Guidelines to use Silicon Lab's Simplicity Studio

Simplicity Studio simplifies the Embedded development process with one-click access to everything developers need to complete their projects using an integrated development environment (IDE) based on Eclipse 4.5.

Steps to create project and program compilation:

Steps:

- Open Simplicity Studio.
- Click File > **Import…**, to import the project.
- Click General > Existing Projects into Workspace and click Next.
- First Select Root Directory of your Eclipse Template for EPB_F340. Then select "Copy projects into workspace" check box and Click Finish.
- Every time you import a project make sure to rename it. So **Right Click Project > Rename** or press **F2** while selecting project to rename it.
- To write source files, right click on Sources, go to New > Source File and you will see New Source File Wizard. Enter Source File name (For example **main.c**) then Click Finish.
- To write header files, right click on Includes, go to New > Header File and you will see New Header File Wizard. Enter Header File name (For example **main.h**) then Click Finish.
- Write your code and then save your Files.

OR

- Copy necessary .c and .h files to your local Sources and Includes project folders respectively. They will be added to your project in Simplicity Studio IDE.
- Click on Build Project in **Project>Build Project** to compile and build the project.
- After successful building the project the .hex file will be generated in the project folder in the workspace.

Steps to use hardware:

- Connect 9V DC Power supply to the educational practice board for F340
- Connect USB cable between PL4 connector of EPBF340 board and PC.
- Using the RUN/PROGRAM mode selection switch set the board in the program mode by pressing switch SW2 and press reset.
- Using download tool (USB Bootloader) download the .HEX file to the target board.
- Wait for the "Successfully loaded image" message in display.
- Connect flat cable between ASK25 and EPBF340 boards according to each experiment's requirement.
- Using the RUN/PROGRAM mode selection switch, set the board in the run mode by releasing Switch SW2. Apply reset to execute the program.
- Observe the expected output of the experiment.

***NOTE:** Use this tool for Experiment No. 3 to 10. Draw interfacing diagram on separate sheet and attach the printout of the tested code.*

F340 Hardware

The EPB-F340Mini is a stand-alone card--allowing developers to evaluate the C8051F340 USB Flash MCU Family to determine if it meets their application requirements. Furthermore, the module is an excellent platform to develop and run software for the 8051 processor. The EPB-F340Mini is shipped with a C8051F340. The EPB-F340Mini allows full speed verification of 8051 code. In addition, an onboard C2 connector provides interface to emulators, with assembly language and ‘C’ high level language debug.

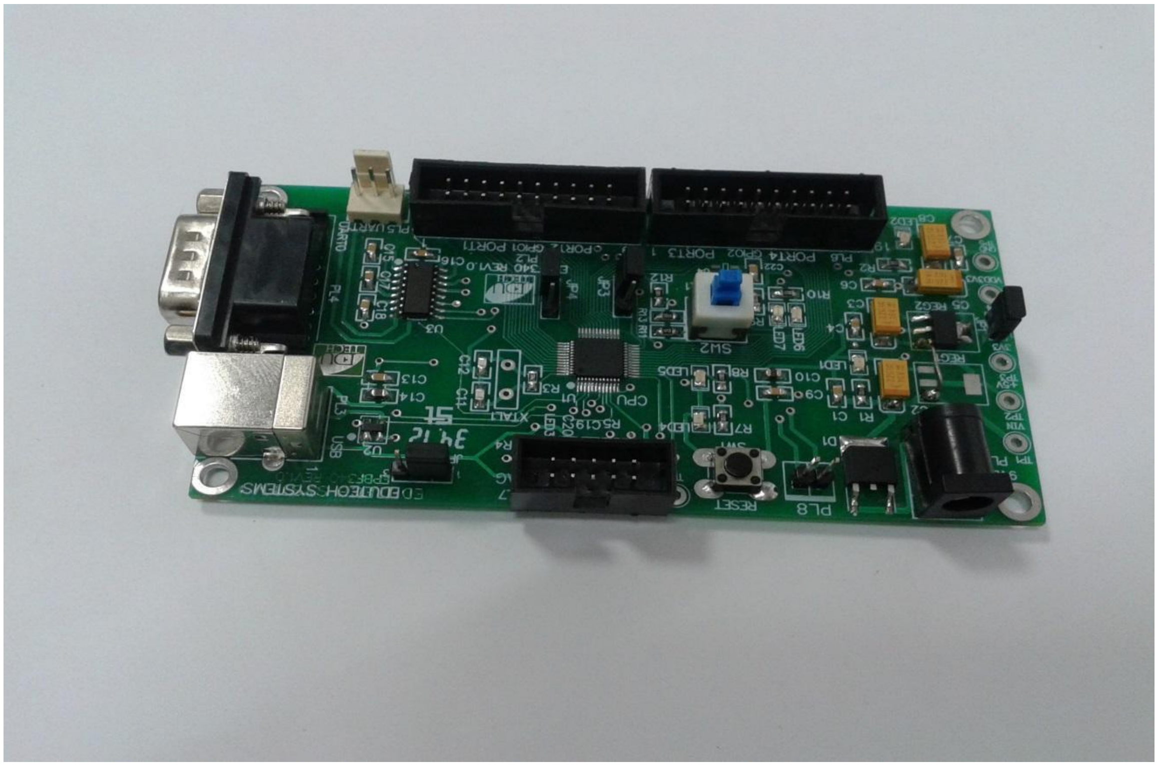


Figure 3.1: The EPB-F340Mini development board

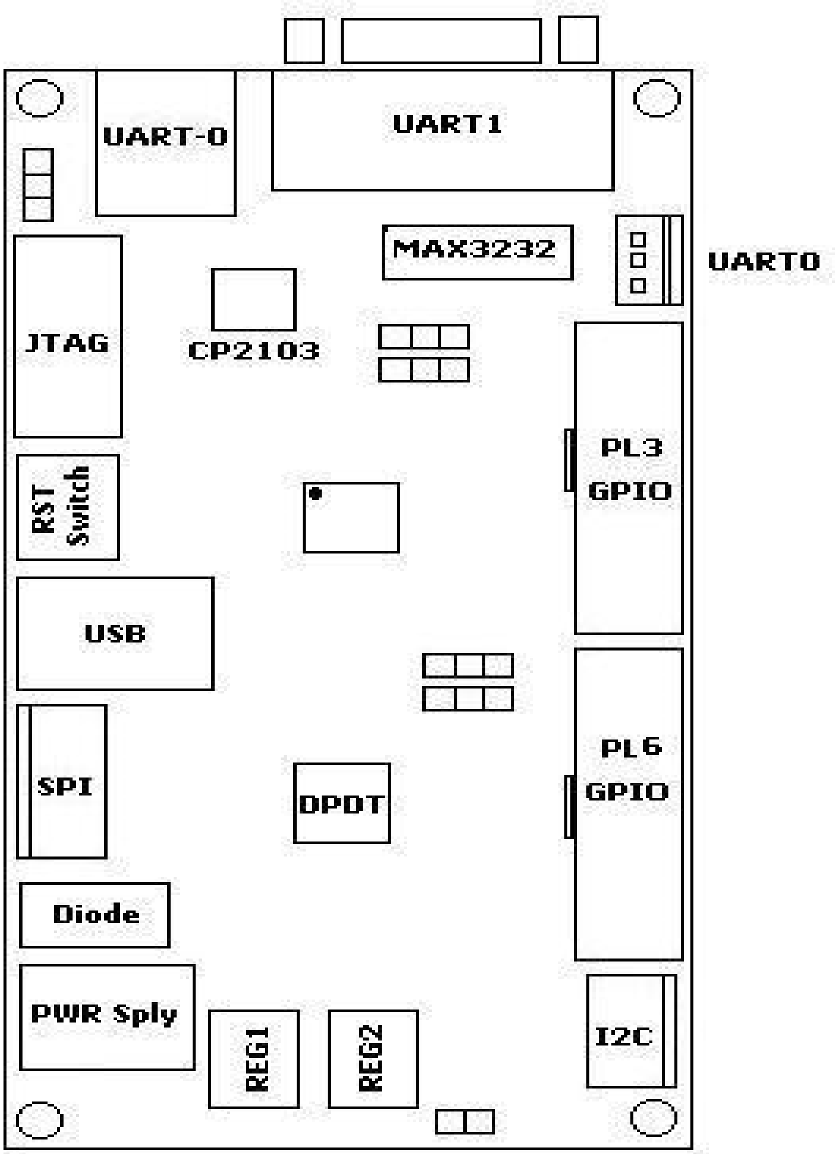


Figure 3.2 The EPB-F340Mini development board layout

The EPB-F340Mini has 8 connectors. The function of each connector is shown in the table below:

Table 3.1 EPB-F340Mini connector description

Unit	Reference	Description
Power jack connector	PL1	Power jack connector
I2C	PL2	4 Pins
I/O Port	PL3	PORT1 and 2 GPIO
USB	PL4	USB Connector
JTAG	PL5	JTAG Interface
I/O Port	PL6	PORT3 and 4 GPIO
SPI	PL7	6 Pins
UART0/Debug	PL8	USB Connector
UART0	PL9	3 Pins Connector
UART1	PL10	DB9 -M Connector

General Purpose I/O Interfacing Kit – ASK 25A

The general purpose I/O interfacing Board is a generic board which focuses on the interfacing of different input and out devices to microcontroller. The board is populated with variety of devices like LED, Key, Relay, LCD, Signal conditioning circuit for ADC, DC motor interface, Stepper Motor interface, I2C EEROM, SPI EEPROM, 2x2 Matrix Key board etc.

