

**Air Conditioning Project**

**Team Members:**

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

Contents

[INTRODUCTION 1](#_Toc132848363)

[High Level Design 1](#_Toc132848364)

[**01)** **Layered Architecture** 1](#_Toc132848365)

[**02)** **Modules Description** 2](#_Toc132848366)

[**03)** **Drivers’ Documentation** 3](#_Toc132848367)

[Low Level Design 12](#_Toc132848368)

[Figure 1: Project Layered Architecture 2](file:///D:\02_workspace\MovingCarProject\Moving_Car.docx#_Toc132213547)

# INTRODUCTION

# High Level Design

## **Layered Architecture**

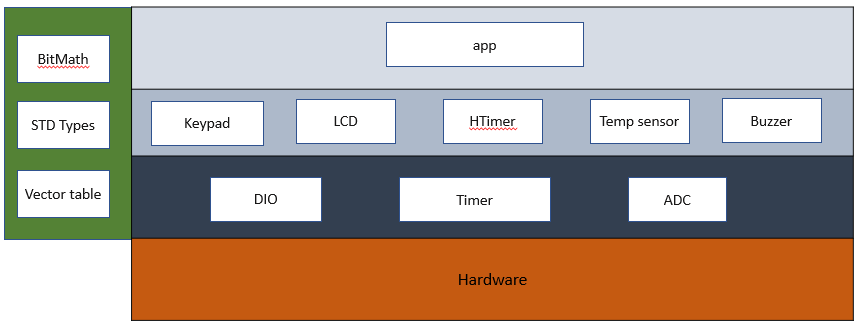


Figure : Project Layered Architecture

## **Modules Description**

**MCAL Layer:**

* **DIO:** For controlling GPIO pins
* **Timer:** Provides an interface with timer 0 low-level capabilities.
* **ADC:** Provides interface to control and read from ADC peripheral.

**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
* **Temp Sensor:** Provides functions to get readings from temperature sensor.
* **Buzzer:** Simple module to control a buzzer.
* **HTimer:** Provides high-level functions using the lower level timer 0 module capabilities.

**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:**

/\*\*

\* \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

\*/

* ADC

/\* Struct Contain all adc information to config it \*/

typedef struct

{

void(\*interruptHandler)(void);

EN\_ADC\_REFERENCE\_SELECTION\_BITS\_t referenceSource;

EN\_ADC\_ADJUST\_RESULT\_t resultAdjust;

EN\_ADC\_PRESCALER\_SELECTION\_t prescalerDivision;

EN\_ADC\_EVENT\_TRIGGER\_SOUREC\_t triggerSource;

}ST\_ADC\_CFG\_t;

/\*\*

\* \brief : This Function Use To Init The Adc It Set Bits For Prescaler , Refrence Source , event trigger resource and Adjust Resualt

\*

\* \param : const ST\_ADC\_CFG\_t \*\_adc

\*

\* \return : Std\_ReturnType

\*/

Std\_ReturnType ADC\_Init(const ST\_ADC\_CFG\_t \*\_adc)

/\*\*

\* \brief : This Function Use To Disable The ADC

\*

\* \param : const ST\_ADC\_CFG\_t \*\_adc

\*

\* \return : Std\_ReturnType

\*/

Std\_ReturnType ADC\_Deinit(const ST\_ADC\_CFG\_t \*\_adc)

/\*\*

\* \brief : This Function Is Used To Select ADC Channel

\*

\* \param : const ST\_ADC\_CFG\_t \*\_adc

\* \param : EN\_ADC\_CHANNEL\_SELECTION\_t \_channel

\*

\* \return : Std\_ReturnType

\*/

Std\_ReturnType ADC\_SetChannel(const ST\_ADC\_CFG\_t \*\_adc , EN\_ADC\_CHANNEL\_SELECTION\_t \_channel)

/\*\*

\* \brief : This Function Use To Start Conversion

\*

\* \param : const ST\_ADC\_CFG\_t \*\_adc

\*

\* \return : Std\_ReturnType

\*/

Std\_ReturnType ADC\_StartConversion(const ST\_ADC\_CFG\_t \*\_adc)

/\*\*

\* \brief : This Function Use To Polling On The ADC Flag To Return The Conversion Resualt

\*

\* \param : const ST\_ADC\_CFG\_t \*\_adc

\* \param : Uint16\_t \*\_ConversionResult

\*

\* \return : Std\_ReturnType

\*/

Std\_ReturnType ADC\_GetConversionResult(const ST\_ADC\_CFG\_t \*\_adc , Uint16\_t \*\_ConversionResult)

