## EXPERIMENT NO: 1

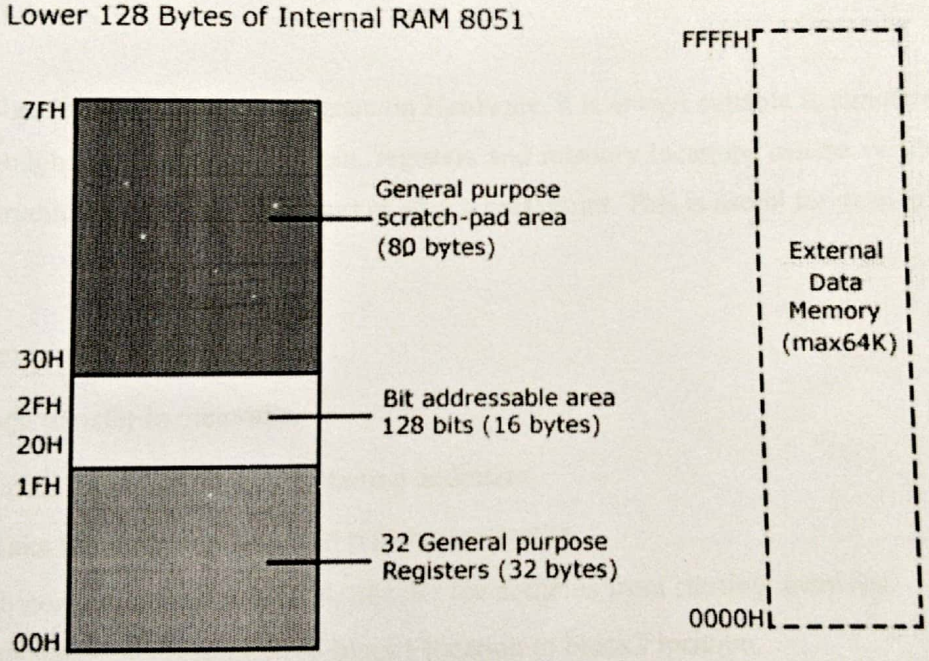
**AIM:-**Simple programmer on Memory transfer.

**APPARATUS:-** PC, Simulation Software.

**THEORY:-**

Today microcontrollers dominated industrial senses &consumer applications. 8951 microcontrollers &other peripherals from Intel are ideal vehicles that are useful to understand microcontroller's architecture & other integrated systems.

The microcontroller 8051 has internal RAM of 128 bytes and external RAM can be of 64 KB. The data can be transferred from internal to external RAM in overlapping and non-overlapping blocks. The microcontroller has 4KB EPROM of flash type so that programs can be erased and programmed again. This is more suitable for program development.




### Program memory organization

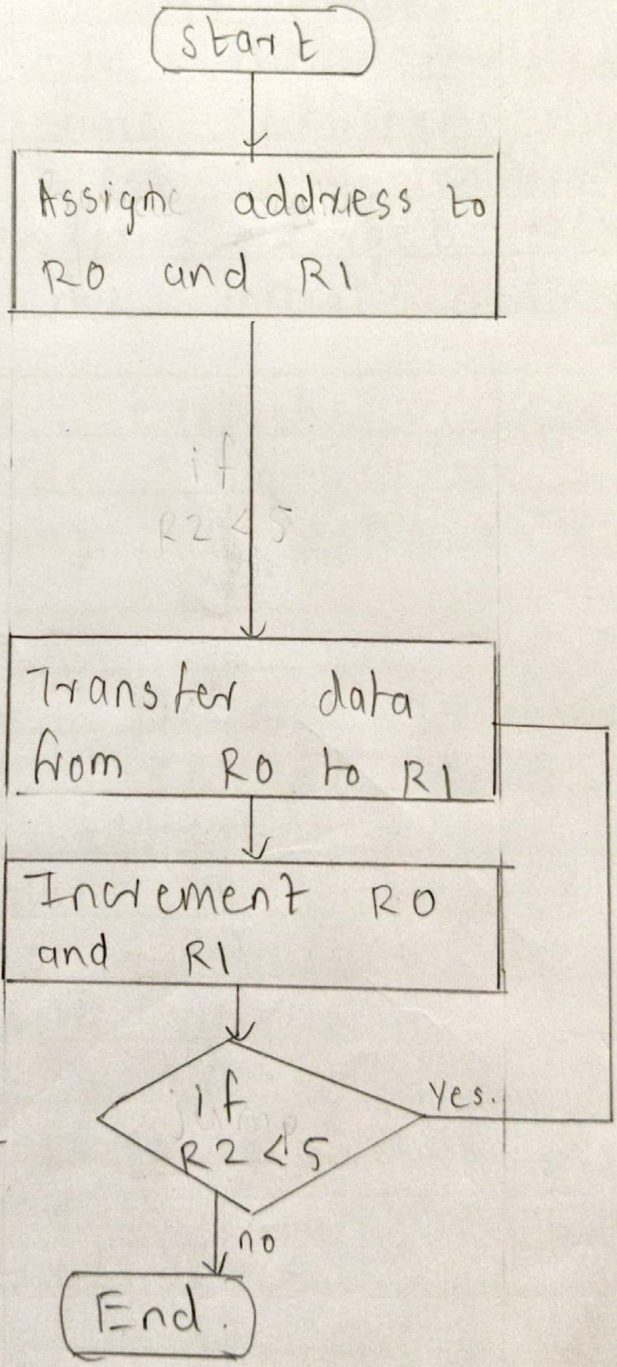
It has an internal program of 4K size and if needed an external memory can be added (by interfacing) of size 60K maximum. So, in total 64K size memory is available for 8051 micro controllers. By default, the External Access (EA) pin should be connected VCC so that instructions are fetched from internal memory initially. If the programmer wants to fetch

**FAQ:**

(i) What are the functions of simulator?

(ii) What are the different addressing modes? Give one example of each.

(iii) Explain the memory map of 8051microcontroller.



## EXPERIMENT NO 2

**AIM:** - Parallel port interfacing of LEDs- Different programs (Flashing, Counter, BCD, Hex display of characteristics)

**APPARATUS:** - 89V51RD2 Development Board, LED Bank

**THEORY:** -

In his experiment we are going to study how to interface a simple LED bank of 8 LED to one of the ports of 8051 microcontroller and write programs to generate different patterns on LED bank such as Flashing of LEDs, Counter, BCD, Hex display of characters.

Light Emitting Diodes are the mostly commonly used components in many applications. They are made of semiconducting material. This article describes basic interfacing of LEDs to the 8051 family microcontrollers.

The main principle of this circuit is to interface LEDs to the 8051-family micro controller. Commonly, used LEDs will have voltage drop of 1.7v and current of 10mA to glow at full intensity. This is applied through the output pin of the micro controller.

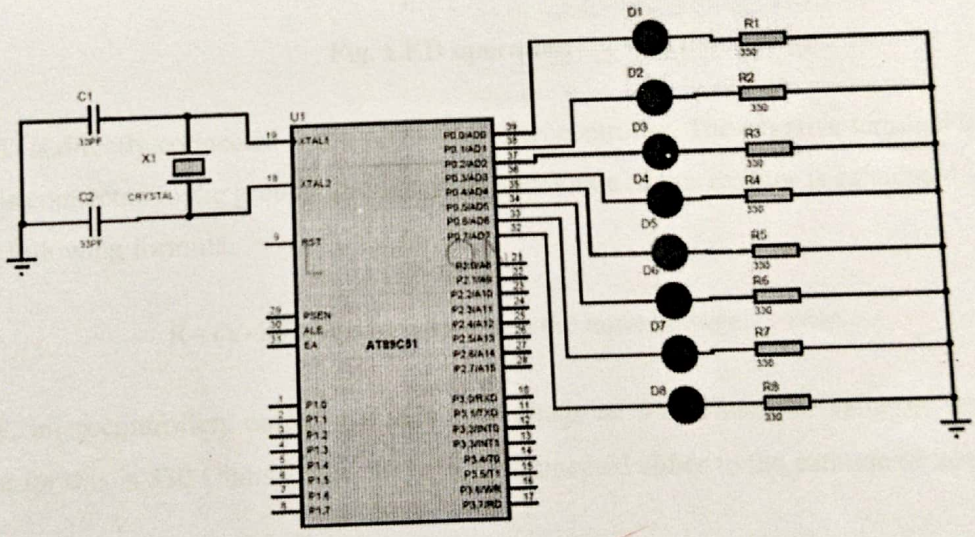
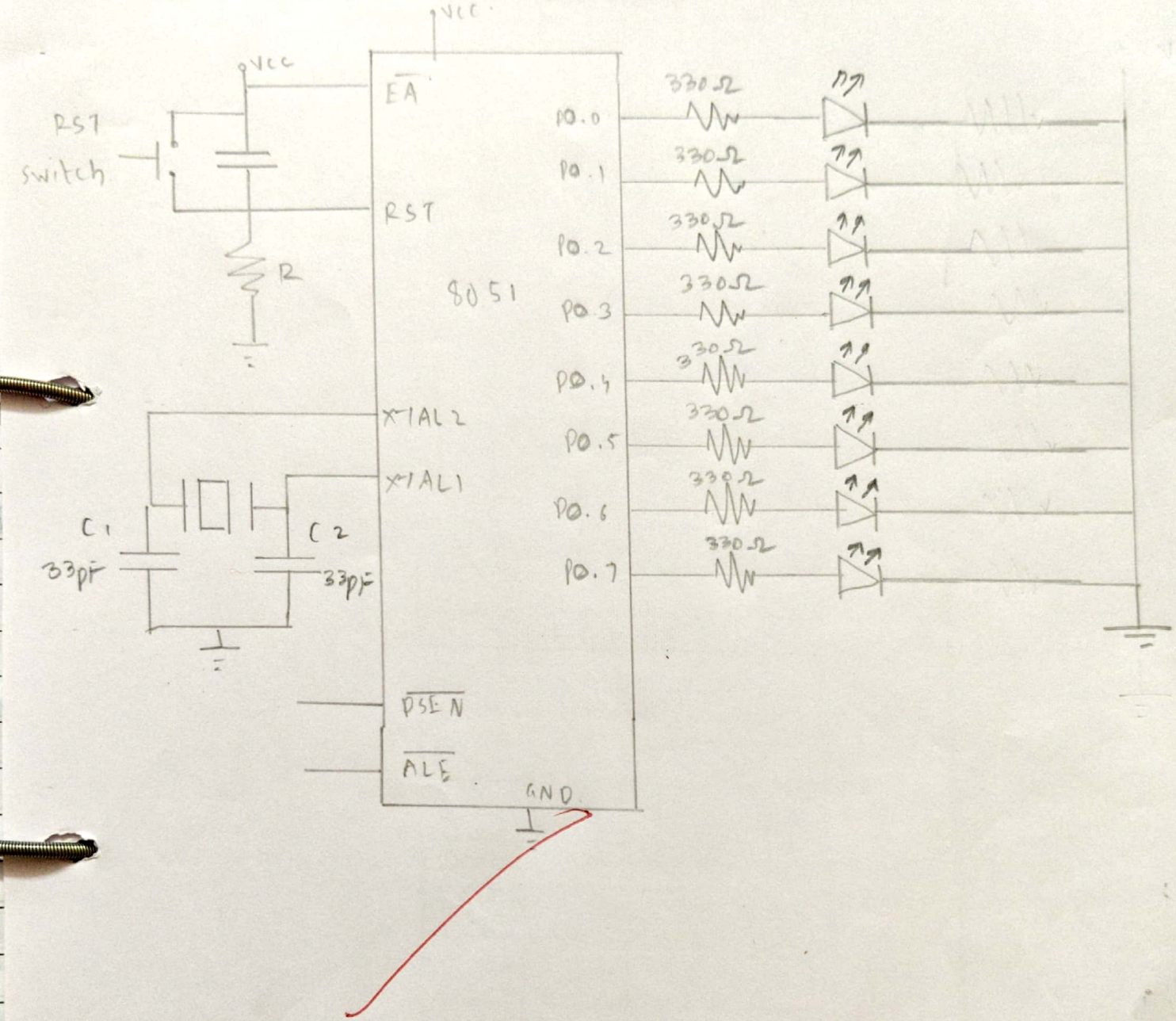


