MICROCONTROLLER & EMBEDDED SYSTEM DESIGN

(EE-328-F)

LAB MANUAL

VI SEMESTER

DRONACHARYA
College of Engineering

Department Of Electronics & Communication Engg
Dronacharya College Of Engineering
Khentawas, Gurgaon – 123506

LIST OF EXPERIMENTS

syllabus

VI SEM - Microcontroller & Embedded System LAB

List of Experiment:

8051/AT 89C51 microcontroller

1. Write an Assembly language Programme (ALP) to generate 10 kHz square wave.
2. To study implementation & interfacing of Display devices Like LCD, LED Bar graph & seven segment display with Microcontroller 8051/AT89C51
3. To study implementation & interfacing of Different motors like stepper motor, DC motor & servo Motors.
4. Write an ALP for temperature & pressure measurement.
5. Write a program to interface a graphical LCD with 89C51.
6. To study Programming and Transmission & reception of data through Serial port & study of Parallel printer port.

PIC Microcontroller

7. To interface PWM based voltage regulator using PIC Microcontroller .
8. Study and analysis of interfacing of Graphical LCD using PIC controller
9. Study and interfacing of IR (RC5 protocol) and RF Communication using PIC controller
10. Study of SD/MMC card Interface using 18F4550

Microcontroller & Embedded System Design EE-328-F

Sr.No.	LIST OF EXPERIMENTS
1.	To study development tools/environment for ATMEL/PIC microcontroller programme and Architecture.
2.	Write an assembly language program to add, subtract, multiply, divide 16 bit data by Atmel microcontroller.
3.	An assembly language program to generate 10 KHz frequency using interrupts on P1.2.
4.	Study and analyze the interfacing of 16 x 2 LCD.
5.	Study of implementation, analysis and interfacing of seven segment display.
6.	Study of implementation of steeper motor angle control.
7.	Study of implementation of DC Motor control using PWM method.
8.	Study and observation of Position control of Servo Motor.
9.	Study of Programming and Transmission and Reception of data through serial port.
10.	To study implementation and programming of Pressure measurement.
11.	To study implementation and programming of Temperature measurement.
12.	Study and analysis of interfacing of graphical LCD using PIC Microcontroller.
13.	To interface PWM based voltage regulator using PIC Microcontroller.
14.	Study and interface of IR (RC5 Protocol) and RF Communication using PIC Microcontroller.

EXPERIMENT NO 1

AIM: To study development tools/environment for ATMEL/PIC microcontroller programme and Architecture.

APPARATUS REQUIRED: µ Vision Keil, ICPROG, AT89C52 Microcontroller, PIC16F877A Microcontroller.

SOFTWARE ENVIRONMENT AND MICROCONTROLLER DESCRIPTION:

Procedure to write the program in µ Vision Keil:

1. Create a **New folder** on the desktop for saving the contents of the program.
2. Double click on the icon of **Keil**.
3. Select the device for the target **(Select Atmel → Select 89C52 →Ok →No)**
4. Go in the **project menu** and click on **µVision Project** after this an edit window will appear on desktop.
5. Write the desired program in the editing window up to end.
6. Right click on **source group** and select **remove start up** in **project workspace**.
7. Go in the **file menu** and click on **save as** and **save** the program with the extinction **.asm** on desktop in the **new folder**.
8. Right click on **source group** → Select **add file to group** → All file → Select file **.asm** → Select **Add.**
9. Now go in the **project menu** and click on options for the target **"Target1"**.
10. Update the **frequency value (eg. 11.0592)** and click on **output** and enable the following.
    a. ®Create Executable
    b. √ or Ok –Debug info
    c. Select Create Hex file
    d. Select Browse info
    Now click on **Ok**
11. Go in the **project menu** and click on **built target**.
12. Go in the **project menu** and click on **Rebuild target**.
13. Go in the **project menu** and click on Run (or Ctrl +F5).
14. After this **Hex file** will be created in the **New Folder**

ATMEL INTRODUCTION 8051 ARCHITECTURE FAMILY

A microcontroller is a single chip microcomputer with on board program ROM and I/O that can be programmed for various control functions. Unlike a general purpose computer, which also includes all of these components, a microcontroller is designed for a very specific task to control a particular system.