/\*\*

\* \brief : This Function Use To Make All Operation Of The Adc

\*

\* \param : const ST\_ADC\_CFG\_t \*\_adc

\* \param : EN\_ADC\_CHANNEL\_SELECTION\_t \_channel

\* \param : Uint16\_t \*\_ConversionResult

\*

\* \return Std\_ReturnType

\*/

Std\_ReturnType ADC\_Conversion(const ST\_ADC\_CFG\_t \*\_adc , Uint16\_t \*\_ConversionResult , EN\_ADC\_CHANNEL\_SELECTION\_t \_channel)

**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);

* **Temperature Sensor**

/\*\*

\* \brief Function to initialize the sensor port/pin

\* \param pst\_a\_sensor reference to sensor info

\* \return void

\*/

void TSENSOR\_Init(st\_TempSensor\_t\* pst\_a\_sensor);

/\*\*

\* \brief Function to get the current sensor reading

\* \param pst\_a\_Sensor reference to sensor info

\* \param f32\_a\_Value reference to variable to store Analog value

\*

\* \return en\_SensorError\_t

\*/

en\_SensorError\_t TSENSOR\_ReadValue(st\_TempSensor\_t \*pst\_a\_Sensor, float32\_t \*f32\_a\_Value);

**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

\*/

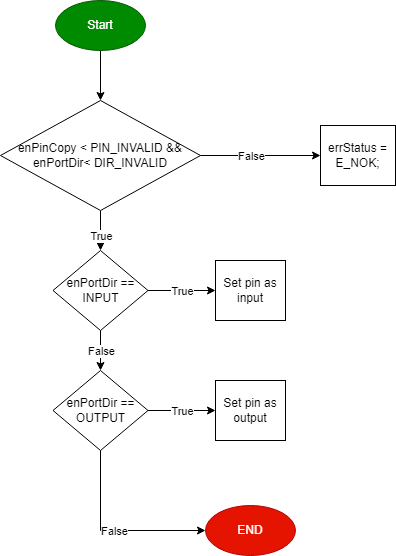
void timeout(void);

# Low Level Design

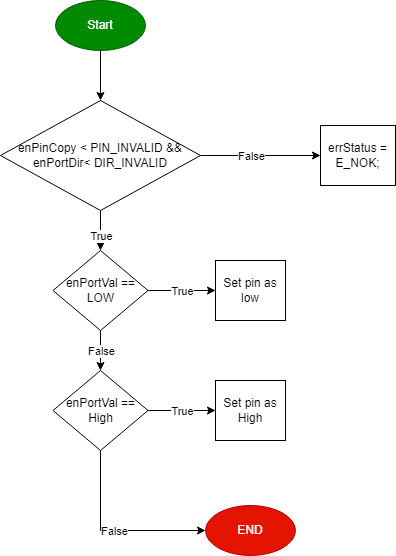
**MCAL Layer:**

* **DIO:**

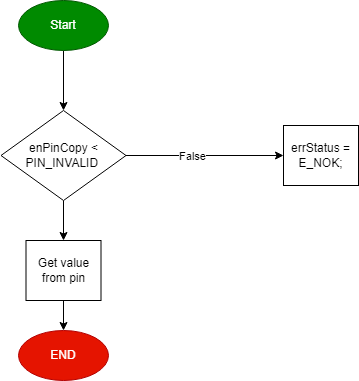
Sint8\_t **DIO\_s8SETPinDir** (enu\_pin enPinCopy, enu\_dir enPortDir)