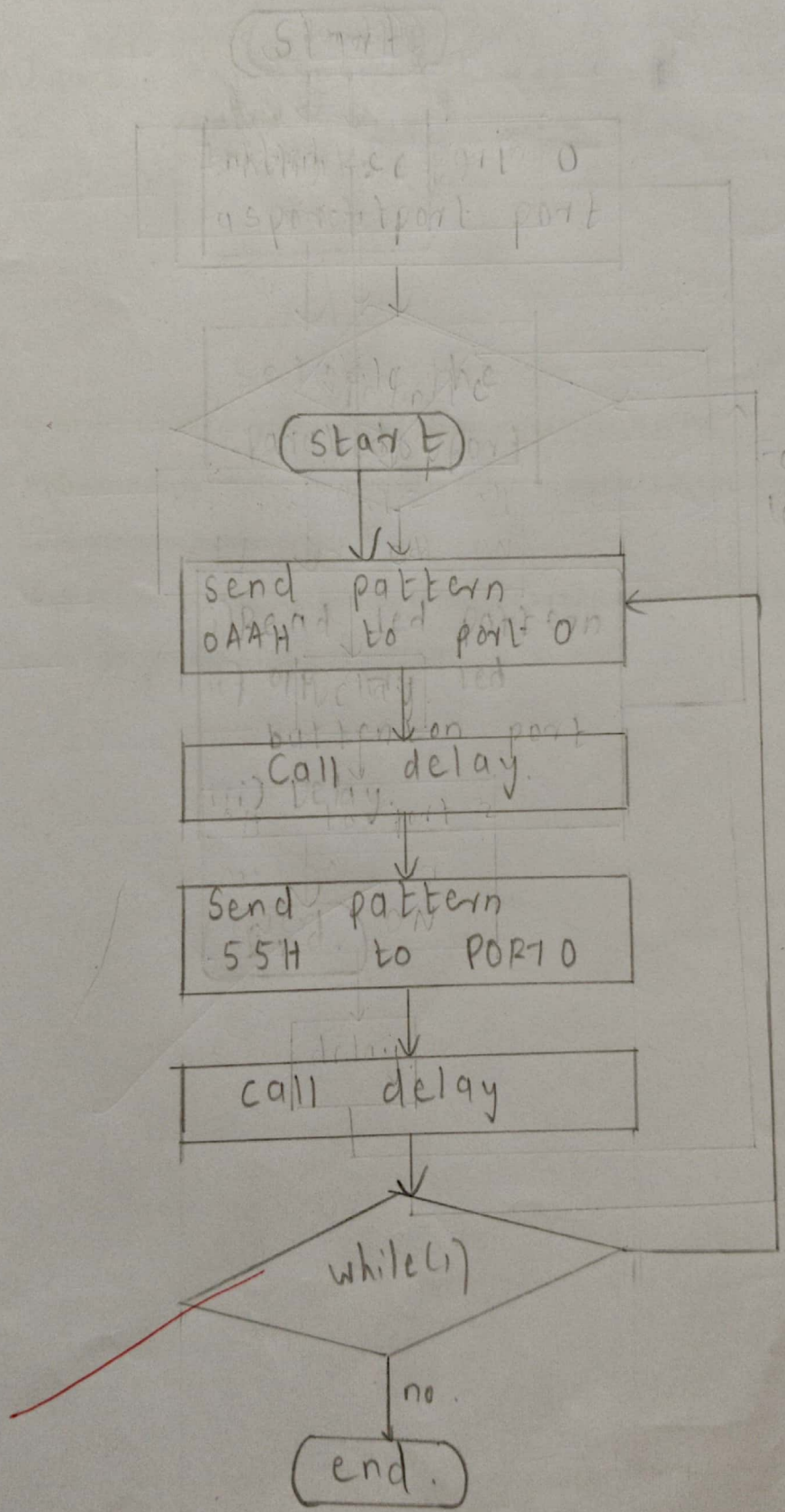

**Fig. Interfacing of LED bank to 8051**

The circuit mainly consists of AT89C51 microcontroller. AT89C51 belongs to the family of 8051 microcontroller. It is an 8-bit microcontroller. This microcontroller has 4KB of Flash Programmable and Erasable Read Only Memory and 128 bytes of RAM. This can be programmed and erased maximum 1000times. It has two 16 bit timers/counters. It supports USART communication protocol. It has 40 pins. There are four ports are designated as P0, P1,

**INTERFACING DIAGRAM:**



**FLOWCHART:**



# EXPERIMENT NO 3

**AIM: -**

Display a digit on seven segment display using 8051 microcontrollers.

**APPARATUS: -** 89V51RD2 Development Board, Seven Segment Display

**THEORY: -**

The 7-segment display consists of seven LEDs arranged in a rectangular fashion. Each of the seven LEDs is called a segment because when illuminated the segment forms part of a numerical digit (both Decimal and Hex) to be displayed. An additional 8th LED is sometimes used within the same package which is the indication of a decimal point (DP), when two or more 7-segment displays are connected numbers greater than ten can be displayed.

So, by forward biasing the appropriate pins of the LED segments in a particular order, some segments will be glowing, and others will remain as it is, allowing the desired character pattern of the number to be generated on the display. This then allows us to display each of the ten decimal digits 0 to 9 on the same 7-segment display.

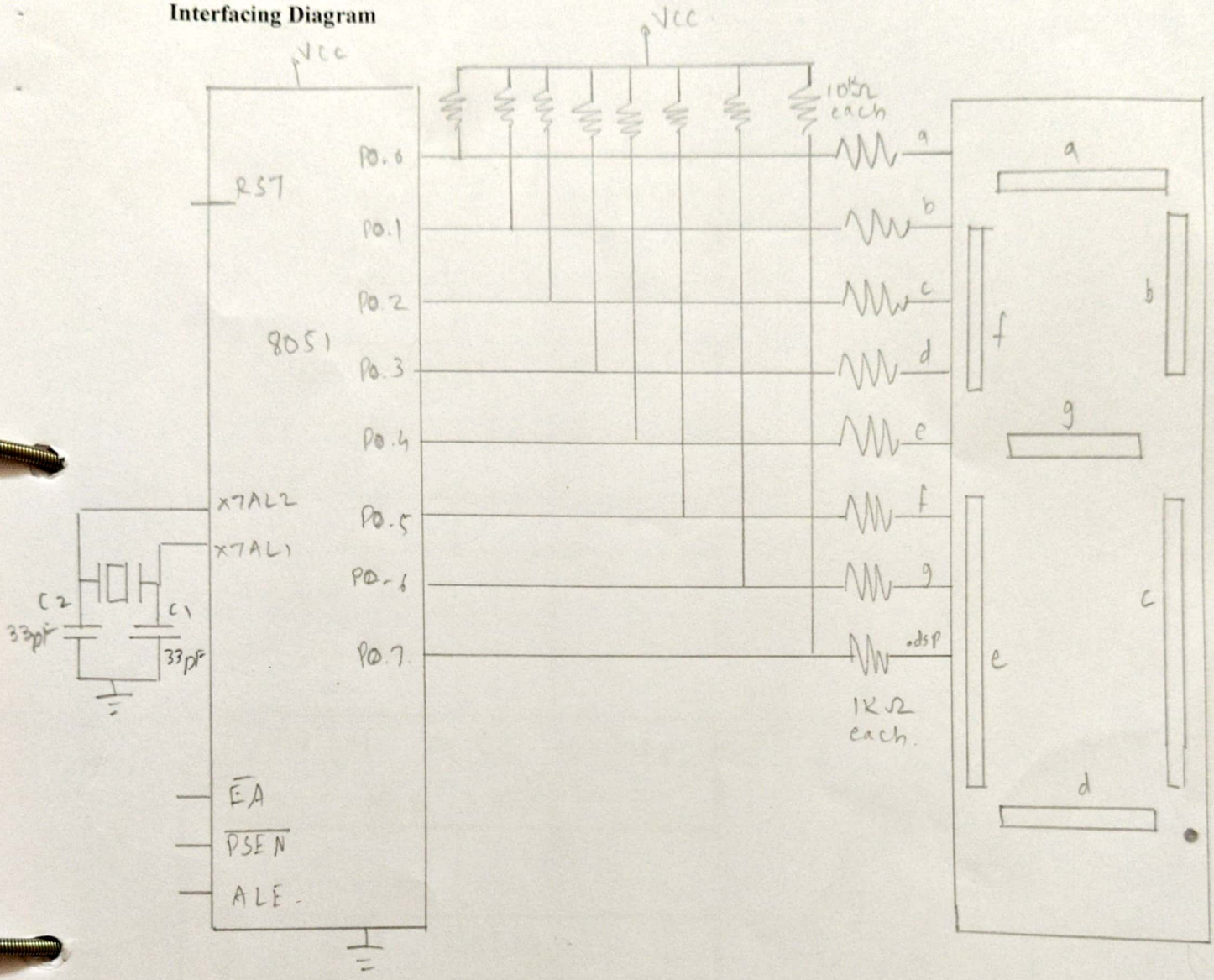
Now accordingly terminals are taken common, so there are two types of display:

