|  |
| --- |
|  |
| AIR CONDITIONER PROJECT  BY: Momen Hassan, Ahmed Atef, and Ahmed Mohamed Hesham |
| |  |  |  | | --- | --- | --- | | 4/13/23 |  | Automotive Boot Camp | |

Contents

[LAYERED ARCHITECTURE 2](#_Toc132826437)

[PROJECT STATE MACHINE DIAGRAM 3](#_Toc132826438)

[INTRODUCTION 4](#_Toc132826439)

[MODULE, PERIPHERALS, & SUPPORTING DRIVERS DESCRIPTION 5](#_Toc132826452)

[DRIVERS’ DOCUMENTATION 6](#_Toc132826453)

[1. DIO 6](#_Toc132826454)

[2. TIMERS 8](#_Toc132826455)

[3. ADC 11](#_Toc132826456)

[4. LCD 12](#_Toc132826457)

[5. KEYPAD 15](#_Toc132826458)

[6. LM35 16](#_Toc132826459)

[7. BUZZEER 17](#_Toc132826460)

[FUNCTIONS’ FLOWCHARTS 18](#_Toc132826461)

[1. DIO 18](#_Toc132826462)

[2. TIMERS 22](#_Toc132826463)

[3. ADC 31](#_Toc132826464)

[4. LCD 33](#_Toc132826465)

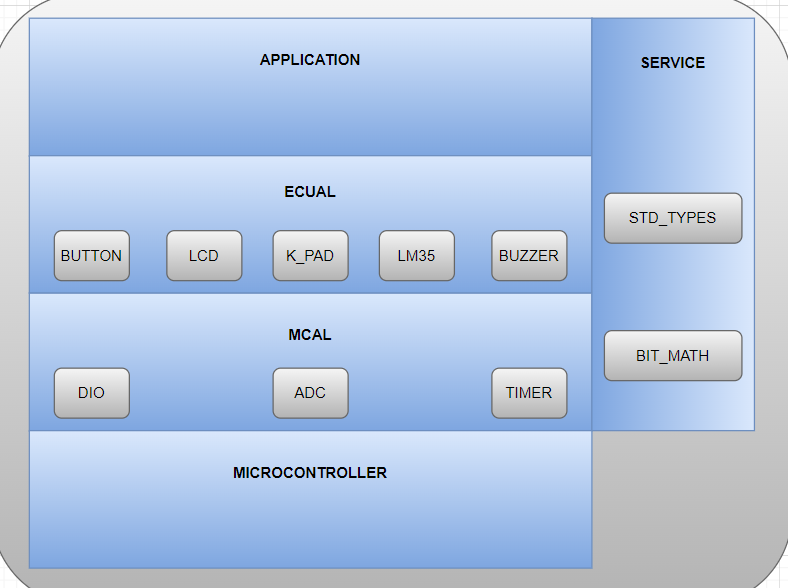
[5. KEYPAD 36](#_Toc132826466)

[6. LM35 38](#_Toc132826467)

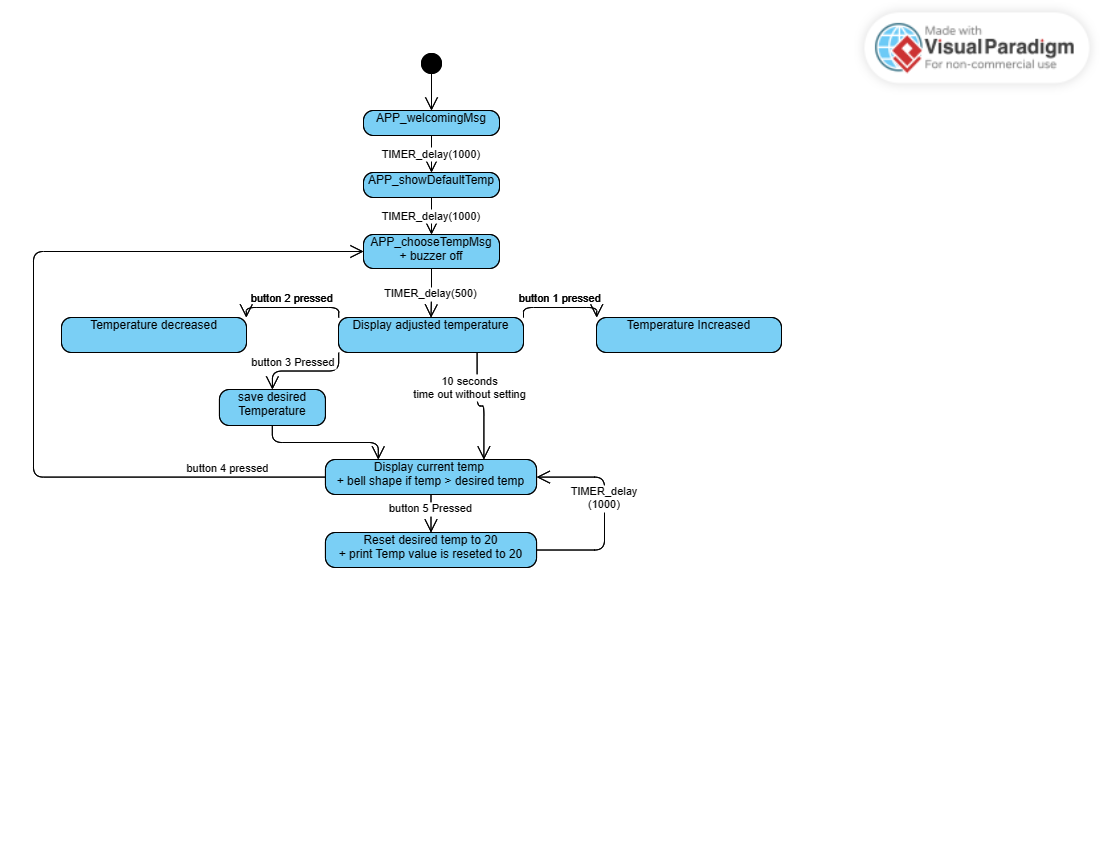
[7. BUZZER 39](#_Toc132826468)

AIR CONDITIONER DESIGN

# LAYERED ARCHITECTURE



# PROJECT STATE MACHINE DIAGRAM



# INTRODUCTION

This project is developing software for temperature sensor integrated with ATMEGA32, 3\*3 keypad, and LCD to make some functionalities similar to what happens with ACs.

So basically, the user will adjust the temperature he/she desires and the MC will compare it with the actual temperature and it buzzes if the actual temperature is higher than the desired temperature.

It consists of four layers:

1- APP

This layer is responsible for integrating application modules and peripherals to perform project functionality via using their APIs

2- ECUAL

In this layer modules' drivers are developed which are the keypad's driver, temperature sensor's driver, and LCD driver.

This layer is like a middle junction between the application layer and the microcontroller abstraction layer

3- MCAL

In this layer, peripherals' drivers are developed: DIO's driver, timers' driver, and ADC.

4- SERVICE

This layer consists of files.h which will serve the main three layers while developing, like it has important data types' type defs, and bit manipulation macros like functions

# MODULE, PERIPHERALS, & SUPPORTING DRIVERS DESCRIPTION

**DIO (Digital Input/Output):** This module deals with the digital input and output operations, such as reading and writing to digital pins of a microcontroller or a microprocessor. It may include functions for setting pin direction, reading and writing digital values, and handling interrupts related to digital pins.

**Timer:** This module deals with timer operations, such as configuring and handling timers in the microcontroller or microprocessor. It may include functions for setting timer intervals, handling timer interrupts, and measuring time. And This module deals with generating PWM signals using normal mode, which are used for controlling the intensity of an output signal, such as controlling the speed of motors or the brightness of LEDs. It may include functions for configuring and controlling PWM signals.

**ADC:** Through this module we can initialize ADC peripheral which is responsible for converting analog input signal to digital signal, and we check on the completion of the conversation by polling on interrupt flag.

**LM35:** This module is the temperature sensor used in our project. Its electrical output is proportional to the temperature in degrees Celsius. It is a linear device which means that the voltage at the output of the LM35 increases proportionally to the temperature. It is rated to operate over a -55°C to 150°C temperature range. The scale factor is 0.01 V/°C.

**LCD:** LCD stands for Liquid Crystal Display. LCD is a type of flat panel display which uses liquid crystals in its primary form of operation. It uses the light-modulating properties of liquid crystals combined with polarizers to display images. It has two pieces of polarized glass (also called substrate) that contain a liquid crystal material between them. A backlight or a reflector creates light that passes through the first substrate and the liquid crystal. The liquid crystals can twist or untwist depending on the electric voltage applied across them. This changes the angle of light passing through the second substrate and the polarizing film. The angel of light determines the color and brightness of the pixels that form the images on the LCD.

**Keypad:** We are using 3\*3 keypad which means we have 3 rows and 3 columns, rows are connected to output high, and columns are connected to be inputs and enable internal pullups. For reading we pass low output simultaneously to the row pins and check if there is any change in the columns.