****

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

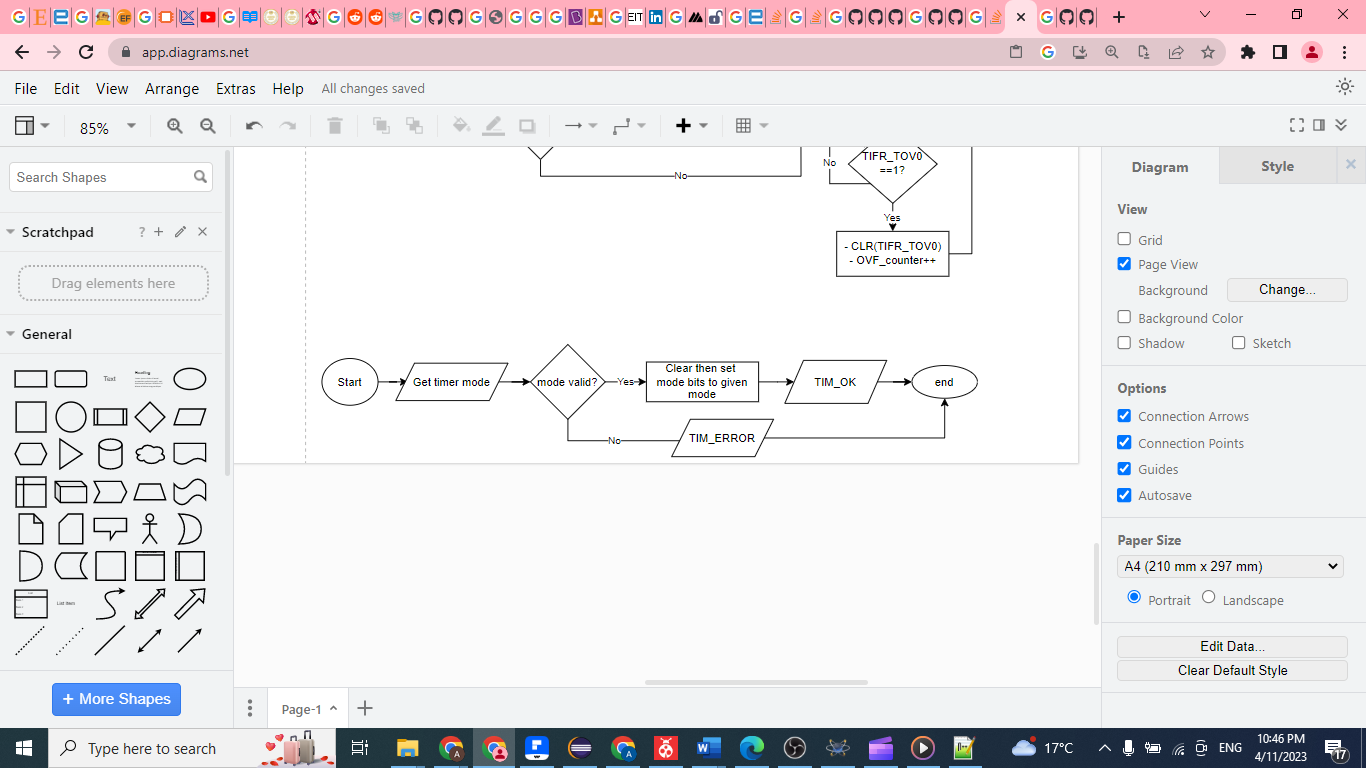
****

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

****

* **Timer:**

**TIM0\_Init**

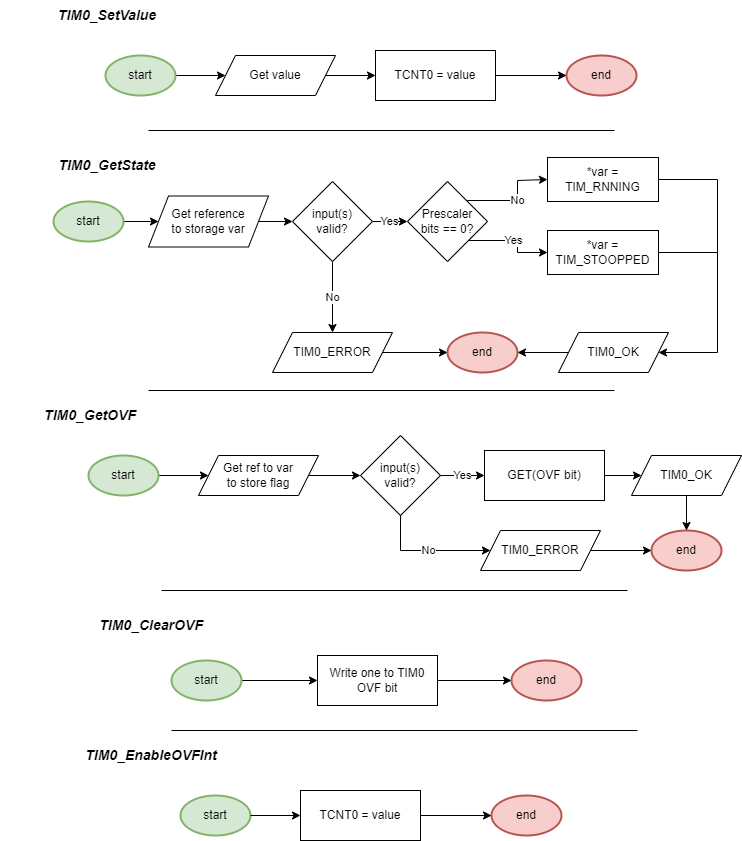


**TIM0\_Start**

****

**TIM0\_Stop**

****

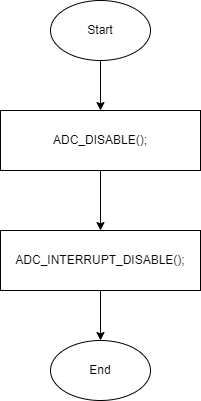


* **ADC**
* ADC\_Init

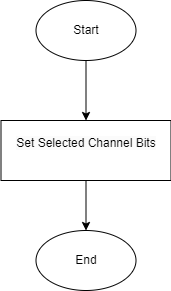