1. Common Cathode display
2. Common Anode display

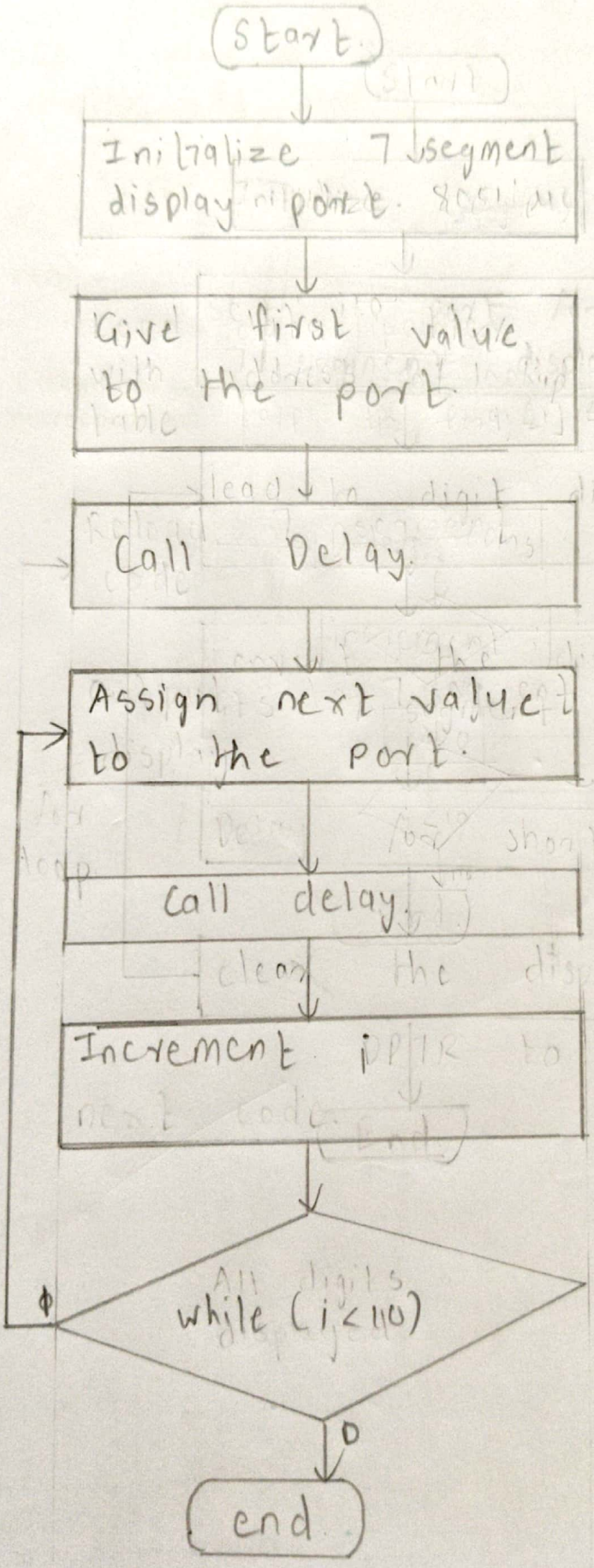
**1. The Common Cathode (CC) –**

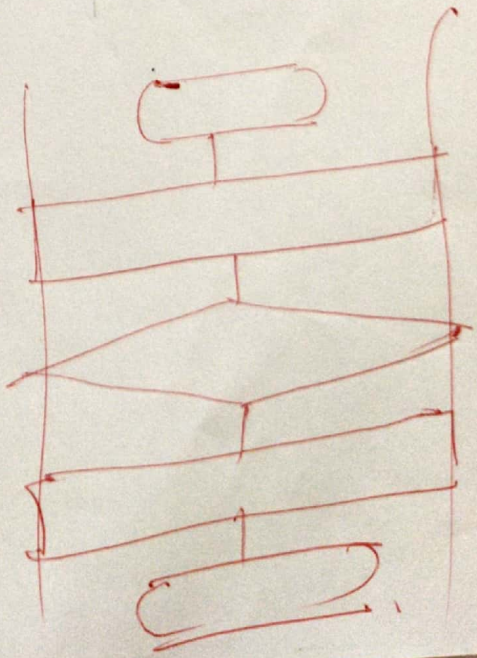
In the common cathode display, all the cathode connections of the LED segments are joined. Together to logic "0" or ground. The individual segments are illuminated by application of a "HIGH", or logic "1" signal via a current limiting resistor to forward bias the individual Anode terminals (a-g).

**Interfacing Diagram**



**Flowchart**



## EXPERIMENT NO 4

**AIM: -**

Interfacing of stepper motor to 8051 (Software delay using timer).

**APPARATUS: -**

89V51RD2 Development Board, stepper motor.

**THEORY: -**

A stepper motor is a brushless and synchronous motor which divides the complete rotation into several steps. Each stepper motor will have some fixed step angle and motor rotates at this angle. Here the interfacing of stepper to 8051 and ULN 2003 is explained.

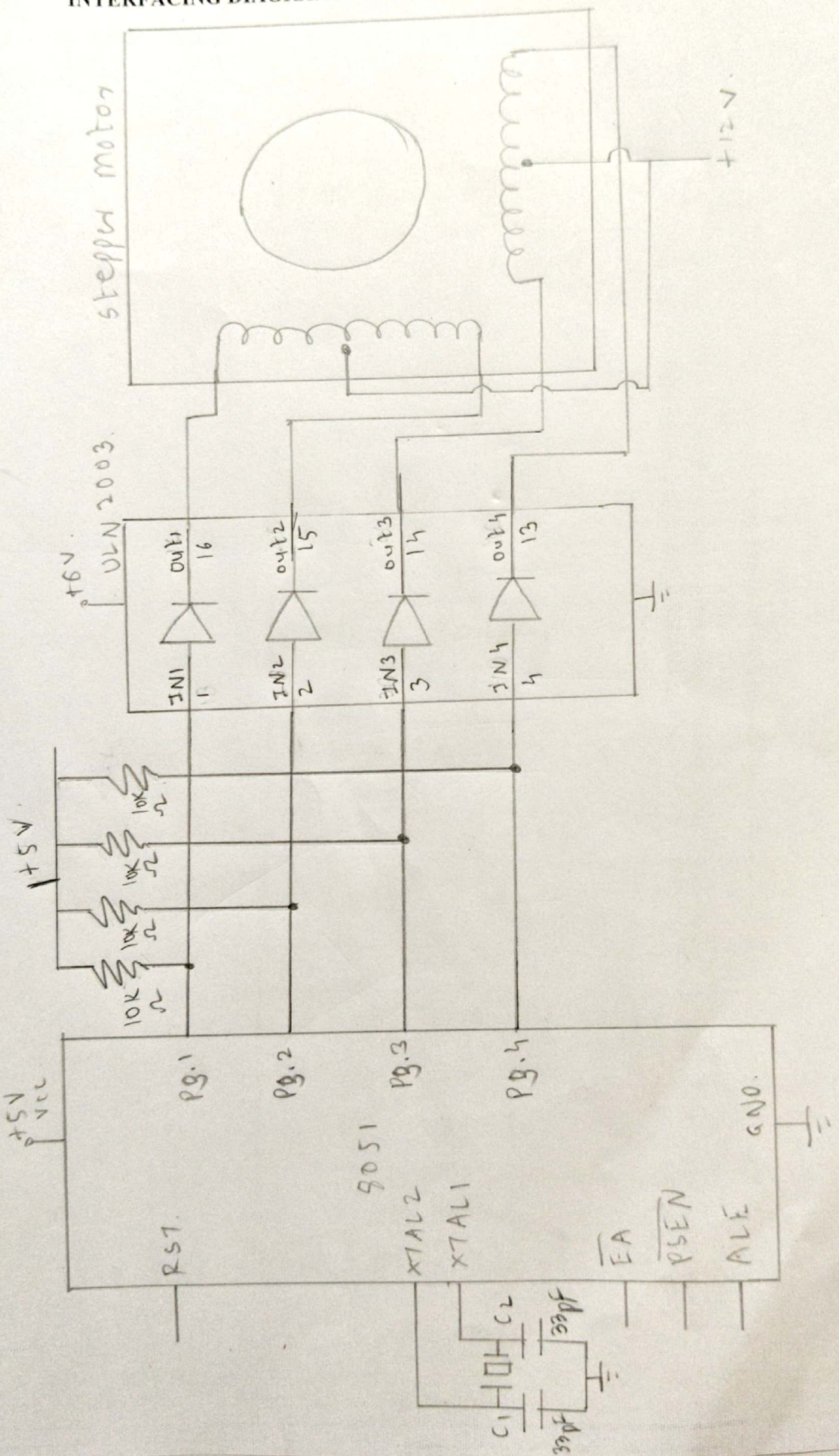
It has many applications in the field of robotics and mechatronics. The total rotation of the motor is divided into steps. The angle of a single step is known as the stepper angle of the motor. There are two types of stepper motors **Unipolar** and **Bipolar**. Due to the ease of operation unipolar stepper motor is commonly used by electronics hobbyists. For more details, please read the article Stepper Motor or Step Motor. Stepper Motors can be easily interfaced with a microcontroller using driver ICs such as L293D or ULN2003.

Interfacing with 8051 is very easy. We just need to give the 0 and 1 to the four wires of stepper motor according to the above tables depending on which mode we want to run the stepper motor. And the rest two wires should be connected to a proper 12v supply (depending on the stepper motor). Here we have used the unipolar stepper motor. We have connected four ends of the coils to the first four pins of port 2 of 8051 through the ULN2003A.

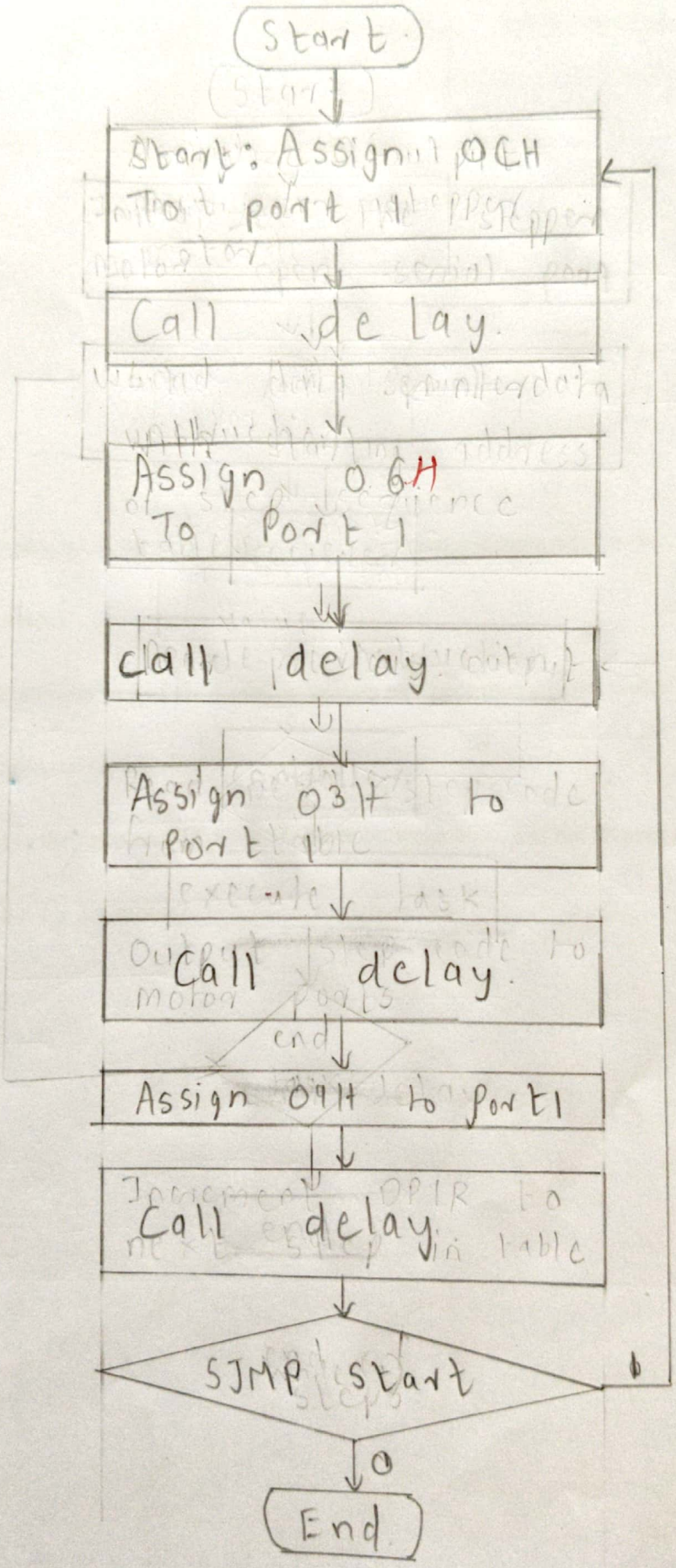
8051 doesn't provide enough current to drive the coils so we need to use a **current driver IC that is ULN2003A**. ULN2003A is the array of seven NPN Darlington transistor pairs. Darlington pair is constructed by connecting two bipolar transistors to achieve high current amplification. In ULN2003A, 7 pins are input pins, and 7 pins are output pins, two pins are for Vcc (power supply) and Ground. Here we are using four input and four output pins. We can also use L293D IC in place of ULN2003A for current amplification.