**Buzzer:** is an output module that buzzes when you write on it HIGH output.

**BIT\_MATH**: This module provides functions for performing bitwise operations, such as AND, OR, XOR, and shifting, which are commonly used for manipulating individual bits in registers or memory locations.

**Standard Types:** This module includes standard data types, such as integer types, floating-point types, and Boolean types, which are used for representing data in a standardized way across the system.

# DRIVERS’ DOCUMENTATION

## DIO

DIO\_init(uint8\_t portNumber, uint8\_t pinNumber, uint8\_t direction);

|  |  |
| --- | --- |
| Function Name | DIO\_init |
| Description | Initializes DIO pins’ direction, output current, and internal attach |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | uint8\_t portNumber, uint8\_t pinNumber, uint8\_t direction |
| Parameters (out) | None |
| Return Value | WRONG\_PORT\_NUMBER, WRONG\_PIN\_NUMBER, WRONG\_DIRECTION, E\_OK |

DIO\_write(uint8\_t portNumber, uint8\_t pinNumber, uint8\_t value);

|  |  |
| --- | --- |
| Function Name | DIO\_write |
| Description | Write on DIO pins’ a specific output High or Low |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | uint8\_t portNumber, uint8\_t pinNumber, uint8\_t value |
| Parameters (out) | None |
| Return Value | WRONG\_PORT\_NUMBER, WRONG\_PIN\_NUMBER, WRONG\_VALUE, E\_OK |

DIO\_toggle(uint8\_t portNumber, uint8\_t pinNumber);

|  |  |
| --- | --- |
| Function Name | DIO\_toggle |
| Description | Toggle the output of a specific pin |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | uint8\_t portNumber, uint8\_t pinNumber |
| Parameters (out) | None |
| Return Value | WRONG\_PORT\_NUMBER, WRONG\_PIN\_NUMBER, E\_OK |

DIO\_read(uint8\_t portNumber, uint8\_t pinNumber, uint8\_t \*value);

|  |  |
| --- | --- |
| Function Name | DIO\_read |
| Description | Read input from a pin and send it back in a pointer to uint8\_t |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | uint8\_t portNumber, uint8\_t pinNumber |
| Parameters (out) | uint8\_t \*value |
| Return Value | WRONG\_PORT\_NUMBER,  WRONG\_PIN\_NUMBER,  E\_OK |

## TIMERS

en\_timerError\_t TIMER\_init(u8 u8\_a\_timerUsed);

|  |  |
| --- | --- |
| Function Name | TIMER\_init |
| Description | Initializes a specific timer to work as a CTC or overflow timer |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | uint8\_t timerUsed |
| Parameters (out) | None |
| Return Value | EN\_timerError\_t |

en\_timerError\_t TIMER\_setTime(u8 u8\_a\_timerUsed, u32 u32\_a\_desiredTime);

|  |  |
| --- | --- |
| Function Name | TIMER\_setTime |
| Description | Used to set time at which the timer interrupt will fires and execute a desired function |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | uint8\_t timerUsed, uint32\_t desiredTime |
| Parameters (out) | None |
| Return Value | EN\_timerError\_t |

en\_timerError\_t TIMER\_start(u8 u8\_a\_timerUsed);

|  |  |
| --- | --- |
| Function Name | TIMER\_start |
| Description | Start specific timer to count |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | uint8\_t timerUsed |
| Parameters (out) | None |
| Return Value | EN\_timerError\_t |

en\_timerError\_t TIMER\_stop(u8 u8\_a\_timerUsed);

|  |  |
| --- | --- |
| Function Name | TIMER\_stop |
| Description | Stop specific timer from counting |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | uint8\_t timerUsed |
| Parameters (out) | None |
| Return Value | EN\_timerError\_t |

en\_timerError\_t TIMER\_pwmGenerator(u8 u8\_a\_timerUsed, u32 u32\_a\_desiredDutyCycle);

|  |  |
| --- | --- |
| Function Name | TIMER\_pwmGenerator |
| Description | Generates PWM signal using normal mode for a specific timer |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | u8\_a\_timerUsed, u8\_a\_desiredDutyCycle |
| Parameters (out) | None |
| Return Value | en\_timerError\_t |

void TIMER\_setCallBack(u8 u8\_a\_timerUsed, void (\*funPtr)(void));

|  |  |
| --- | --- |
| Function Name | TIMER\_setCallBack |
| Description | Initializes Sends pointer to function to be called when the timer’s interrupt fires |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | uint8\_t portNumber, uint8\_t pinNumber, uint8\_t direction |
| Parameters (out) | None |
| Return Value | None |

en\_timerError\_t TIMER\_stopInterrupt(u8 u8\_a\_timerUsed);

|  |  |
| --- | --- |
| Function Name | TIMER\_stopInterrupt |
| Description | Disable a specific timer’s peripheral interrupt |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | u8\_a\_timerUsed |
| Parameters (out) | None |
| Return Value | en\_timerError\_t |

en\_timerError\_t TIMER\_delay(u8 u8\_a\_timerUsed, u32 u32\_a\_timeInMS);

|  |  |
| --- | --- |
| Function Name | TIMER\_enableInterrupt |
| Description | Generates a delay using a specific timer |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | u8\_a\_timerUsed, u32\_a\_timeInMS |
| Parameters (out) | None |
| Return Value | en\_timerError\_t |

en\_timerError\_t TIMER\_enableInterrupt(u8 u8\_a\_timerUsed);

|  |  |
| --- | --- |
| Function Name | TIMER\_enableInterrupt |
| Description | Enables a specific timer’s peripheral interrupt |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | u8\_a\_timerUsed |
| Parameters (out) | None |
| Return Value | en\_timerError\_t |

## ADC

void ADC\_init (void);

|  |  |
| --- | --- |
| Function Name | ADC\_init |
| Description | Initialize ADC according to preprocessed configured definitions |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | None |

u16 ADC\_read (void);

|  |  |
| --- | --- |
| Function Name | ADC\_read |
| Description | Pulls on ADC interrupt flag till it rises to start returning a conversion to the user |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | u16 |

## LCD

void LCD\_Init(void);

|  |  |
| --- | --- |
| Function Name | LCD\_Init |
| Description | Initialize LCD according to preprocessed configured definitions |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | None |

void LCD\_PinsInit ();

|  |  |
| --- | --- |
| Function Name | LCD\_PinInit |
| Description | Initialize LCD pins directions according to preprocessed configured definitions |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | None |

void LCD\_WriteChar(u8 u8\_a\_ch);

|  |  |
| --- | --- |
| Function Name | LCD\_WriteChar |
| Description | Prints Character on LCD |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | U8\_a\_ch |
| Parameters (out) | None |
| Return Value | None |

void LCD\_WriteString(u8 \*u8\_a\_str);

|  |  |
| --- | --- |
| Function Name | LCD\_WriteString |
| Description | Prints string on LCD |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | \*u8\_a\_str |
| Parameters (out) | None |
| Return Value | None |

void LCD\_WriteNumber(i32 i32\_a\_num);

|  |  |
| --- | --- |
| Function Name | LCD\_WriteNumber |
| Description | Prints a specific number on LCD |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | i32\_a\_num |
| Parameters (out) | None |
| Return Value | None |

void LCD\_SetCursor(u8 u8\_a\_line,u8 u8\_a\_cell);

|  |  |
| --- | --- |
| Function Name | LCD\_SetCursor |
| Description | Changes Cursor’s Location |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | u8\_a\_line, u8\_a\_cell |
| Parameters (out) | None |
| Return Value | None |

void LCD\_Clear(void);

|  |  |
| --- | --- |
| Function Name | LCD\_Clear |
| Description | Clears LCD’s screen and set cursor at line 0 cell 0 |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | None |

void LCD\_ClearLoc(u8 u8\_a\_line ,u8 u8\_a\_cell,u8 u8\_a\_num);

|  |  |
| --- | --- |
| Function Name | LCD\_ClearLoc |
| Description | Clear specific cells from a specific location |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | u8\_a\_line, u8\_a\_cell\_, u8\_a\_num |
| Parameters (out) | None |
| Return Value | None |

void LCD\_CustomChar(u8 u8\_a\_loc,u8 \*u8\_a\_pattern);

|  |  |
| --- | --- |
| Function Name | LCD\_CustomChar |
| Description | Creates a customized character |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | u8\_a\_loc, \*u8\_a\_pattern |
| Parameters (out) | None |
| Return Value | None |

## KEYPAD

void KEYPAD\_init (void);

|  |  |
| --- | --- |
| Function Name | KEYPAD\_init |
| Description | Initialize KEYPAD according to preprocessed configured definitions |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | None |

u8 KEYPAD\_read (void);

|  |  |
| --- | --- |
| Function Name | KEYPAD\_read |
| Description | returns 0 if there is no key pressed or equivalent value for the key if there is a key pressed |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | U8 |

