Obstacle avoidance robot documentation

Written by: Khaled Mustafa Anwar

Table of Contents

Obstacle avoidance robot

1 Project Introduction

1.1 Hardware components

Hardware components name	Number of items required
ATMega32 micro-controller	1 Unit
DC motor	4 Units
Push button	1 Unit
Keypad	1 Unit
Ultrasonic sensor	1 Unit
LCD	1 Unit

1.2 System requirements

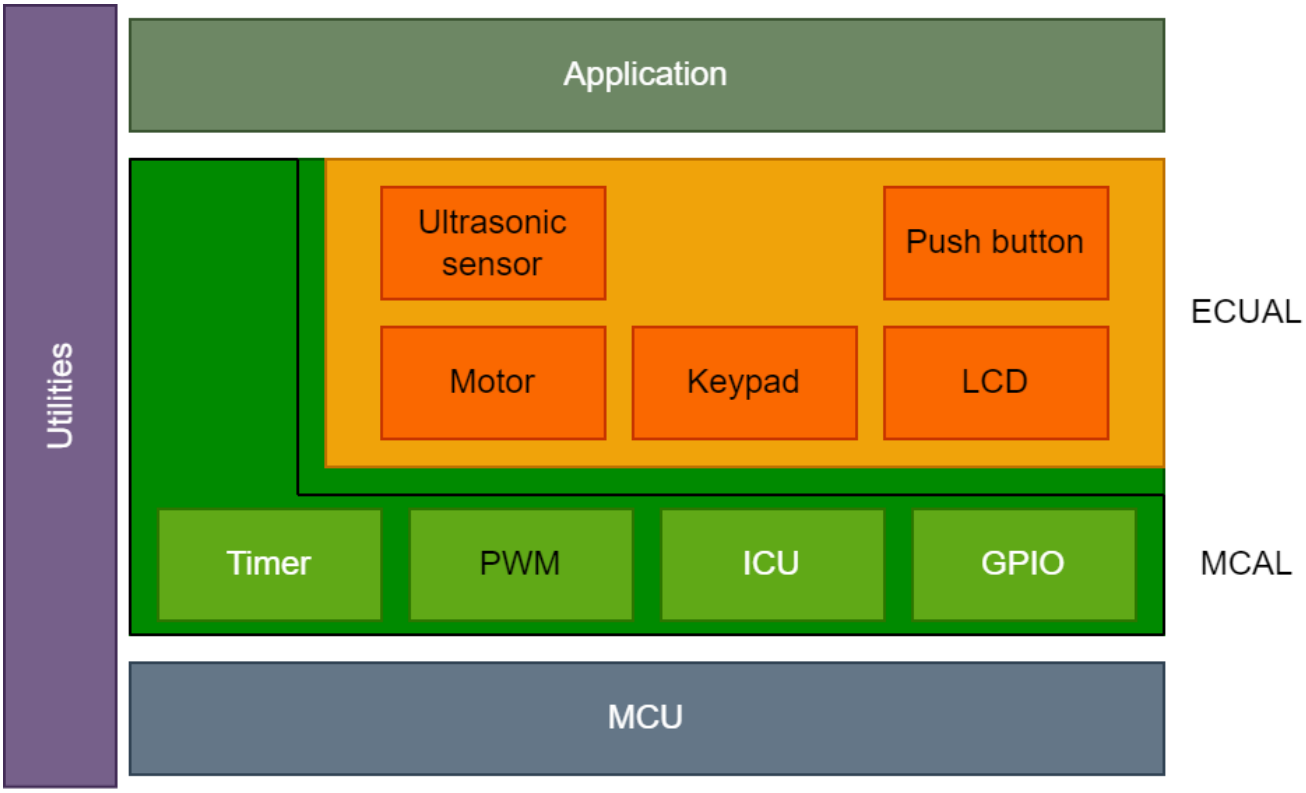
1. The car starts initially from 0 speed.
2. The default rotation direction is to the ***right***.
3. Press (***Keypad button 1***), (***Keypad button 2***) to start or stop the robot respectively.
4. After Pressing Start:
    1. The LCD will display a centered message in line 1 `“Set Def. Rot.”`
    2. The LCD will display the selected option in line 2 `“Right”`
    3. The robot will wait for ***5 seconds*** to choose between Right and Left:
        1. When ***PBUTTON0*** is pressed ***once***, the default rotation will be *Left* and the LCD line 2 will be updated.
        2. When ***PBUTTON0*** is pressed again, the default rotation will be *Right* and the LCD line 2 will be updated.
        3. For each press the default rotation will changed and the LCD line 2 is updated.
        4. After the 5 seconds the default value of rotation is set.
    4. The robot will move after 2 seconds from setting the default direction of rotation.
5. For No obstacles or object is far than 70 cm:
    1. The robot will move forward with 30% speed for 5 seconds
    2. After 5 seconds it will move with 50% speed as long as there was no object or objects are located at more than 70 centimeters distance
    3. The LCD will display the speed and moving direction in line 1:
       `“Speed:00% Dir: F/B/R/S”`
       where **F**: forward, **B**: Backwards, **R**: Rotating, and **S**: Stopped
    4. The LCD will display Object distance in line 2 “`Dist.: 000 Cm”`

6. Obstacles located between 30 and 70 centimeters
    1. The robot will decrease its speed to 30%
    2. LCD data is updated
7. For Obstacles located between 20 and 30 centimeters:
    1. The robot will stop and rotates 90 degrees to right/left according to the chosen configuration.
    2. The LCD data is updated.
8. For Obstacles located less than 20 centimeters:
    1. The robot will stop, move backwards with 30% speed until distance is greater than 20 and less than 30.
    2. The LCD data is updated.
    3. Then preform *point 8.*