You need to find out four coil wires and two common wires very carefully otherwise the motor will not rotate. You can find it out by measuring resistance using multimeter, multimeter won't show any readings between the wires of two phases. Common wire and the other two wires in the same phase should show the same resistance, and the two end points of

**INTERFACING DIAGRAM:**



**FLOWCHART:**



# EXPERIMENT NO.5

**AIM:**

write a program for interfacing button, LED, relay & buzzer as follows:

**A.**

When button 1 is pressed relay and buzzer is turned ON and LED's start chasing from Left to right.

**B.**

When button 2 is pressed relay and buzzer is turned OFF and Led start chasing from Right to left.

**OBJECTIVE:**

To understand the concept of interfacing button, LED, relay & buzzer with port of PIC microcontroller.

**HARDWARE USED: Universal board for PIC 18F4520.**

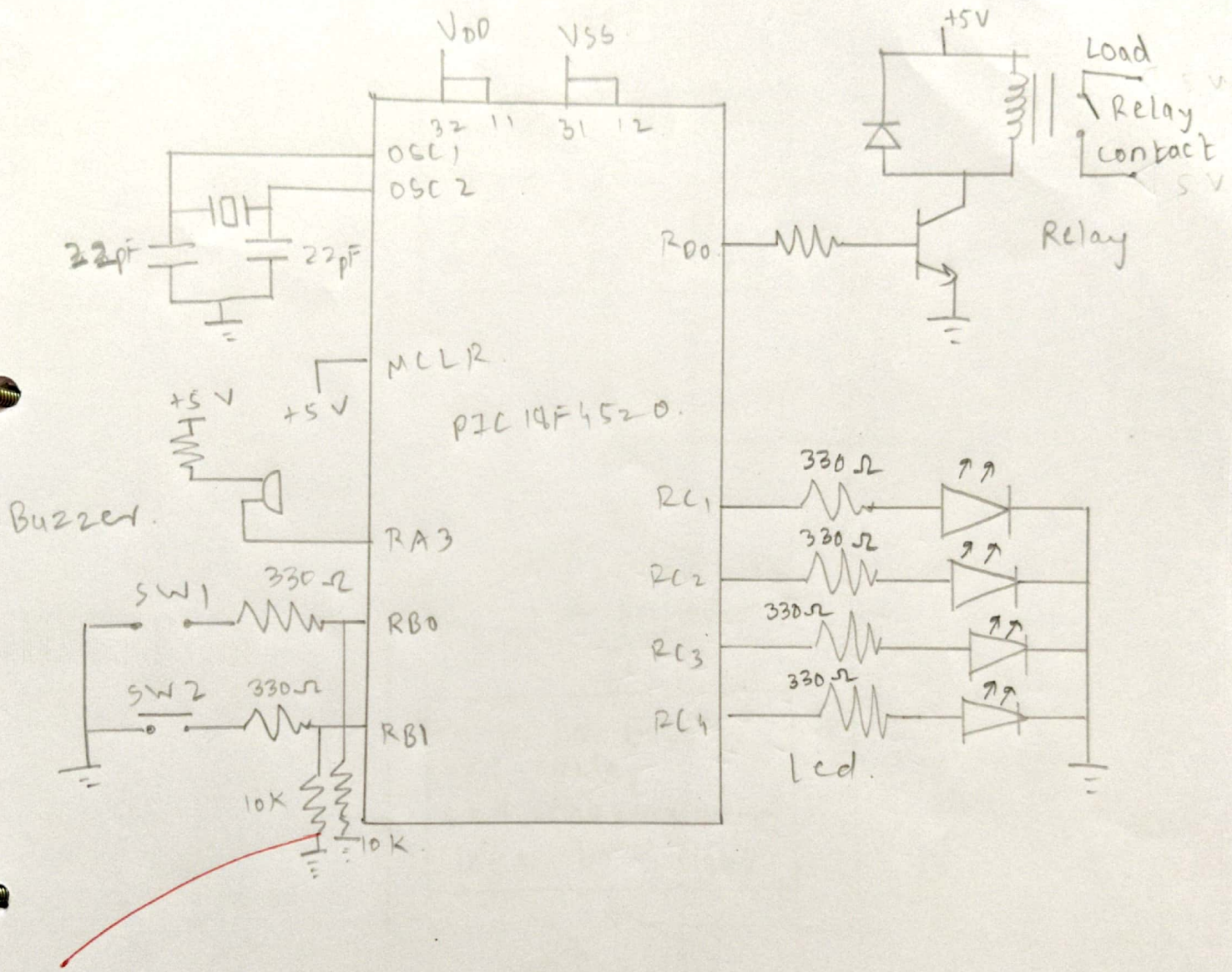
**SOFTWARE USED: MPLAB IDE v8.30, PIC KIT3**

**THEORY:**

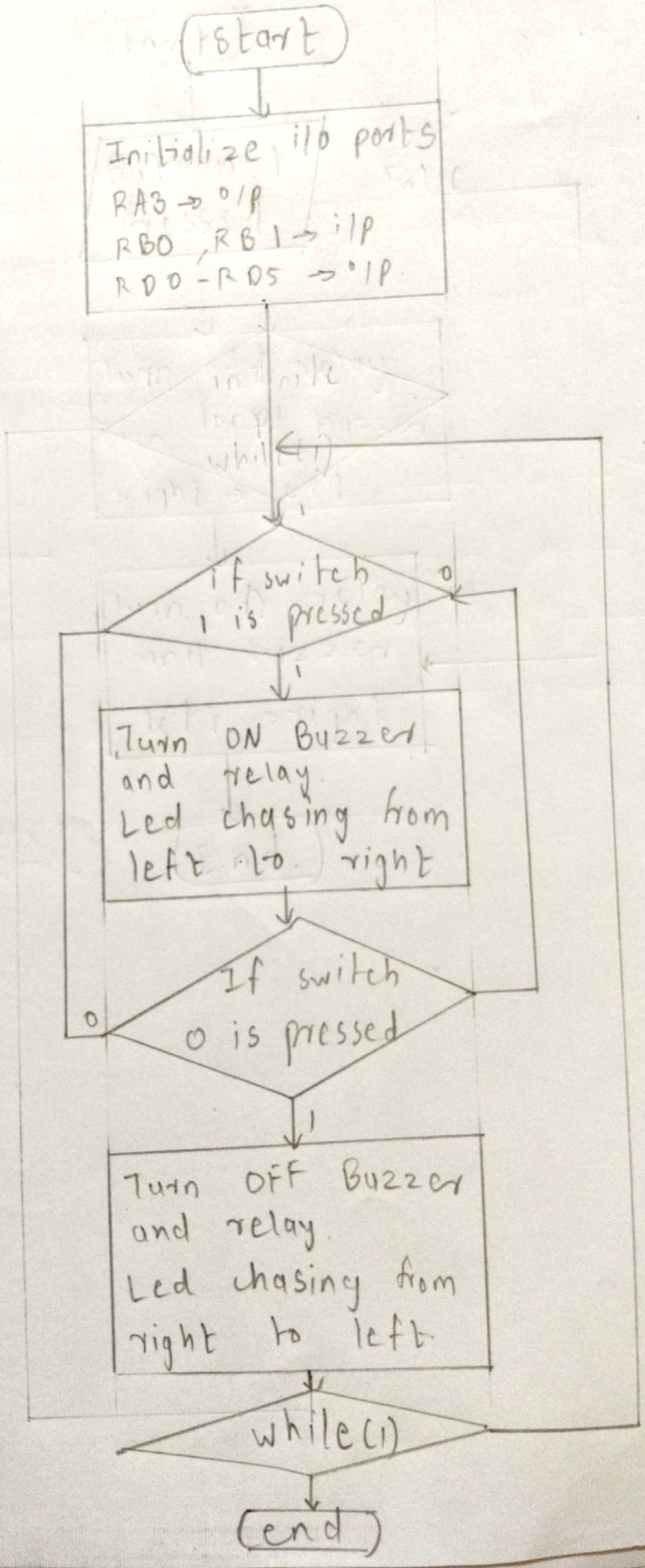
**1.LED:**

- A **light-emitting diode (LED)** is a two-lead semiconductor light source. It resembles a basic PN junction diode, which emits light when activated.
- When a fitting voltage is applied to the leads, electrons are able to recombine with electron holes within the device, releasing energy in the form of photons.
- This effect is called electroluminescence, and the color of the light (corresponding to the energy of the photon) is determined by the energy band gap of the semiconductor.

**INTERFACING DIAGRAM:**



Flow chart:



## Experiment NO: 6

**Aim: Interfacing 16x2 LCD with PIC18F Microcontroller**

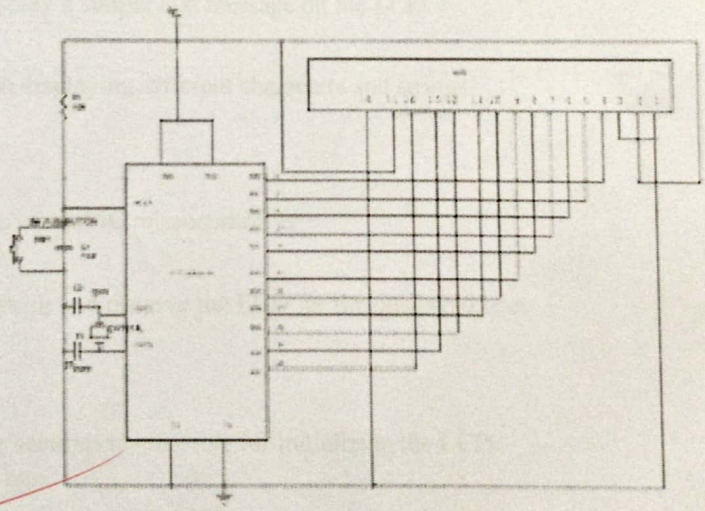
**Objective:**

1. To understand the basics of LCD (Liquid Crystal Display) technology.
2. To learn the process of interfacing a 16x2 LCD with a PIC18F microcontroller.
3. To display text and characters on the LCD screen using embedded C code.