## LM35

void LM35\_init (void);

|  |  |
| --- | --- |
| Function Name | LM35\_init |
| Description | Initialize LM35 according to preprocessed configured definitions |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | None |

u16 LM35\_read (void);

|  |  |
| --- | --- |
| Function Name | LM35\_read |
| Description | returns Degree Celsius after Conversion |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | None |
| Parameters (out) | None |
| Return Value | u16 |

## BUZZEER

en\_buzzerError\_t BUZZER\_init (u8 u8\_a\_buzzerNumber);

|  |  |
| --- | --- |
| Function Name | BUZZER\_init |
| Description | Initialize Buzzer according to preprocessed configured definitions |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | u8\_a\_buzzerNumber |
| Parameters (out) | None |
| Return Value | BUZZER\_OK, WRONG\_BUZZER. |

en\_buzzerError\_t BUZZER\_on (u8 u8\_a\_buzzerNumber);

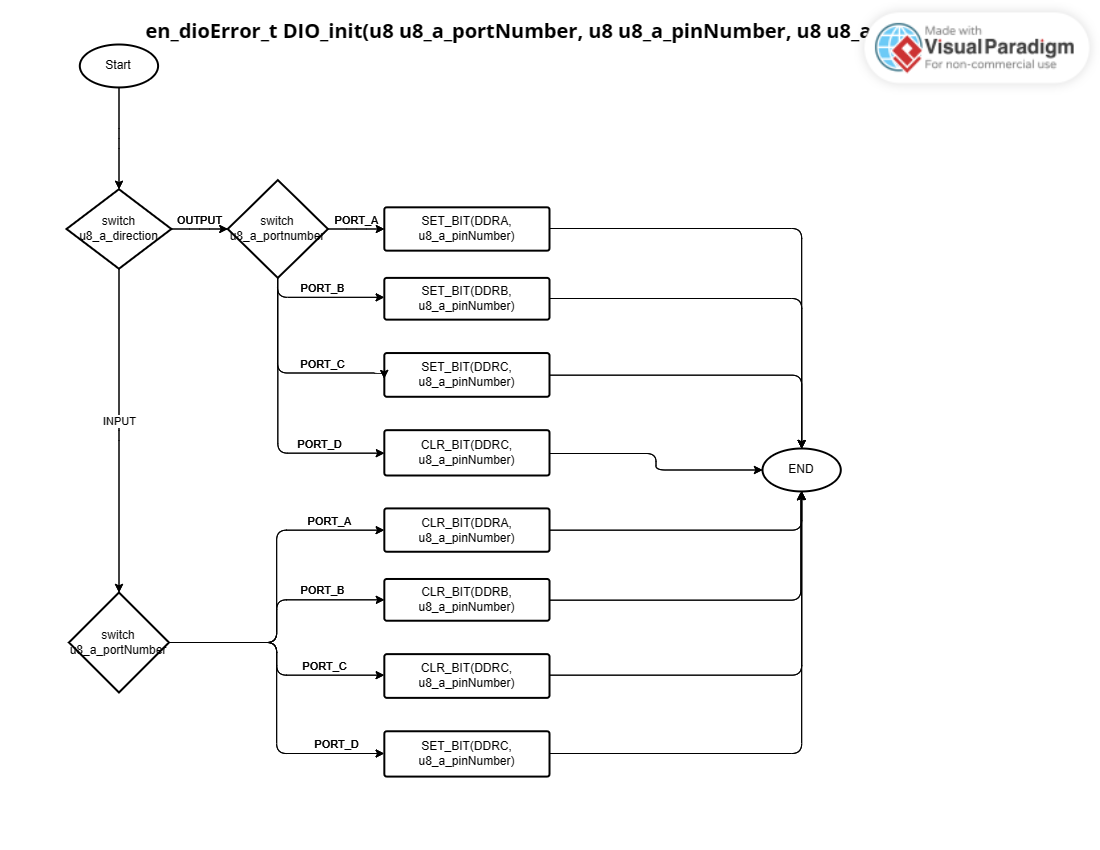
|  |  |
| --- | --- |
| Function Name | BUZZER\_on |
| Description | Switches Buzzer On |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | u8\_a\_buzzerNumber |
| Parameters (out) | None |
| Return Value | BUZZER\_OK, WRONG\_BUZZER. |

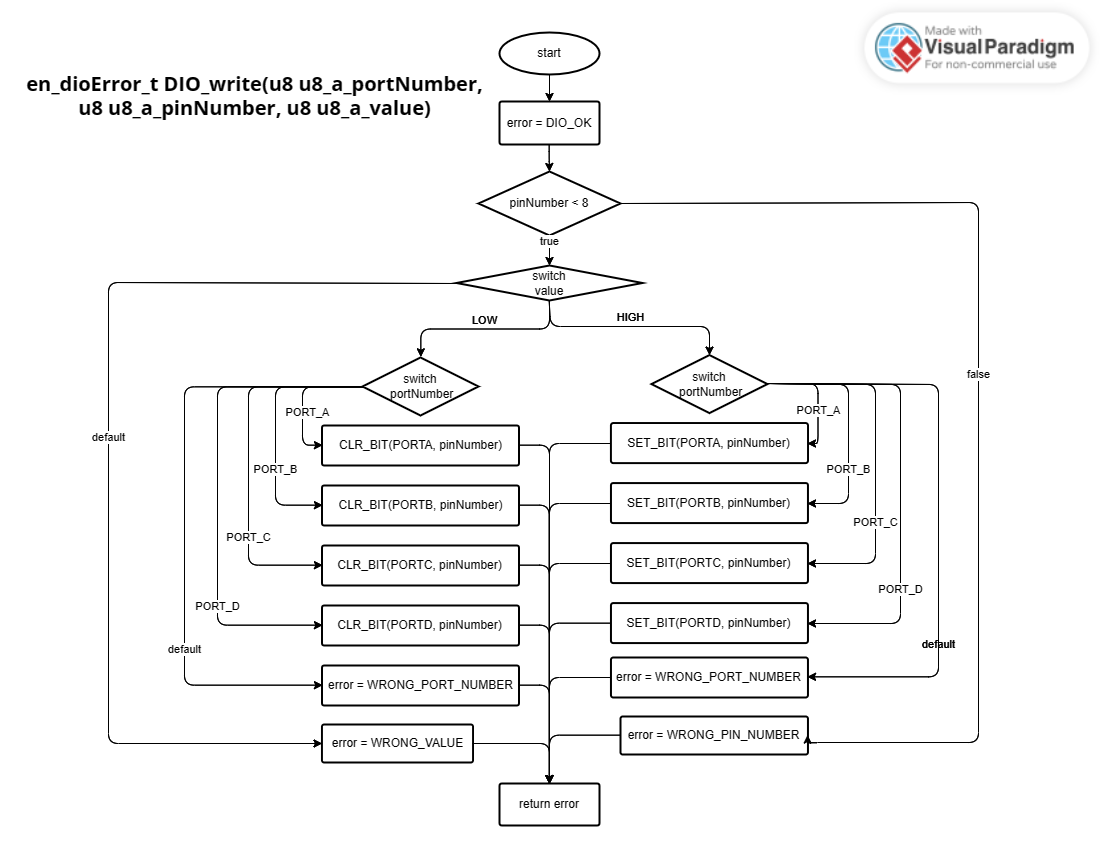
en\_buzzerError\_t BUZZER\_off (u8 u8\_a\_buzzerNumber);

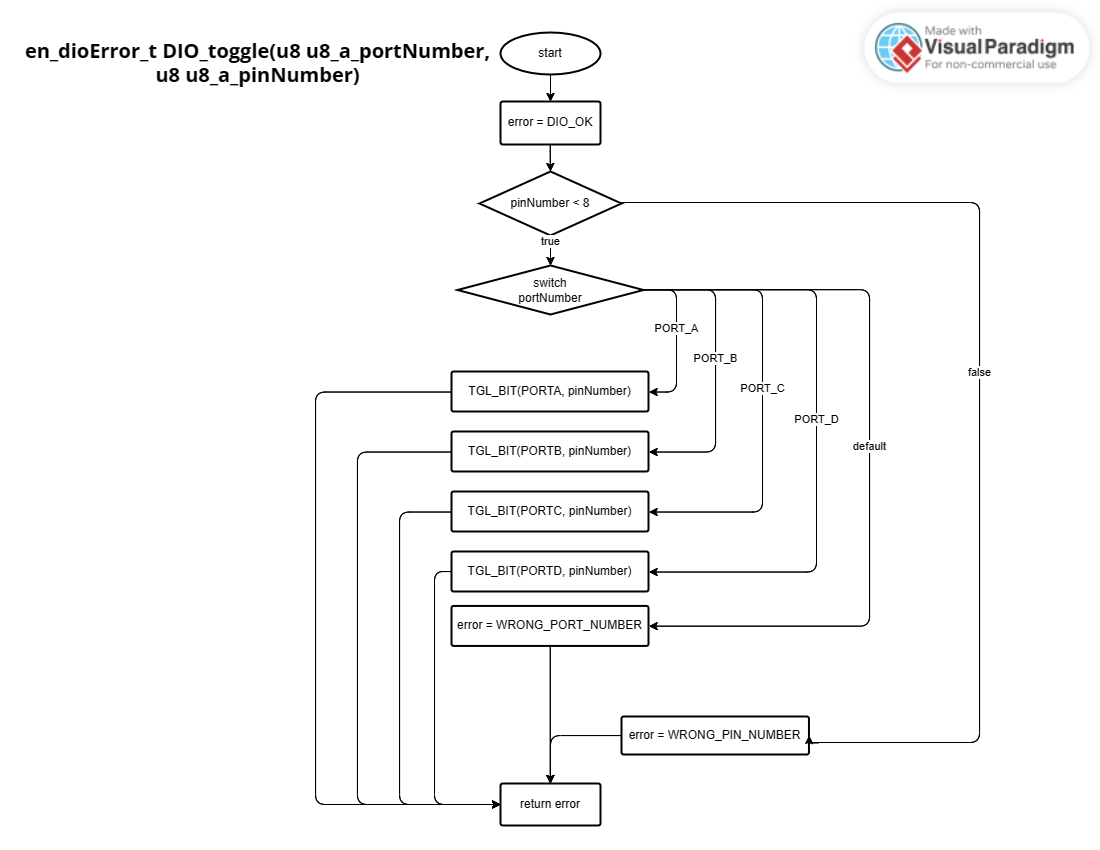
|  |  |
| --- | --- |
| Function Name | BUZZER\_off |
| Description | Turns Buzzer off |
| Sync\Async | Synchronous |
| Reentrancy | Reentrant |
| Parameters (in) | u8\_a\_buzzerNumber |
| Parameters (out) | None |
| Return Value | BUZZER\_OK, WRONG\_BUZZER. |