The AT89C52 is a low power, high performance CMOS 8 bit microcomputer with 8K bytes of Flash Programmable and Erasable Read Only Memory. The on chip flash allows the program memory to be reprogrammed in system or by a conventional non-volatile memory programmer. The AT89C52 provides 256 Bytes of RAM, 32 I/O lines, three 16 bit timer/counters, and six vector two levels interrupt.

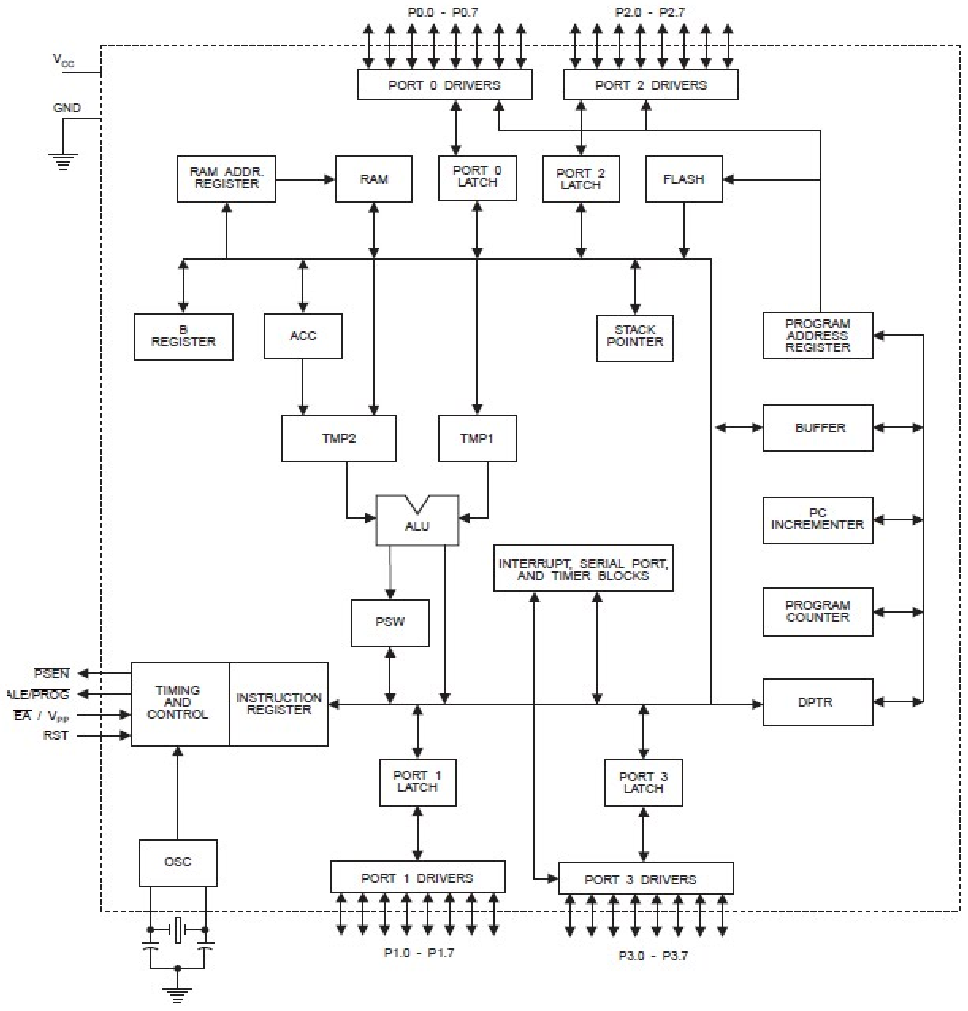
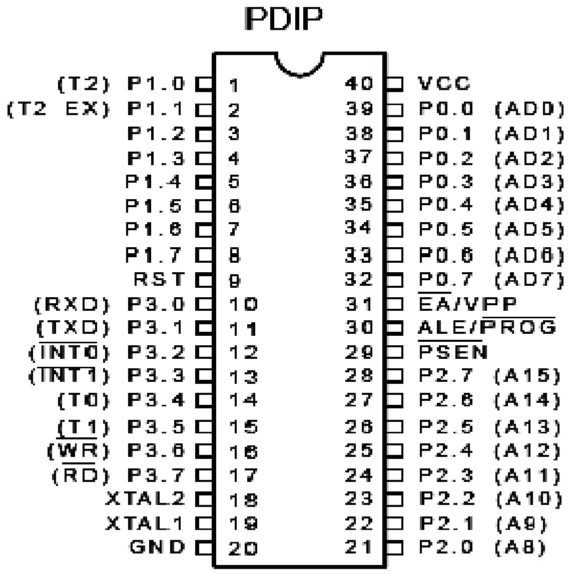