**Chart, diagram, box and whisker chart

Description automatically generated**

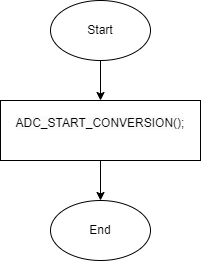
* **ADC\_Deinit**

****

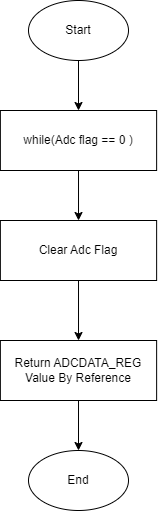
* **ADC\_SetChannel**

****

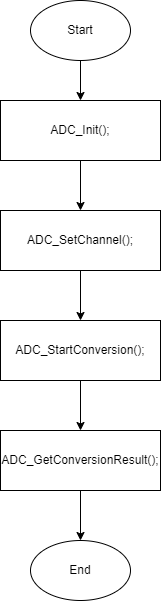
* **ADC\_StartConversion**

****

* **ADC\_GetConversionResult**

****

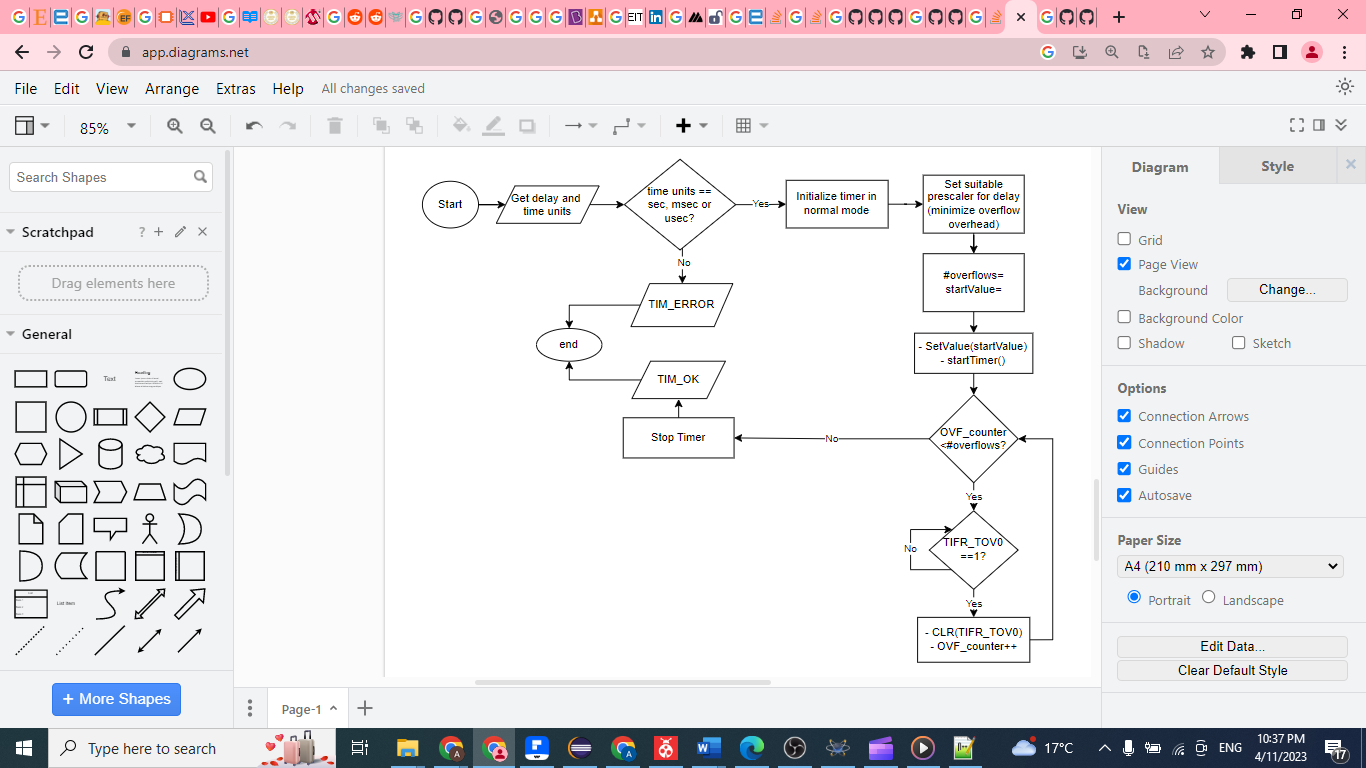
* **ADC\_Conversion**

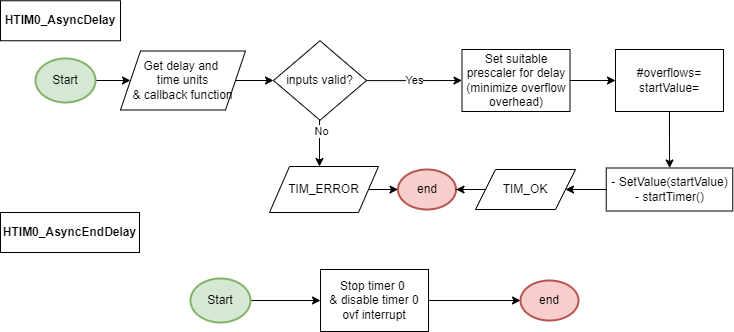
****

**HAL Layer:**

* **Timer0**