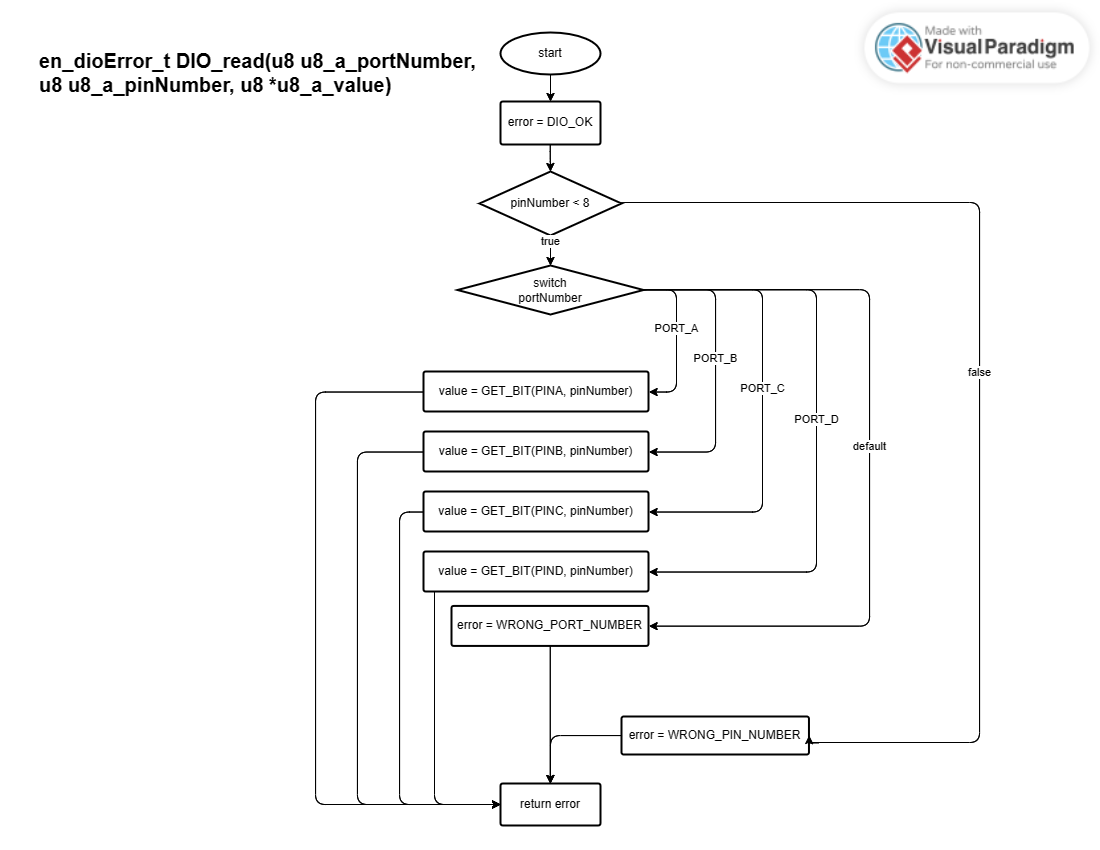
# FUNCTIONS’ FLOWCHARTS

## DIO



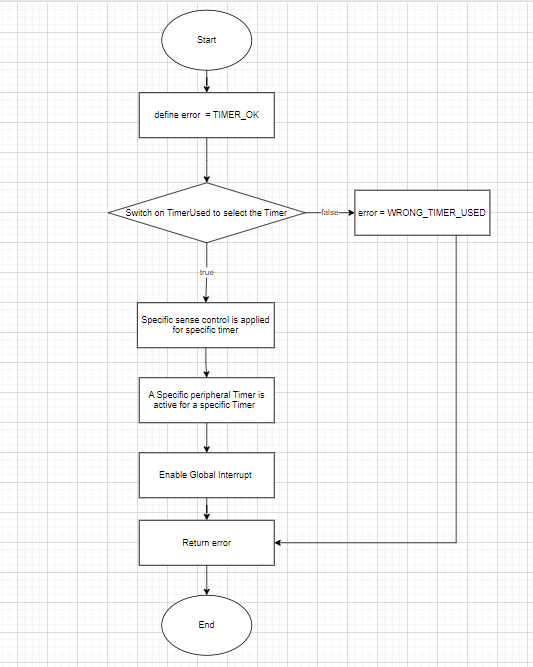




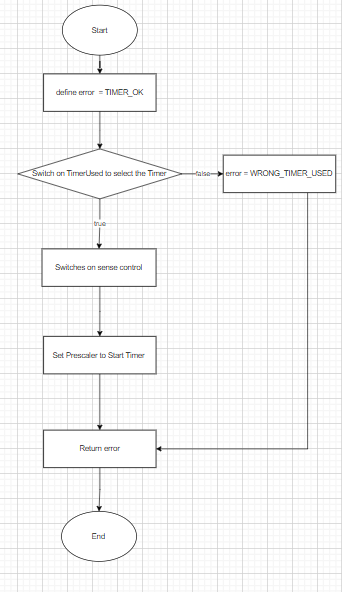


## TIMERS

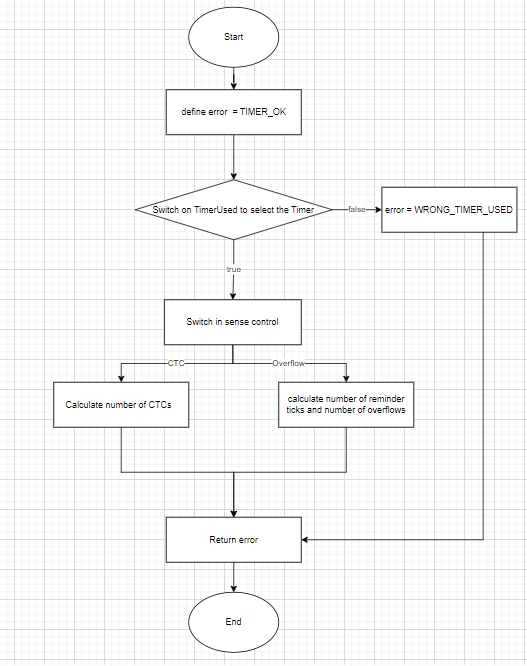
en\_timerError\_t TIMER\_init(u8 u8\_a\_timerUsed);



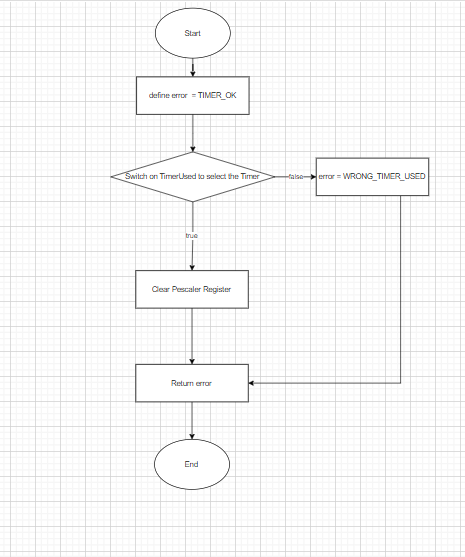
en\_timerError\_t TIMER\_start(u8 u8\_a\_timerUsed);



