Department of

# Electrical Engineering

## LAB MANUAL
## MC & PLC LAB

## B.Tech– VI Semester


## KCT College OF ENGG AND TECH.
## VILLAGE FATEHGARH
## DISTT.SANGRUR

# TABLE OF CONTENTS


| S No. | Topic |
|---|---|
| **1.** | Exp No. 1: Study of 8051/8031 Micro controller kits |
| **2.** | Exp No. 2: Write a program to add two numbers lying at two memory locations and display the result. |
| **3.** | Exp No. 3: Write a program for multiplication of two numbers lying at memory location and display the result |
| **4.** | Exp No. 4: Write a program to check a number for being ODD or EVEN and show the result on display |
| **5.** | Exp No. 5: Write a program to split a byte in two nibbles and show the two nibbles on display |
| **6.** | Exp No. 6: Write a Program to arrange 10 numbers stored in memory location in Ascending and Descending order |
| **7.** | Exp No. 7: Write a program to find a factorial of a given number |
| **8.** | Exp No. 8: Study of Interrupt structure of 8051/8031 micro controllers |
| **9.** | Exp No. 9: Write a program to show the use of INT0 and INT1. |
| **10.** | Exp No. 10: Write a program of Flashing LED connected to port 1 of the Micro controller |
| **11.** | Exp No. 11: Write a program to generate a Ramp waveform using DAC with micro controller. |
| **12.** | Exp No. 12: Write a program to interface the ADC |
| **13.** | Exp No. 13: Write a program to control a stepper motor in direction, speed and number of steps. |
| **14.** | Exp No. 14: Write a program to control the speed of DC motor. |
| **15.** | Exp No. 15: Interfacing of high power devices to Micro-controller port-lines, LED, relays and LCD display. |
| **16.** | EXP NO.16: Programmed Logic Controller (PLC) Experiment for CHEN 4570 – Instrumentation and Process Control Programmed Logic Controller (PLC) Experiment for CHEN 4570 – Instrumentation and Process Control |

## Experiment No:1

Study of 8051/8031 Micro controller kits

**AIM:** - To study 8051 micro controller kit.

**APPARATUS**: - 8051 Micro controller kit.

**THEORY: -**
The system has got 8031/8051/89C51 as central processing unit. The clock of system is 10MHz and is generated from a crystal of 10MHz.

**MEMORY: -**
VMC-8031/8051/89C51 provide 32KB of RAM using 62256 chip and 32 KB of EPROM for monitor. The various chips which can be used 2732, 2764, .27128, 27256, 6116 and 6264. There is one memory space provided on VMC8031/8051/89C51. This one space can be defined any address slots from 3000 - FFFF depending upon the size of the memory chip to be used.

**1/0 DEVICES:-**
The various 1/0 chips used in VMC-8031/8051/89C51 are 8279, 8255, 8251 & 8253. The functional role of all these chips is given below: -

**8279 (KEYBOARD & DISPLAY CONTROLLER): -**
8279 is a general-purpose programmable keyboard and display 1/0 interface device designed for use with the 8031/8051/89C51 microprocessor. It provides a scanned interface to 28 contact key matrix provided in VMC-8031/8051/89C51 and scannd interface for the six seven segment displays.

**8255 (Programmable Peripheral Interface): -**
8255 is a programmable peripheral interface (PPI) designed to use with 8251/8231/89C51 Microprocessor. This basically acts as a general purpose 1/0 device to interface peripheral equipment to the system bus. It is not necessary to have an external logic to interface with peripheral devices since the functional configuration of 8255 is programmed by the system software. Any 1/0 'combination of port A, Port B, Port C upper and lower can be defined using the appropriate software commands.

**8251(USART): -**
This chip is a programmable communication interface and is used as a peripheral device. This device accepts data format and convert them into serial data characters for CPU. This chip will signal the CPU. Whenever it can accept a new character for CPU.

**8253(Programmable Interjal Timer):-**

This chip is a programmable interval Timer/Counter and can be used for the generation of accurate time delays under software control. Various other functions that can be implemented with this chip are programmable rate generator, Even Counter, Binary rate, Multiplier, Real Time Clock etc. This chip
has got three in dependent 16 bit counter each having a count rate of up to 2MHz.The first Timer/Counter (i.e. Counter 0) is being used for single step operation. However, its connections are also brought at connector space CN4. For single step operation CLK0 signal of counter 0 is getting a clock frequency of 1.535 MHz. The counter 1 is used to generate clock for 8251. The clock1 is also Jed with 1.535MHz.

**DISPLAY:-**

VMC-8031/8051/89C51 provide six digits of seven segment display. Four digits are for displaying the address of any location or name of any register, whereas the rest of two digits are meant for displaying the contents of memory location or of a register. All the six digits of the display are in hexadecimal notation.

BATTERY BACK-UP: - The VMC-8031/8051/89C51 provide a battery back up for RAM area.

**COMMAND DESCRIPTION: -**

**KEYBOARD DESCRIPTION: -**

VMC-8051 has 28 keys and six seven segment display to communicate with outside world. As VMC-8051 switches on, a message 'UP-51' is displayed after pressing reset. The Keyboard is as shown.

**RESET: -** Reset the system.

**SHIFT: -** Provides second level command to some keys.

**GO: -** To execute the programme.

**SI: -** To execute the program in single step mode.

**EXREG:** - Examine Register; allows user to examine and modify the contents of different registers.

**EXMEM:** - Examine Program Memory; allows user to examine/modify any data memory location.

**PRE: -** Previous is used as an intermediate terminator in case of Examine Memory. It decrements the PC contents and writes the contents of data field to the address displayed in address location.

