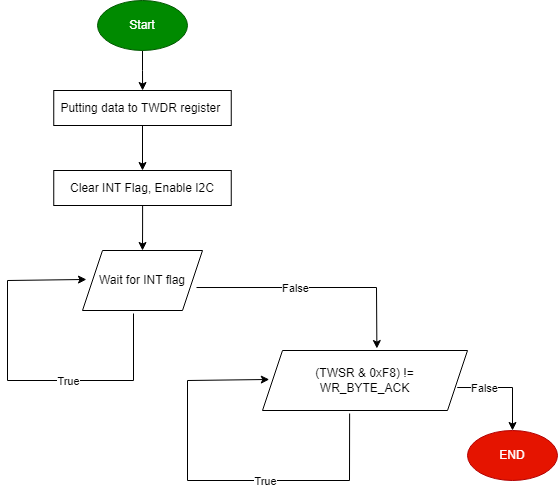
**void** **i2c\_send\_slave\_address\_with\_write\_req**(Uint8\_t slave\_address);



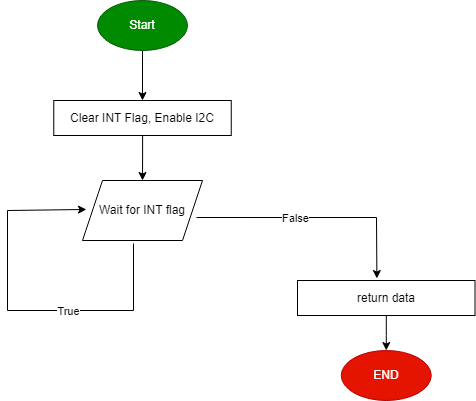
**void** **i2c\_send\_slave\_address\_with\_read\_req**(Uint8\_t slave\_address);



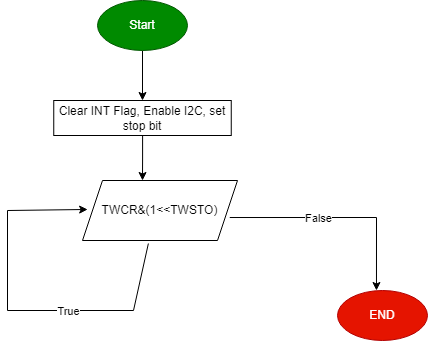
**void** **i2c\_write\_byte**(Uint8\_t byte);