**Apparatus:**

- PIC18F microcontroller (e.g., PIC18F4520)
- 16x2 LCD display
- Breadboard and jumper wires
- Power supply for the PIC microcontroller
- MikroC for PIC compiler

**Circuit Diagram:**



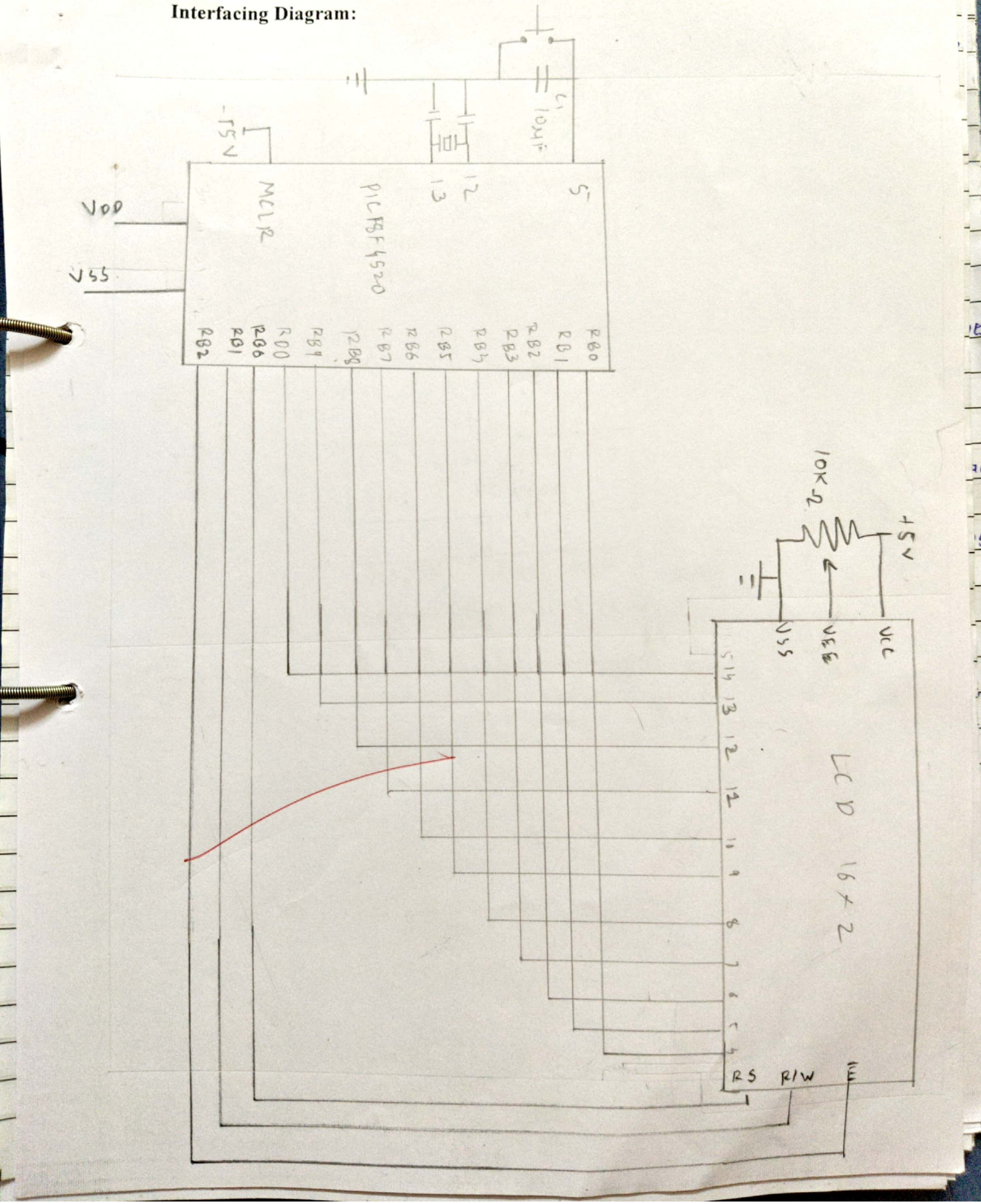
**Procedure:**

**1. Setup:**

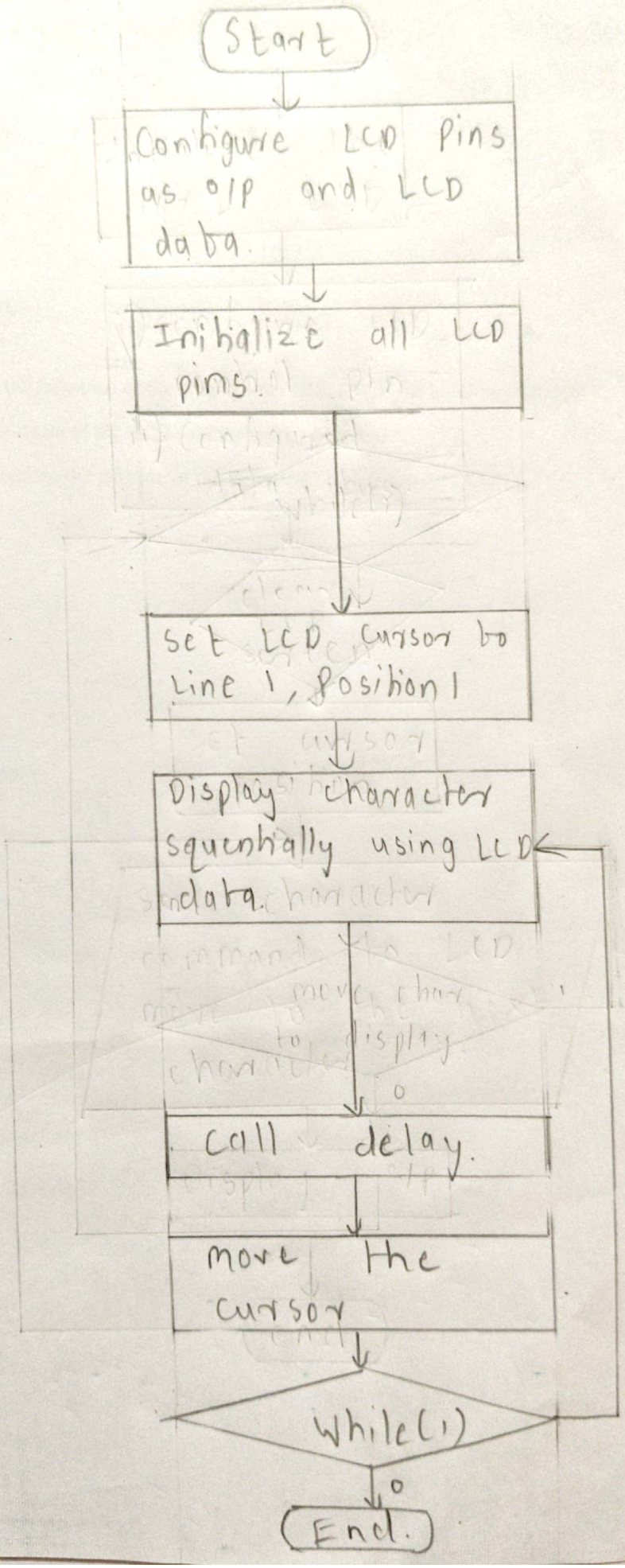
a. Connect the PIC18F microcontroller to the breadboard.

b. Connect the 16x2 LCD display to the appropriate ports on the PIC microcontroller.

**Interfacing Diagram:**



**Flowchart:**



## Experiment no: 7

## ADC INTERFACING WITH PIC 18F4520 MICROCONTROLLER.

**AIM:**

**Write embedded C program to interface analog voltage 0V to 5V to internal ADC and display value on LCD.**

**OBJECTIVE:**

To understand the concept of ADC in PIC microcontroller.

Write programs in embedded C for displaying digital value converted by PIC microcontroller.

**THEORY:**

The ADC peripheral of the PIC18 has the following characteristics.

(a) It is a 10-bit ADC.

(b)It can have 5 to 15 channels of analog input channels, depending on the family member.
In PIC18452/458, pins RA0-RA7 of PORT A are used for the 8 analog channels.

(c)The converted output binary data is held by two special function registers called ADRESL (A/D Result Low)

and ADRESH (A/D Result High)

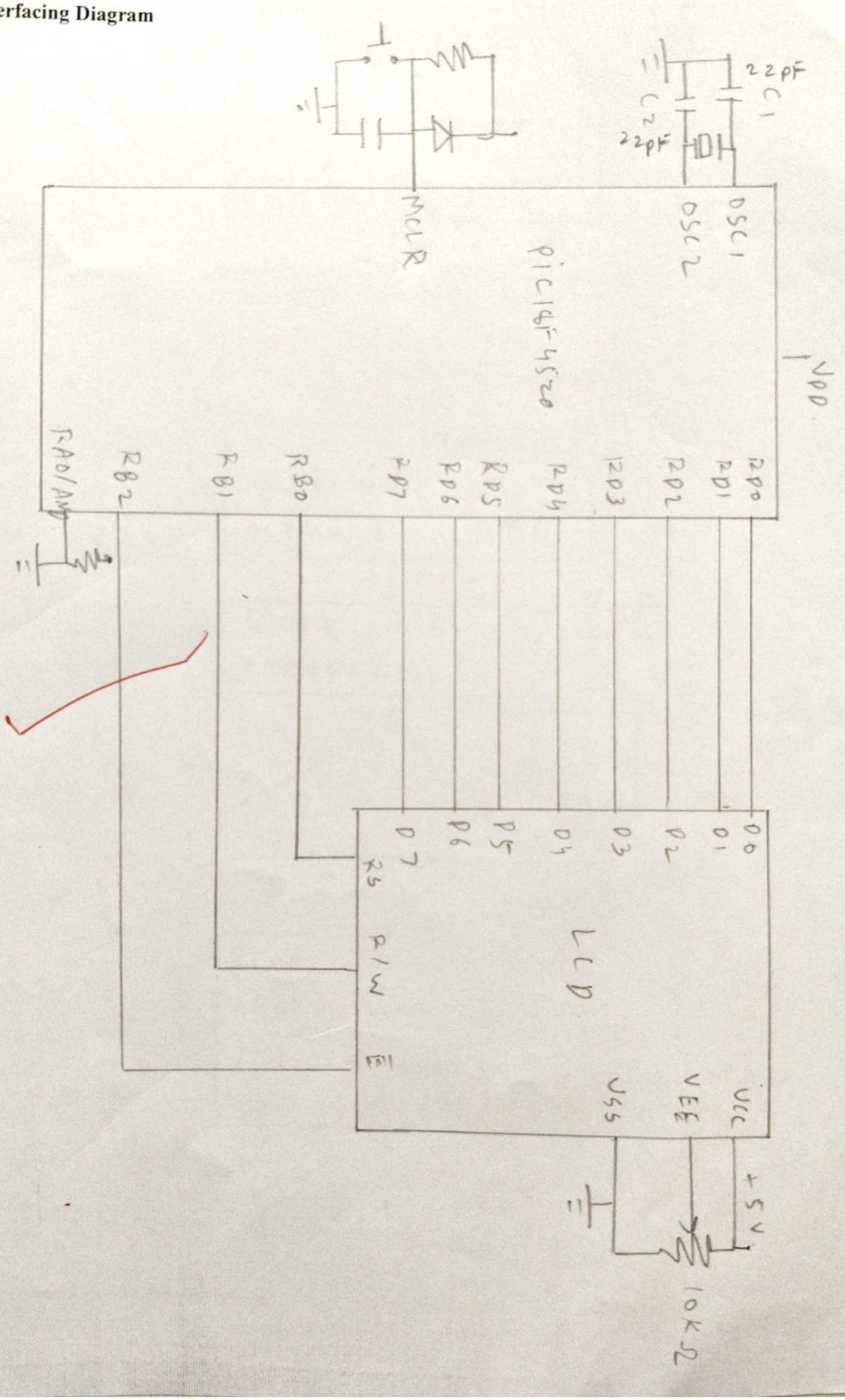
- Because the ADRESH: ADRESL registers give us 16 bits and the ADC data out is only 10 bits wide,6 bits of the 16 are unused. We have the option of making either the upper 6 bits or the lower 6 bits unused.