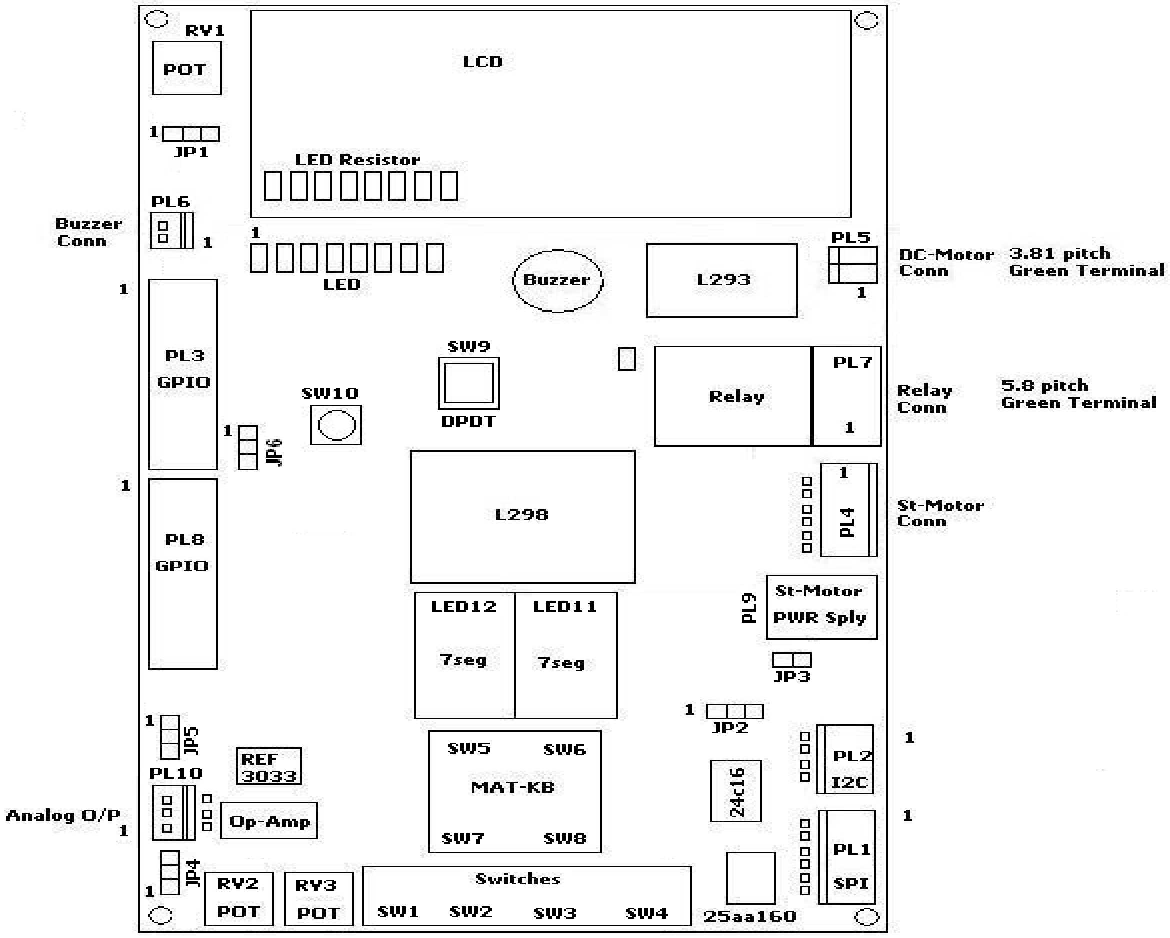


Figure 3.3 The ASK-25 board layout

The ASK-25A has 10 connectors. The function of each connector is shown in the table below:

Table 3.2 ASK-25A connector description

Unit	Reference	Description
SPI	PL1	6 Pins
I2C	PL2	4 Pins
I/O Port	PL3	GPIO
Stepper Motor Connector	PL4	6 Pins Connector
DC Motor Connector	PL5	2 Pin Green Terminal
Buzzer Connector	PL6	2 Pin Connector
Relay Connector	PL7	3 Pin Green Terminal
I/O Ports	PL8	GPIO

Stepper Motor Power Supply	PL9	Power Socket
Analog Output	PL10	3 Pin Connector

T. Y. B. Tech (Electrical and Computer Engineering)

Trimester: V **Subject: Microcontroller and Applications**
Name: Shreerang Mhatre **Class: TY**

Roll No: 52 **Batch: A3**

Experiment No: 03

Name of the Experiment: Interfacing of LED, Buzzer, Relay and Switch with C8051F340

Performed on: 03/10/2023

Submitted on: 07/10/2023

Marks	Teacher's Signature with date

Aim: Write C program for interfacing of LED, Buzzer, Relay and Switch with C8051F340 to turn it ON when key is pressed.

Apparatus: EPBF340 Board, ASK25 board, Connectors

Theory:

Ports in C8051F340:

Digital and analog resources are available through 40 I/O pins. These pins are available on ports. C8051F340 has 5 ports, each port has eight pins. Each of the Port pins can be defined as general-purpose I/O (GPIO) or analog input. This resource assignment flexibility is achieved through the use of a Priority Crossbar Decoder. Registers XBR0, XBR1, and XBR2 are used to assign the digital I/O resources to the physical I/O Port pins.

Port I/O Initialization

Port I/O initialization consists of the following steps:

Step 1. Select the input mode (analog or digital) for all Port pins, using the Port Input Mode register (PnMDIN).

Step 2. Select the output mode (open-drain or push-pull) for all Port pins, using the Port Output Mode register (PnMDOUT).

Step 3. Select any pins to be skipped by the I/O Crossbar using the Port Skip registers (PnSKIP).

Step 4. Assign Port pins to desired peripherals (XBR0, XBR1) as shown in figure 2.3.

Step 5. Enable the Crossbar (XBARE = ‘1’).

SFR definitions:

1. **PnMDIN: Portn Input Mode**

0: Corresponding P0.n pin is configured as an analog input.

1: Corresponding P0.n pin is not configured as an analog input.

2. **PnMDOUT: Portn Output Mode**

0: Corresponding P0.n Output is open-drain.

1: Corresponding P0.n Output is push-pull.

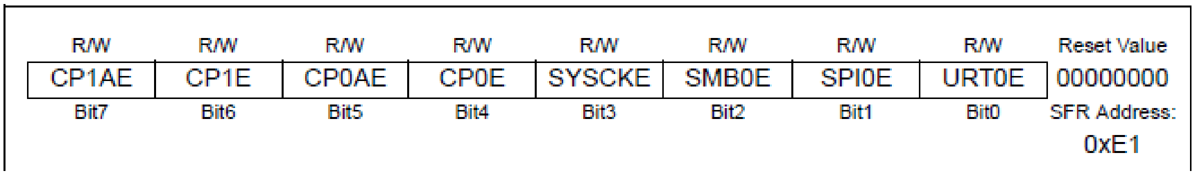
3. **PnSKIP: Portn Skip**

0: Corresponding P0.n pin is not skipped by the Crossbar.

1: Corresponding P0.n pin is skipped by the Crossbar.

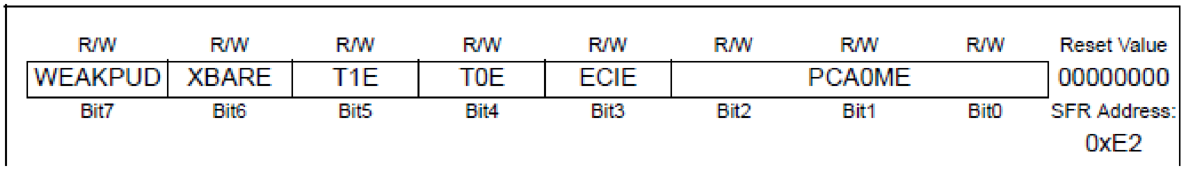
4. **XBR0, XBR1, and XBR2 are used to assign the digital I/O resources to the physical I/O Port pins**

SFR Definition 15.1. XBR0: Port I/O Crossbar Register 0



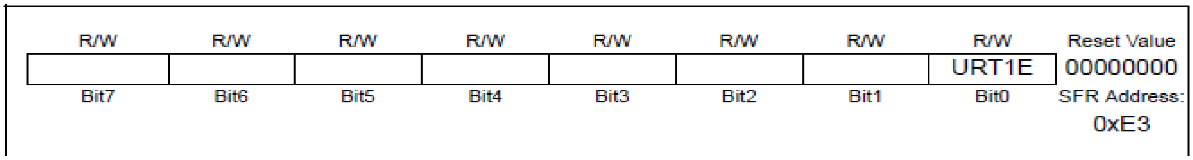
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
CP1AE	CP1E	CP0AE	CP0E	SYSCKE	SMB0E	SPI0E	URT0E	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE1

SFR Definition 15.2. XBR1: Port I/O Crossbar Register 1



R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
WEAKPUD	XBARE	T1E	T0E	ECIE	PCA0ME			00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE2

SFR Definition 15.3. XBR2: Port I/O Crossbar Register 2



R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
							URT1E	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xE3

LED: LED is an output device. 8 LED's are connected to 8 port pins. The LED's are connected in common anode configuration. Thus, to turn ON the LED logic '0' must be given and to turn OFF the led logic '1' must be given.