**NEXT: -** Increment is used as an intermediate terminator in case of Examine Memory, Examine Register etc. It increments the PC contents and writes the data lying in the data field at the location displayed at address field.

11.19: - Terminator is used to terminate the command and write the data in the data field at the location displayed in the address field.

**BM.DD:** - Allows user to move a block of data memory to another data memory.

**BM.PP:** - Allows user to move a block of Program memory to another Program Memory.

**BM.DP:** - Allows user to move a block of Data Memory to Program Memory. **FILL: -** Allows user to fill RAM area with a constant.

**INS: -** Inserts one or more data bytes from in the user's program/ data area. **DELD: -** Deletes one or more data bytes from the user's program/data area.

**SETBR**: -Set Break point allows user to set a break point anywhere in the user program.

**CLRBR: -** Clear Break point allows user to clear a break point anywhere in the user program.

**ENBR: -** Enable break point allows user to enable a breakpoint anywhere in the user program.

**DPBR: -** Display Break point allows user to see the address where the break point was set.

**P.PRG: -** This Key is for further expansion.

**SERIAL: -** This key is used for Serial Communication with PC. -All command are followed by a set•or numeric parameters separated by PREY, NEXT and "." (Execute) to work as delimiters.

## Experiment No:2

Write a program to add two numbers lying at two memory locations and display the result.

**AIM:** Write a program to add 2 nos. lying at two-memory location and display the result

**APPARATUS:** Micro controller Kit

**PROGRAM—**

| Memory location | OP code | Label | Mnemonics | Operands | Comments |
|---|---|---|---|---|---|
| 2000 | 90,30,00 | | MOV | DPTR#3000 | Move 16-bit address to DPTR |
| 2003 | E0 | | MOVX | A, @DPTR | Move the contents of memory location, which is saved in DPTR, to the ACC. |
| 2004 | F8 | | MOV | $R_0$,A | Move the contents of acc. to $R_0$ |
| 2005 | A3 | | INC | DPTR | Increment the DPTR |
| 2006 | E0 | | MOVX | A, @DPTR | Move the contents of memory location, which is saved in DPTR, to the ACC. |
| 2007 | 28 | | ADD | A,$R_0$ | ADD the contents of acc.& $R_0$. |
| 2008 | A3 | | INC | DPTR | Increment DPTR. |
| 2009 | F0 | | MOVX | @DPTR,A | Move the contents of acc. to the memory location saved in DPTR. |
| 200A | 80,FE | HERE | SJMP | HERE | End of the program. |

**PROCEDURE: -**

1. Connect the power supply to the Micro controller Kit.
2. Load the data first
3. Then load the program starting from 2000H with opcodes.
4. To execute the program press the keys In the following sequence: RESET---GO----1st MEMORY LOCATION----FILL.
5. 'E' will be displayed indicating execution of program.
6. For getting the result ,press the keys as follows:
7. RESFT---EXMEM---MEMORY LOCATION WHERE THE RESULT IS STORED- NEXT
8. Switch off Die power supply.

**DATA:**
1. 3000 —23H
2.3001 —33H

**RESULT:**
3002 — 56H

**PRECAUTIONS:**
1. Never touch the IC'S during operation.
2. Opcodes must be fed properly and carefully.
3. Program must be executed before result before result is displayed.

## Experiment No:3

**AIM:-** Write a program to multiply two numbers lying at two memory location & display the resin.

**APPARATUS:-** Micro controller kit.

**PROGRAM:**

| Memory location | OP code | Label | Mnemonics | Operands | Comments |
|---|---|---|---|---|---|
| 2000H | 90,30,00 | | MOV | DPTR, #3000 | Move 16 -bit address to DPTR |
| 2003H | E0 | | MOVX | A, @DPTR | Move the contents of memory location, which is saved in DPTR, to the ACC. |
| 2004H | F5, F0 | | MOV | 0F0,A | Move data from A to B having address 0F0 |
| 2006H<br>2007H | A3<br>E0 | | INC<br>MOVX | DPTR<br>A, @DPTR | Increment DPTR by 1<br>Move the contents of memory location, which is saved in DPTR, to the ACC. |
| 2008H | A4 | | MUL | AB | Multiply A&B |
| 2009H | A3 | | INC | DPTR | Increment DPTR by 1 |
| 200AH | F0 | | MOVX | @DPTR, A | Move the contents of acc. to the memory location saved in DPTR. |
| 200BH | A3 | | INC | DPTR | Increment DPTR by 1 |
| 200CH | E5, F0 | | MOV | A, 0F0H | Move data from B having address 0F0 to A |
| 200EH | F0 | | MOVX | @DPTR, A | Move the contents of acc. to the memory location saved in DPTR. |
| 200FH | 80,FE | HERE | SJMP | HERE | End program |

**THEORY:**

In multiplying two numbers In 80511 the use of register A and B is required since the multiplication instruction work only with these two registers. Multiplications operations uses register A & 0 es both source and destination addresses for the operation. The unsigned number In register B multiplies the unsigned number in register A, as follows
MUL AB: multiply A by B
By this, low order byte of the product in A and high order byte is stored in B. The result Is a 16-bit data. There Is no Instruction for moving the data In register B so in program we write the address (RAM) of B which Is OFO it is direct
addressing mode and by using this Instruction data 2CH are loaded in the register B whose address is given.