- We have the option of using VDD (VCC),the voltage source of the PIC18 chip itself, as the Vref or connecting it to an external voltage source for the Vref.

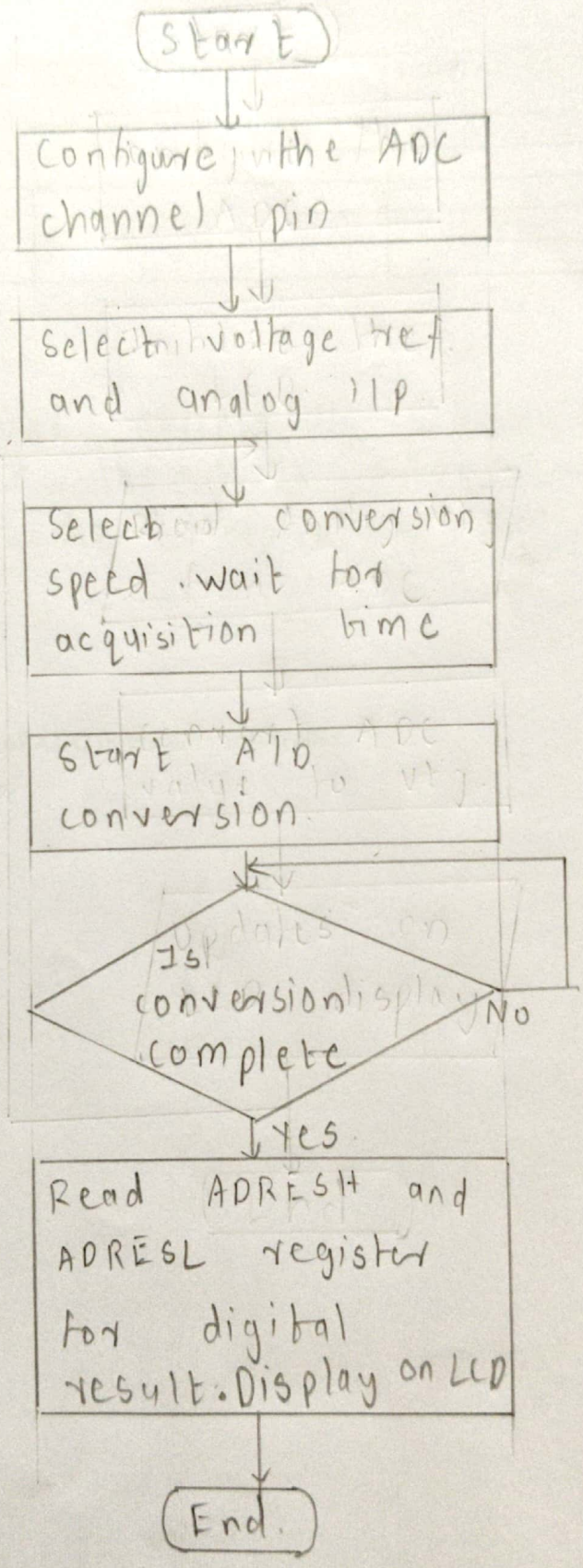
(f)The conversion time is dictated by the Fosc of crystal frequency connected to the OSCs pins.While the Fosc for PIC18 can be as high as 40 Mhz,the conversion time cannot be shorter than 1.6ms.

(g)It allows the implementation of the differential Vref voltage using the Vref(+) and Vref(-) pins,where Vref=Vref(+)-Vref(-).

**Interfacing Diagram**



**Flowchart**



## Experiment NO:8

**Lab Title: PWM Signal Generation for DC Motor Control**

**Objective:**

1. To understand the concept of Pulse Width Modulation (PWM).
2. To learn how to generate a PWM signal using a PIC microcontroller.
3. To control the speed of a DC motor using PWM.

**Apparatus:**

- PIC18F4520 microcontroller
- DC motor
- Motor driver circuit (if needed)
- Breadboard and jumper wires
- Oscilloscope (optional)
- MikroC for PIC compiler

**Procedure:**

**1. Setup:**

Connect the PIC18F4520 to the breadboard. b. Connect the DC motor to the motor driver circuit and the output of the motor driver to an appropriate port on the PIC microcontroller.

**2. Circuit Connections:**

Connect the PWM output pin of the PIC microcontroller to the input of the motor driver. b. Make sure the power supply to the motor is properly connected.

**3. MikroC Setup:**

Open the MikroC for PIC compiler. b. Create a new project for the PIC18F4520 microcontroller. c. Write the PWM initialization and motor control code. (Refer to the provided code snippet.)

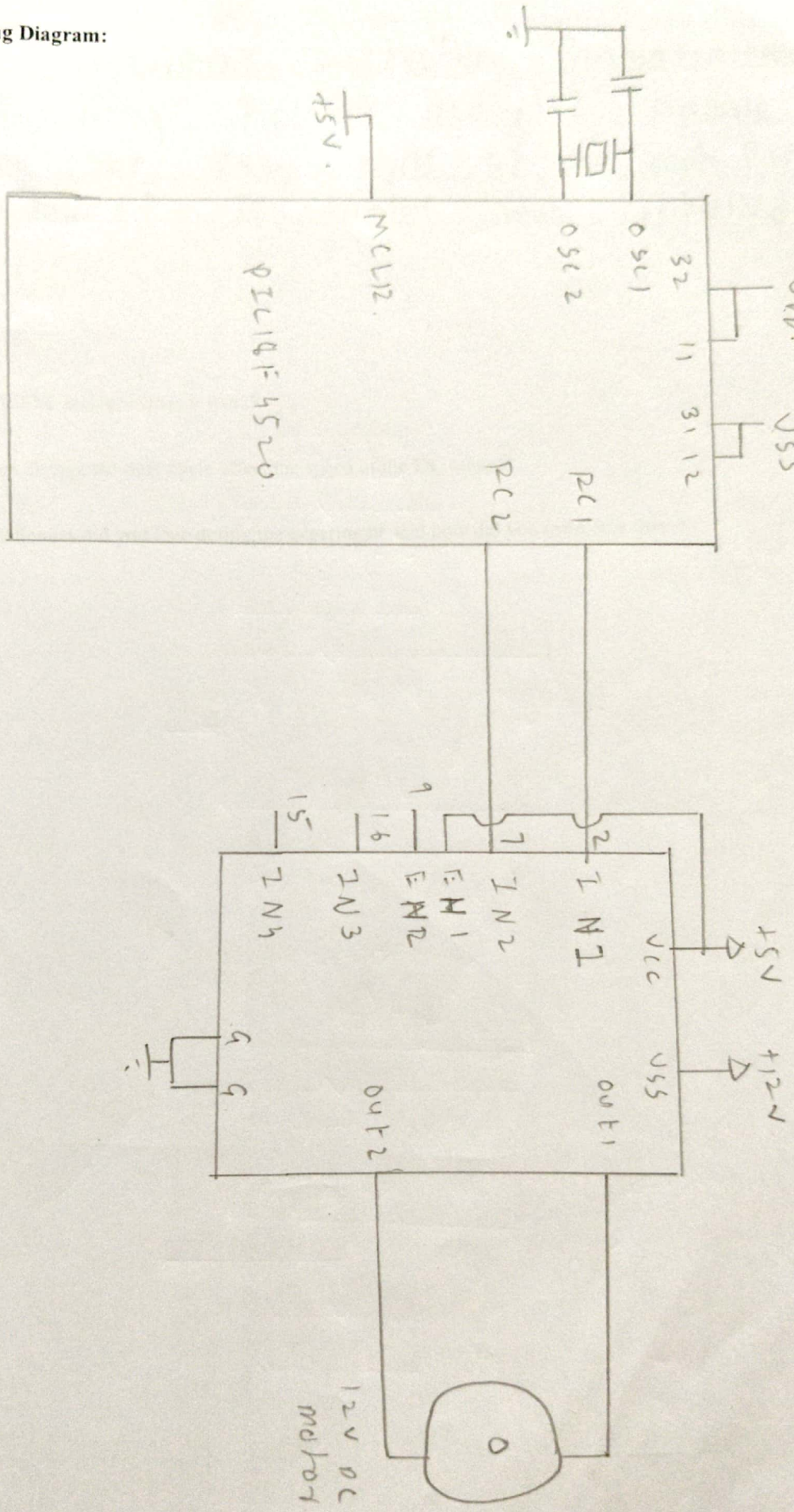
**4. PWM Initialization:**

In the code, initialize the PWM module for the desired pin. b. Set the PWM period and any other necessary parameters.

**5. Motor Control:**

Write code to control the duty cycle of the PWM signal. b. Use the PWM signal to control the speed of the DC motor.

**Interfacing Diagram:**



**6. Testing:**

a. Upload the code to the PIC microcontroller.