Buzzer: Buzzer is an output device. The buzzer is turned ON when logic '1' is applied to the port pin and turned OFF when logic '0' is applied to port pin.

Relay: Relay is an output device. The relay is turned ON when logic '0' is applied to port pin and turned OFF when logic '1' is applied to the port pin.

Switch/button: Switch is an input device. Push button switches are used. When the switch is released the port pin has logic '1' and when the switch is pushed the port pin has logic '0'.

Algorithm:

Interfacing Diagram:

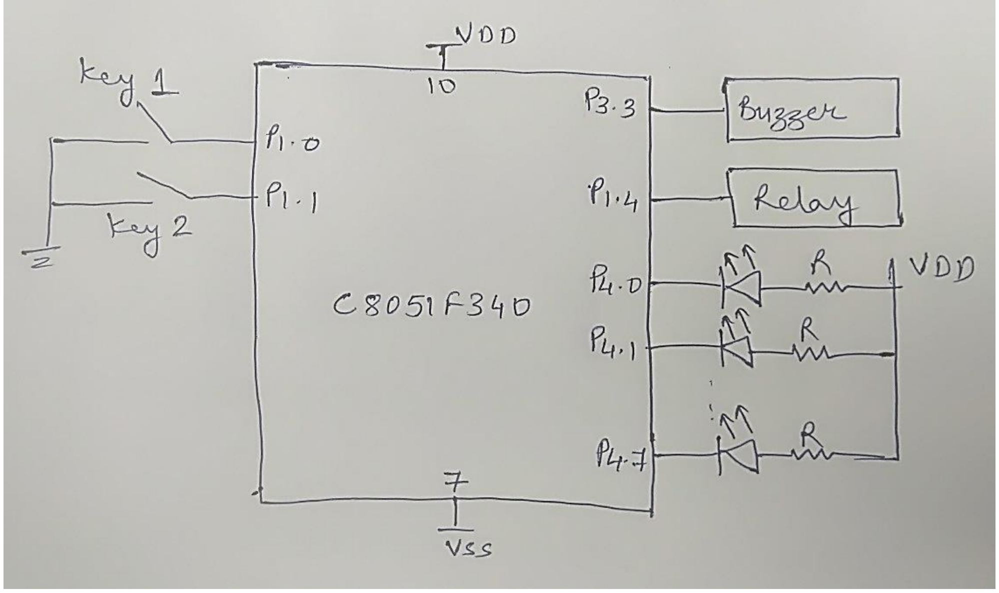


Figure 3.4 LED, Buzzer, Relay, and Switch Interfacing Diagram

Hardware Connections:

Connect flat cable between PL8 connector of ASK25 and PL6 connector of EPBF340 board. Connect other flat cable between PL3 connector of ASK25 and PL3 connector of EPBF340 board.

Table 3.3 Hardware connections between EPBF340 and ASK25 board

Pin Connection	PL8 Connector of ASK25	PL6 Connector of EPBF340	PL3 Connector of ASK25	PL3 Connector of EPBF340
1			SW1	P1.0
2			SW2	P1.1
3			SW3	P1.2
4	BUZZER	P3.3	SW4	P1.3
5			RELAY	P1.4
10	LED1	P4.0		
11	LED2	P4.1		
12	LED3	P4.2		
13	LED4	P4.3		
14	LED5	P4.4		
15	LED6	P4.5		
16	LED7	P4.6		
17	LED8	P4.7		
18		3.3 V		3.3 V
19	5V	5.0 V	5V	5.0 V
20	GROUND	GND	GROUND	GND

Program: Attach printout of the tested code.

Expected Result:

LEDs, Buzzer and Relay should turn on when respective key is pressed.

Conclusion:

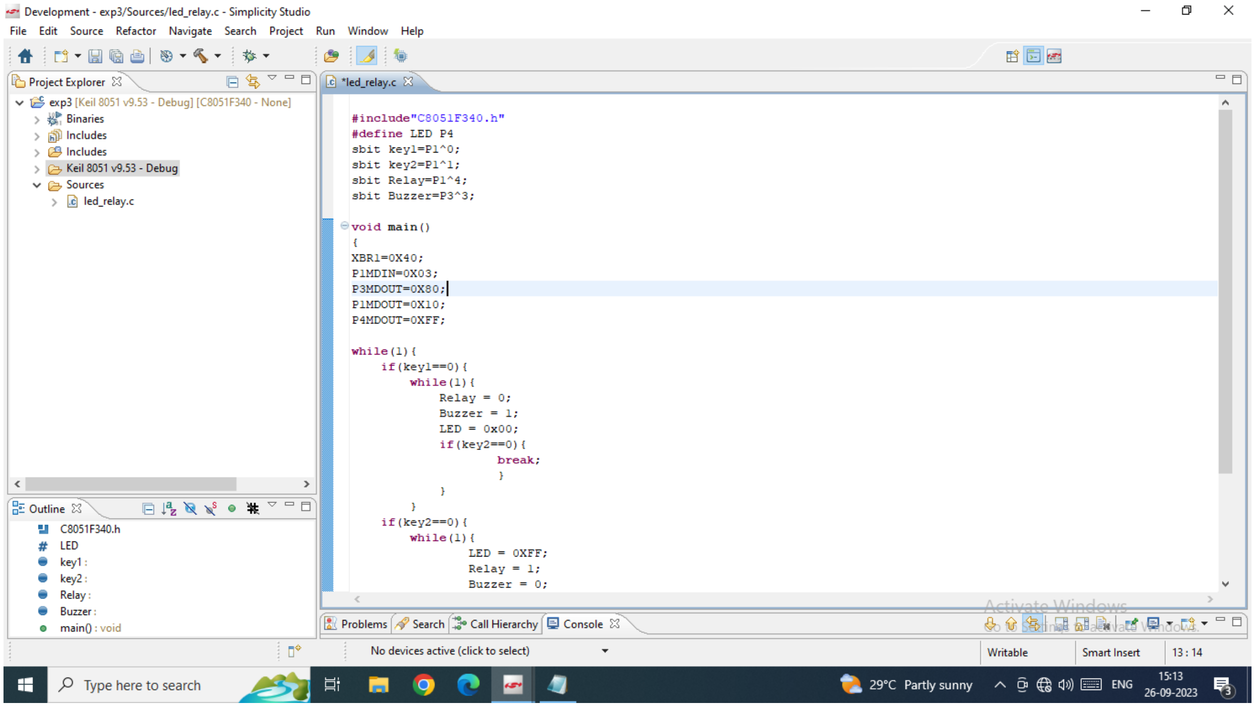
Study Question:

1. Explain the common anode and common cathode configuration of LED.
2. How many ports are available in C8051F340?
3. How to configure port as an input/output?
4. Explain Priority Crossbar Decoder.

Additional links:

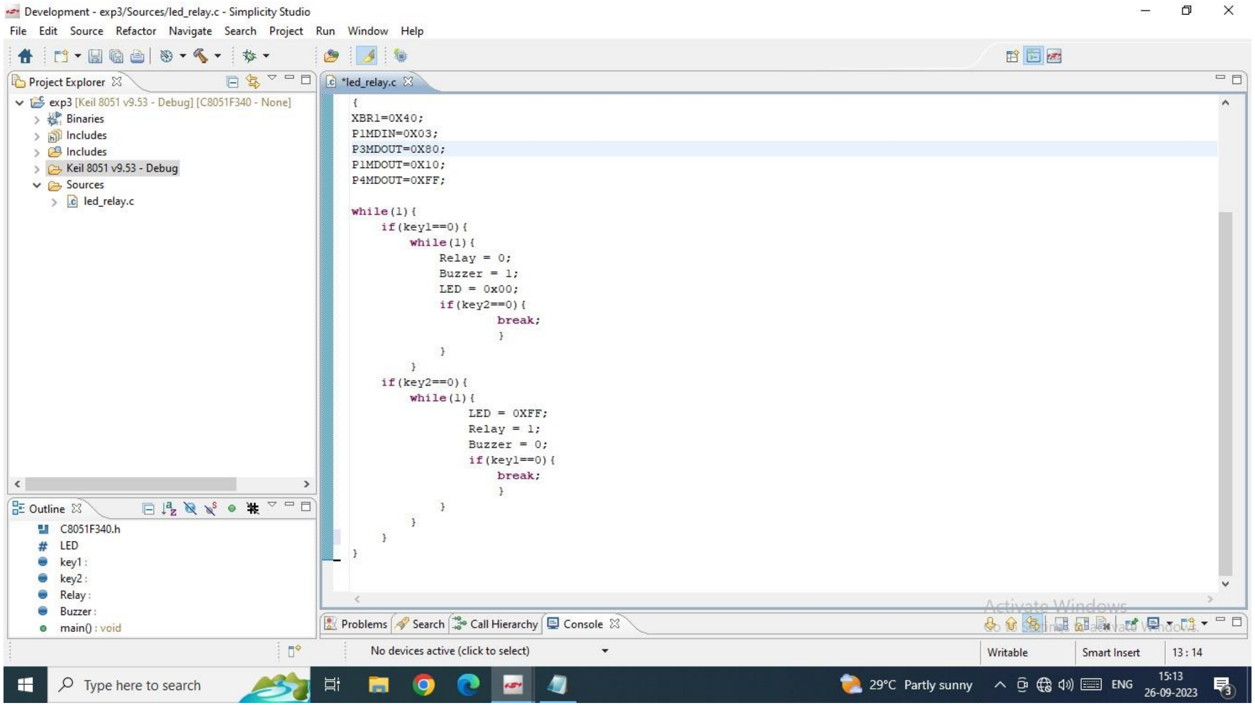
1. *https://www.silabs.com/documents/public/data-sheets/C8051F34x.pdf*
2. *https://aticleworld.com/interfacing-of-switch-and-led-using-the-8051/*