Figure. The architecture of the 8051 family of Microcontrollers



Pin Diagram of AT89C52

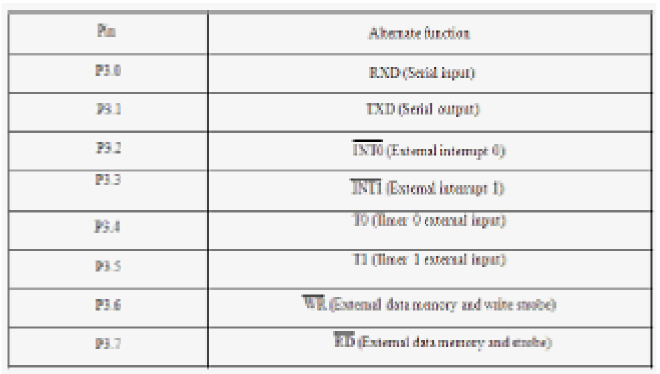
Port 0 (P0.0 - P0.7): If designated as output, each of these pins can be connected up to 8 TTL input circuits. If designated as input, they are high impedance inputs as their potential is undefined with respect to the ground. If external memory is used, these pins are used for alternate transfer of data and addresses (A0-A7) for accessing the extra memory chip. Signal on ALE pin determines the mode of transfer on port.

Port 1 (P1.0 - P1.7): If designated as output, each of these pins can be connected up to 4 TTL inputs. If designated as input, these pins act like standard TTL inputs (that is, they have an internal resistor connected to the positive supply pole and a +5V voltage). Also, pins of Port 1 have alternate functions according to the following table:

Pin	**Alternate function**
P1.0	T2 (Timer 2 input)
P1.1	T2EX (Timer 2 control input)

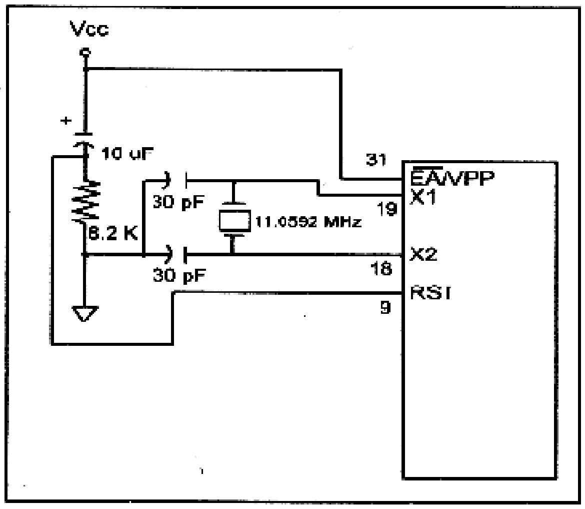
Port 2 (P2.0 - P2.7): If designated as input or output, this port is identical to Port 1. If external memory is used, Port 2 stores the higher address byte (A8-A15) for addressing the extra memory chip.