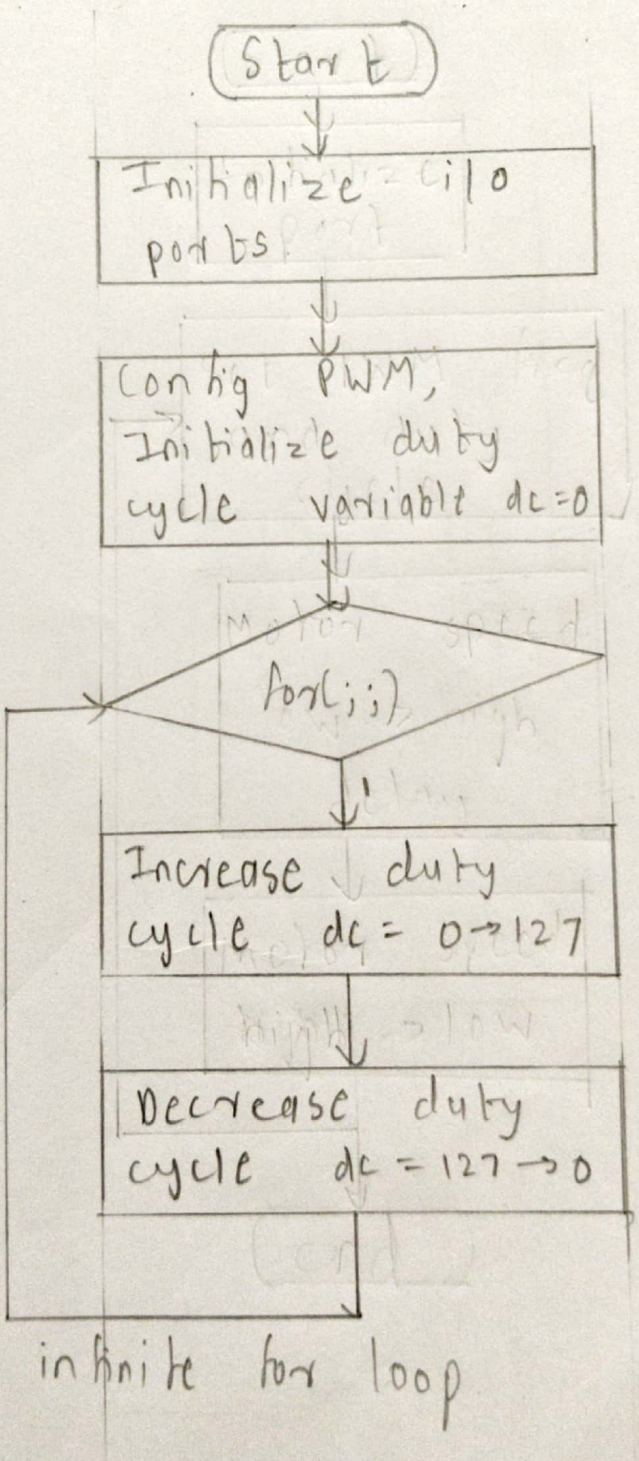
b. Power on the circuit and observe the motor's behaviour.

c. If available, use an oscilloscope to measure and analyse the PWM signal.

**7. Analysis and Observations:**

Record the duty cycle values used and their corresponding motor speeds. b. Note any observations about the motor's performance.

**Flowchart:**



# EXPERIMENT NO.9

**AIM:** Write a program for Interfacing serial port with PC both side communication.

**OBJECTIVE:** To understand the concept of serial port interfacing and communication with port of PIC microcontroller.

**HARDWARE USED: Universal board for PIC 18F4520.**

**SOFTWARE USED: MPLAB IDE v8.30, PIC KIT3**

**THEORY:**

## ENHANCED UNIVERSAL SYNCHRONOUS RECEIVER TRANSMITTER (EUSART) :

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module is one of the two serial I/O modules. (Generically, the USART is also known as a Serial Communications Interface or SCI.) The EUSART can be configured as a full-duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers. It can also be configured as a half-duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs, etc.

The Enhanced USART module implements additional features, including automatic baud rate detection and calibration, automatic wake-up on Sync Break reception and 12-bit Break character transmit. These make it ideally suited for use in Local Interconnect Network bus (LIN bus) systems. The EUSART can be configured in the following modes:

- Asynchronous (full duplex) with:
  - Auto-wake-up on character reception
  - Auto-baud calibration
  - 12-bit Break character transmission
- Synchronous – Master (half duplex) with selectable clock polarity
- Synchronous – Slave (half duplex) with selectable clock polarity

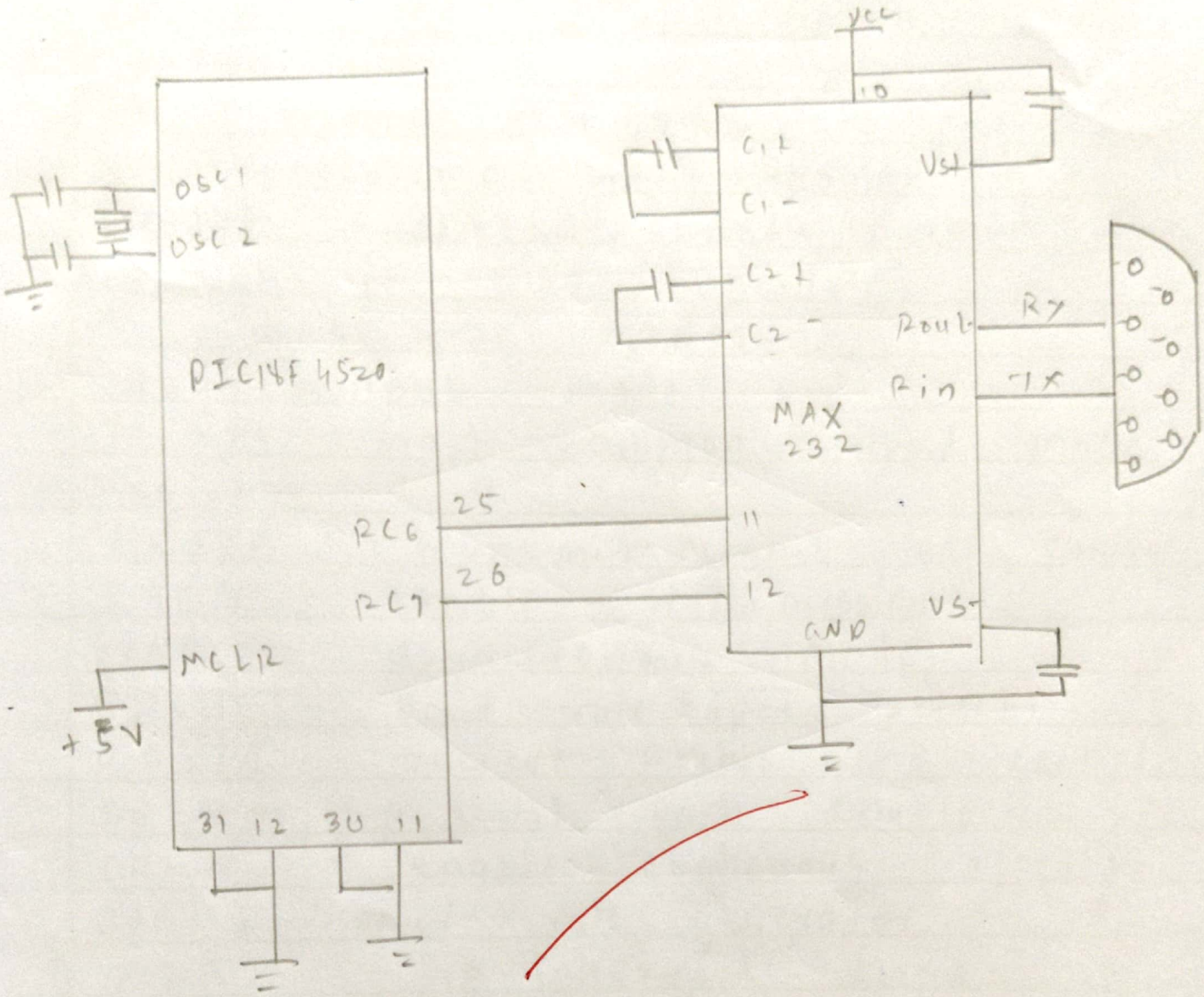
The pins of the Enhanced USART are multiplexed with PORTC. In order to configure RC6/TX/CK and RC7/RX/DT as a USART:

(i) bit SPEN (RCSTA<7>) must be set (= 1)
(ii) bit TRISC<7> must be set (= 1)
(iii) bit TRISC<6> must be set (= 1)

The operation of the Enhanced USART module is controlled through three registers:

- Transmit Status and Control (TXSTA)
- Receive Status and Control (RCSTA)
- Baud Rate Control (BAUDCON)

**Interfacing Diagram:**



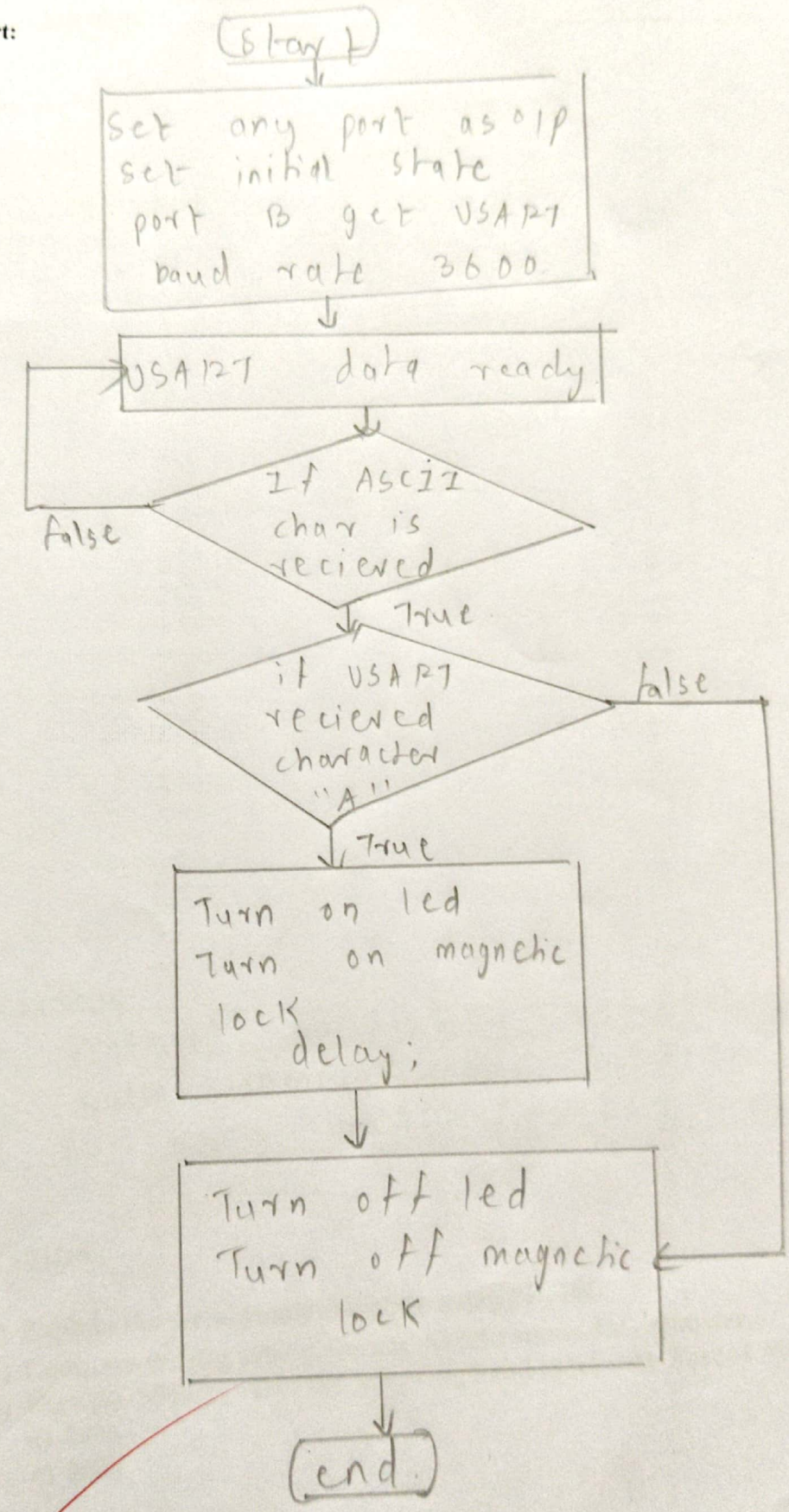

**CONCLUSION:**

The practical successfully established 2 way serial communication between Pic 18 MC and PC using USART / EUSART. module.