CODE:



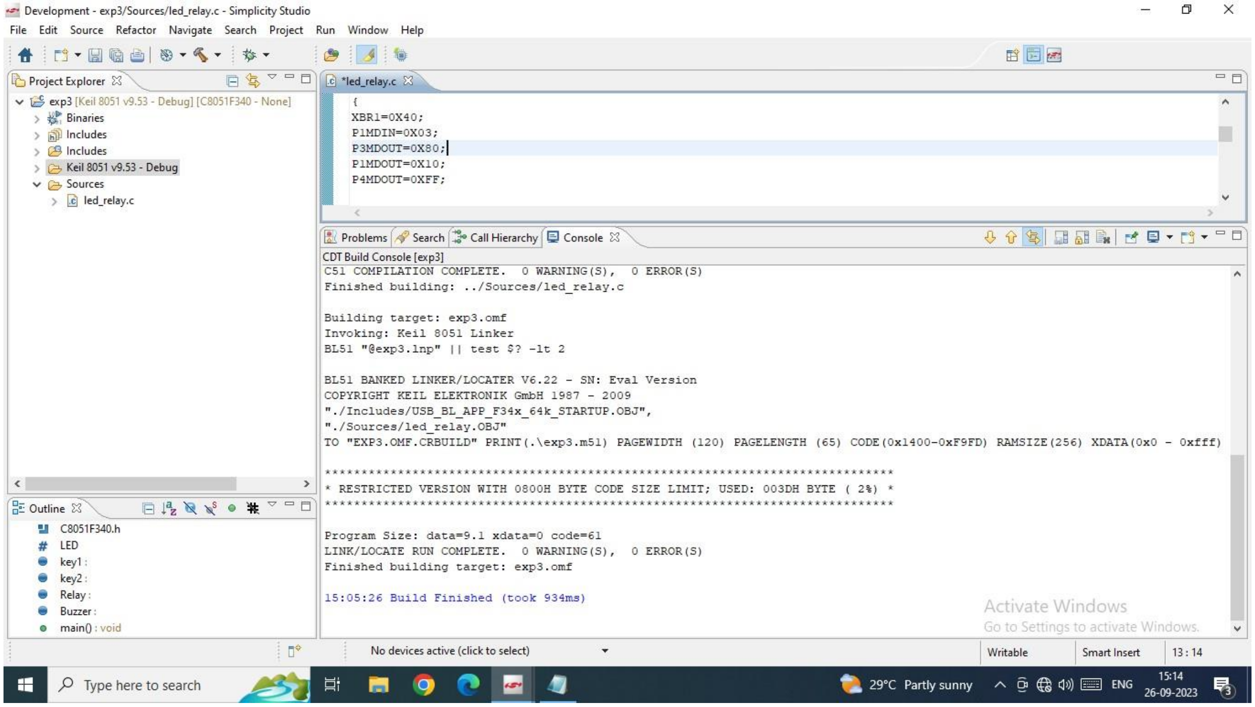
Development - exp3/Sources/led_relay.c - Simplicity Studio

```c
#include"C8051F340.h"
#define LED P4
sbit key1=P1^0;
sbit key2=P1^1;
sbit Relay=P1^4;
sbit Buzzer=P3^3;

void main()
{
XBR1=0X40;
P1MDIN=0X03;
P3MDOUT=0X80;
P1MDOUT=0X10;
P4MDOUT=0XFF;

while(1){
    if(key1==0){
        while(1){
            Relay = 0;
            Buzzer = 1;
            LED = 0x00;
            if(key2==0){
                    break;
                    }
            }
        }
    if(key2==0){
        while(1){
                LED = 0XFF;
                Relay = 1;
                Buzzer = 0;
```



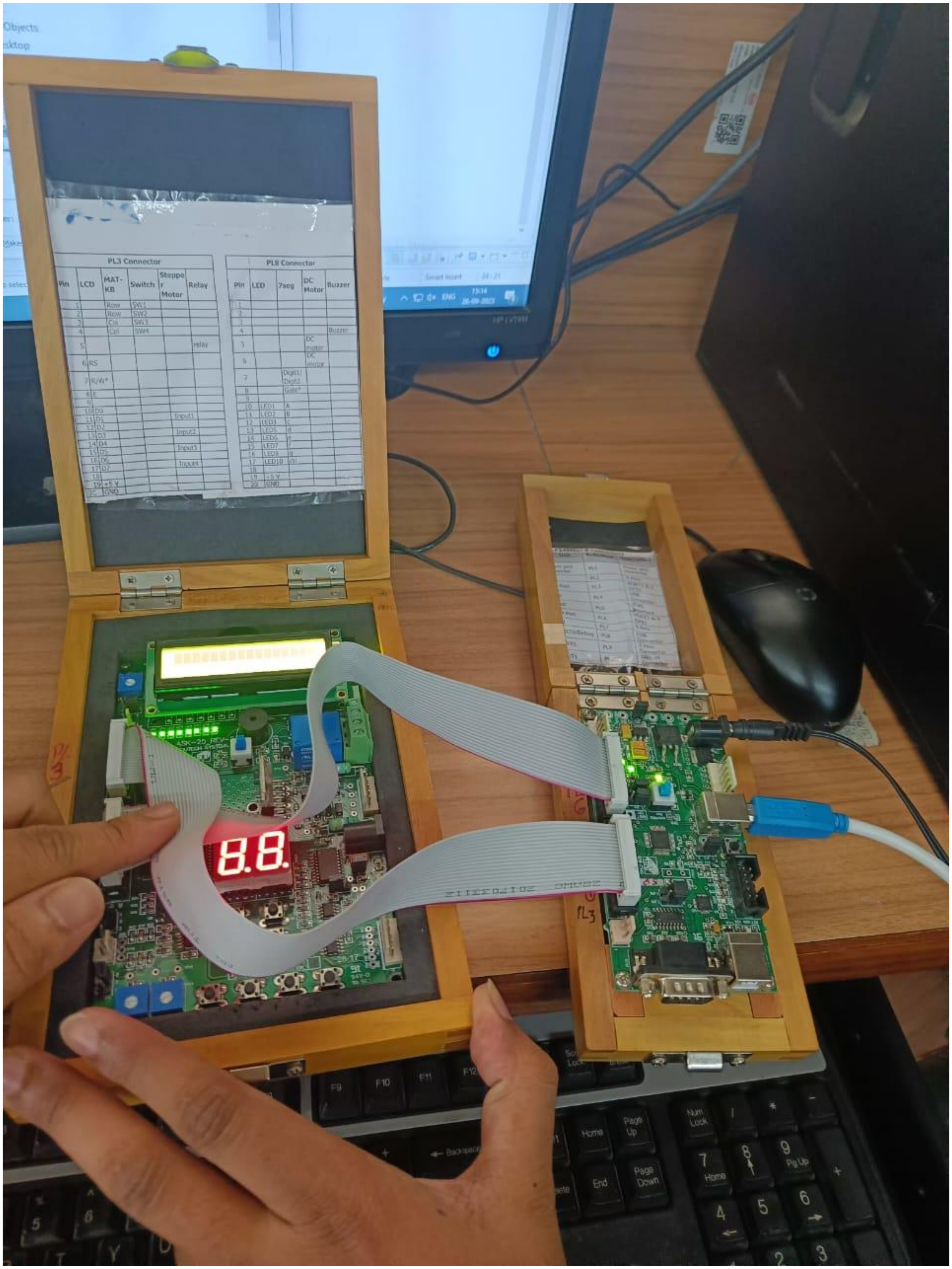
Development - exp3/Sources/led_relay.c - Simplicity Studio