Port 3 (P3.0 - P3.7): Similar to Port 1, Port 3 can also be used as universal I/O, but pins of Port 3 also have alternate functions.



Pin	Alternate function
P3.0	RXD (Serial input)
P3.1	TXD (Serial output)
P3.2	$\overline{INT0}$ (External interrupt 0)
P3.3	$\overline{INT1}$ (External interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{WR}$ (External data memory and write strobe)
P3.7	$\overline{RD}$ (External data memory and strobe)

RST: Reset input: A high on this pin for two machine cycles while the oscillator is running resets the device and terminate all activities. This is often referred to as a power on reset. Activating a power on reset will cause all values in the register to be lost. Figure A and figure B shows two ways of connecting the RST pin to the power on reset circuitry.



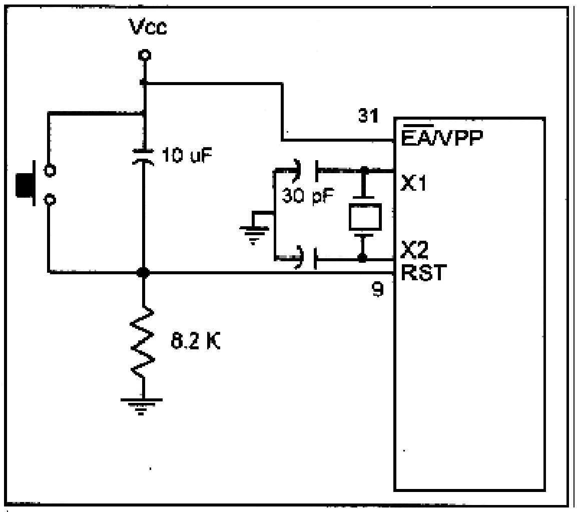


Figure A. Power On Reset Circuit Figure B. Power On with Debounce

Positive voltage impulse on RST pin resets the MCU. In order to be detected, this impulse needs to have duration of at least two operating cycles (this cycle represents the time necessary to execute one instruction and lasts for 12 oscillator signals).

EA/VPP: External Access Enable; When this pin is connected to the ground, MCU gets program instructions from external program memory. In case that internal program memory is used (common case); this pin should be connected to the positive supply pole (VCC). During the loading of program to internal Flash memory, this pin is at +12V.

ALE/PROG: This pin emits an impulse sequence with a frequency equal to 1/6 of the frequency generated by the main oscillator. If external memory is used, signal from this pin controls the additional register for temporary storage of the lower address byte (A0 - A7). This pin also serves as a control input during the writing of program to MCU.

PSEN: Program Store Enable; This pin is used for reading from external program memory (ROM). When the AT89C52 is executing code from external program memory, PSEN is activated twice each machine cycle, except that two PSEN activations are skipped during each access to external data memory.

XTAL1 and XTAL2: XTAL1 is the Input to the inverting oscillator amplifier and input to the internal clock operating circuit. XTAL2 is Output from the inverting oscillator amplifier. AT89C52 has an on chip oscillator but requires an external clock to run it(See Figure. C). Most often quartz crystal is connected to XTAL1 and XTAL2.The quartz crystal oscillator connected to XTAL pins also needs two capacitors of 30pf value.

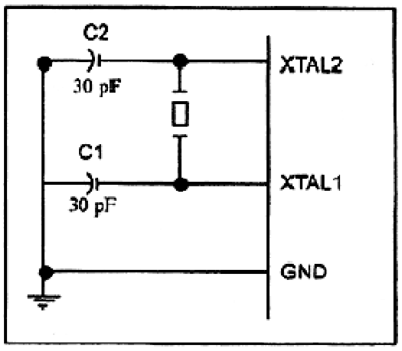


Figure. C. XTAL Connection in 8051

VCC: Power Supply (4 – 6V)
GND: Negative supply pole (ground)