**QUESTION:**

1) **Explain the serial communication using PIC18f.**
2) **Compare the Asynchronous and Synchronous data transfer.**
3) **Find the SPBRG value for following baud rates with Fosc=4 MHz**
   a) **2400**
   b) **9600**

**Flow Chart:**



# EXPERIMENT NO.10

**AIM:** To interface an LED with PIC18F4520 and blink it using Timer interrupt.

**APPARATUS:** PIC18F4520 microcontroller, .LED and 330 Resistor, MPLAB IDE with C18 compiler ,PIC Kit 2 programmer.

**THEORY:**

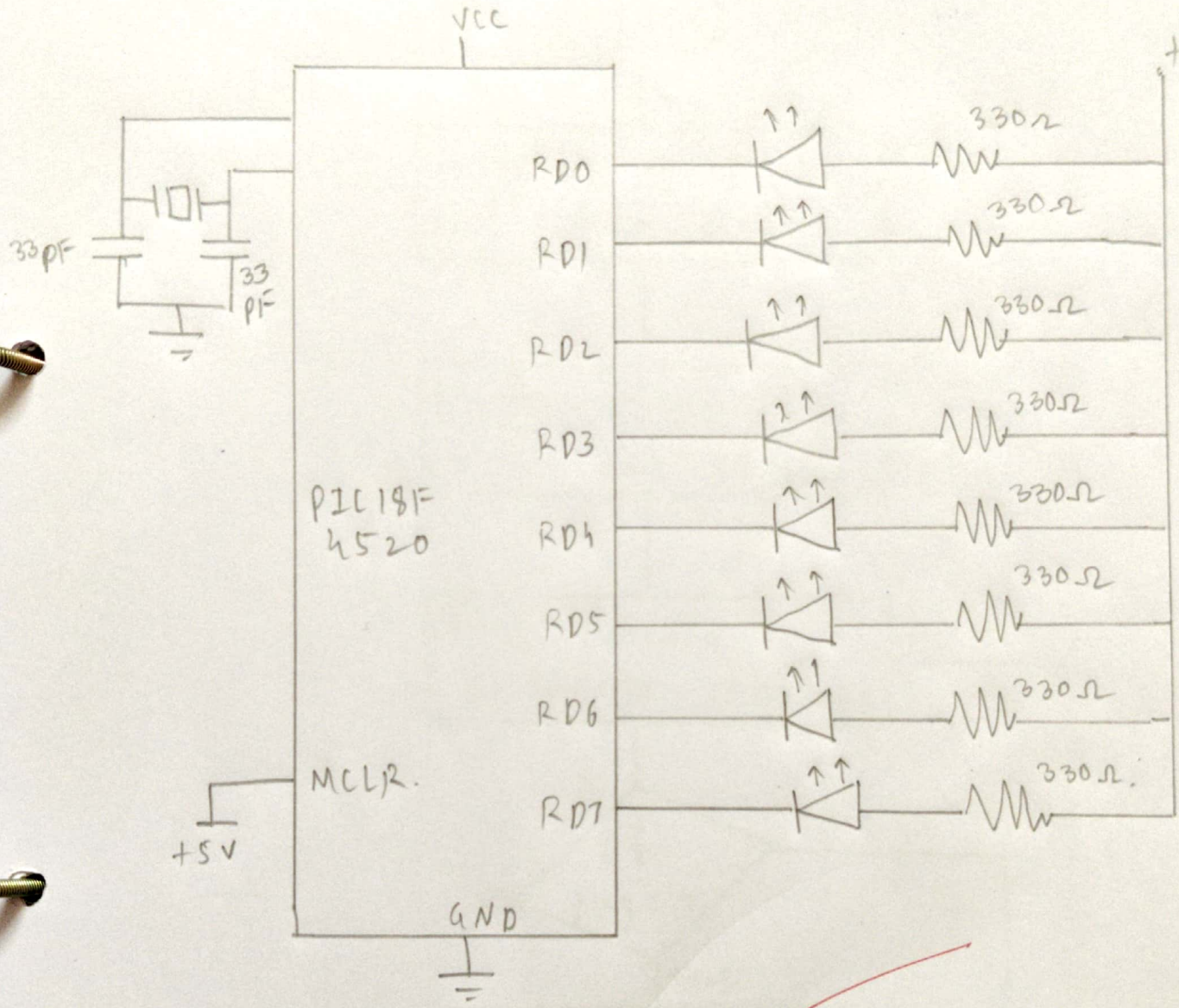
Microcontrollers like the **PIC18F4520** contain several built-in **timers** that can be used to generate precise time delays, measure pulse widths, or count external events. A timer is essentially a hardware counter that increments at a known rate derived from the system clock or an external source. The main advantage of using a hardware timer over a software delay loop is **accuracy and efficiency**: once configured, the timer operates independently of the CPU, allowing the processor to perform other tasks or enter low-power modes while the timer runs.In the PIC18F4520, there are multiple timers (Timer0–Timer3), each with unique features. **Timer0** is one of the most commonly used; it can operate in both **8-bit** and **16-bit** modes. When configured as a timer, it counts internal instruction cycles (Fosc/4) that are optionally divided by a prescaler. When it reaches its maximum count (0xFF for 8-bit or 0xFFFF for 16-bit mode), it **overflows**, and the microcontroller sets an **interrupt flag (TMR0IF)**. If timer interrupts are enabled, this overflow automatically triggers the **Interrupt Service Routine (ISR)**. The ISR is a special block of code that executes immediately whenever the interrupt occurs.By toggling an LED inside the ISR, we can make the LED blink at a steady rate. The blink frequency depends on the timer overflow period, which can be calculated from the oscillator frequency, prescaler, and preload values of TMR0H:TMR0L registers. For example, with a 10 MHz crystal, the instruction cycle frequency is 2.5 MHz (Fosc/4), meaning one instruction cycle equals 0.4 µs. Using a prescaler increases this time base, allowing us to achieve longer delays before overflow.

This method provides several benefits:

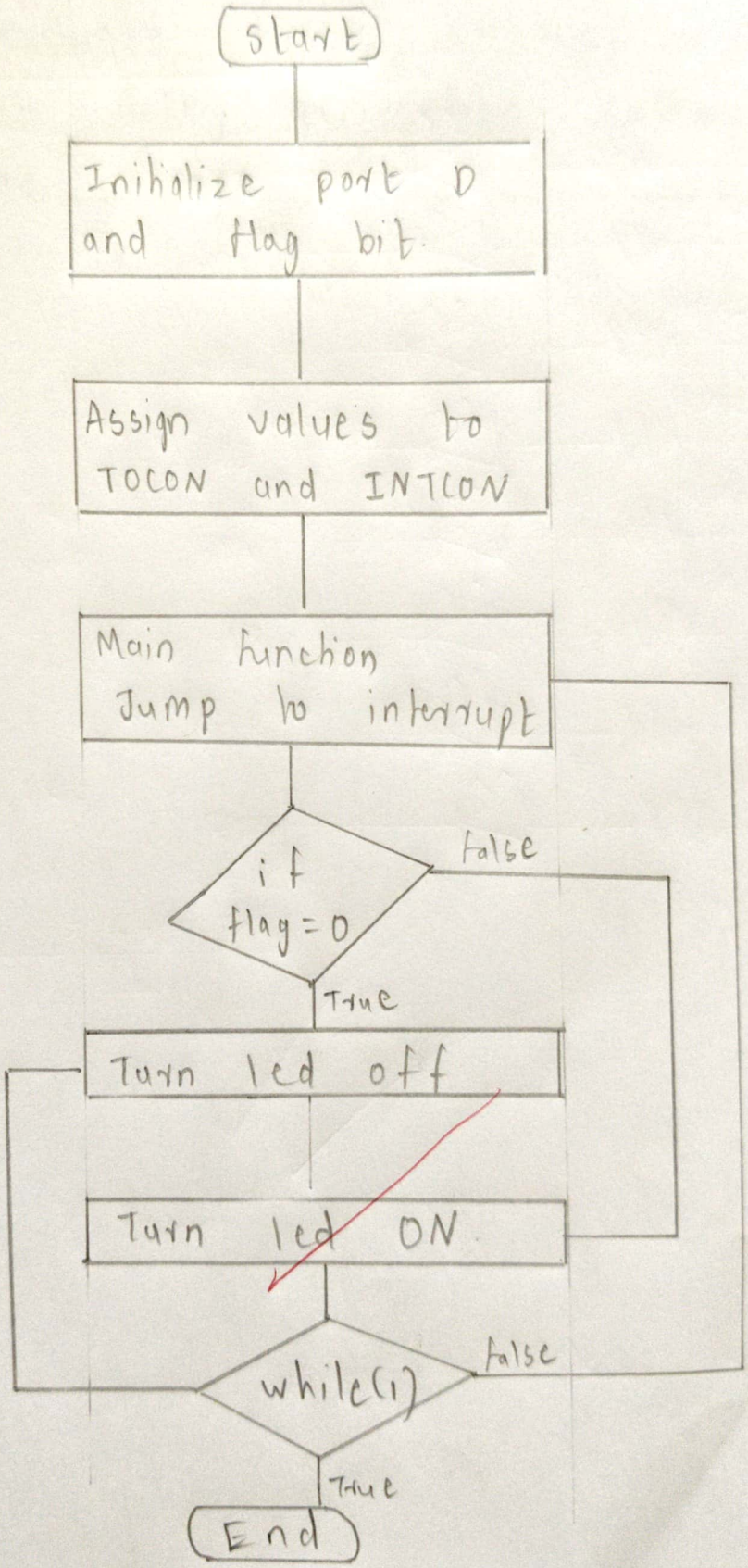
- **Precise timing:** The delay depends on hardware counters, not on CPU instruction execution time.
- **Interrupt-driven operation:** The LED toggling occurs automatically through interrupts, freeing the CPU for other work.

# Interfacing





# Flow chart



**CONCLUSION:**

The led successfully blinked using ~~conti~~ hardware timer interrupt, demonstrating efficient delay generation without software loops.