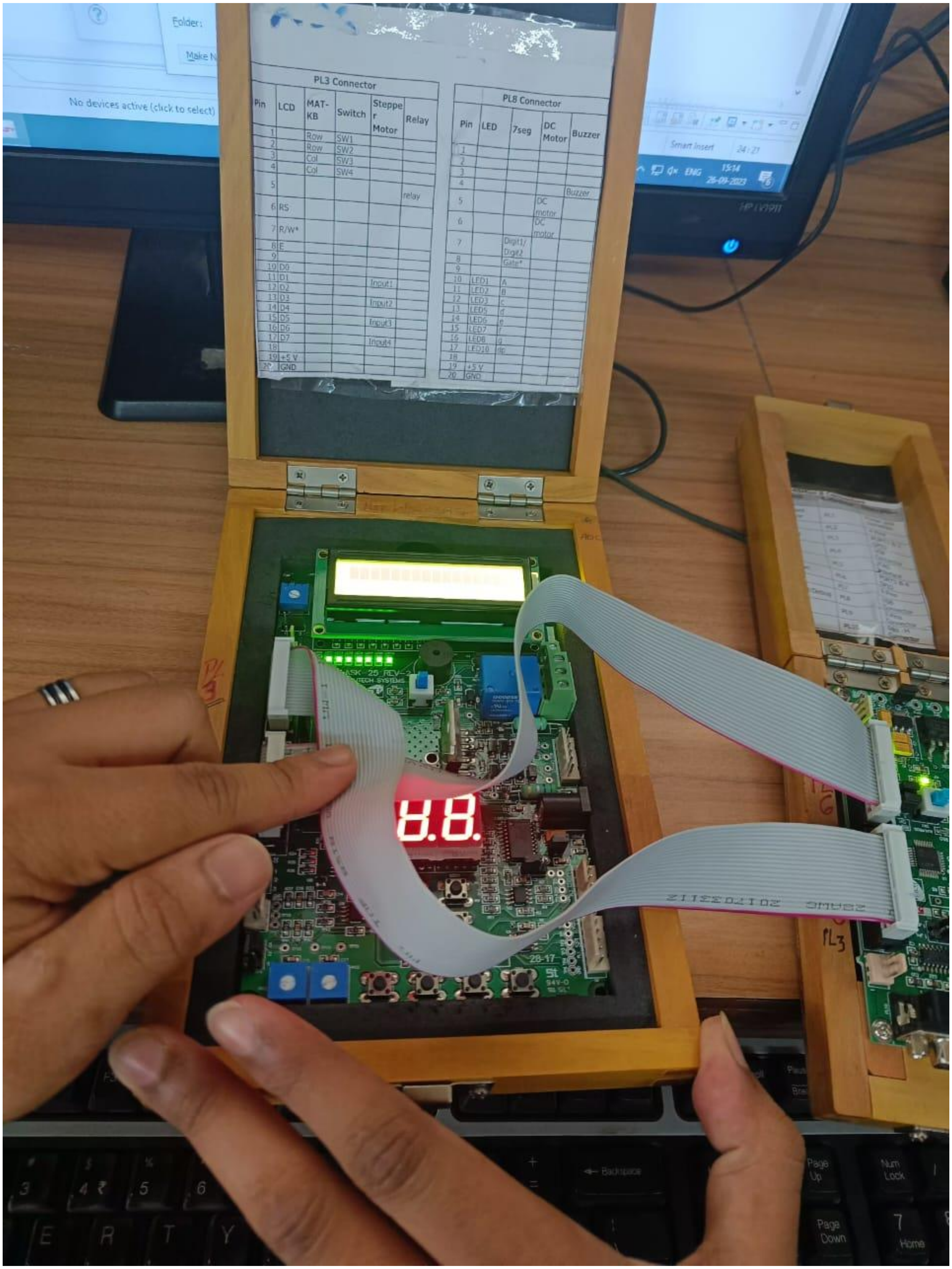
```c
{
XBR1=0X40;
P1MDIN=0X03;
P3MDOUT=0X80;
P1MDOUT=0X10;
P4MDOUT=0XFF;

while(1){
    if(key1==0){
        while(1){
            Relay = 0;
            Buzzer = 1;
            LED = 0x00;
            if(key2==0){
                    break;
                    }
            }
        }
    if(key2==0){
        while(1){
                LED = 0XFF;
                Relay = 1;
                Buzzer = 0;
                if(key1==0){
                    break;
                    }
            }
        }
    }
}
```



```
{
XBR1=0X40;
P1MDIN=0X03;
P3MDOUT=0X80;
P1MDOUT=0X10;
P4MDOUT=0XFF;
```

```
C51 COMPILATION COMPLETE.  0 WARNING(S),  0 ERROR(S)
Finished building: ../Sources/led_relay.c

Building target: exp3.omf
Invoking: Keil 8051 Linker
BL51 "@exp3.lnp" || test $? -lt 2

BL51 BANKED LINKER/LOCATER V6.22 - SN: Eval Version
COPYRIGHT KEIL ELEKTRONIK GmbH 1987 - 2009
"./Includes/USB_BL_APP_F34x_64k_STARTUP.OBJ",
"./Sources/led_relay.OBJ"
TO "EXP3.OMF.CRBUILD" PRINT(.\exp3.m51) PAGEWIDTH (120) PAGELENGTH (65) CODE(0x1400-0xF9FD) RAMSIZE(256) XDATA(0x0 - 0xfff)

******************************************************************************
* RESTRICTED VERSION WITH 0800H BYTE CODE SIZE LIMIT; USED: 003DH BYTE ( 2%) *
******************************************************************************

Program Size: data=9.1 xdata=0 code=61
LINK/LOCATE RUN COMPLETE.  0 WARNING(S),  0 ERROR(S)
Finished building target: exp3.omf

15:05:26 Build Finished (took 934ms)
```


PL3 Connector
PL8 Connector
8.8.



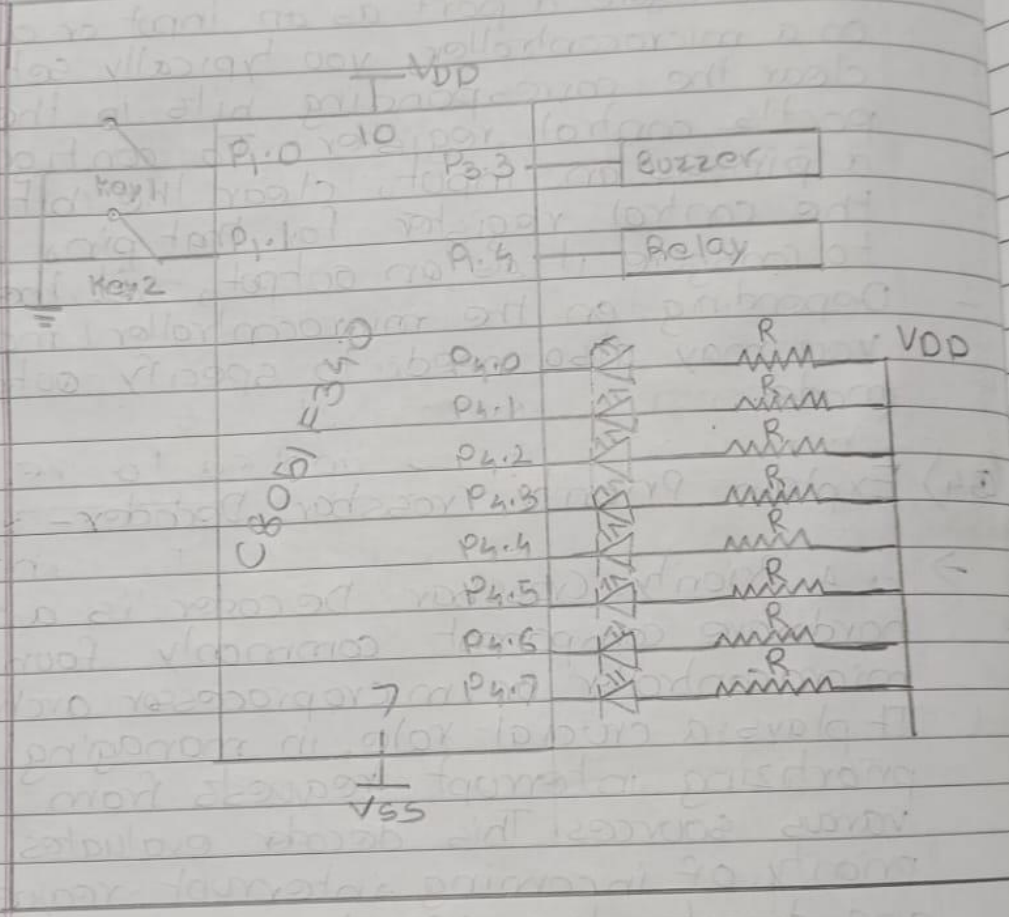
PL3 Connector

Pin	LCD	MAT-KB	Switch	Stepper Motor	Relay
1		Row	SW1		
2		Row	SW2		
3		Col	SW3		
4		Col	SW4		
5					relay
6	RS				
7	R/W*				
8	E				
9					
10	D0				
11	D1			Input1	
12	D2				
13	D3			Input2	
14	D4				
15	D5			Input3	
16	D6				
17	D7			Input4	
18					
19	+5 V				
20	GND				

PL8 Connector

Pin	LED	7seg	DC Motor	Buzzer
1				
2				
3				
4				
5			DC motor	Buzzer
6			DC motor	
7		Digit1/ Digit2		
8		Gate*		
9				
10	LED1	A		
11	LED2	B		
12	LED3	c		
13	LED5	d		
14	LED6	e		
15	LED7	f		
16	LED8	g		
17	LED10	dp		
18				
19	+5 V			
20	GND			

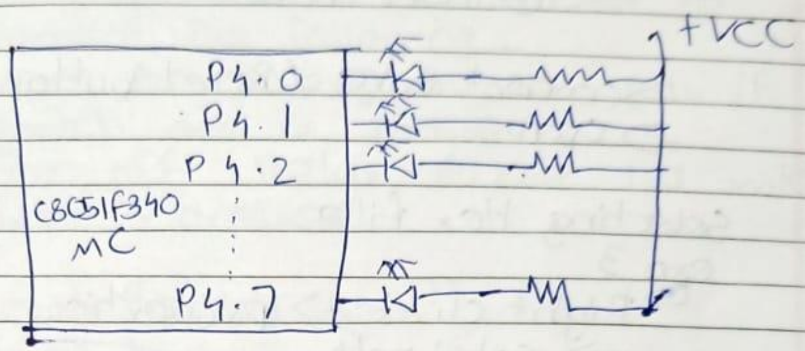
* Interfacing of LED, Relay, Buzzer
Diagram