HOW TO USE ICPROG SOFTWARE FOR PIC Microcontroller

1. Click ICPROG software.
2. Select device PIC 16F877 A.
3. Click <settings> Select <Hardware> Select JDM programmer and Windows API in place of Direct I/O & click OK.
4. Click <Setting> then <options> then <programming> then select the option <verify after programming>.
5. Set configuration in ICPROG software as mentioned below:
6. Oscillator----- XT
7. Write Enable----OOOO-OFFFH
8. Deselect all Fuses.
9. Set swl in Un pressed condition for IAP mode. Jumper in 1 2 position.
10. Select <command> then Program all.

PIC INTRODUCTION & ARCHITECTURE:

PIC is the name for the Microchip microcontroller (MCU) family, consisting of a microprocessor, I/O ports, timer(s) and other internal, integrated hardware. The main advantages of using the PIC are low external part count, a wide range of chip sizes available, nice choice of compilers (assembly, C, Basic, etc.) good wealth of example/tutorial source code and easy programming. Once bought, the PIC's program memory is empty, and needs to be programmed with code (usually HEX files) to be usable in a circuit. For the purpose, a wide range of simple programmer hardware docs and software is downloadable from the net.

PIC is a family of Harvard architecture microcontrollers made by Microchip Technology, derived from the PIC1650 originally developed by General Instrument's Microelectronics Division.

The PIC architecture is distinctively minimalist. It is characterized by the following features:

- Separate code and data spaces (Harvard architecture)
- A small number of fixed length instructions
- Most instructions are single cycle execution (4 clock cycles), with single delay cycles upon branches and skips

- A single accumulator (W), the use of which (as source operand) is implied (i.e is not encoded in the opcode)
- All RAM locations function as registers as both source and/or destination of math and other functions.
- A hardware stack for storing return addresses
- A fairly small amount of addressable data space (typically 256 bytes), extended through banking
- Data space mapped CPU, port, and peripheral registers
- The program counter is also mapped into the data space and writable (this is used to synthesize indirect jumps).

Unlike most other CPUs, there is no distinction between "memory" and "register" space because the RAM serves the job of both memory and registers, and the RAM is usually just referred to as the register file or simply as the registers.

PIC16F877A Specifications & Architecture:

High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program Branches, which are two-cycle
- Operating speed : DC–20 MHz clock input DC–200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory, Up to 368 x 8 bytes of Data Memory (RAM), Up to 256 x 8 bytes of EEPROM Data Memory
- Pin out compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers.