1.3 Systems requirements flowchart

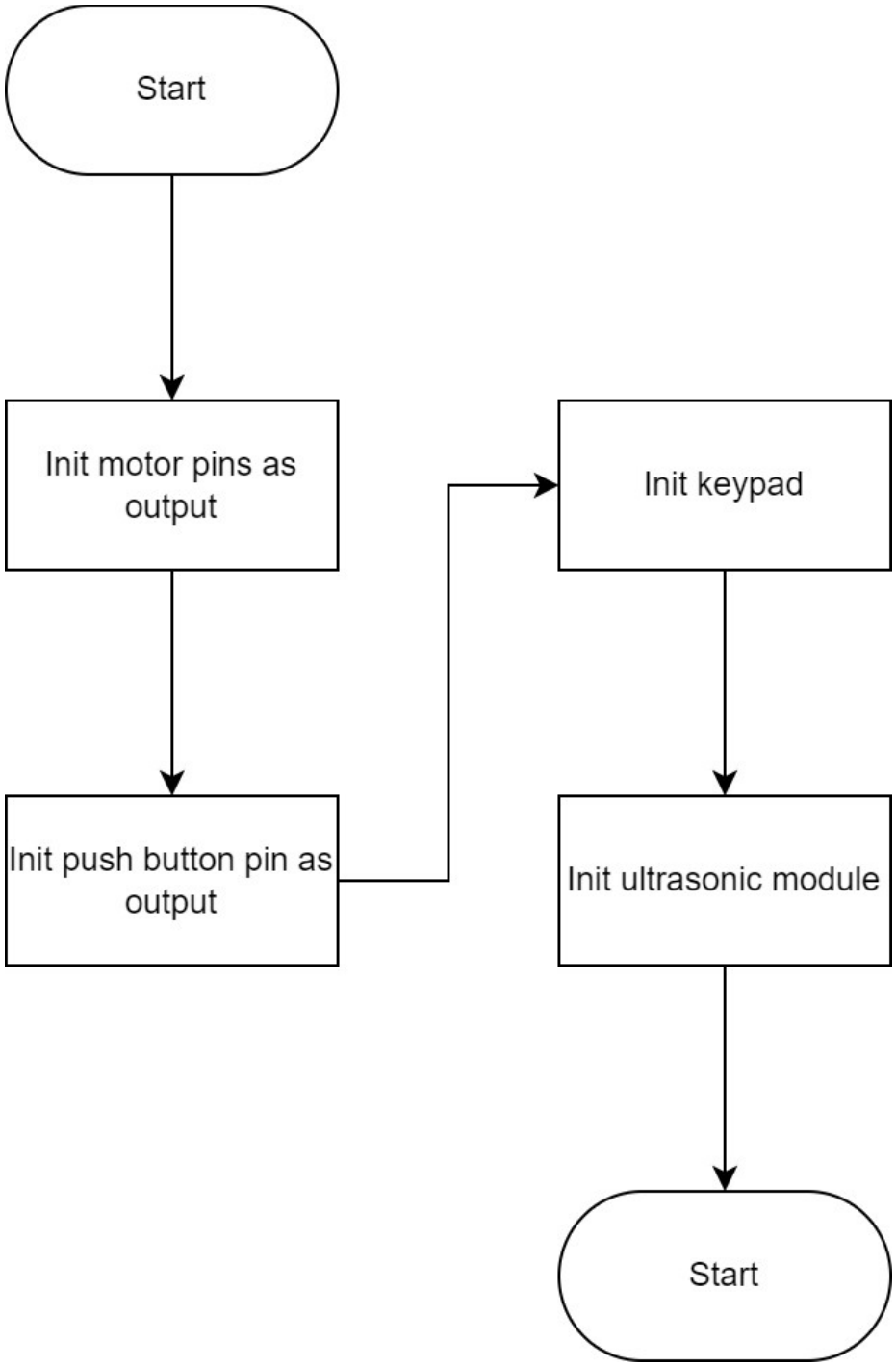
2 High level design

2.1 Layered architecture

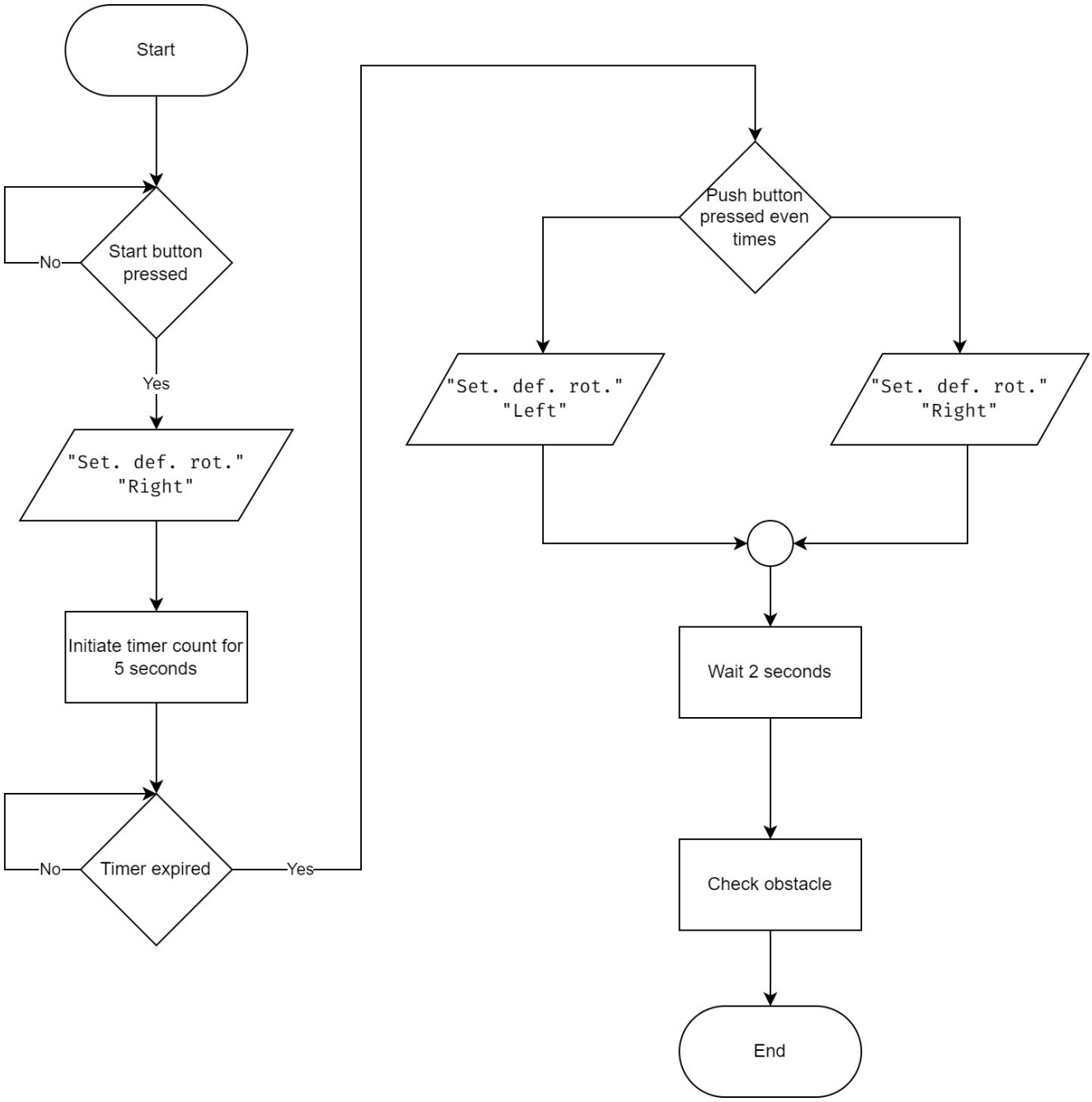


3 System’s flowchart

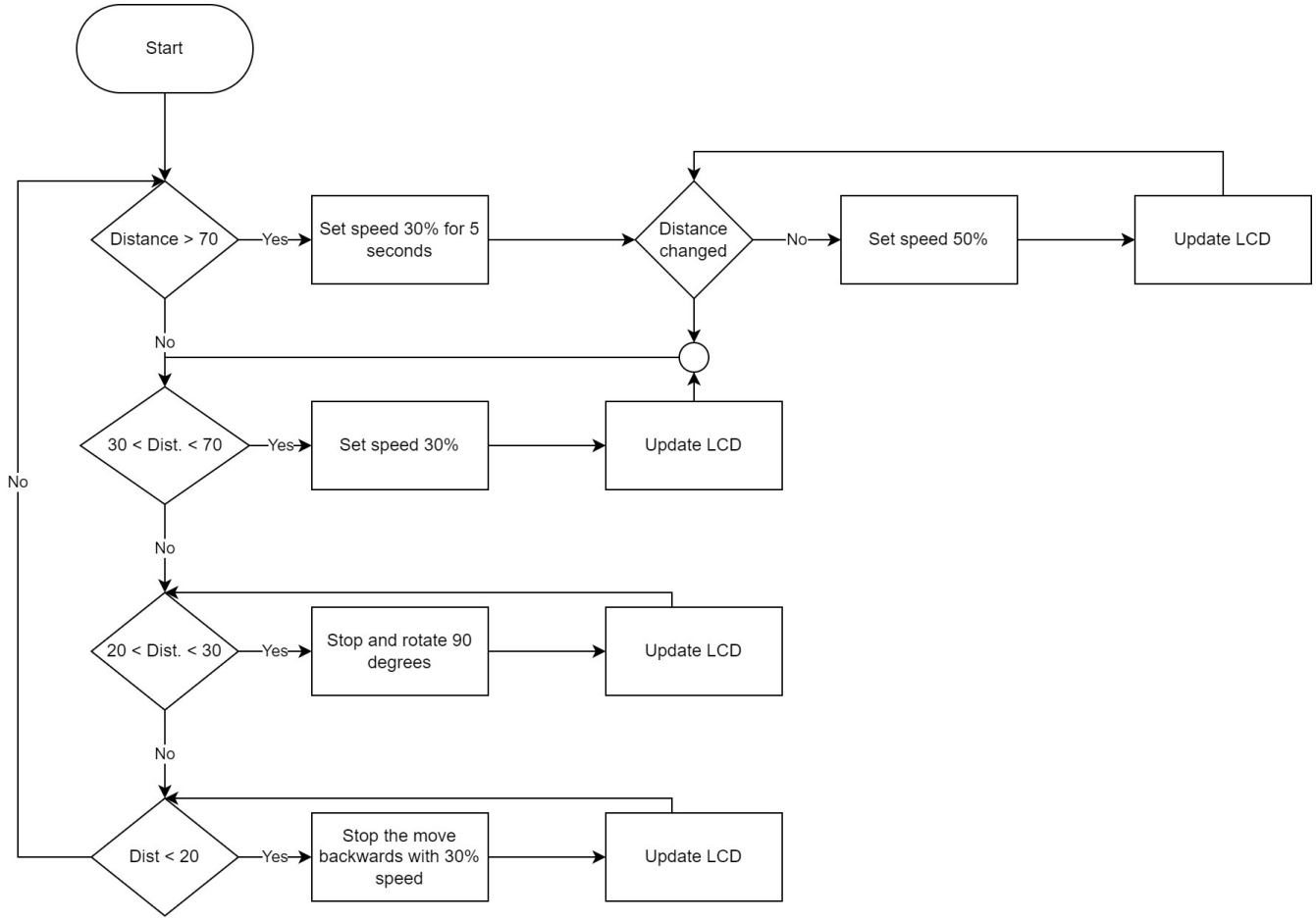
3.1 System initialization



3.2 Warm-up stage



3.3 Obstacle checking



4 Drivers description

4.1 DIO (Digital Input/Output)

This driver performs the following functionalities:

1. Set the pin direction as either input or output.
2. Set the logical state of the pin as either high or low.
3. Read the current value of the pin.
4. Toggle the logical state of the pin.

4.2 Timer

Used to set a time delay or counting events.

4.3 PWM (Pulse Width Modulation)

Used for controlling the speed of the motors.

4.4 Keypad driver

Perform the following:

1. Initialize the keypad.
2. Get the key pressed.

4.5 Push button driver

Perform the following:

1. Initialize the push button.
2. Check the current status of the push button.

4.6 LCD driver

Perform the following:

1. Initialize the LCD module.
2. Sends LCD specific commands to the LCD module.
3. Display a single character.
4. Display a whole string.

4.7 Ultrasonic sensor module

Detecting the distance between the platform and an obstacle in the environment.

4.8 Application

The logic necessary to meet the system’s specifications.

5 Drivers' APIs

5.1 DIO driver

- `en_dioError_t DIO_initpin (DIO_Pin_type pin,DIO_PinStatus_type status);`

Function name	`DIO_initpin`
Description	Sets the pin direction as either input or output.

- `en_dioError_t DIO_writepin (DIO_Pin_type pin,DIO_PinVoltage_type volt);`

Function name	`DIO_writepin`
Description	Set the logical state of the pin as either high or low.

- `en_dioError_t DIO_readpin (DIO_Pin_type pin,DIO_PinVoltage_type *volt);`

Function name	`DIO_readpin`
Description	Read the current value of the pin.

- `en_dioError_t DIO_togglepin(DIO_Pin_type pin);`

Function name	`DIO_togglepin`
Description	Toggle the logical state of the pin.