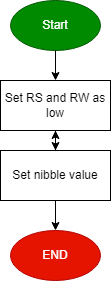
**HTIM0\_SyncDelay**

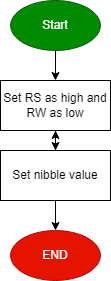




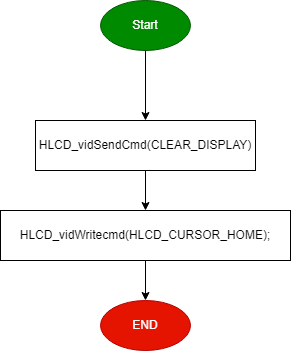
* **LCD:**

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

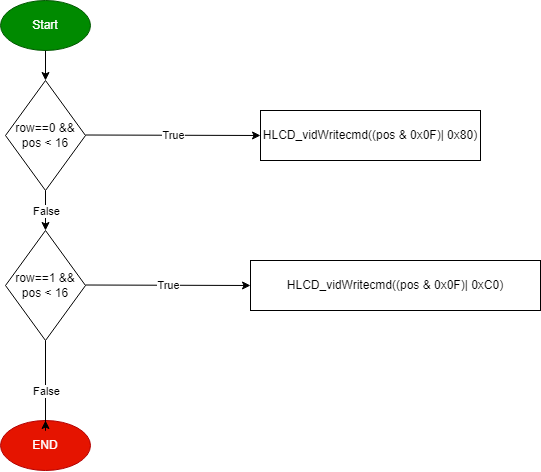
**void** **HLCD\_vidWritecmd**(Uint8\_t u8commandCopy)

****

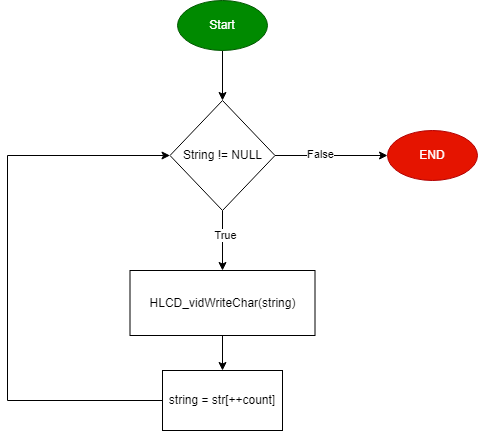
**void** **HLCD\_vidWriteChar**(Uint8\_t u8CharCopy)