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



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

****

### **HAL Layer**

#### **HTimer0**

**HTIM0\_SyncDelay**

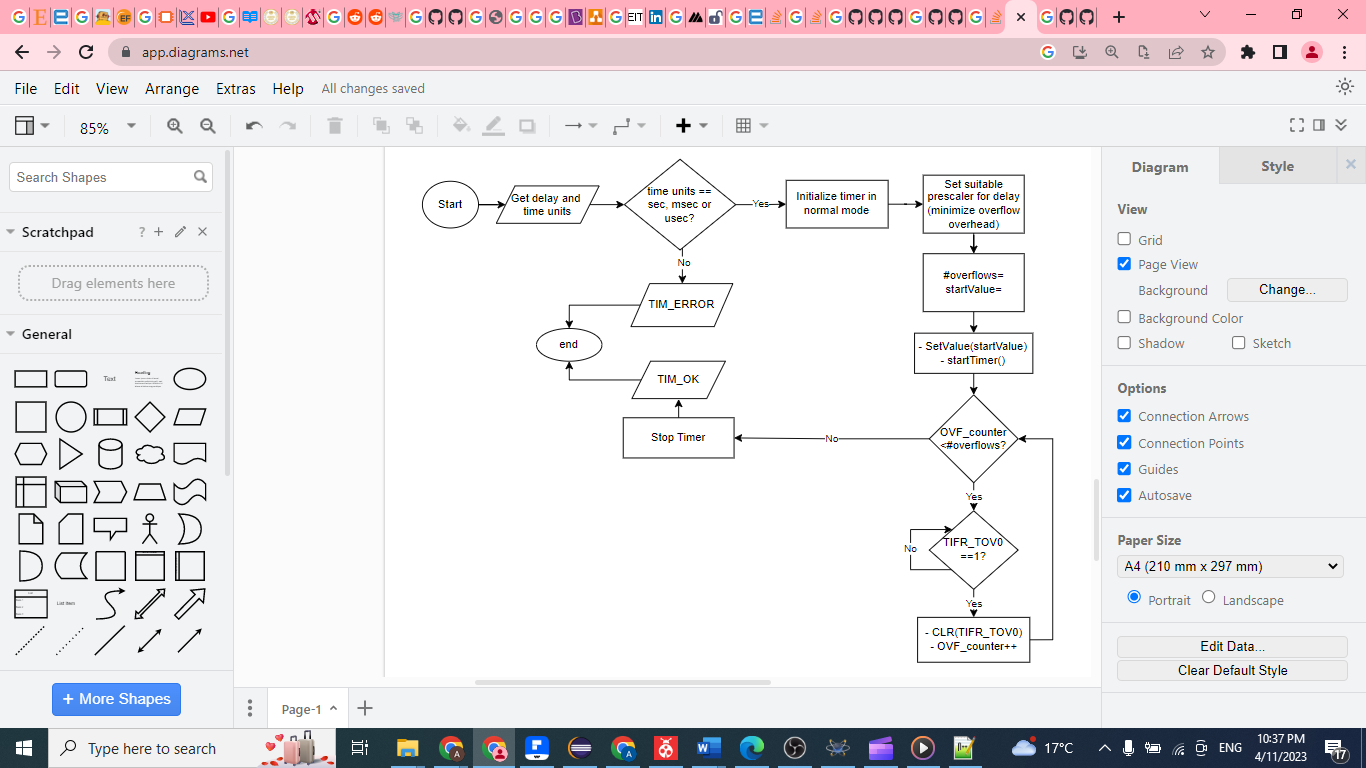


Figure 15 HTIM0\_SyncDelay Flow Chart

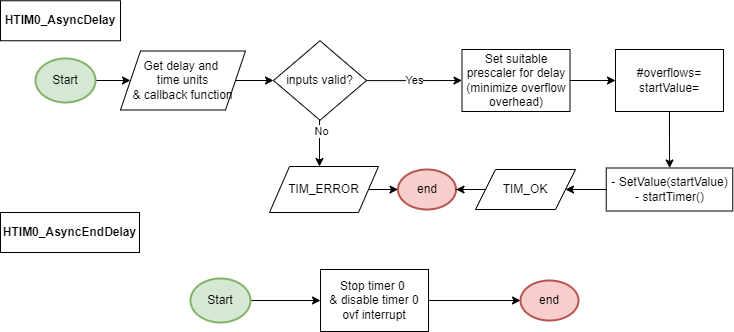


Figure 16 HTIM0\_AsyncDelay and EndDelay

#### **LCD**

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

Figure HLCD\_vidInit Flow Chart

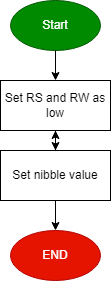
**void** **HLCD\_vidWritecmd**(Uint8\_t u8commandCopy)

Figure HLCD\_vidWritecmd Flow Chart

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

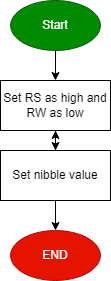
****

Figure HLCD\_vidWriteChar Flow Chart

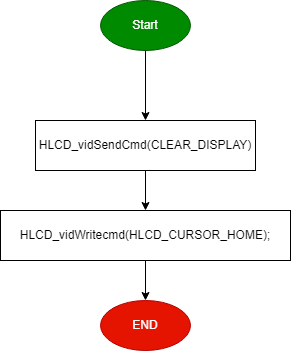
**void** **HLCD\_ClrDisplay**(**void**)