**PROCEDURE:-**

1. Connect the power supply to the Micro controller kit.
2. Load the program starting from 2000h with opeodes.
3. To execute a program press the keys in the following soquence. RESET--- 00--o Ist MEMORY LOCATION---TERMINATE.
4. 'E' will be displayed indicating execution of program.
5. For getting the result, press the keys as follows: ReSET---EXMEM—MEMORY LOCATION WHERE RESULT IS STORED---*NEXT
6. Switch off the power supply.

**DATA:** 3000: FFH
3001: 2CH

**RESULT:**
3002: D4H
3003: 2BH

**PRECAUTION:**

1. Supply and ground pin must be connected properly.
2. Never touch the IC's during operation.
3. Opcodes must be fed property and carefully.
4. Program must be executed before result is displayed.

## Experiment No:4

**AIM:** Write a program to check a no. for being ODD or EVEN and show the result on Display.

**APPARATUS**: Micro controller kit, Manuel

**PROGRAM:**

| MEMORY LOCATION | OP-CODE | LABEL | MNEMONICS | OPERANDS | COMMENTS |
|---|---|---|---|---|---|
| 2000H | 90, 30, 00 | | MOV | DPTR, #3000H | Move 16-bit address to DPTR |
| 2003H | E0 | | MOVX | A, @DPTR | Move the contents of memory location which is saved in DPTR, to the ACC. |
| 2004H | 54, 01 | | ANI | A#01H | AND the contents of acc. with 01 |
| 2006H | 60, 05 | | JZ | NEXT | Jump if the above result in acc. =0 to next |
| 2008H | 74, 01 | | MOV | A#01H | Move 01 to acc. to show that no. is an odd no. |
| 200AH | F0 | | MOVX | @DPTR, A | Move the contents of acc. to the memory location saved in DPTR. |
| 200BH | 80, FE | HERE | SJMP | HERE | End of the program if no. is ODD |
| 200DH | 74, 00 | NEXT | MOV | A, #00H | Move 00 to acc. to show that no. is an EVEN no. |
| 200FH | F0 | | MOVX | @DPTR, A | Move the contents of acc. to the memory location saved in DPTR. |
| 2010H | 80, FE | HERE | SJMP | HERE | End of the program if no. is EVEN |

**DATA:** 1) 40
2)55

## Experiment No:5

**AIM**: WRITE a program to spilt a byte into two nibble and show the two nibbles on Display.

**APPARATUS**: Micro controller kit

**PROGRAM:**

| MEMORY LOCATION | OP-CODE | LABEL | MNEMONICS | OPERANDS | COMMENTS |
|---|---|---|---|---|---|
| 2000H | 90, 30, 00 | | MOV | DPTR, #3000H | Move 16-bit address to DPTR |
| 2003H | E0, | | MOVX | A, @DPTR | Move the contents of memory location, which is saved in DPTR, to the ACC. |
| 2004H | F8 | | MOV | $R_0$, A | Store the no. in Reg. $R_0$ |
| 2005H | 54, F0 | | ANL | A, # F0H | AND the contents of acc. with F0 or Mask off the lower nibble |
| 2007H | C4 | | SWAP | A | Make MSB digit=LSB digit |
| 2008H | A3 | | INC | DPTR | Increment DPTR |
| 2009H | F0 | | MOVX | @DPTR, A | Move the contents of acc to the memory location saved in DPTR=3001. |
| 200AH | E8 | | MOV | A,$R_0$ | Load the no. back in acc. |
| 0BH | 54,0F | | ANL | A, # 0FH | AND the contents of acc. with 0F or Mask off the upper nibble |
| 200DH | A3 | | INC | DPTR | Increment DPTR |
| 200EH | F0 | | MOVX | @DPTR, A | Move the contents of acc to the memory location saved in DPTR=3002. |
| 200FH | 80,FE | HERE | SJMP | HERE | End of the program. |

**Data :** 93

**RESULT:** 09

## Experiment No:6(A)

**AIM: -**Write a pro ram to sort the numbers in ascending order.

**Apparatus: -** Top viewer simulator, Personal computer.

**Program:**

```
        org Oh
        mov r5,#09h
start:  mov r0,#60h
        mov r7,#09h.
back:   mov r2,00
        mov a,@r0
        mov 0f0h,a
        inc r0
        mov a,@r0
        cjne a,0f0h,next
        ajmp skip
next:   jnc skip
        mov r0,02h
        mov @r0,a inc r0
        mov a,0f0h
        mov @r0,a
skip:   djnz 07h,back
        djnz 05h,
start:  end
```

**Data: number**

60h-73h
61 h-22h
62h-55h
63h-68h
64h-31h
65h-82h
66h-11 h
67h-95h
68h-25h
69h-09h

**Reasult:**

60h-09h
61h-11h
62h-22h
63h-25h
64h-31 h
65h-55h
66h-68h
67h-73h
68h-82h
69h-95h

## Experiment No:6(B)

**AIM: -**Write a program to sort the numbers in descending order.

**Apparatus: -** Top viewer simulator, Personal computer.

**Program:**

```
        org Oh
        mov r5,#09h
start:  mov r0,#60h
        mov r7,#09h.
back:   mov r2,00
        mov a,@r0
        mov 0f0h,a
        inc r0
        mov a,@r0
        cjne a,0f0h,next
        ajmp skip
next:   jc skip
        mov r0,02h
        mov @r0,a inc r0
        mov a,0f0h
        mov @r0,a
skip:   djnz 07h,back
        djnz 05h,
start:  end
```