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

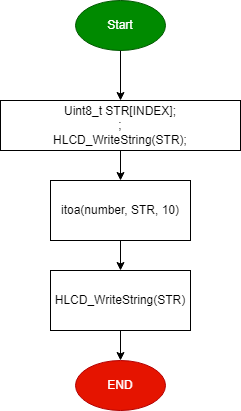
**void** **HLCD\_gotoXY**(Uint8\_t row, Uint8\_t pos)

****

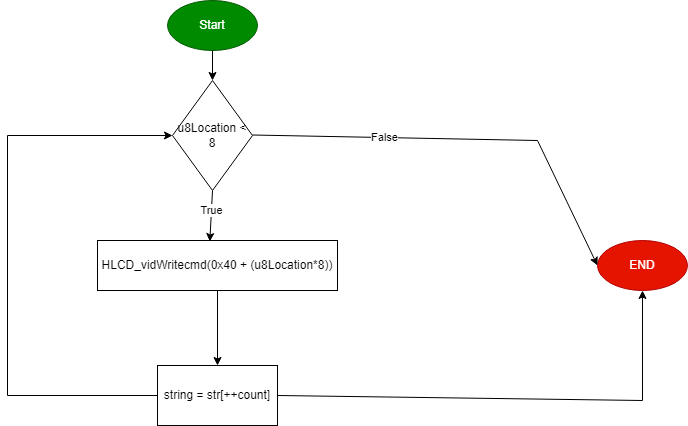
**void** **HLCD\_WriteString**(Uint8\_t\* str)

****

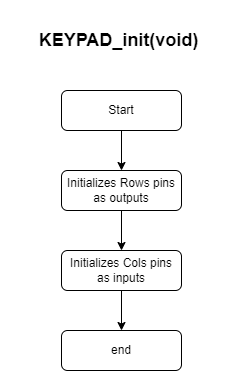
**void** **HLCD\_WriteInt**(Uint32\_t number)