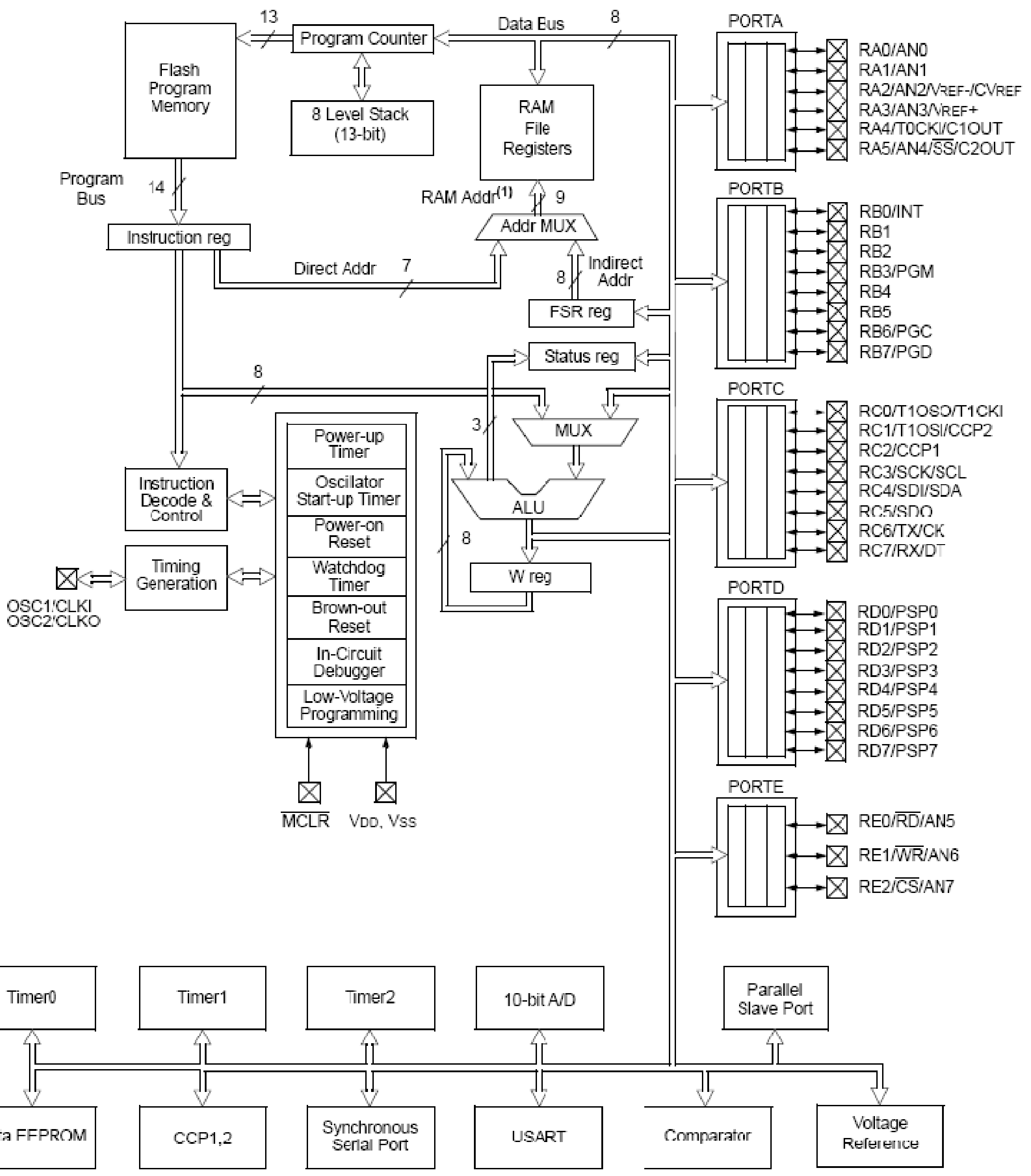
Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  a. Capture is 16-bit, max. Resolution is 12.5 ns
  b. Compare is 16-bit, max. Resolution is 200 ns
  c. PWM max. Resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP)–8 bits wide with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
  A. Two analog comparators
  B. Programmable on-chip voltage reference (VREF) module
  C. Programmable input multiplexing from device inputs and internal voltage reference
  D. Comparator outputs are externally accessible

Block diagram of PIC16F877A Controller



1.What is meant by micro controller?

Ans A device which contains the microprocessor with integrated peripherals like memory, serialports, parallel ports, timer/counter, interrupt controller, data acquisition interfaces like ADC, DAC is called micro controller.

2.List the features of 8051 micro controllers?

Ans
- Single supply +5v operation using HMOS technology.
- 4096 bytes program memory on-chip.
- 128 data memory on chip.

· 4 register banks
· 2 multiple modes, 16 bit timer/counter
· Extensive Boolean processing capabilities.
· 64KB external RAM size.
· 32 bi-directional I/O lines.

3. Explain the operating mode 0 of 8051 serial port?

Ans In this mode serial data enters and exists through RXD, TXD outputs the shift clock. 8-bits are transmitted or received:8-data bits(LSB first). The baud rate is fixed at 1/12 the oscillator frequency.