**Data: number**

60h-10h
61 h-50h
62h-70h
63h-90h
64h-11h
65h-31h
66h-51 h
67h-71h
68h-92h
69h-98h

**Reasult:**

60h-98h
61h-92h
62h-90h
63h-71h
64h-70 h
65h-51h
66h-50h
67h-31h
68h-11h
69h-10h

## Experiment No:7

**AIM: -** Write a program to find a factorial of a given number.

**Apparatus: -** Top viewer simulator, Personal computer.

**Program:**

```
        org Oh
        mov r1,#05h
        mov r5,#04h
        mov a,r1
again: dec r1
        mov 0f0h,rl
        mul ab
        djnz r5,again
        mov r3,0f0h
        mov r4,a
        end
```

**Data:** number : 5h

**Result:**

A=78
B (0f0h)=00
R4=78
R3=00

## Experiment No:8

**AIM: -** Study of Interrupt structure of 8051/8031 micro controllers

.

**Apparatus: -** Top viewer simulator, Personal computer.

**Theory:**

- Whenever the micro controller is executing a program an if a user wants service to an 1/0 device then an external asynchronous input would inform the micro controller that it should complete the execution of current instruction and then fetch a new routine that will service the requesting 1/0 device. Once, the 1/0 device is serviced, the micro controller. resumes operation from the point whenever it had stopped. The external asynchronous input applied to the micro controllers is an Interrupt.
- The interrupts may be generated by internal chip operations or they may be provided by external sources. An interrupts causes the micro controller to enter an interrupt handling routine. The interrupt handling routine is located at a predetermined absolute address in the program memory.
- The micro controller supports five interrupts. Three interrupts are automatically generated by internal operations and two interrupts are generated by external signals provided. The three interrupts that are automatically generated by internal operations are Timer flag 0 (TFO), Timer flag 1 (TF1) and serial port interrupt (R1 or T1)
- The two interrupts that are triggered by external signals are INT 0 and INT 1
- All the interrupt functions are under the program control. The programmer is able to change the control bits in the Interrupt Enable register (IE), the interrupt priority register (IP), and the Timer control register (TCON). By setting or clearing the bits in these registers the program can block any or all of the interrupts.

**Timer flag interrupt:**

When the timer/ counter overflows, the corresponding timer flag TFO or TF1 is set to 1. The flag is cleared to 0 when the interrupt generates program call to the timer subroutine in the memory.

**Serial port Interrupt:**

- The serial port interrupt is generated because of two bits R1 and T1. These bits are logically 0Red, to provide a single interrupt to the processor.
- The T1 bit in the SCON register is set when a data is transmitted and the R1 bit in the SCON register is set whenever a data byte is received.
- The serial port Interrupts R1 or T1 are not cleared like the Timer interrupt when the interrupt generates program call. So, the program which deals with serial communication must reset or clear the R1 or T1 bits to 0 to enable next data communication.

**External Interrupts:**

The two interrupts that are generated by external circuits are INTO and INT1. The inputs on the pins of these interrupts sets. the interrupt flags IEO IE1 in the ICON register. These interrupts may be edge triggered or they may be level triggered.

**The Interrupt Enable Register (IE):**

| EA | - | ET2 | ES | ET1 | EX1 | ET0 | EX0 |
|---|---|---|---|---|---|---|---|

**Enable bit = 1 enables the interrupt** **Enable Bit = 0 disable it.**

| SYMBOL | POSITION | FUNCTION |
|---|---|---|
| EA | IE.7 | DISABLE ALL INTERRUPTS. If EA = 0, no interrupt will be acknowledged. If EA=1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit. |
| - | IE.6 | Reserved |
| ET2 | IE.5 | For future expansion |
| ES | IE.4 | Serial Port interrupt enable bit |
| ET1 | IE.3 | Timer 1 interrupt 1 enable bit |
| EX1 | IE.2 | External interrupt 1 enable bit |
| ET0 | IE.1 | Timer 0 interrupt enable bit |
| EX0 | IE.0 | External interrupt 0 enable bit |

**The Interrupts Priority Register (IP):**

The register IP bits determine whether the interrupt is to have a high or low priority. If a bit is set then the interrupt has high priority otherwise it has low priority. Once the high priority interrupt is serviced, then the lower priority interrupts will be serviced. (MSB)

| | - | PT2 | PS | PT1 | PX1 | PT0 | PX0 |
|---|---|---|---|---|---|---|---|

Priority bit =1 assigns high priority, Priority bit = 0 assigns low priority

| SYMBOL | POSITION | FUNCTION |
|---|---|---|
| - | IE.7 | RESERVED |
| - | IE.6 | RESERVED |
| PT2 | IE.5 | FOR 8052 |
| PS | IE.4 | SERIAL PORT INTERRUPT PRIORITY BIT |
| PT1 | IE.3 | TIMER 1 INTERRUPT PRIORITY BIT |
| PX1 | IE.2 | EXTERNAL INTERRUPT 1 PRIORITY BIT |
| PT0 | IE.1 | TIMER 0 INTERRUPT PRIORITY BIT |
| PX0 | IE.0 | EXTERNAL INTERRUPT 0 PRIORITY BIT |

**Interrupt priority register (IP)**

- If two interrupts having the same priority occur at the same time then the priorities assigned to them are shown in the Table 1.

| NUMBER | SOURCE | PRIIORITY |
|---|---|---|
| 1 | IE0 | HIGHEST |
| 2 | TF0 | |
| 3 | IE1 | |
| 4 | TF1 | |
| 5 | RI OR TI | LOWEST |

- These priorities are assigned to the registers by default but if the programmer wishes to change the priority can be changed by the IP register.