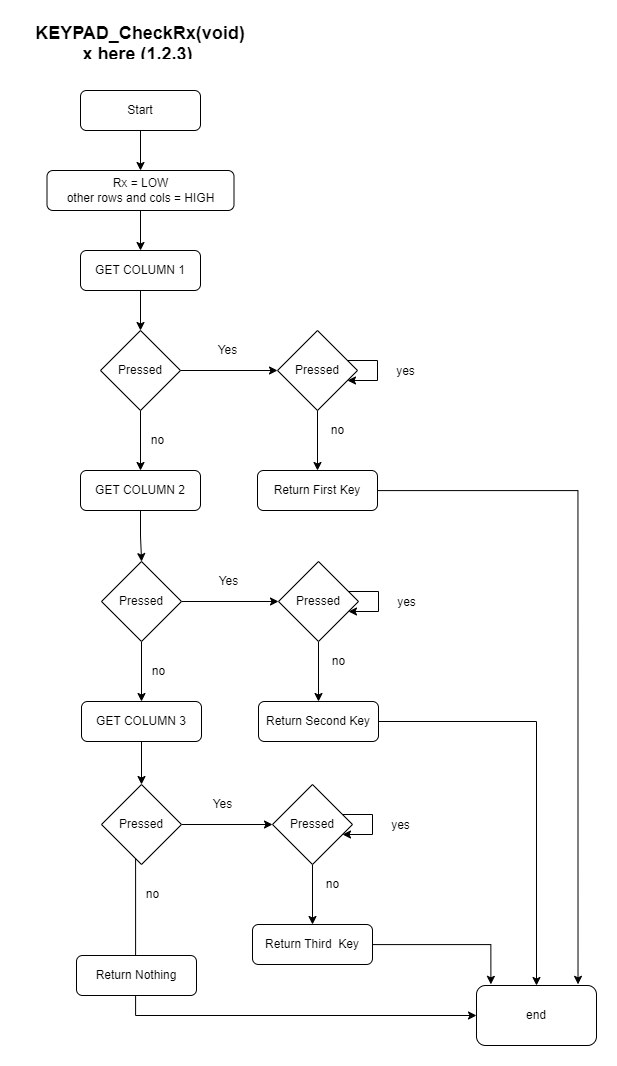
****

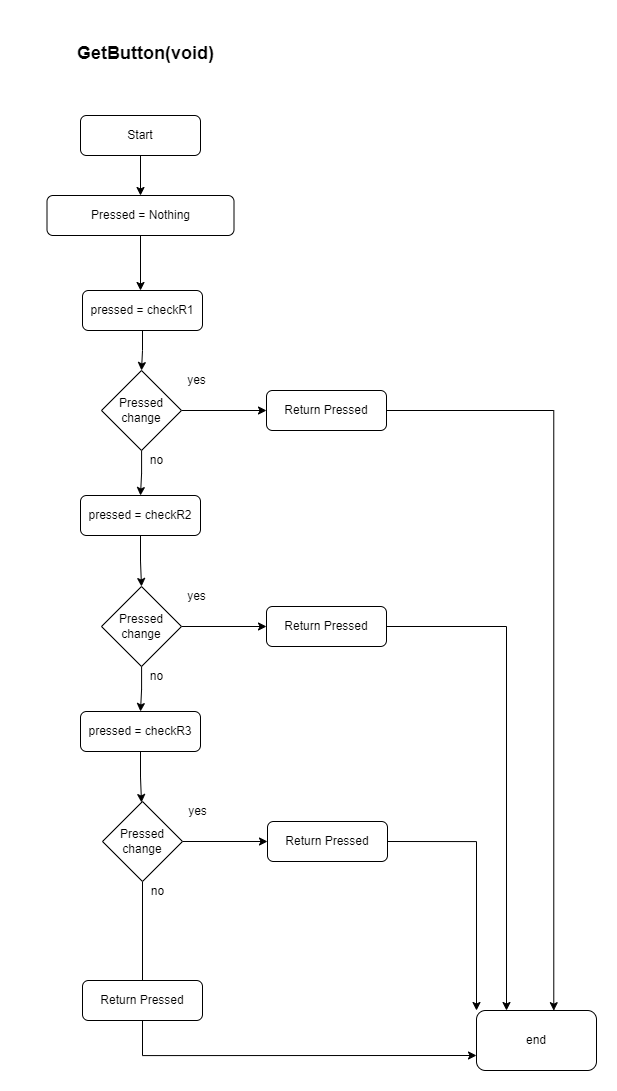
**void** **HLCD\_vidCreatCustomChar**(Uint8\_t\* pu8custom, Uint8\_t u8Location)

****

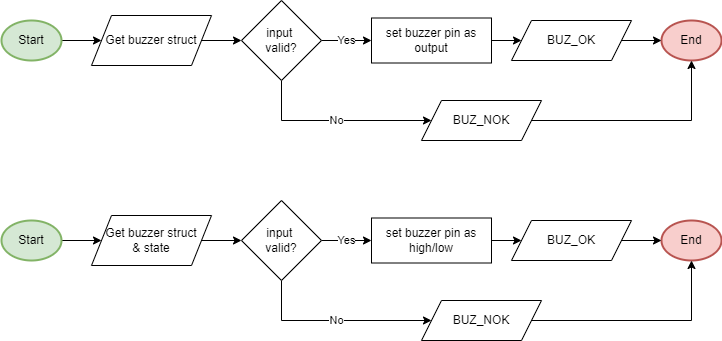
* **Keypad :**





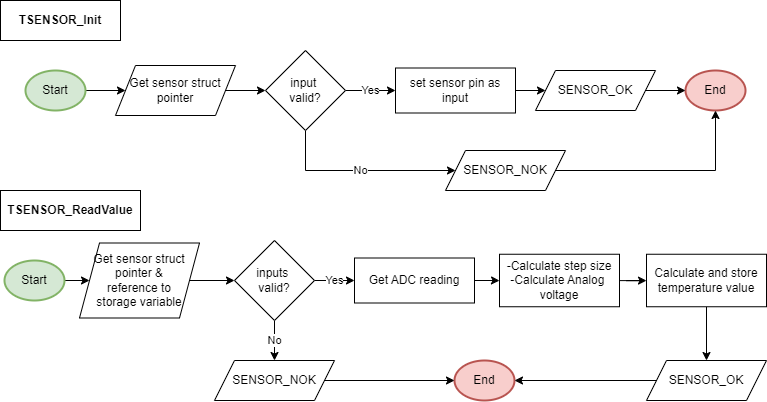


* **Buzzer**

****

**BUZ\_SetState**

**BUZ\_Init**

* **Temperature Sensor**

**Application Layer:**