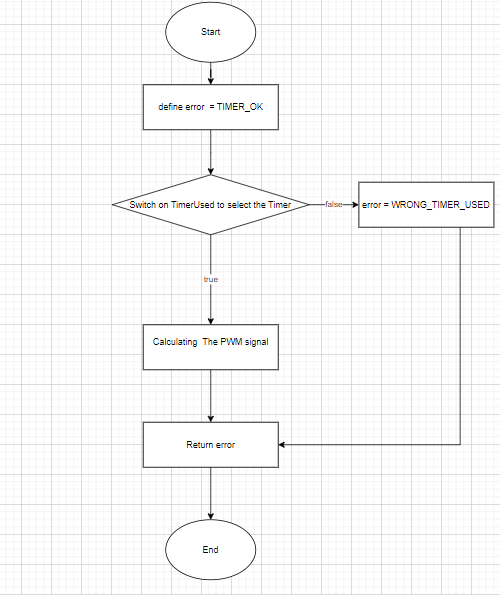
en\_timerError\_t TIMER\_setTime(u8 u8\_a\_timerUsed, u32 u32\_a\_desiredTime);



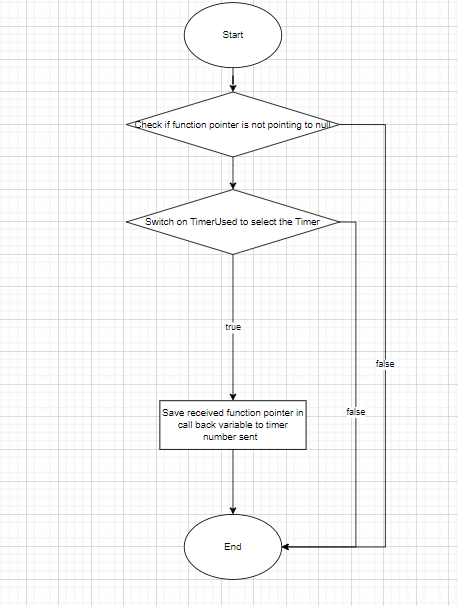
en\_timerError\_t TIMER\_stop(u8 u8\_a\_timerUsed);



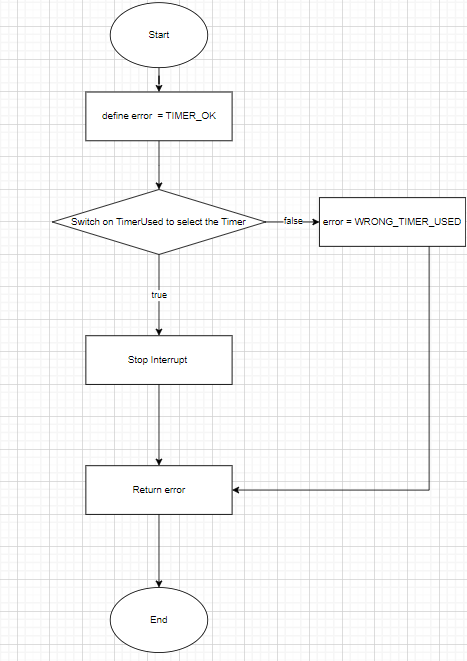
en\_timerError\_t TIMER\_pwmGenerator(u8 u8\_a\_timerUsed, u32 u32\_a\_desiredDutyCycle);



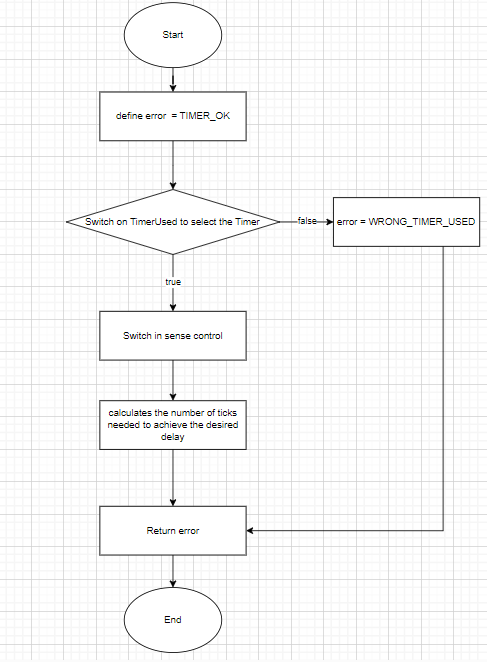
Void TIMER\_setCallBack(u8 u8\_a\_timerUsed, void (\*funPtr)(void));



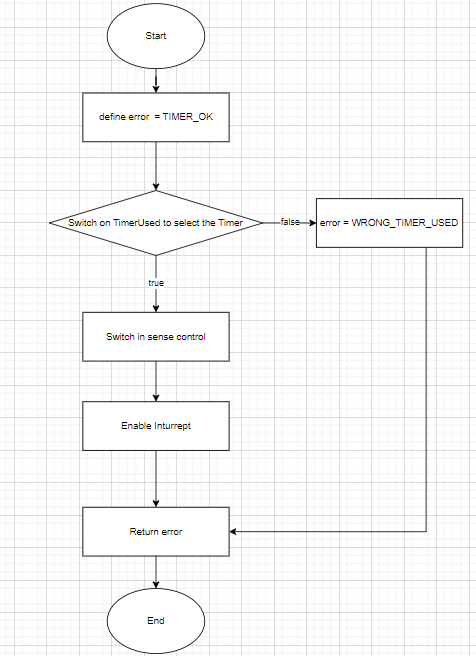
en\_timerError\_t TIMER\_stopInterrupt(u8 u8\_a\_timerUsed);



en\_timerError\_t TIMER\_delay(u8 u8\_a\_timerUsed, u32 u32\_a\_timeInMS);

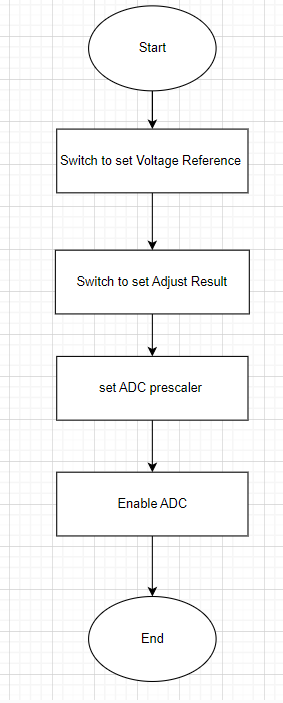


en\_timerError\_t TIMER\_enableInterrupt(u8 u8\_a\_timerUsed);

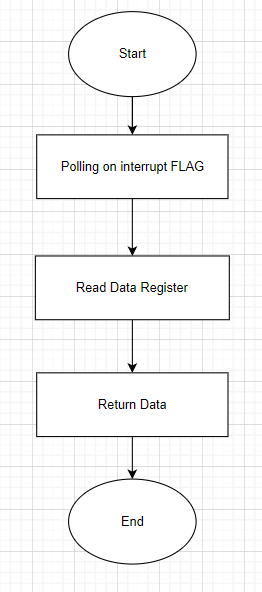


## ADC

ADC\_init()