- `en_dioError_t DIO_WritePort (DIO_Port_type port,u8 value);`

Function name	`DIO_WritePort`
Description	Write a logical level as either high or low to the whole port

5.2 Timer

- `void Timer0_Init(TIMER0_Config *TIMER0_UserConfig);`

Function name	`Timer0_Init`
Description	Initialize Timer0 with the user configuration and starts the timer.

- `void Timer0_DeInit(void);`

Function name	`Timer0_DeInit`
Description	Disables Timer0 by disengage of the clock source.

- `void Timer0_SetValue(uint8 Timer0_InitValue);`

Function name	`Timer0_SetValue`
Description	Sets an initial value for Timer0 to start counting from.

- `TIMER0_Status Timer0_GetStatus(void);`

Function name	`Timer0_GetStatus`
Description	Gets the overflow and compare status of Timer0.

- `void Timer0_SetDelay(uint32 delay_ms);`

Function name	`Timer0_SetDelay`
Description	Sets a delay assuming that the prescaler value is 1024.

- `void Timer0_Int_Callback(void(*p_func)(void));`

Function name	`Timer0_Int_Callback`
Description	Sets the call-back function when Timer0 triggers an interrupt.

5.3 Keypad driver

- `void KEYPAD_init(void);`

Function name	`KEYPAD_init`
Description	Initializes the direction of the row keypad pins as output pins, and the columns keypad pins as input pins.

- `uint8 KEYPAD_getKeyPressed(void);`

Function name	`KEYPAD_getKeyPressed`
Description	Returns the number of position of the key pressed on the keypad matrix.

5.4 Push button driver

- `en_PB_State PB_init(en_PB_Names pb_name);`

Function name	`PB_init`
Description	Initializes the direction of the PB pins as input pins.

- `en_PB_State PB_status(en_PB_Names pb_name, uint8 *p_pb_status);`

5.5 LCD driver

- `void LCD_init(void);`

Function name	`LCD_init`
Description	Initializes the pins connected to the LCD display, clears the display and sets the cursor to the home position.

- `void LCD_sendCommand(uint8 lcd_command);`

Function name	`LCD_sendCommand`
Description	Sends a command to the LCD display.