## Interrupt Destination:

Table 2 depicts the vector location for different interrupt sources.

e.g. : If TF0 interrupt is generated by Timer 0, then the micro- controller corresponds to location 000BH. The execution will proceed from that location till a return instruction is encountered.

TABLE 2:

| SOURCE | VECTOR ADDRESS |
|---|---|
| IE0 | 0003H |
| TF0 | 000BH |
| IE1 | 0013H |
| TF1 | 001BH |
| RI OR TI | 0023H |

## Experiment No:9

**AIM: -** Write a program to show the use of INT0 and INT1.

**Apparatus: -** Top viewer simulator, Personal computer.

**Program:**

```
        Instruction          Comment
        ORG  0000 H
        LJMP MAIN            ; BY-PASS INTERRUPT VECTOR TABLE
                              ; ISR FOR INTERRUPT INT 0 TO TURN ON THE LED
        ORG  0003 H          ; INT 0 ISR
        SETB P1.3            ; TURN ON LED
        MOV R3, #255
L1:     DJNZ R3 , L1         ; LET LED BE ON
        CLR P1.3             ; TURN OFF LED
        RETI                 ; RETURN FROM ISR
                             ; MAIN PROGRAM FOR INITIALISATION
        ORG 30 H
        MOV IE , #81 H       ; ENABLE INT 0
L2 :    SJMP L2
        END
```

## Experiment No:10

**AIM: -** Write a program of Flashing LED connected to port 1 of the Micro controller.

**Apparatus: -** Top viewer simulator, Personal computer.

**Program:**

```
Start: mov a,#00h
again: mov p0,a
       inc a
       jc start        #jump when a=FF+ 1
       Ijmp again
       end
```

## Experiment No:11

**AIM:** - Write a program to generate a Ramp waveform using DAC with micro controller.

**Apparatus: -** Top viewer simulator, Personal computer.

**Program:**

```
Port A      EQU      0000 H
Port BEAU   0002 H
Port C      EQU      0004 H
CWR         EQU      0006 H

            MOV A, #80H          ; Initialize Port A of 8255 as output port.
            MOV DPTR , # Port A  ; Load DPTR with port address of port A.
            MOV A, FFH           ; Initialize A
            MOV B, A
            ANL A, # 80H         ; Check for MSB JZ SKIP
            MOV A, B             ; Initialize A
Ll:         MOV @DPTR , A        ; Output contents on port A.
            INC A
            CJNE A ,# FFH , Ll   ; Compare A with FFH.
L2:         DEC A                ; Decrement A.
            MOV @DPTR , A
            JNZ L2.
            JNZ L1.
SKIP :      MOV A, B             ;Initialize A hack.
            JMP  L2.
```