Figure HLCD\_ClrDisplay Flow Chart

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

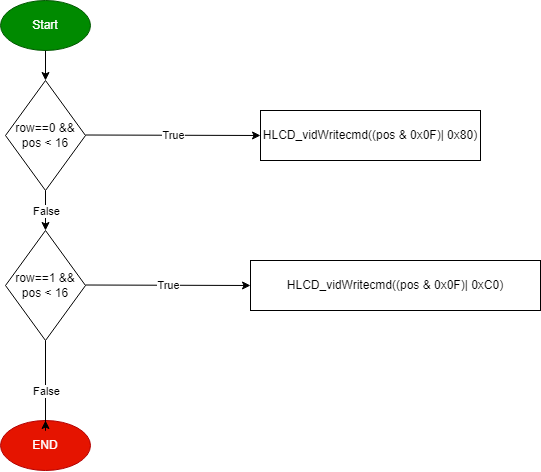
****

Figure 21 HLCD\_gotoXY Flow Chart

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

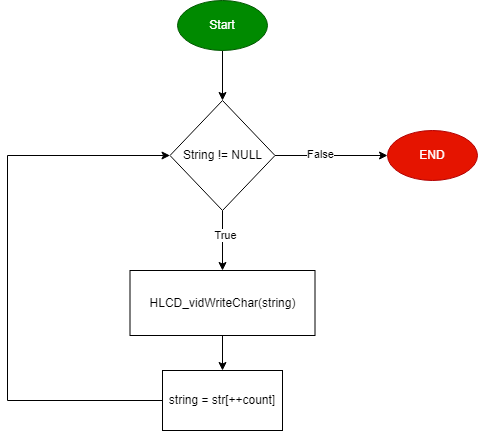
****

Figure 22 HLCD\_WriteString Flow Chart

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

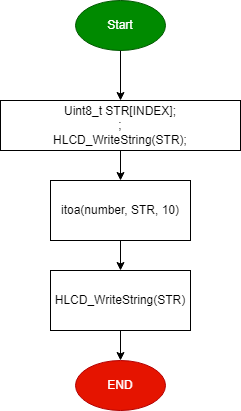
****

Figure 23 HLCD\_WriteInt Flow Chart

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

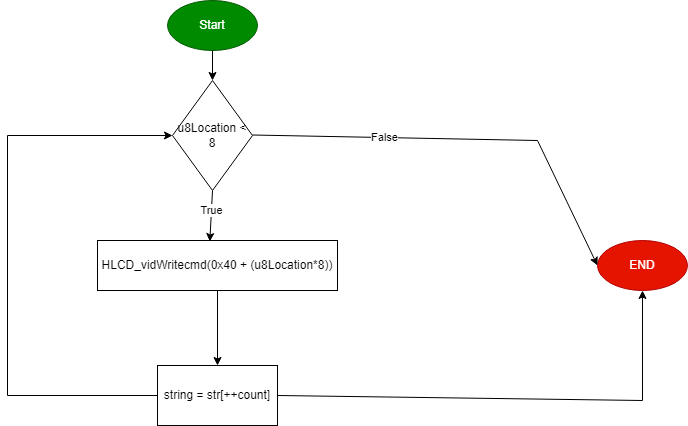
****

Figure 24 HLCD\_vidCreatCustomChar Flow Chart

#### **Keypad**

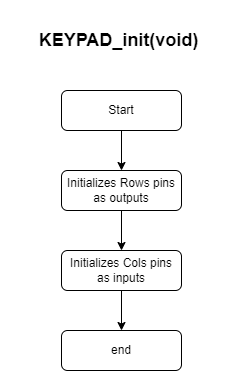


Figure 25 KEYPAD\_Init Flow Chart

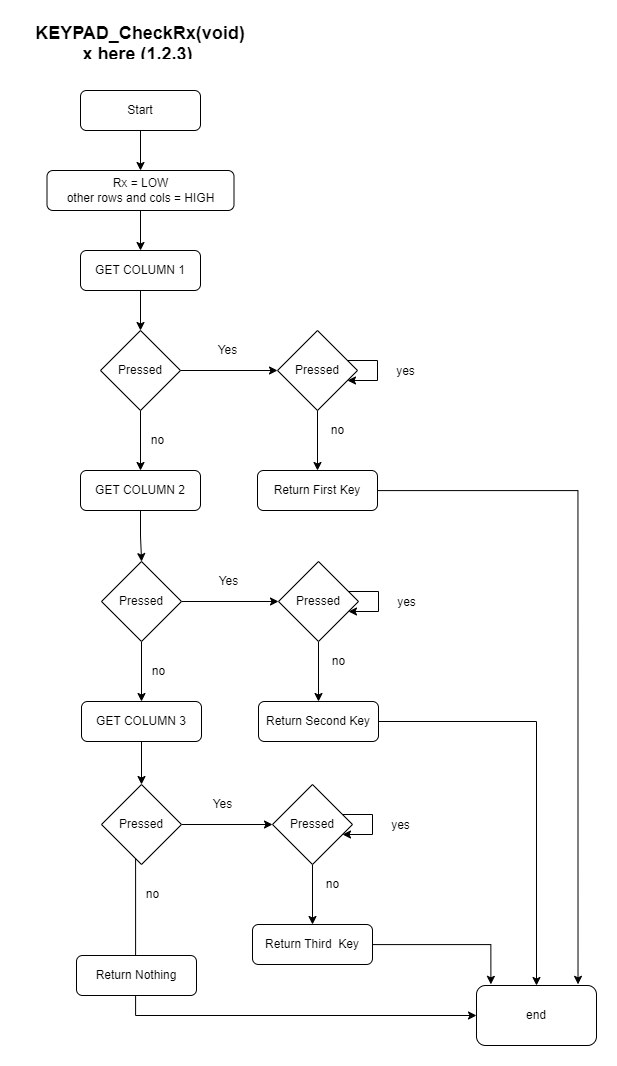


Figure 26 KEYPAD\_CheckRx Flow Chart