- `void LCD_displayCharacter(uint8 character);`

Function name	`LCD_displayCharacter`
Description	Displays the specified character on the LCD display.

- `void LCD_displayString(uint8 *p_string);`

Function name	`LCD_displayString`
Description	Displays a whole string on the LCD.

- `void LCD_moveCursor(uint8 row,  uint8 col);`

Function name	`LCD_moveCursor`
Description	Moves the cursor to a specific row and column on the LCD display.

5.6 Ultrasonic driver

- `void ULTRASONIC_init(st_ULTRASONIC_CONFIG *st_ultrasonic_config);`

Function name	`ULTRASONIC_init`
Description	This function initializes the ultrasonic module.

- `void ULTRASONIC_triggerFire(st_ULTRASONIC_CONFIG *st_ultrasonic_config);`

Function name	`ULTRASONIC_triggerFire`
Description	Fires the the trigger pulse.

- `u32 ULTRASONIC_distCalc(st_ULTRASONIC_CONFIG *st_ultrasonic_config);`

Function name	`ULTRASONIC_distCalc`
Description	Calculates the distance.

5.7 PWM driver

- `void PWM_init();`

Function name	`PWM_init`
Description	Initialize the PWM module.

- `void PWM_startSignal(u16 freq, u8 duty_cycle);`

Function name	`PWM_startSignal`
Description	Starts the PWM signal.

- `void PWM_stopSignal();`

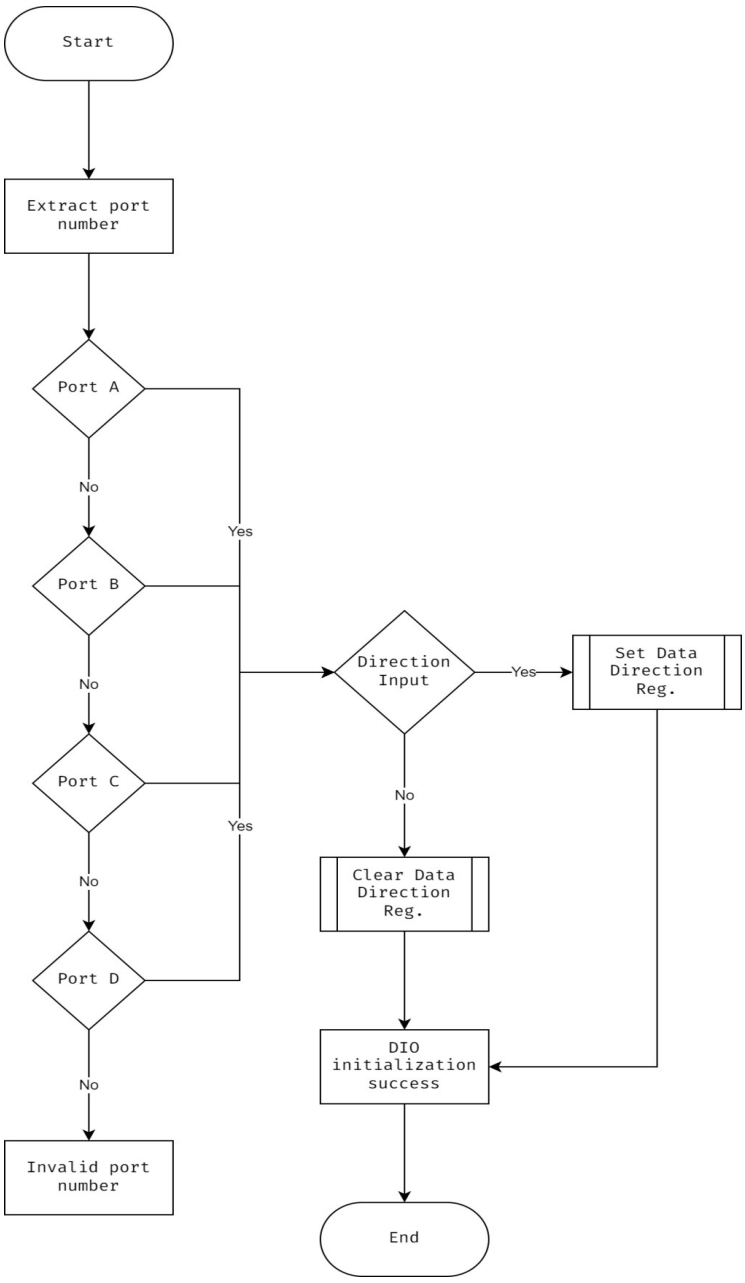
Function name	`PWM_stopSignal`
Description	Stops the PWM signal.