4. Explain the operating mode 2 of 8051 serial port?

Ans In this mode 11 bits are transmitted (through TXD) or received (through (RXD): a start bit(0), 8 data bits(LSB first), a programmable 9th data bit and a stop bit(1). On transmit, the 9th data bit can be assigned the value 0 or 1. On receive, the 9th data bit go into the RB8 in special function register SCON, while the stop bit is ignored.
The baud rate is programmable to either 1/32 or 1/64 the oscillator frequency.

5. Explain the mode 3 of 8051 serial port?

Ans In this mode, 11 bits are transmitted (through TXD) or (received (through RXD): a start bit(0), 8 data bits(LSB first), a programmable 9th data bit and a stop bit(1).It is same as mode 2 except the baud rate. The baud rate in mode 3 is variable.

6. Explain the interrupts of 8051 micro controller?

Ans · External interrupt 0 (IE0) – Highest priority,· Timer interrupt 0 (TF0)
· External interrupt 1 (IE1),· Timer interrupt 1 (TF1)
· Serial port Interrupt,Receive interrupt (RI) – lowest priority
Transmit interrupt (TI)

7. How many bytes of internal RAM and ROM supported by 8051 micro controller?

Ans 128 bytes of internal RAM and 4 bytes of ROM.

8. Define machine cycle of 8051?

Ans 8051 machine cycle consists of 6 states, S1 through S7. One state is made up of 2 clock pulses. Thus 12 clock period constitute one machine cycle. Two clock periods in a state is termed as phase 1 and phase 2.

9. What are the special function of port 0 of 8051?

Ans Port 0 is used as a multiplexed low order address/data bus during the external memory access. When ALE is enabled, the address on port 0 pins are latched and bus is ready to act as a data bus when ALE is low.

10. What are the alternative function of port 3 of 8051?

Ans Serial data input (P3.0), serial data output (P3.1), external interrupt 0 (P3.2), external interrupt 1 (P3.3), external input for timer 0(P3.4), external input for timer 1 (P3.5), external memory write pulse (P3.6), external memory read (P3.7) are the alternative functions of port 3.

EXPERIMENT NO 2

IM: Write an assembly language program to add, subtract, multiply and divide 16 bit data by Atmel icrocontroller.

APPARATUS: M51-02 trainer kit, keyboard and power cord.

PROGRAM:

Addition:

```
          ORG 0000H
          CLR   C                  ; make CY=0
          MOV A, #0E7H             ; load the low byte now A=E7H
          ADD A, #8DH              ; add the low byte now A=74H and CY=1
          MOV R6, A                ; save the low byte of the sum in R6
          MOV A, #3BH              ; load the high byte
          ADDC A, #3BH             ; add with carry (3B+3C+1=78)
          MOV R7, A                ; save the high byte of the sum
```

Subtraction:

```
          ORG 3000H
          CLR C                    ; make CY=0
          MOV A, #50H              ; load the low byte now A= 50H
          MOV R1, #30H             ; load the byte now R1=30H
          SUBB A, R1               ; subtract contents of A and R1
          JNC Next
          CPL A
          INC A
    Next: MOV R2, A
          SJMP 3000H
```

Multiply:

```
          ORG 4000H
          MOV A, #03H              ; move the first no. into acc
          MOV B, #02H              ; move the second no. into B
          MUL AB                   ; multiply the contents of acc with B
          SJMP 4000H
```

Divide:

```
          ORG 5000H
          MOV A, #25H              ; move the first no. into acc.
          MOV B, #5H               ; move the second no. into B
          DIV AB                   ; divide the contents of acc with B
          SJMP 5000H
```

RESULT: Addition, Subtraction, Multiplication, Division of 16 bit data has been performed successfully
on the kit.

PRECAUTIONS: Make sure correct power supply is given to the kit/Equipment. Wrong power supplies
may cause damage to your equipments.