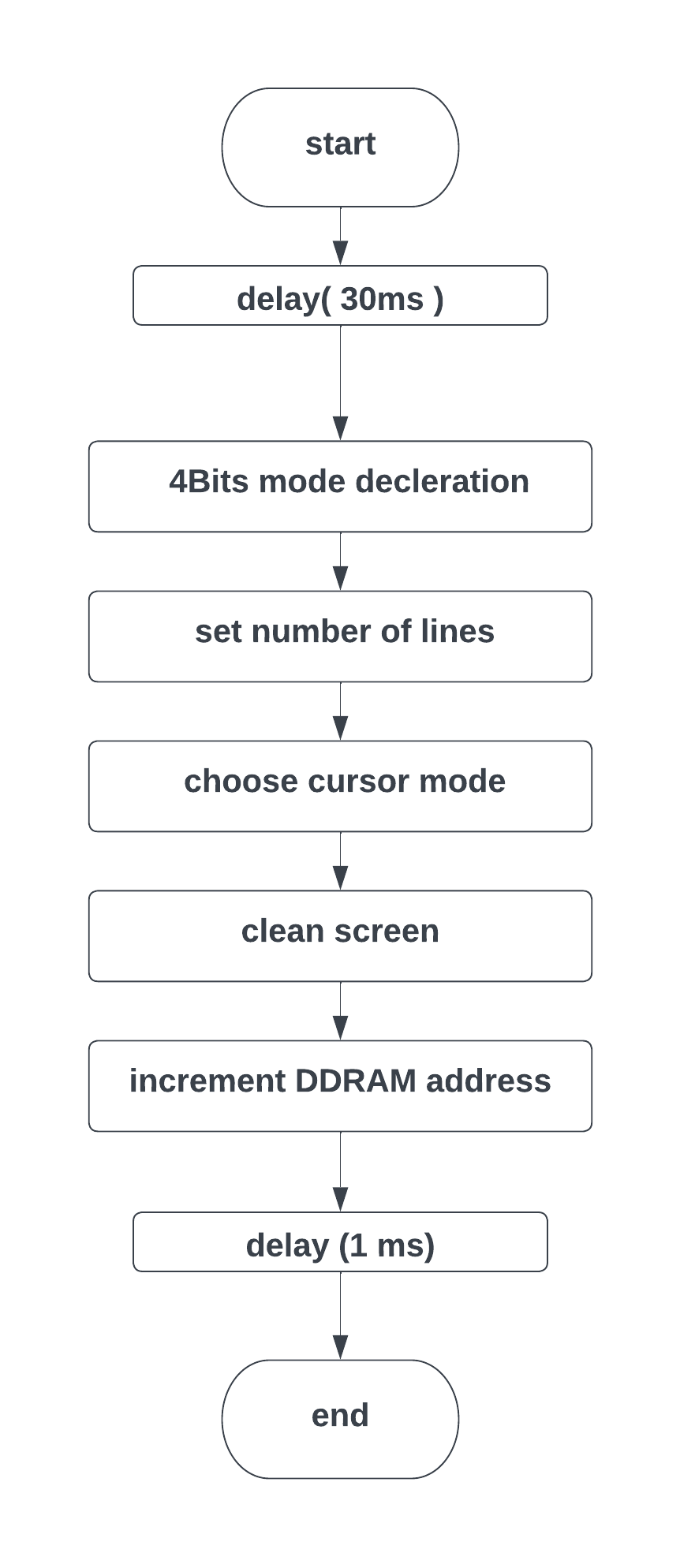


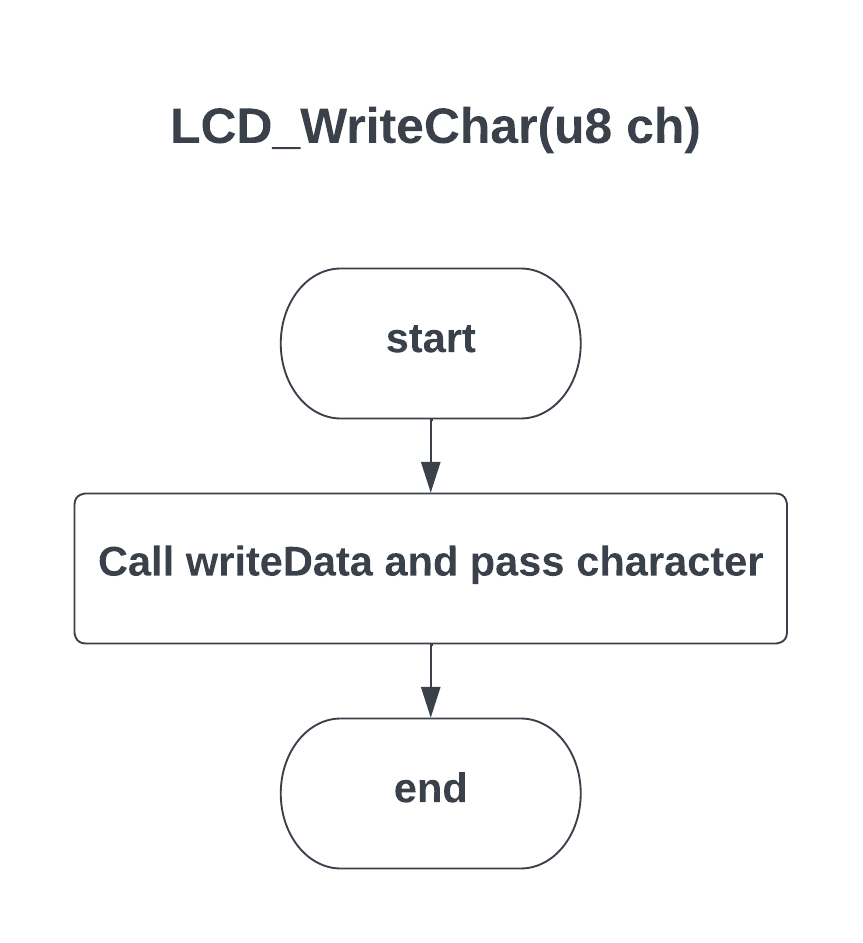
u16 ADC\_read()

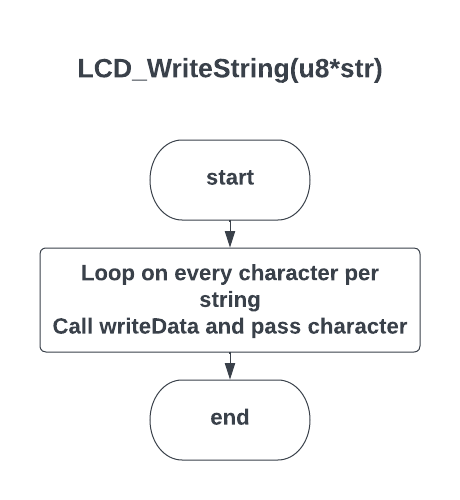


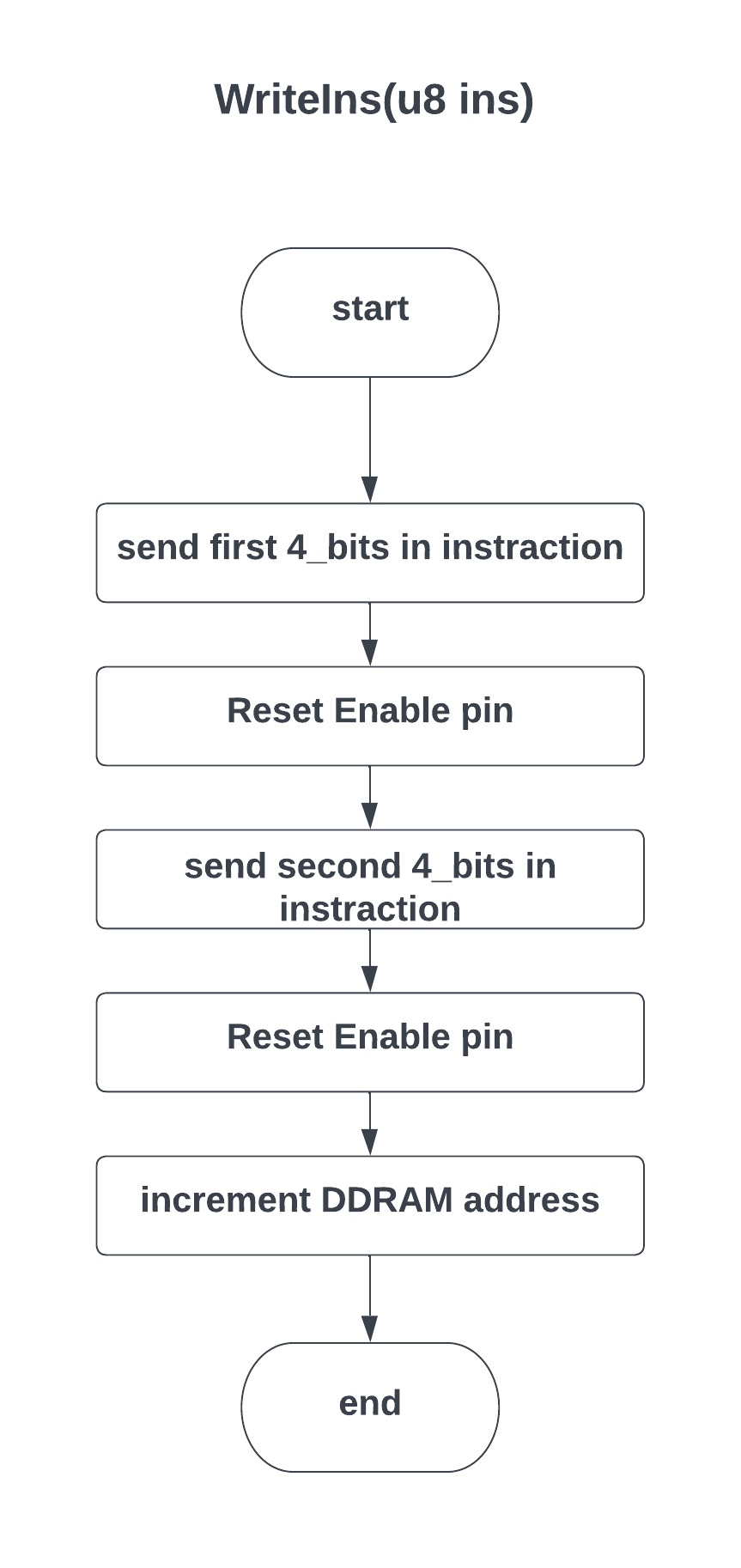
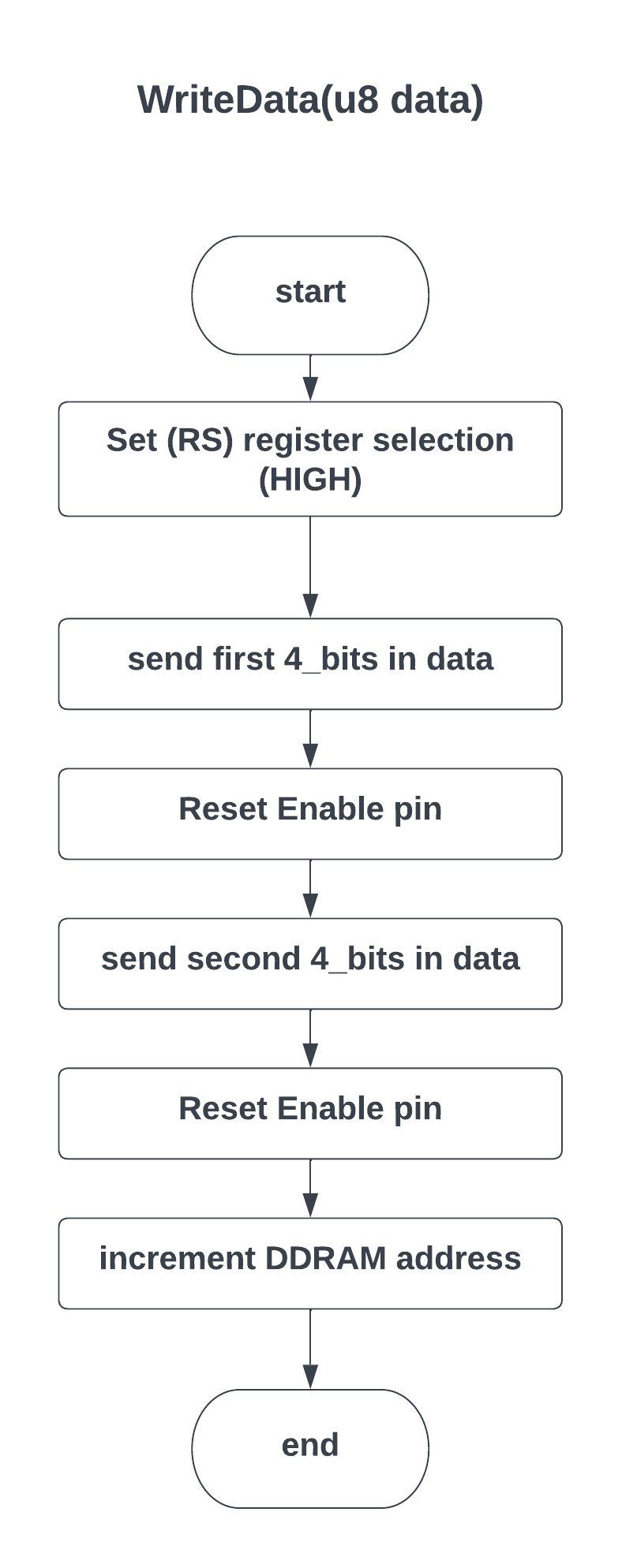
## LCD