6 Low-level design

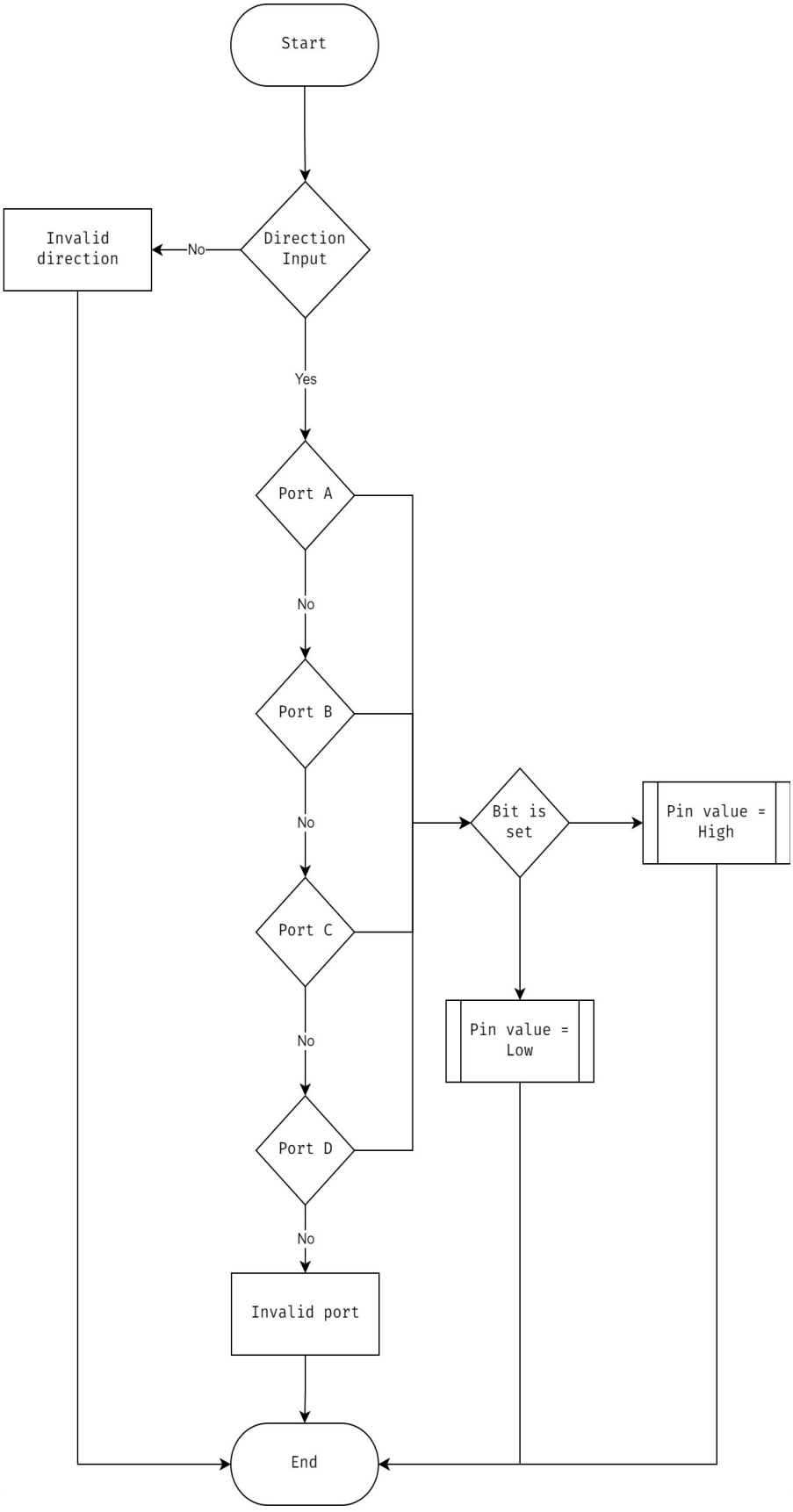
6.1 Drivers API flowchart

6.1.1 DIO API

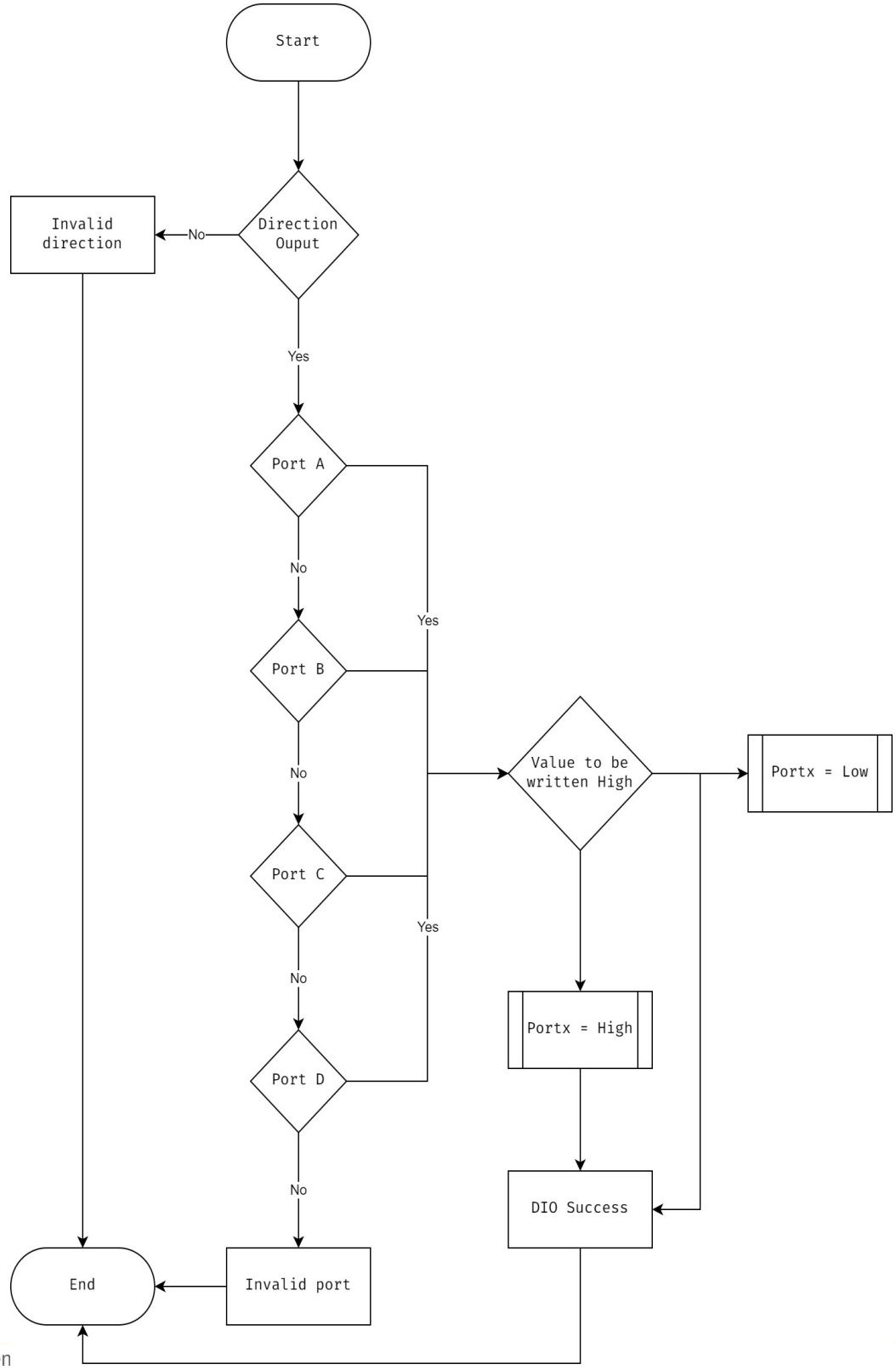
6.1.1.1 DIO Initialize



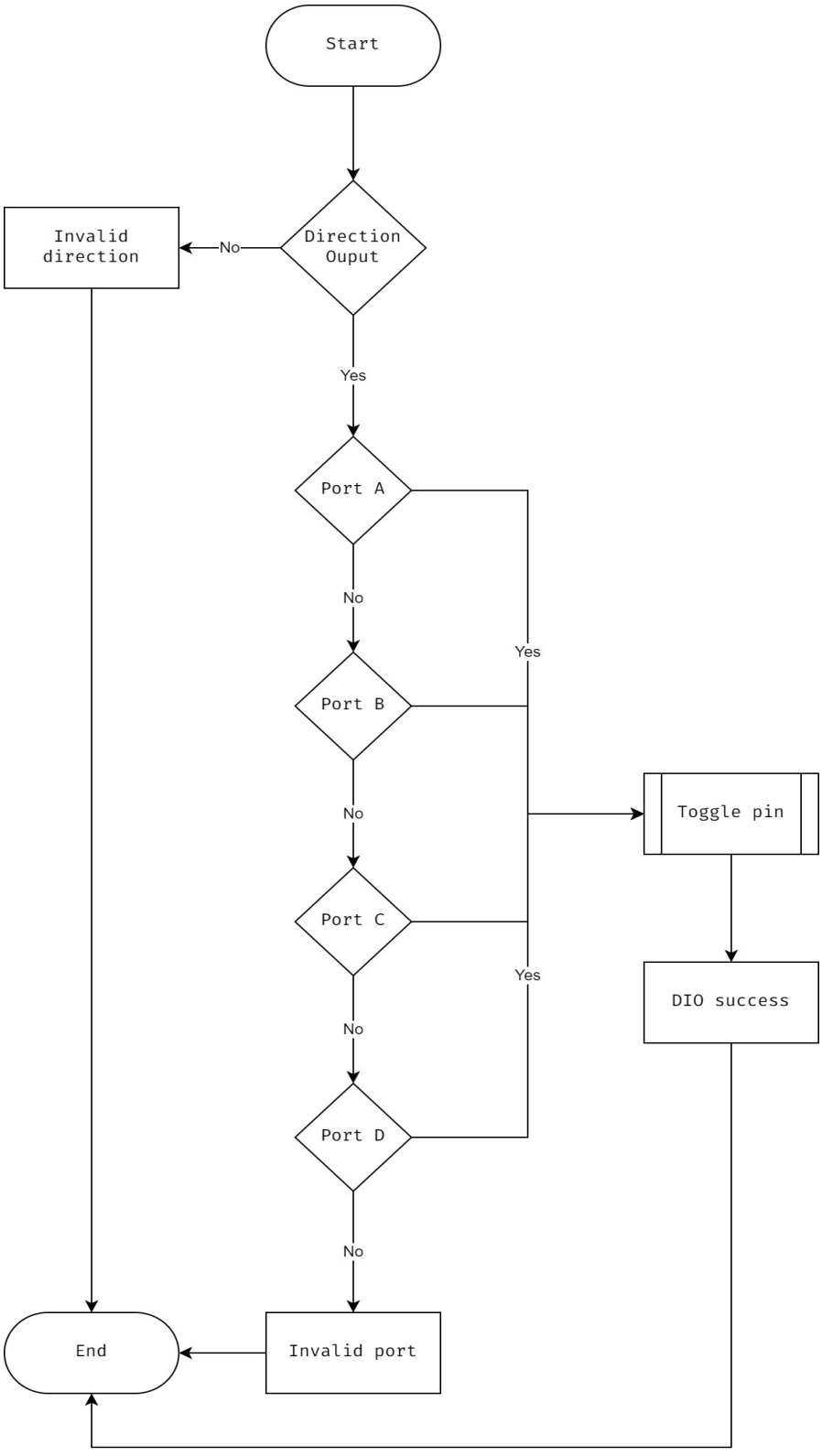
6.1.1.2 DIO Read Pin



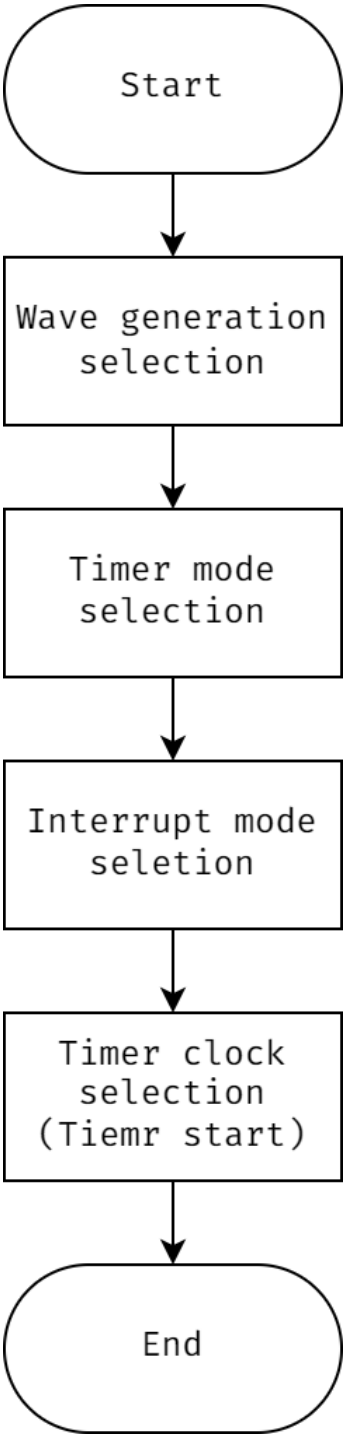