PAGE No.
DATE 12/9/23

Exp 3

Interfacing diagram

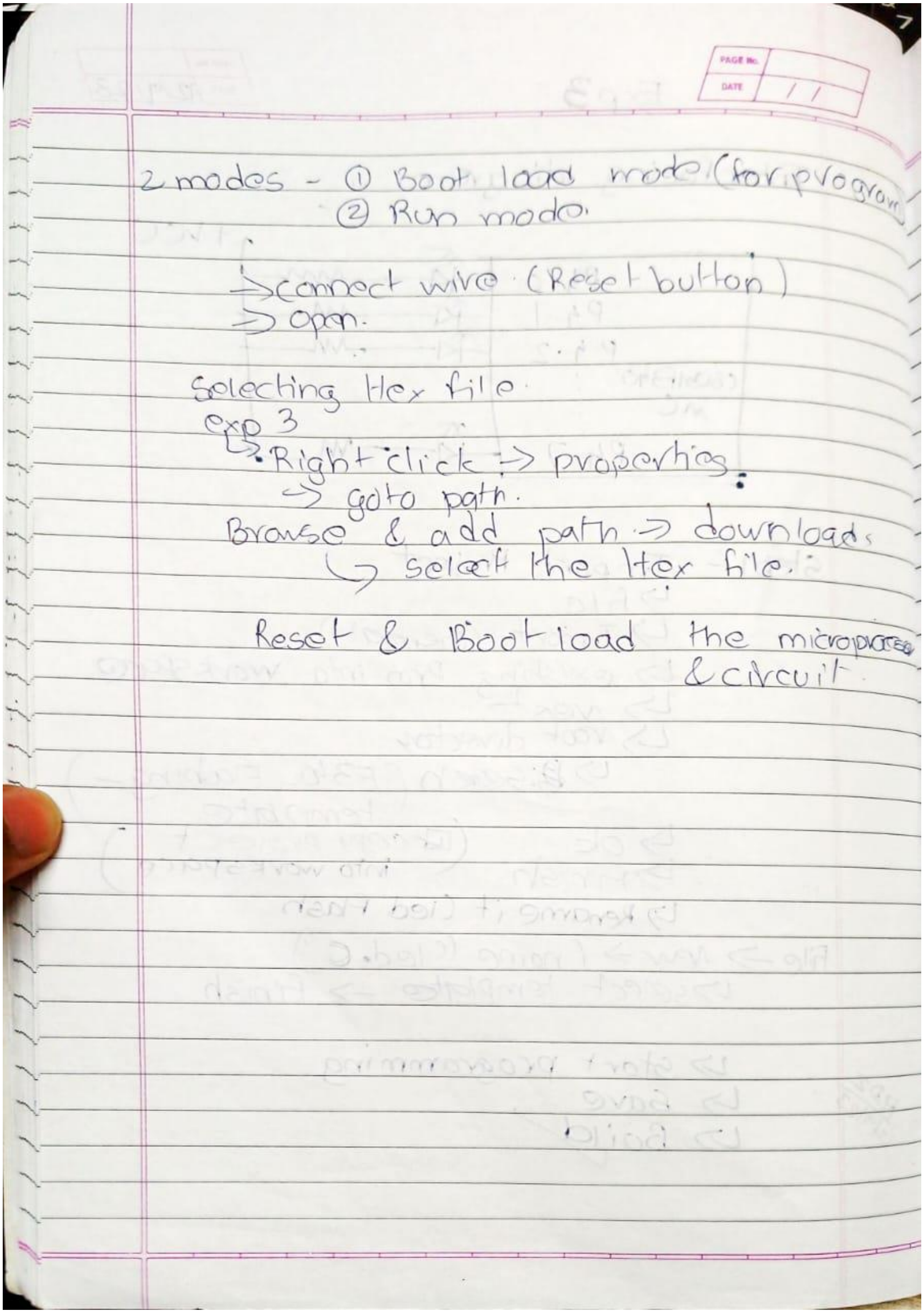


Step1 - Import Project
↳ file
↳ Import (General)
↳ existing Proj into workspace
↳ next
↳ root directory
↳ ~~B~~: search (F340_ Flashing - template)
↳ ok (☑ copy project into workspace)
↳ Finish.
↳ Rename it Cled Flash

File → New → (name "led.c")
↳ select template → finish

↳ start programming
↳ save
↳ Build

HDZ
12/9/23



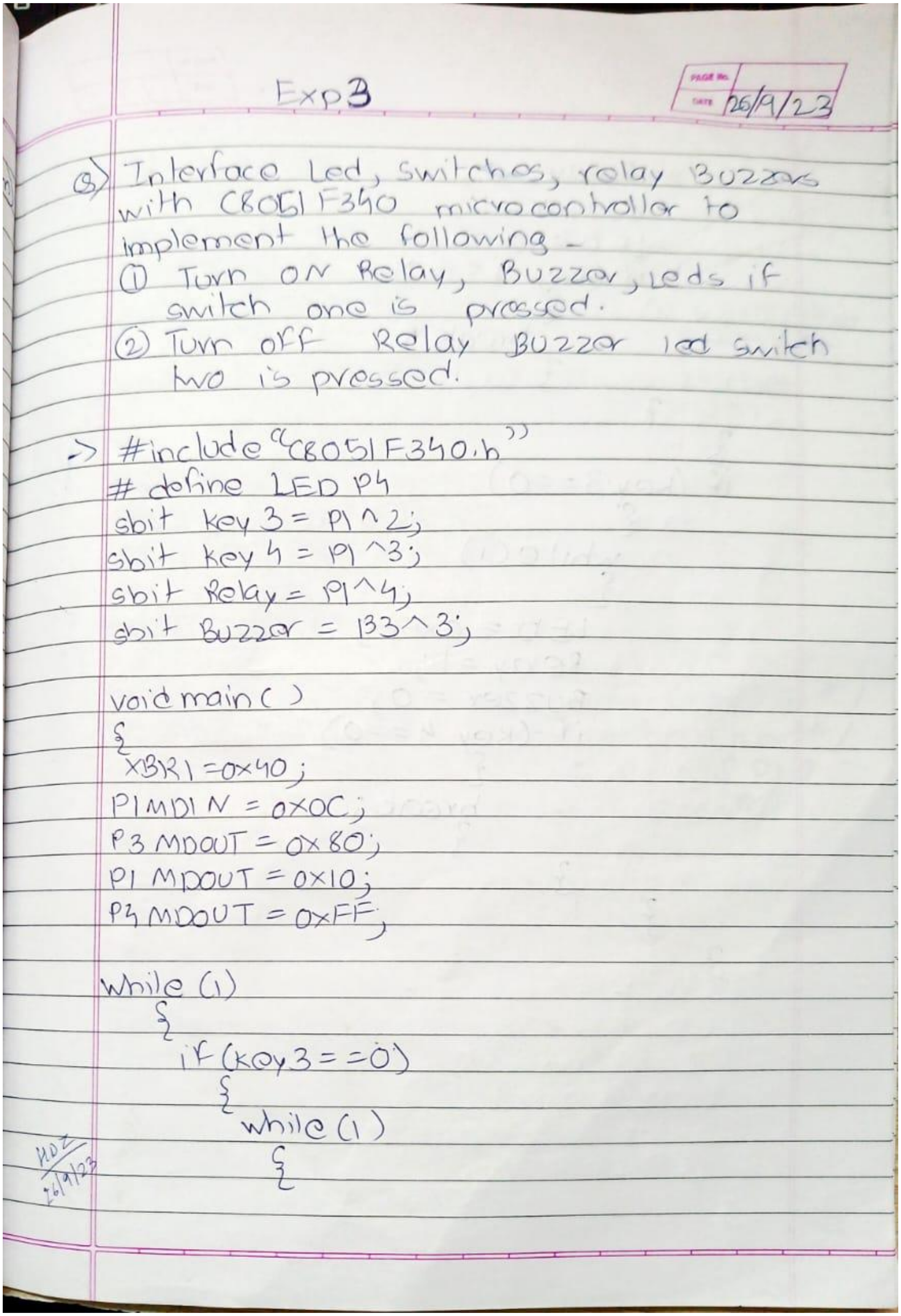
PAGE No.
DATE / /

2 modes - ① Boot load mode (for program
② Run mode.

→ connect wire (Reset button)
→ open.

Selecting Hex file.
exp 3
↳ Right click → properties.
→ goto path.
Browse & add path → download.
↳ select the Hex file.

Reset & Boot load the microprocess
& circuit.