LCD\_Init()

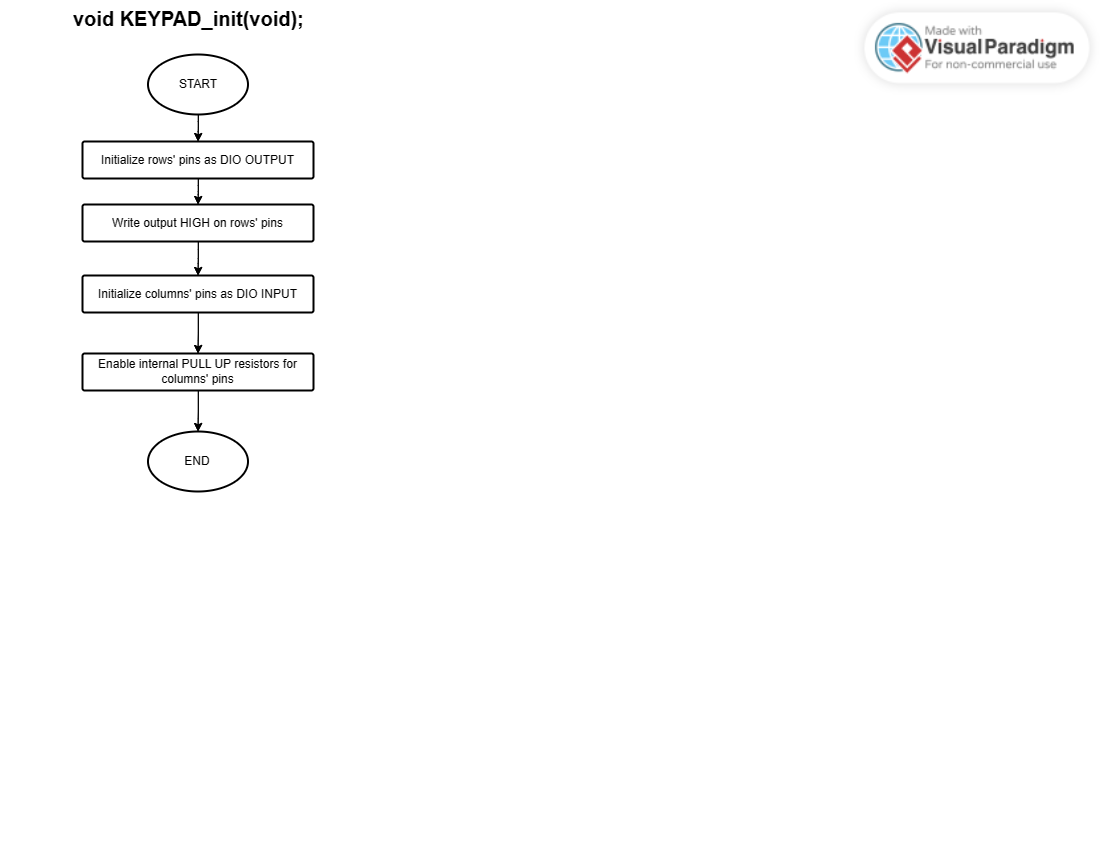


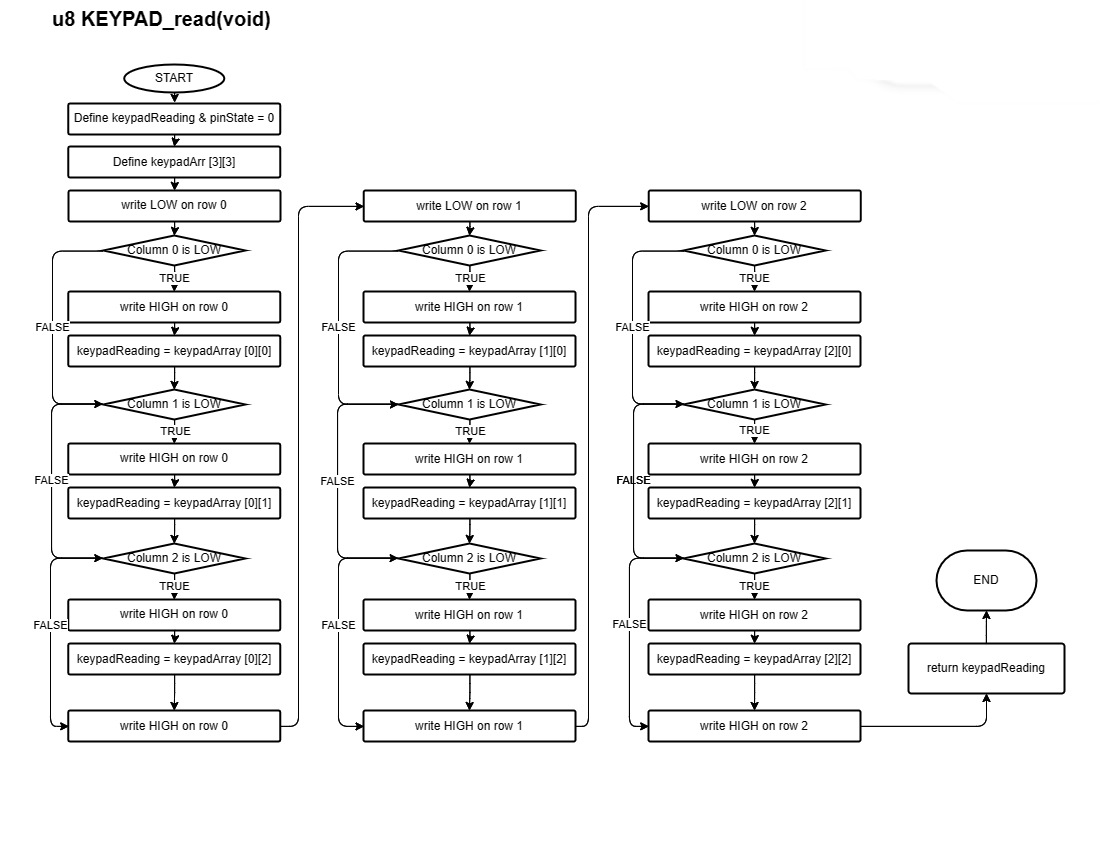




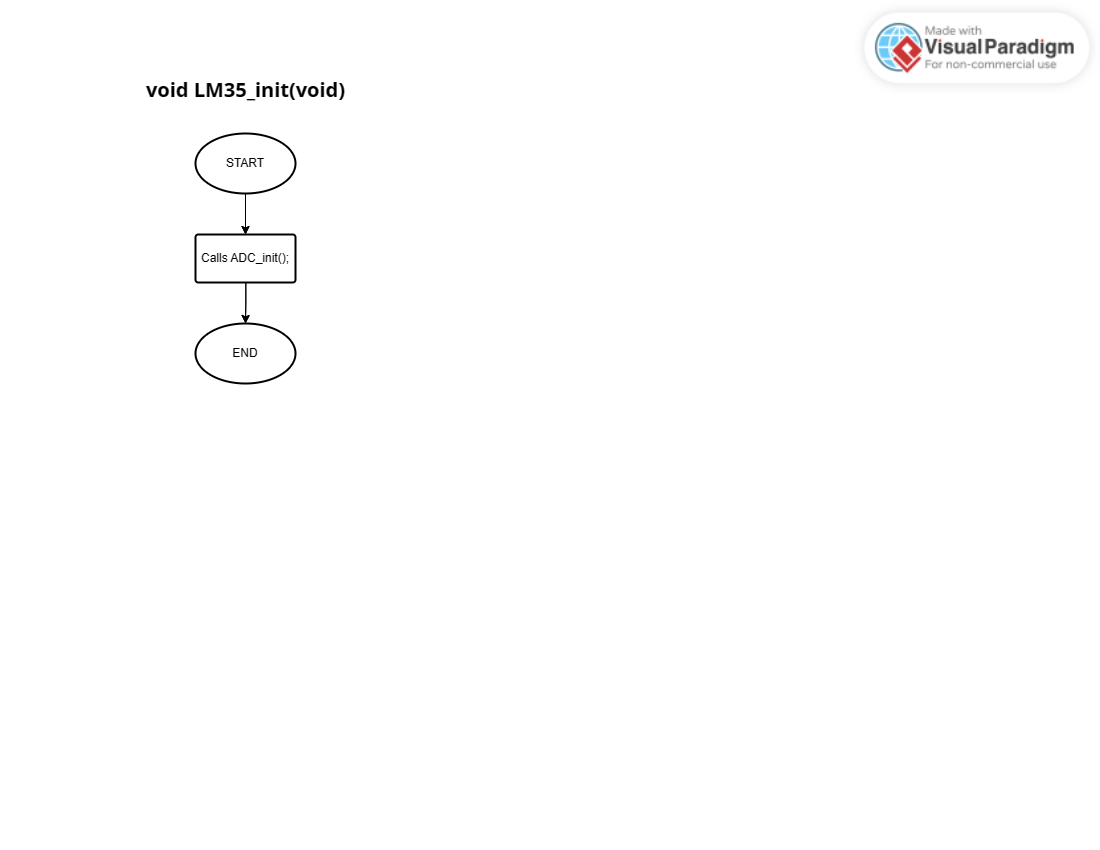


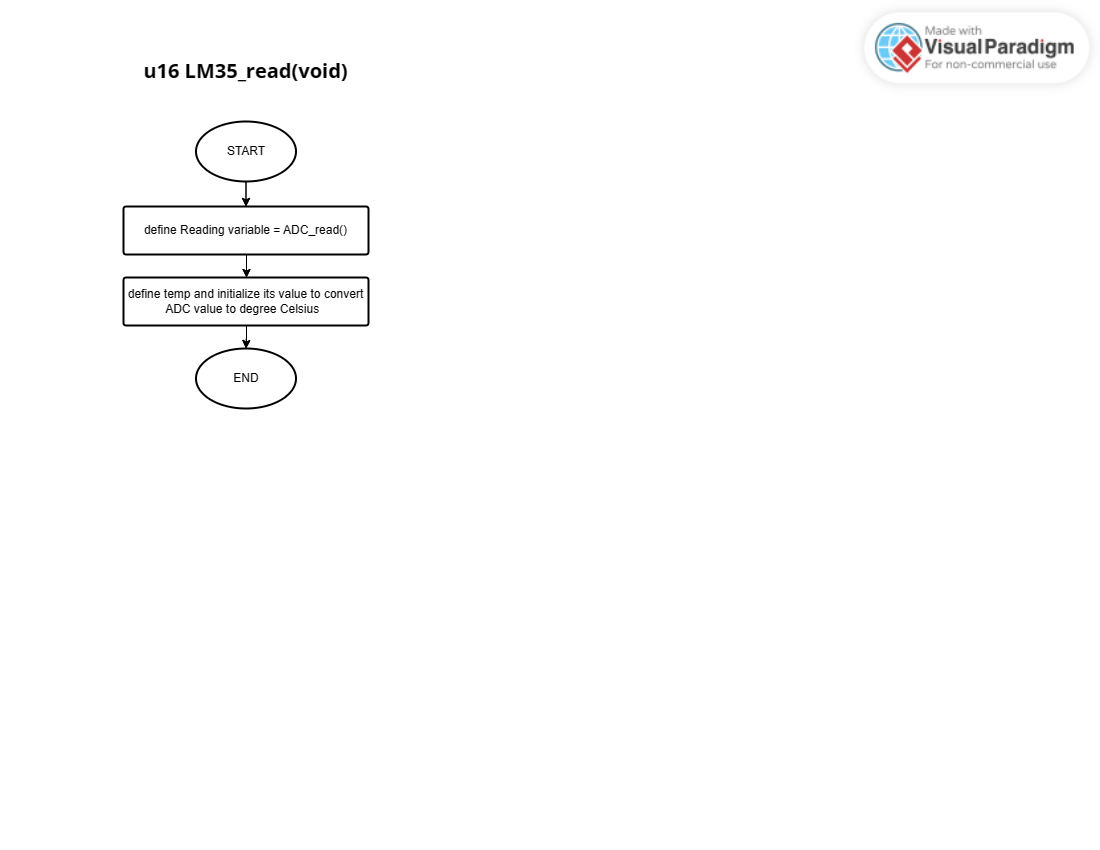