6.1.1.3 DIO Write Pin



6.1.1.4 DIO Toggle pin

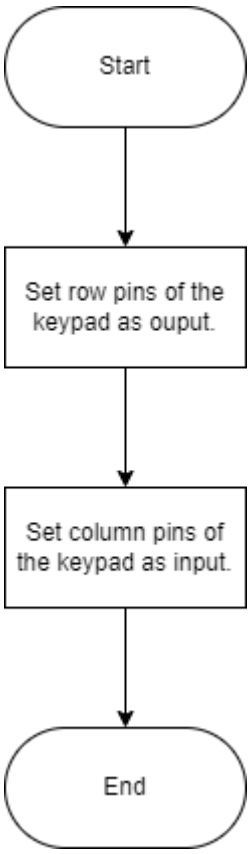


6.1.2 Timer

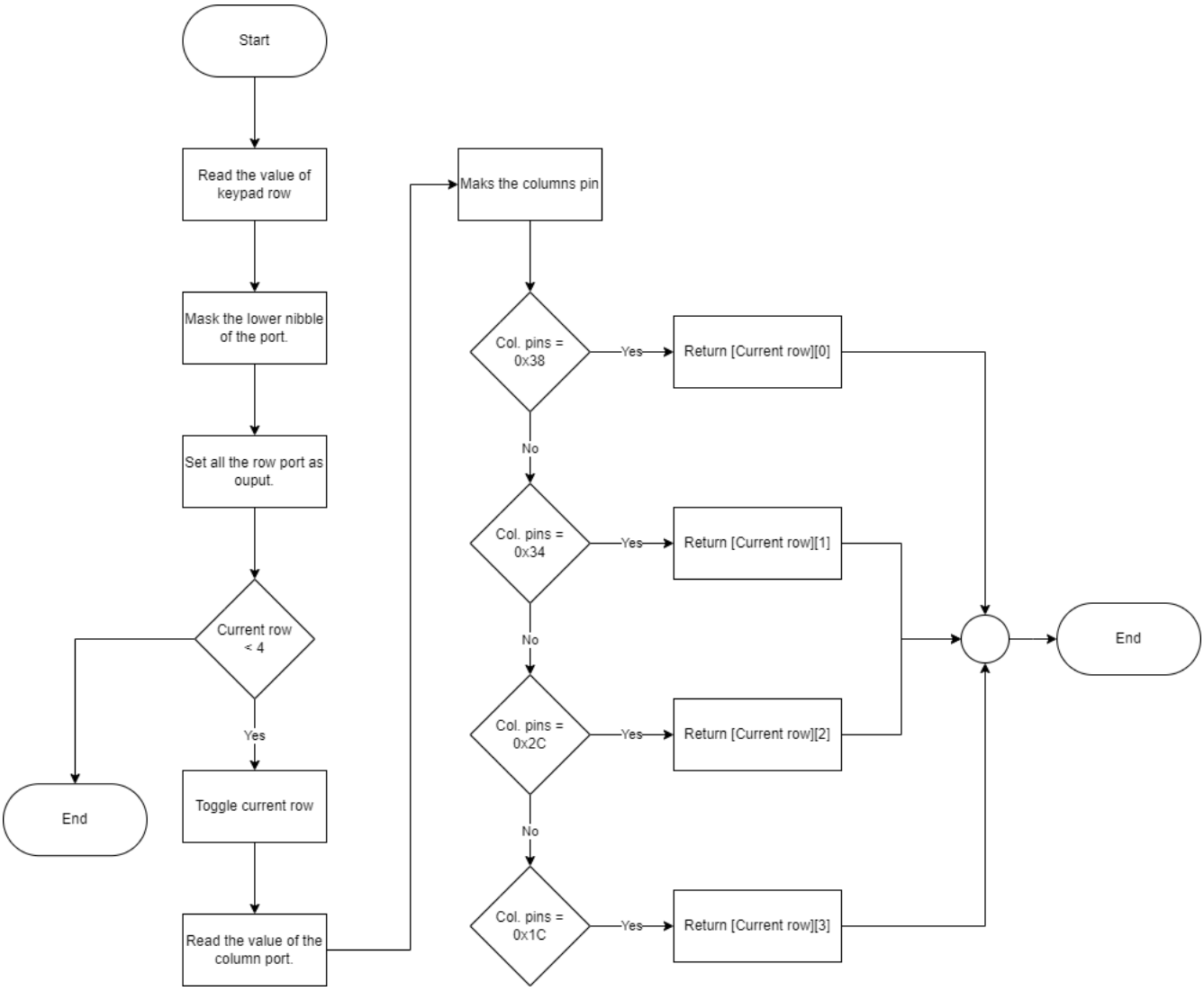

Start
Wave generation selection
Timer mode selection
Interrupt mode seletion
Timer clock selection (Tiemr start)
End