Exp 3

PAGE No.
DATE 26/9/23

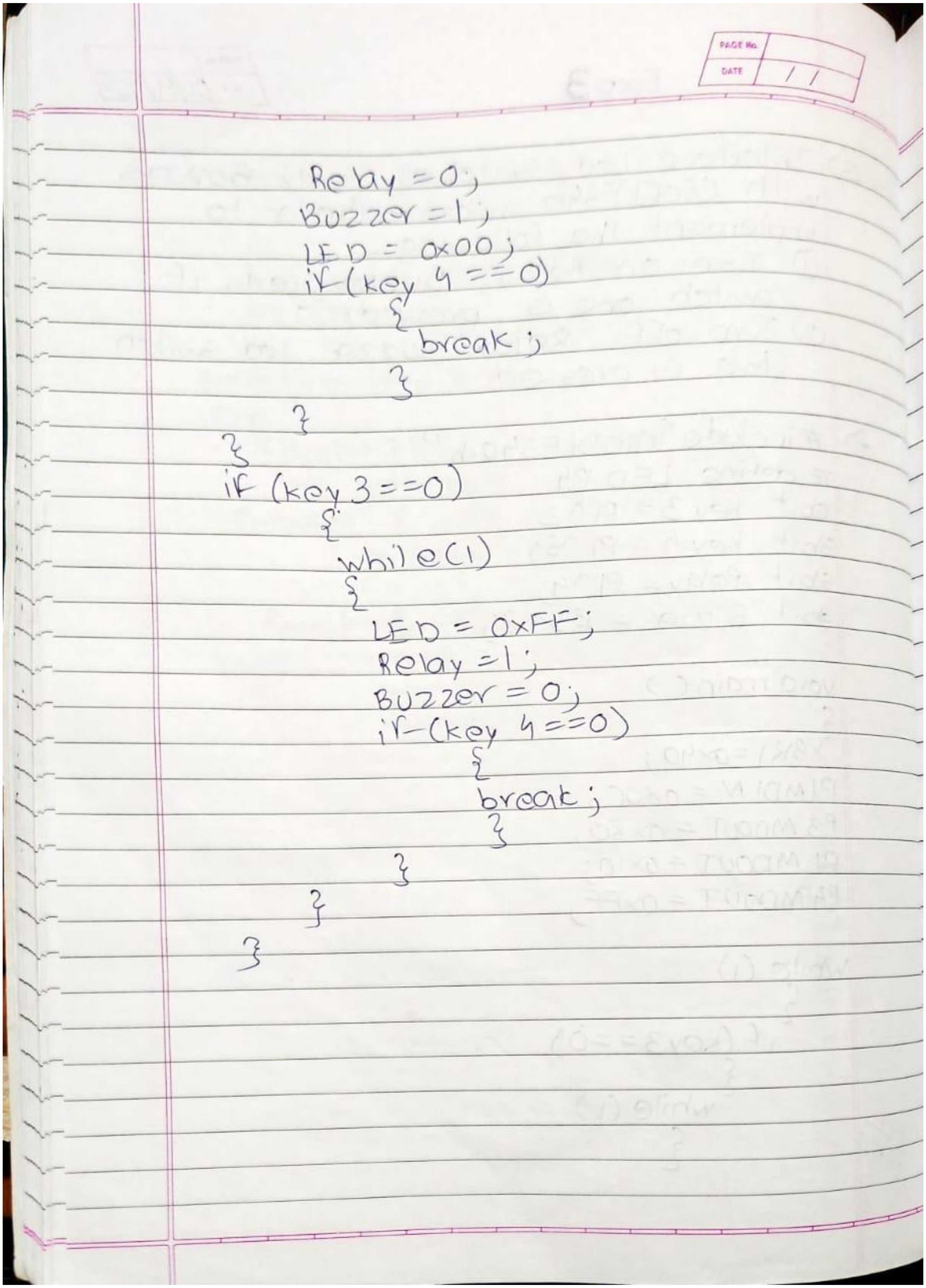
Q) Interface Led, switches, relay Buzzers with C8051F340 microcontroller to implement the following -

① Turn ON Relay, Buzzer, Leds if switch one is pressed.

② Turn off Relay Buzzer led switch two is pressed.

```
-> #include "C8051F340.h"
# define LED P4
sbit key 3 = P1 ^ 2;
sbit key 4 = P1 ^3;
sbit Relay = P1^4;
sbit Buzzer = P3^3;

void main ( )
{
 XBR1 =0x40;
P1MDIN = 0x0C;
P3 MDOUT = 0x 80;
P1 MDOUT = 0x10;
P4 MDOUT = 0xFF;

while (1)
   {
     if (key3 = =0)
        {
          while (1)
            {
```

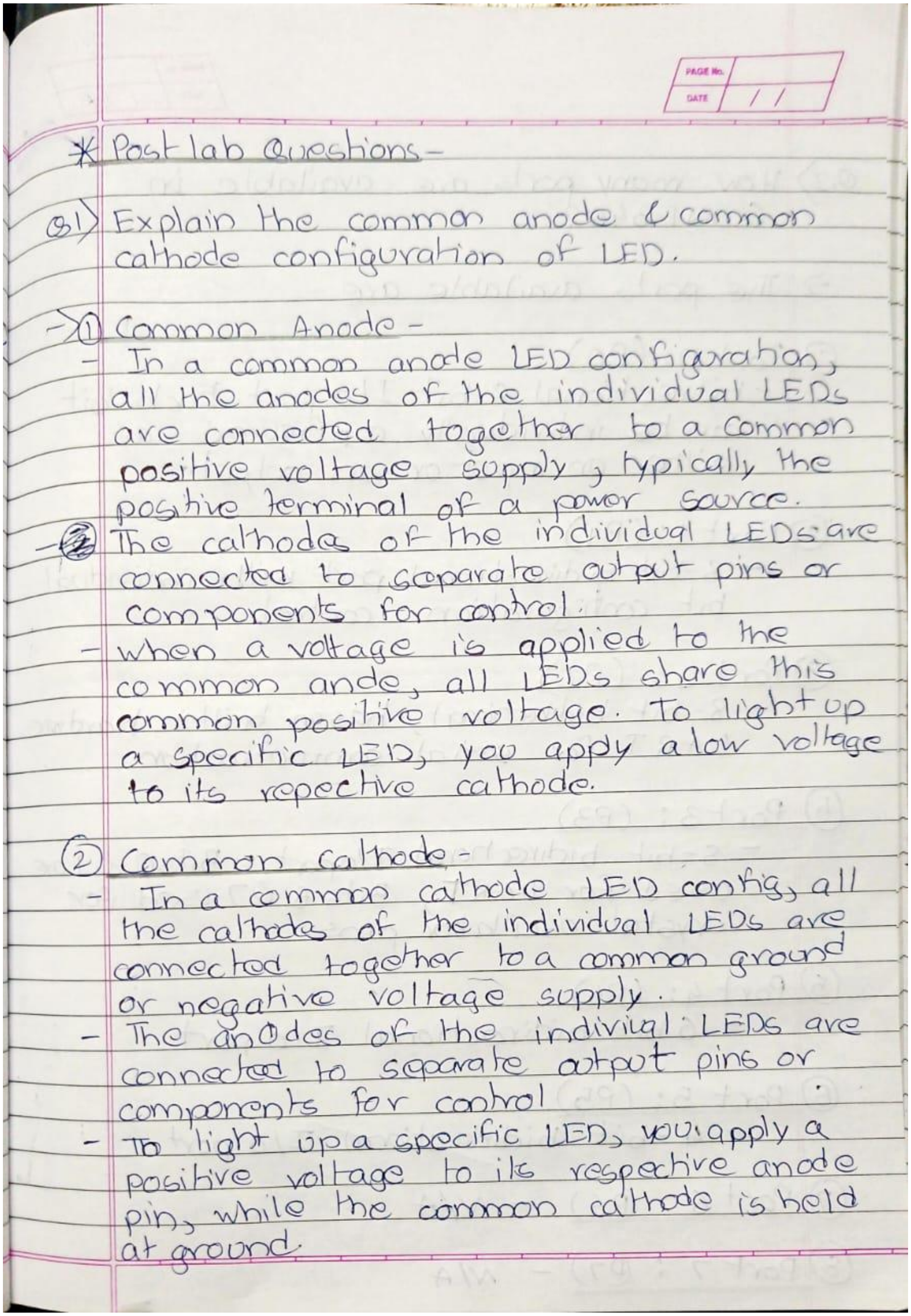
HOD
26/9/23



PAGE No.
DATE / /

```
        Relay =0;
        Buzzer =1;
        LED = 0x00;
        if(key 4 ==0)
              {
                break;
              }
      }
  }
  if (key 3==0)
     {
       while(1)
       {
         LED = 0xFF;
         Relay =1;
         Buzzer = 0;
         if(key 4==0)
               {
                 break;
               }
           }
       }
  }
```

POST LAB QUESTIONS:



PAGE No.
DATE / /

* Post lab Questions -

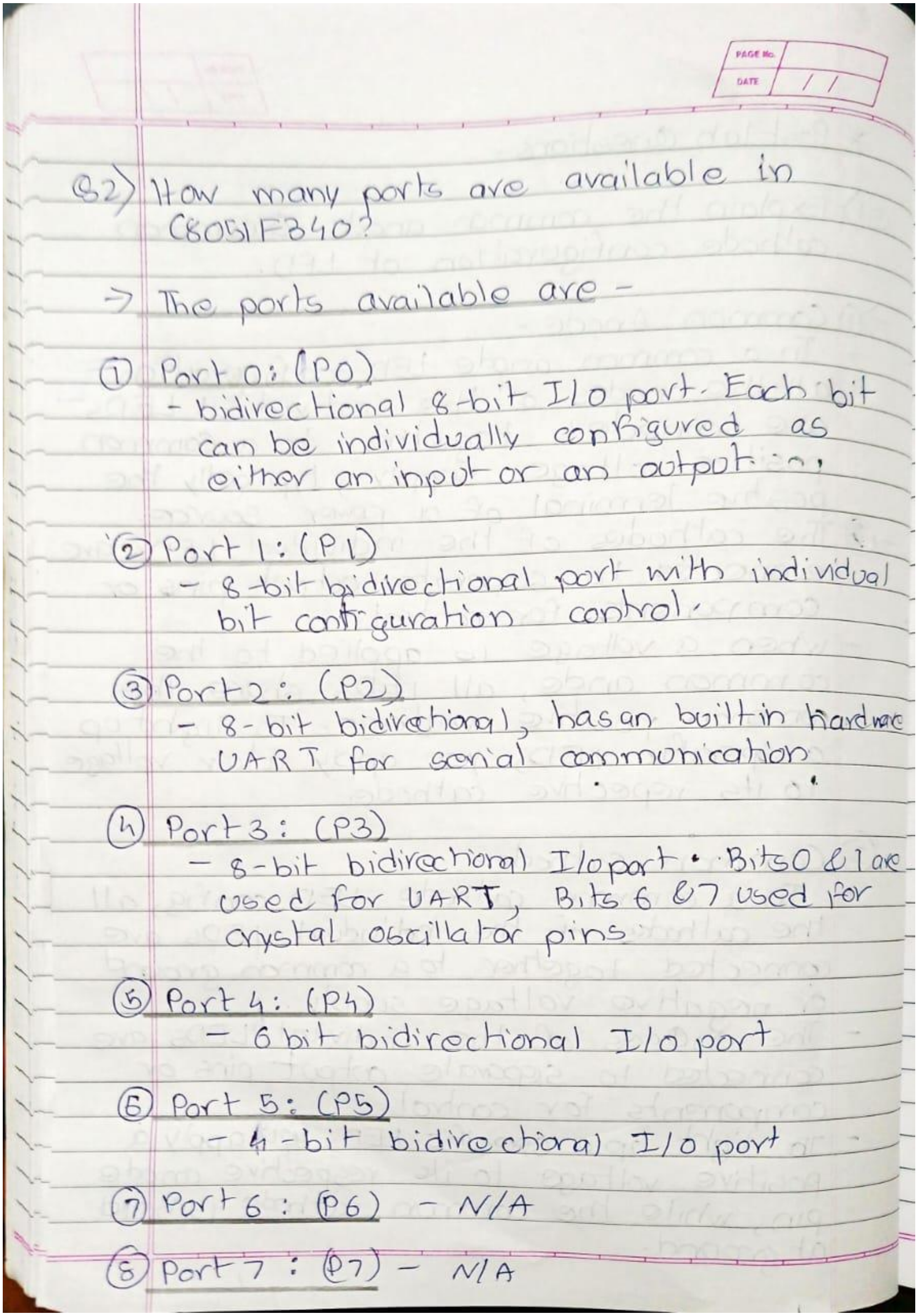
Q1) Explain the common anode & common cathode configuration of LED.

-> ① Common Anode -

- In a common anode LED configuration, all the anodes of the individual LEDs are connected together to a common positive voltage supply, typically the positive terminal of a power source.
- ② The cathodes of the individual LEDs are connected to separate output pins or components for control.
- when a voltage is applied to the common ande, all LEDs share this common positive voltage. To light up a specific LED, you appy a low voltage to its repective cathode.

② Common Cathode -

- In a common cathode LED config, all the cathodes of the individual LEDs are connected together to a common ground or negative voltage supply.
- The anodes of the indiviual LEDs are connected to separate output pins or components for control
- To light up a specific LED, you apply a positive voltage to its respective anode pin, while the common cathode is held at ground.



PAGE No.
DATE / /

Q2) How many ports are available in C8051F340?

→ The ports available are -

① Port 0: (P0)
- bidirectional 8-bit I/o port. Each bit can be individually configured as either an input or an output.

② Port 1: (P1)
- 8-bit bidirectional port with individual bit configuration control.

③ Port 2: (P2)
- 8-bit bidirectional, has an built in hardware UART for serial communication.

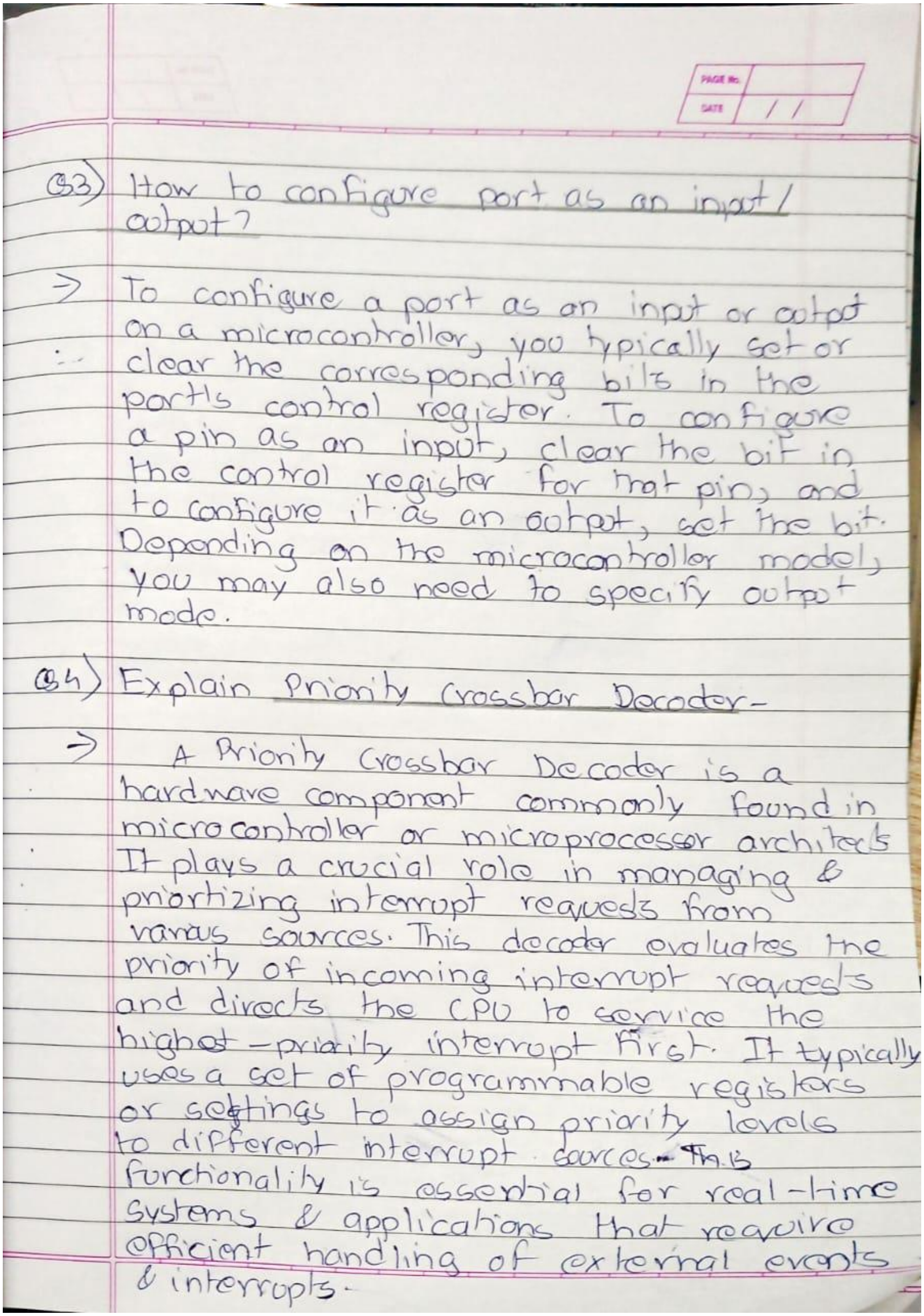
④ Port 3: (P3)
- 8-bit bidirectional I/o port. Bits 0 & 1 are used for UART, Bits 6 & 7 used for crystal oscillator pins.

⑤ Port 4: (P4)
- 6 bit bidirectional I/o port

⑥ Port 5: (P5)
- 4-bit bidirectional I/o port

⑦ Port 6: (P6) - N/A

⑧ Port 7: (P7) - N/A



PAGE No.
DATE / /

Q3) How to configure port as an input/ output?

-> To configure a port as an input or output on a microcontroller, you typically set or clear the corresponding bits in the port's control register. To configure a pin as an input, clear the bit in the control register for that pin, and to configure it as an output, set the bit. Depending on the microcontroller model, you may also need to specify output mode.

Q4) Explain Priority Crossbar Decoder-

-> A Priority Crossbar Decoder is a hardware component commonly found in microcontroller or microprocessor architects It plays a crucial role in managing & priortizing interrupt requests from various sources. This decoder evaluates the priority of incoming interrupt requests and directs the CPU to service the highest-priority interrupt first. It typically uses a set of programmable registers or settings to assign priority levels to different interrupt sources. This functionality is essential for real-time systems & applications that require efficient handling of external events & interrupts.