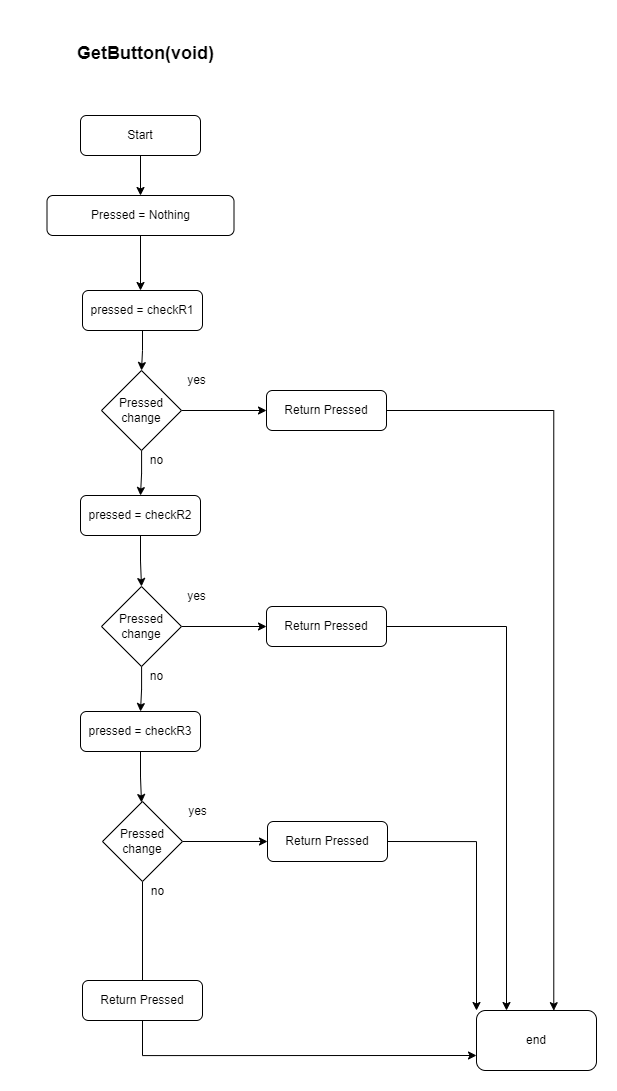
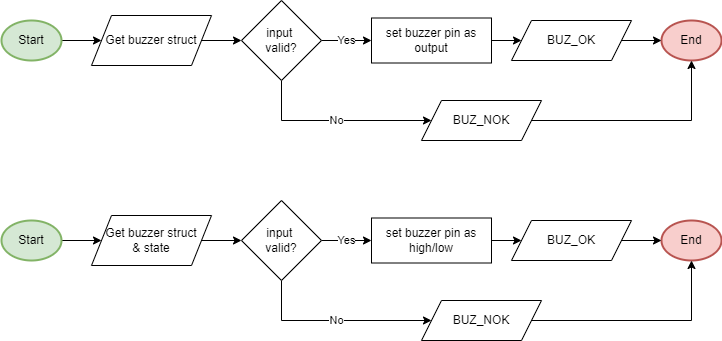


Figure 27 GetButton Flow Chart

#### **Buzzer**

****

**BUZ\_SetState**

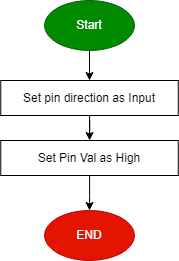
Figure Buzzer Init & SetState Flow Charts

**BUZ\_Init**

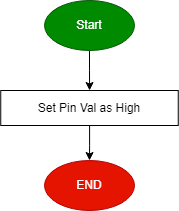
#### **HEXTINT:**

#### **Button:**

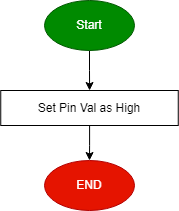
enu\_buttonError\_t **HButton\_Init**(enu\_pin en\_pinx);



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



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

****

### **Application Layer:**

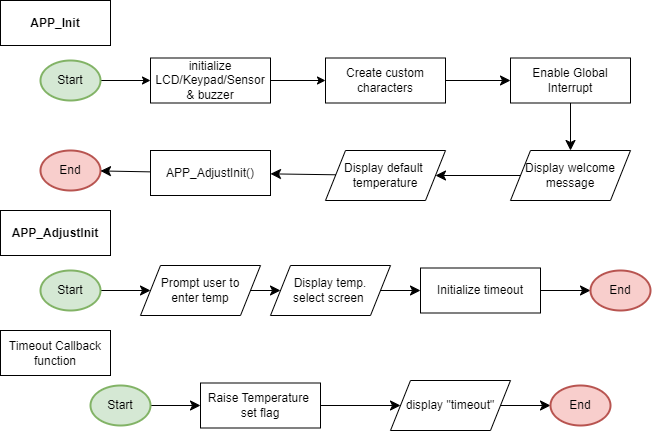


Figure 30 App. APIs Flow Charts



Figure 31 APP\_Start & App States Flow Charts



Figure 32 Cont. App States Flow Charts