## KEYPAD





## LM35

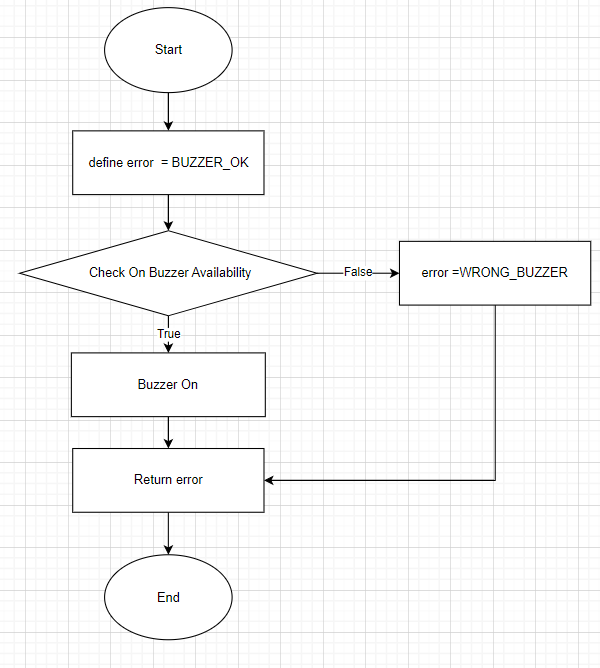


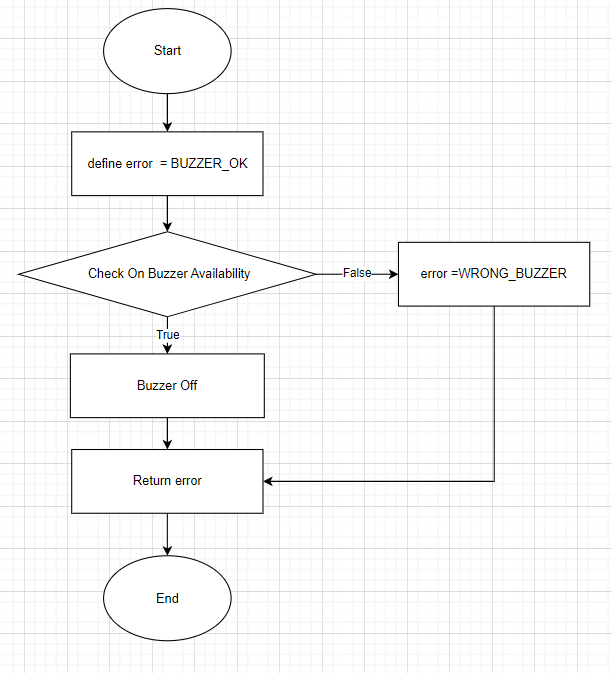


## BUZZER

BUZZER\_init()

BUZZER\_on



BUZZER\_OFF