## Experiment No:12

**AIM: -** Write a program to interface the ADC.

**Apparatus: -** Top viewer simulator, Personal computer.

**Program:**

| | | | |
|---|---|---|---|
| | 00.00V | 00 | |
| | **01.25V** | 3F | |
| | 02.50V | 7F | |
| | 05.00V | FF | |
| 2000 | 75 81 50 | MOV 81 H,#50H | ;INITIALIZE THE ;POINTER |
| 2003 | 74 98 | MOV A,#98H | ;CONTROLWORDFOR8255 |
| 2005 | 90 FF 03 | MOV DPTR,#0FF03H | ;CONTROL WORD AD ;DRESS OFF03H |
| 2008 | FO | MOVX @DPTR,A | ;OUT THE CONTROL ;WORD AT OFF03H |
| 2009 | 74 00 | LOOP1:MOV A,#00H | ;SELECT THE |
| 200B | 90 FF 01 | MOV DPTR,#0FF01 H | |
| 200E | F0 | MOVX @DPTR,A | |
| 200F | 74 00 | MOV A,#00H | ;SEND SOC SIGNAL |
| 2011. | 90 FF 02 | MOV DPTR,#0FF02H | |
| 2014 | F0 | MOVX @DPTR,A | |
| 2015 | 74 03 | MOV A,#03H | |
| 2017 | 90 FF 02 | MOV DPTR,#0FF02H | |
| 201A | F0 | MOVX @DPTR,A | |
| 201B | 74 00 | MOV A,#00H | |
| 201D | 90 FF 02 | MOV DPTR,#0FF02H | |
| 2020. | F0 | MOVX @DPTR,A | |
| 2021 | 90 FF 02 | LOOP2:MOV DPTR,#0FF02H | ;CHECK FOR END OF ;CONVERSION |
| 2024 | EO | MOVX A,@DPTR | |
| 2025 | 5410 | ANL A,#1 0H | |
| 2029 | 7404 | MOV A,#04H | ;O/P ENABLE |
| 202B | 90 FF 02 | **MOV DPTR,#0FF02H** | |
| 202E | FO | MOVX @DPTR,A | |
| 202F | 90 FF 00 | **MOV DPTR,#0FF00H** | |
| 2032 | E0 | MOVX A,@DPTR | ;READ ADC |

```
2033    F5 40              MOV 40H,A          ;STORE THE RESULT
                                              ;AT 40H LOC.
2035    12 20 41    LCALL  DISPLAY            ;CALL DISPLAY
2038    90 31 00           MOV DPTR,#3100H    ;LOAD DISPLAY
203B    12 06 F7    LCALL  06F7H              ;DISPLAY READING
                                              ;DATA SEG.
203E    02 20 09    LJMP   LOOP1              ;GO FOR NEXT
                                              ;SAMPLE OF DATA
2041    90 31 00    DISPLAY:MOV               ;DISPLAY LOC. 3100H
                                              ;LOADED IN DPTR
2044    74 FF              MOV A,#0FFH
2048    F0                 MOVX @DPTR,A
2347    90 31 01           MOV DPTR,#3101H
204A    74 FF              MOV A,#0FFH
204C    F0                 MOVX @DPTR,A
204D    90 31 02           MOV DPTR,#3102H
2050    74 FF              MOV A,#0FFH
2052    F0                 MOVX @DPTRA
2053    90 31 03           MOV DPTR,#3103H
2058    74 FF              MOV A,#0FFH
2058.   F0                 MOVX @DPTR,A
2059    E5 40              MOV A,40H          ;MAKE CODE FOR
205B    54 F0              ANL A,#0F0H        ;FROM LOOKUP
205D    C4                 SWAP A
205E    90 30 00           MOV DPTR,#3000H
2061    93                 MOVC A,@A+DPTR
2062    90 31 04           MOV DPTR,#3104H
2065    F0                 MOVX CDPTR,A
2066    E5 40              MOV A,40H
2068    54 0F              ANL A,#0FH
206A    90 30 00           MOV DPTR,#3000H
206D    93                 MOVC A,@A+DPTR
206E    90 31 05           MOV DPTR,#3105H
2071    F0                 MOVX @DPTR,A
2072    90 31 06           MOV DPTR,#3106H
2075    74 FF              MOV A,#0FFH
2077    F0                 MOVX @DPTR,A
2078    90 31 07           MOV DPTR,#3107H
207B    74 FF              MOV A,#0FFH
207D    F0                 MOVX @DPTR,A
207E    22                 RET
207F    7E 01      DELAY:MOV R6,#01 H
2081    7D 25        LP2:  MOV R5,#25H
2083    00           LP1:  NOP
2084    00                 NOP
2085    DD FC              DJNZ R5,LP1
2087    DE F8              DJNZ R6,LP2
2089    22                 RET
```

```
3000            ORG 3000H
3000    03      DFB 03H
3001    9F      DFB 9FH
3002    25      DFB 25H
3003    0D      DFB ODH
3004    99      DFB 92H
3005    49      DFB 49H
3006    41      DFB 41 H
3007    1F      DFB 1FH
3008    01      DFB 01H
3009    09      DFB 09H
300A    11      DFB 11H
300B    C1      DFB 001H
300C    83      DFB 63H,
300D    85      DFB 85H
300E    61      DFB 61H
300F    71      DFB 71 H
```

## Experiment No:13

**AIM: -** Write a program to control a stepper motor in direction, speed and number of steps.

**Apparatus: -** Top viewer simulator, Personal computer.

**Program:**

```
2000   90 FF 03          MOV DPTR,#OFF03H
2003   7480              MOV A,#80H
2005   F0                MOVX @DPTR,A

2006   90 FF 00 START:   MOV DPTR,#0FF00H
2009   74 FA              MOV A,#0FAH
200B   FO                MOVX @DPTR,A
200C   122020                LCALL DELAY
200F   90 FF 0C          MOV DPTR,#0FF00H
2012   74 F6              MOV A,#0F6H
2014   FO                MOVX @DPTR,A
2015   12 20 2C              LCALL DELAY
2018   90 FF OC          MOV DPTR,#0FF00H
201B   74 F5              MOV A,#0F5H
201 D  F0                MOVX @DPTR,A
201E    122020           LCALL DELAY
2021   90 FF 00          MOV DPTR,#0FF00H
2024   74 F9              MOV A.#0F9H
2026   FO                MOVX @DPTR,A
2027   12 20 2D          LCALL DELAY
202A   02 20 06          LJMP START
202D   7F 32            DELAY: MOV R7,#32H
202F   7E F9        LP1: MOV R6,#0F9H
2031   00           LP2: NOP
2032   DE FD             DJNZ R6,LP2
2034    DF F9             DJNZ R7,LP1
2036    22                RET
```

## Experiment No:14

**AIM: -** Write a program to control the speed of DC motor.

**Apparatus: -** Top viewer simulator, Personal computer.

**Program:**

```
3000                        ORG 3000H
3000    75 8150             MOV 81H,#50H
3003    7430                MOVA,#80H
3005    9C FF 03            MOV DPTR,#OFF03H.
3008    FC                  MOVX @DPTR,A
3009    74 FF       START:  MOVA,#OFFH
300B    90 FF 02            MOV DPTR,#OFF02H
300E    F0                  MOVX @DPTR,A
30OF    90 FF 00            MOV DPTR,#OFF,00h
3012    FC.                 MOVX @DPTR,A
3013    12 30 26            LCALL DELAY
3016    74 00               MOVA.#OOH
3018    90 FF 02            MOV DPTR,#OFF02H
301B    F0                  MOVX@DPTR,A
301C    90 FF 00            MOV DPTR,#OFF00H
301F    F0                  MOVX @DPTR,A
3020    12 30 26            LCALL DELAY
3023    02 30 09            LJMP START
3026    7F 64       DELAY:  MOV R7,#064H
3028    7E          LPl:    MOV R6,#0F9H
302A    0C          LP2:    NOP
302B    DE FD               DJNZR6,LP2
302D    DIF F9              DJNZ R7,LP1 302F22 RET
```