6.1.3 Keypad driver

6.1.3.1 Keypad initialization

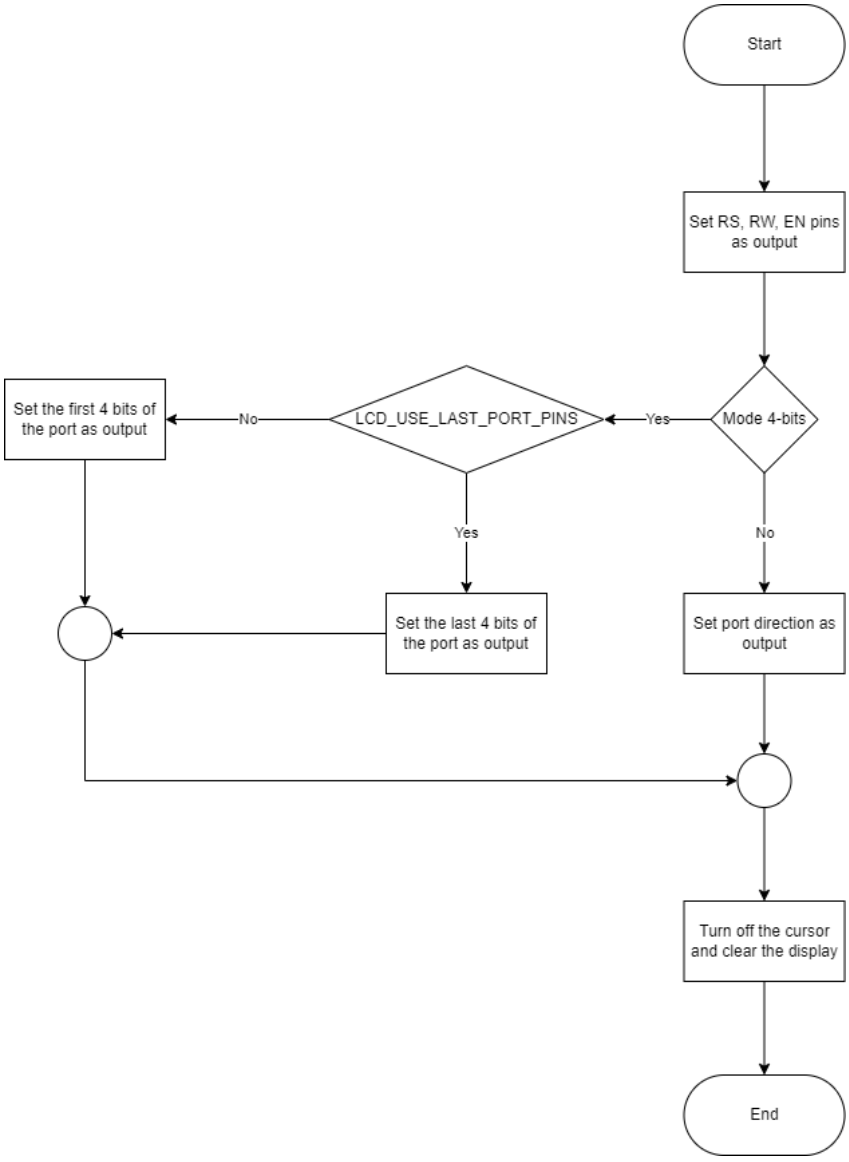


6.1.3.2 Keypad get key pressed

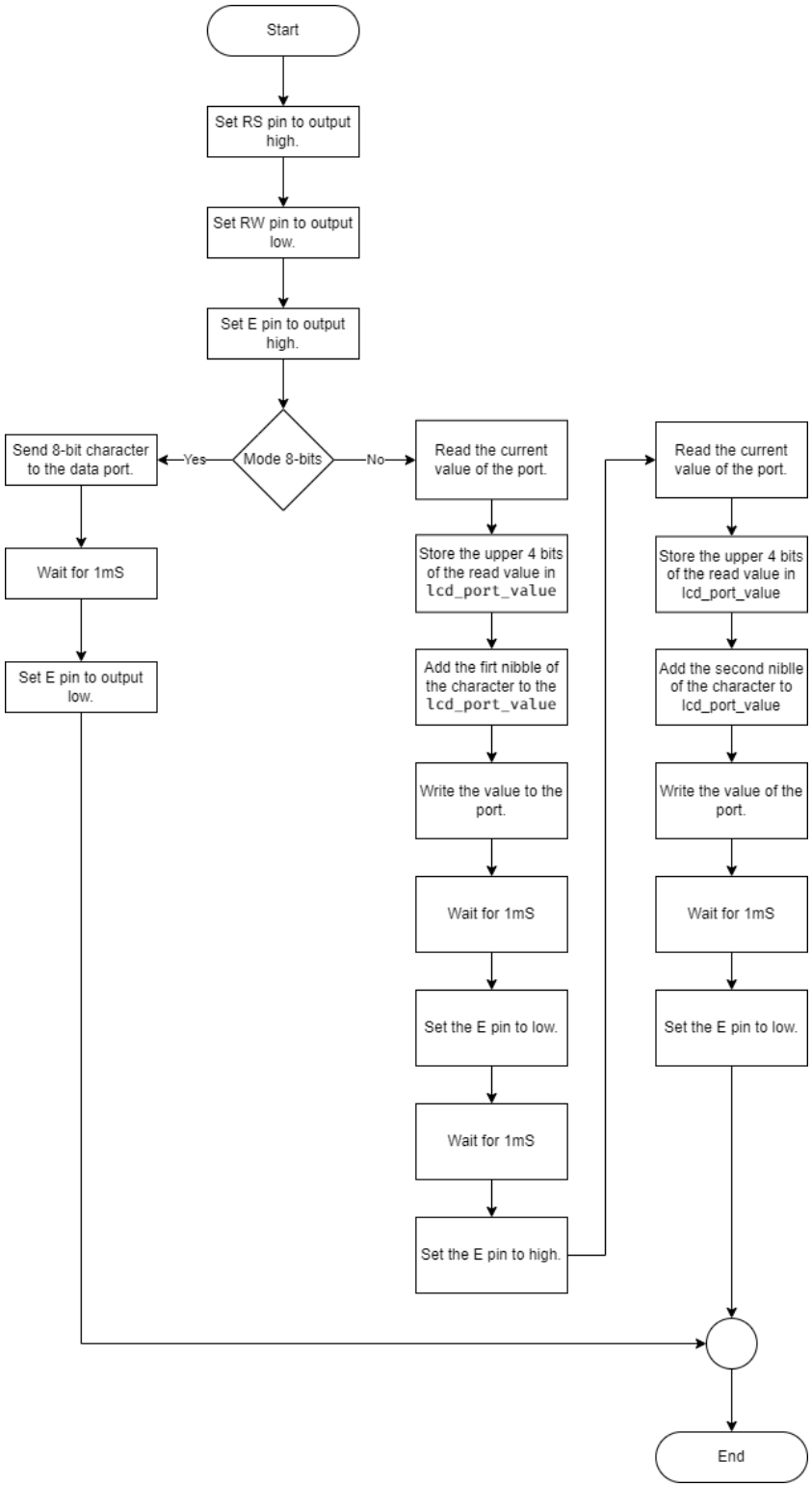


6.1.4 LCD driver