## Experiment No:15

**AIM: -** Interfacing of high power devices to Micro-controller port-lines, LED, relays and LCD display.

**Apparatus: -** Top viewer simulator, Personal computer.

**Program:**

### LED MATRIX

PORT ADDRESS

| | |
|---|---|
| Port A | 00 |
| Port B | 01 |
| Port C | 02 |
| Control Word Register | 03 |

PROGRAM

| Address | Opcodes | Mnemonics |
|---|---|---|
| 2000 | 90 FF 03 | MOVDPTR,#0FF03H |
| 2003 | 7480 | MOVA,#80H |
| 2005 | FO | MOVX@DPTR,A |
| 2006 | 90 FF 00 START: | MOVDPTR,#0FF00H |
| 2009 | 74 E0 | MOVA,#0E0H |
| 200B | F0 | MOVX@DPTR,A |
| 200C | 90 FF 01 | MOV DPTR,#0FF01H |
| 200F | 74 7F | MOVA,#7FH |
| 2011 | F0 | MOVX@DPTR,A |
| 2012 | 12 20 96 | LCALL DELAY1 |
| 2015 | 90 FF 00 | MOVDPTR,#OFFOOH |
| 2018 | 74 FF | MOVA,#OFFH |
| 201A | F0 | MOVX@DPTR,A |
| 201 B | 90 FF 01 | MOV DPTR,#0FF01 H |
| 201E | 74 BF | MOV A,#OBFH |
| 2020 | F0 | MOVX@DPTR,A |
| 2021 | 90 FF 00 | MOV,DPTR,#0FF00H |
| 2024 | 74 DF | MOVA,#0DFH |
| 2026 | F0 | MOVX@DPTR,A |
| 2027 | 12 20 96 | LCALL DELAY1 |
| 202A | 90 FF 00 | MOVDPTR,#0FF00H |

| | | |
|---|---|---|
| 202D | 74 FF | MOV A.#0FFH |
| 202F | F0 | MOVX, @DPTR,A |
| 2030 | 90 FF 01 | MOV DPTR,#0FF01H |
| 2033 | 74 DF | MOVA,#0DFH |
| 2035 | F0 | MOVX@DPTR,A |
| 2036 | 90 FF 00. | MOVDPTR,#0FF00H |
| 2039 | 74 BF - | MOV A.#OBFH |
| 203B | F0 | MOVX@DPTR,A |
| 203C | 12 20 96 | LCALL DELAY1 |
| 203F | 60 FF 00 | MOVDPTR,#0FF00H |
| 2042 | 4 FF | MOV A,#OFFH |
| 2044 | FJ | MOVX@DPTR,A |
| 2045 | 90 FF 01 | MOVDPTR,#0FF01H |
| 2048 | 74 E7 | MOV A.#0E7H |
| 204A | F0 | MOVX@ D P T R , A |
| 204B | 90 FF 00 | MOV DPTR,#0FF00H |
| 204E | 47 7F | MOV A,#7FH |
| 2050 | F0 | MOVX@DPTR, |
| 2051 | 12 20 96 | LCALLDELAY1 |
| 205A | 90 FF 01 | MOVDPTR,#0FF01H |
| 205D | 74 FB | MOVA,#0FBH |
| 205F | F0 | MOVX@DPTR,A |
| 2050 | 90 FF 00 | MOVDPTR,#OFFOOH |
| 2065 | FO | MOVX@DPTR,A |
| 2066 | 12 20 98 | LCALL DELAY1 |
| 2065 | FO | MOVX@DPTR,A |
| 2066 | 12 20 98 | LCALL DELAY1 |
| 2069 | 90 FF 00 | MOVDPTR,#0FF00H |
| 205F | F0 | MOVX@DPTR,A |
| 2050 | 90 FF 00 | MOVDPTR,#OFFOOH |
| 2065 | FO | MOVX@DPTR,A |
| 2066 | 12 20 98 | LCALL DELAY1 |
| 2065 | FO | MOVX@DPTR,A |
| 2066 | 12 20 98 | LCALL DELAY1 |
| 206C | 74 FF | MOVA,#0FFH |
| 206E | F0 | MOVX@DPTRA |
| 206F | 90 FF 01 | MOVDPTR,#0FF01H |
| 2072 | 74 FD | MOVA.#0FDH |
| 2074 | F0 | MOVX@DPTR,A |

```
2075    90 FF 00          MOV DPTR,#0FF0H
2078    74 DF             MOVA.#0DFH
207A    F0                MOVX@DPTR,A
207B    12 20 96          LCALL DELAYI
207E    90 FF 00          MOV DPTR.#0FF00H
2081    74 FF             MOVA,#0FFH
2083    F0                MOVX@DPTR,A
2084    90 FF 01          MOVDPTR,#0FF01H
2087    74 FE             MOVA,#0FEH
2089    F0                MOVX@DPTR,A
208A    90 FF 00          MOV DPTR,#0FF00H
208D    74 E0             MOVA,#0E0H
208F    F0                MOVX@DPTP,A
2090    12 20 96          LCALL DELAY1
2093    02 20 06          LJMP START
2096    7F 09    DELAY1   MOV R7,#09H
2098    7E 64      LP1:   MOV R6,#64H
209A    00         LP2:   NOP
209B    DE FD             DJNZ R6,LP2
209D    DF F9             DJNZ R7,LP1
209F    22                RET
```