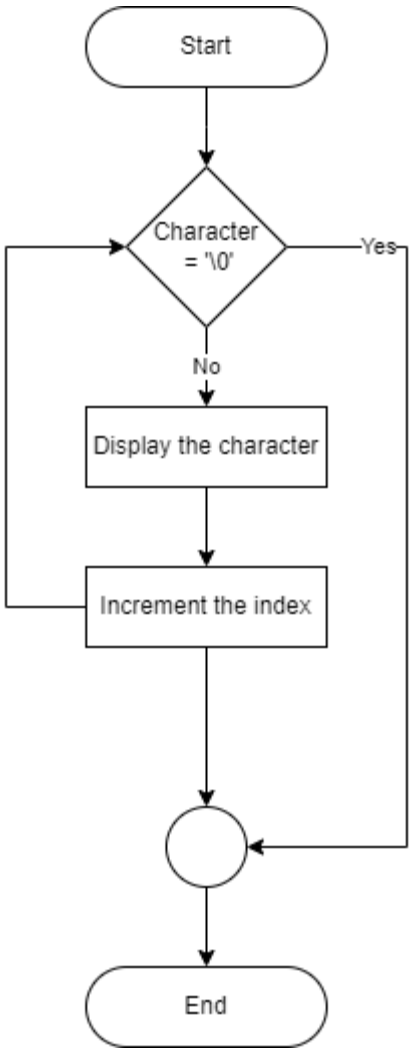
6.1.4.1 LCD initialization



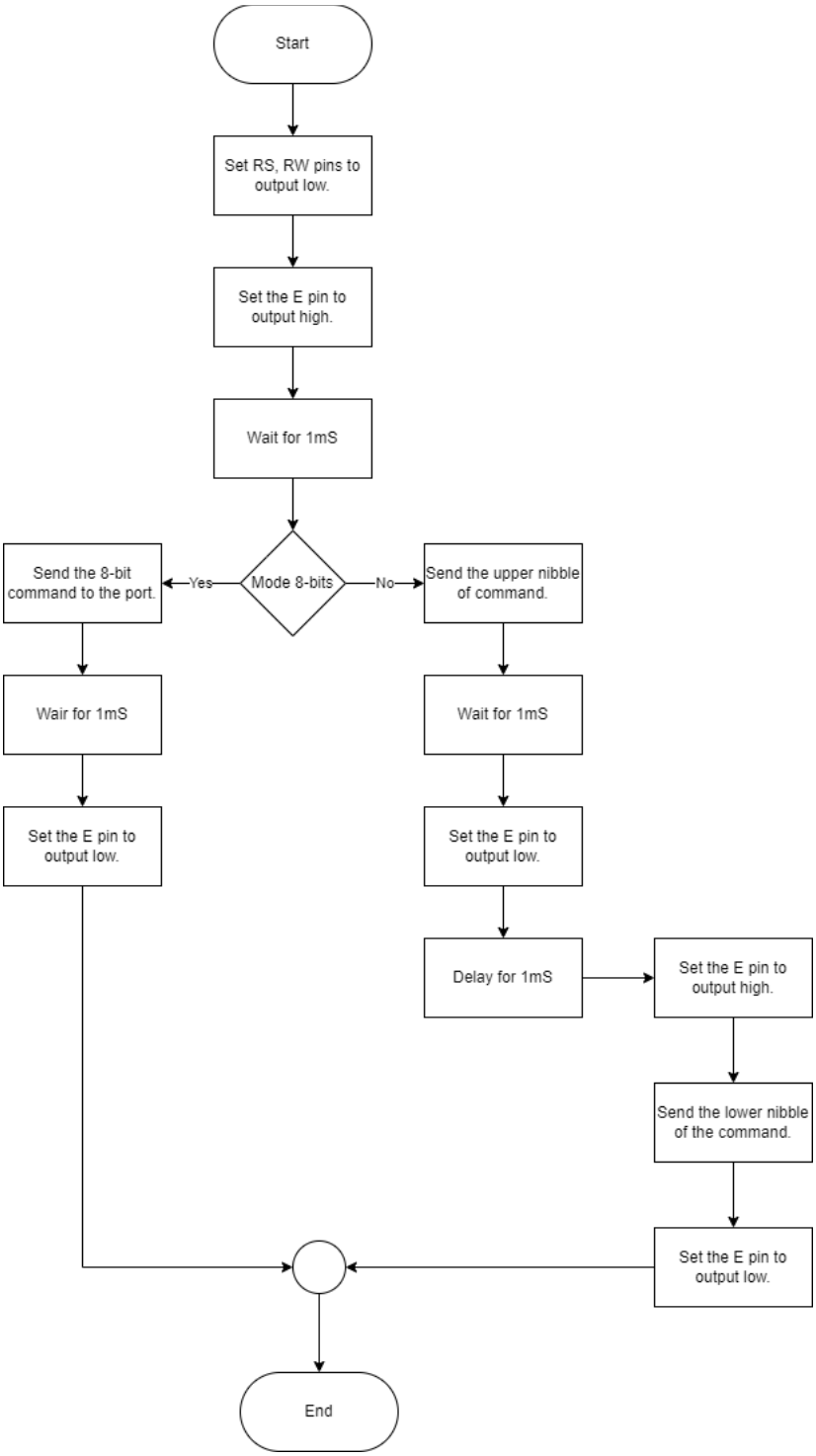
6.1.4.2 LCD display character



6.1.4.3 LCD display string



6.1.4.4 LCD send command



6.1.4.5 LCD move cursor