RELAY

PORT ADDRESS

| | |
|---|---|
| Port A | FF00 |
| Port B | FF01 |
| Port C | FF02 |
| Control Word Register | FF03 |

**PROGRAM**

```
Address  Opcodes          Mnemonics              Comments
3000     74 80            MOVA,#80H
3002     90 FF 03         MOV DPTR,#0FF03H       ;control word 80H
3005     F0               MOVX @DPTR,A           ;make all ports as
                                                 ;output
         START:
3006     74 0F            MOVA,#0FH              ;output OFH on port-
3008     90 FF 00         MOVDPTR,#0FF00H
```

```
300B   F0              MOVX@DPTR,A
300C   12 30 18        LCALL DELAY          ;generate a delay
300F   74 00           MOVA,#00H            ;output 00H on port-A
3011   90 FF 00        MOVDPTR,#0FF00H
3014   F0              MOVX@DPTR,A
3015   12 30 18        LCALL DELAY          ;generate a delay
3018   02 30 06        LJMP START           A chitring sound of
                                            ;relay is generated
301B   7F 64    DELAY:  MOV R7,#64H
301D   7E F9    LP1:    MOV R6,#0F9H
301F   00       LP2:    NOP
3020   DE FD           DJNZ R6,LP2
3022   DF F9           DJNZ R7,LP1
3024   22               RET
```

**Experiment No:16**

**Aim**:- Programmed Logic Controller (PLC) Experiment

for CHEN 4570 – Instrumentation and Process Control
Programmed Logic Controller (PLC) Experiment
for CHEN 4570 – Instrumentation and Process Control

**BUDGET SUMMARY:**
Total Project Budget $ 1992
EEF Request $ 1600
Outside funding:
**Source** *Instrumentation and Process Control* (CHEN 4570) is a required course for the B.S. Chemical Engineering and B.S. Chemical & Biological Engineering degree programs at CU-Boulder. In the mid- 1990's, this course was expanded from three to four credit hours to include a laboratory component. Since then, students meet for lab three hours per week throughout the semester and complete 14 laboratory experiments. These experiments are carried out in the ITLL and in the Chemical Engineering undergraduate laboratory (ECCH 1B70). As career opportunities evolved for ChE graduates in the 80's and 90's, more of our alumni were working in pharmaceutical, specialty chemical, and manufacturing positions where they encountered programmed logic controllers (PLC's). Based on this feedback from alumni, we modified the course in the late 90's to include three (of the 14) laboratory exercises that introduce our students to the concepts and practice of PLC's. We acquired PLC hardware and software from Bytronics, Ltd., in England and have used that for the past ten years. The feedback from our alumni over the past ten years has been positive and reinforcing; however, lately, we have heard that our hardware and software are outdated when compared to current industrial practice. We have also recognized this because the software (LadSim) will only run on a Windows 2000 operating system, which means that we cannot use the computers in the ITLL and ChE undergraduate labs and must use older machines loaded with Windows 2000. The purpose of this request then it to allow us to upgrade, modernize and expand our PLC hardware and software and thereby continue this important component of our laboratory education in instrumentation and process control. We have used commercial Allen-Bradley MicroLogix PLC's in student projects for over five years now. These were suitable for these projects because student teams could dedicate the time required to learn the use of the hardware and, in particular, the support software (Rockwell RSLogix 500), which is quite complex ("industrial strength"). In recent developments, the cost of the PLC hardware and software has reduced to a manageable level. Additionally, Rockwell has released a scaled-down version of the software (RS Logix Micro) that is appropriate to the time constraints of the CHEN 4570 lab. And, software is now available from a separate vendor in Canada, Logix Pro, that allows students to learn PLC fundamentals and programming in a simulated environment prior to encountering the MicroLogix/RS Logix Micro "real world" equipment. So, we now believe that we can acquire, at modest cost, a set of PLC's and software that will allow us, at once, to modernize and make great improvement in the PLC laboratory exercises in CHEN 4570. To do this, we require four Allen-Bradley MicroLogix 1000 PLC units, four RS Logix Micro software licenses, and 20 network-based licenses of the Logix Pro simulator software. This acquisition also expands our capability to accommodate the increased enrollments in our two B.S. degree programs.

All ChE and ChBE undergraduate students are required to take CHEN 4570. Projected enrollments are shown in the table below. These are based on current enrollments in prior classes.
Spring 2009 66
Spring 2010 78
Spring 2011 150
Spring 2012 150
Enrollment Projection --
Project Impact
Project Budget

*Confirmed? [Y/N]*
**Total Amount**
Department Contribution Y $392
College Contribution 0
**BUDGET BREAKDOWN:**
Equipment and Materials:
**Item Name / Description**
**Unit Price Quantity Total Amount**
Allen-Bradley MicroLogix 1000 PLC $199 4 $796
Rockwell RS Logix Micro Software 129 4 516
Logix Pro Simulation Software 34 20 680
Total $ $1992