Question & Answer:

1.What are Pseudo instructions .
Ans. **Assembler Directives**
2.Number of the times the instruction sequence below will loop before coming out of loop is

```
 MOV AL, 00h
A1:  INC AL
 JNZ A1                                                    :
```

Ans.256
3.With which instructionsDirection flag is used .
Ans. **String instruction**
4.A Bus cycle is equal to how many clocking period.
Ans.**4**
5.What is NMI input.
Ans.**Edge Sensitive**
6.What do the symbols [ ] indicate.
Ans.**Indirect addressing**
7.The internal RAM memory of the 8051 is.
Ans.**128 Bytes**
8.The I/O ports that are used as address and data for external memory are.
Ans.**Ports 0 and 2**
9.The total external data memory that can be interfaced to the 8051 is.
Ans.**64K**
10.What is the 8-bit address bus allows access to an address range.
Ans.**00 to FFH**

EXPERIMENT NO 3

AIM: Write an assembly language program to generate 10 KHz frequency using interrupts on P1.2.

APPARATUS: E89-01 KIT, power cord, CRO and connecting leads.

PROGRAM:

```
        ORG 0H                      ; Wake up ROM reset Location
        LJMPMAIN                    ; Bypass interrupt vector table
        ORG 000BH                   ; ISR for Timer 0
        CPL P1.2                    ; Complement P1.2
        MOV TL0, #0D2H
        MOV TH0, #0FFH              ; Reload timer value
        RETI
        ORG 0030H                   ; starting location for program
MAIN:   MOV TMOD, #01H              ; timer 0, mode 1
        MOV THO, #0FFH
        MOV TLO, #0D2H              ; Enable timer 0 interrupt
        MOV IE, #82H
        SETB TR0                    ; Start timer 0
HERE:   SJMP HERE                   ; Stay here until interrupted
        END                         ; End of the Program
```

RESULT: Square wave has been generated on P1.2 and displayed on CRO.

PRECAUTIONS: Make sure correct power supply is given to the kit/Equipment. Wrong power supplies may cause damage to your equipments.

Question & Answer:

1.Device pins XTAL1 and XTAL2 for the 8051 are used for connections to an external oscillator or crystal.

Ans. **TRUE**

2.When the 8051 is reset and the $\overline{EA}$ line is HIGH, the program counter points to the first program instruction in the:

Ans.Internal Code Memory

3.An alternate function of port pin P3.4 in the 8051 is:

Ans.Timer 0

4.Bit-addressable memory locations are:

Ans.20H through 2FH

5.The number of data registers is:

Ans.32

6.The total amount of external code memory that can be interfaced to the 8051 is:\

Ans. **64 K**

7.An alternate function of port pin P3.0 (RXD) in the 8051 is:

Ans.Serial Port Input

8.MOV A, @ R1 will:

Ans.copy the contents of memory whose address is in R1 to the accumulator

9.The start-conversion on the ADC0804 is done by using the:

Ans.SC

10. The number of data registers is:

Ans.**32**

EXPERIMENT NO 4

AIM: Study and analyze the interfacing of 16 x 2 LCD.

APPARATUS: NV5001 Microcontroller development Board, MC-04 Kit, power cord and connecting leads.

PROGRAM: for displaying ‘NVIS Technologies’ on LCD.

```
RS_LCD EQU P3.5
RW_LCD EQU P3.6
E_LCD EQU P3.7
-----------------------------------------------------------------------------------------------------------------
ORG 0000H
JMP START
ORG 0200H
START: MOV P0, #00H
MOV A, #00H
LCALL LCDINIT
MOV A, #086H
LCALL COMMAND
MOV A, #"N"
LCALL DISPLAY
MOV A, #"v"
LCALL DISPLAY
MOV A, #"i"
LCALL DISPLAY
MOV A, #"s"
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
-----------------------------------------------------------------------------------------------------------------
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
------------------------------------------------------------------------------------------------------------
MOV A, #082H
LCALL COMMAND

MOV A, #"T"
LCALL DISPLAY
MOV A, #"e"
LCALL DISPLAY
MOV A, #"c"
LCALL DISPLAY
MOV A, #"h"
LCALL DISPLAY
MOV A, #"n"
LCALL DISPLAY
MOV A, #"o"
LCALL DISPLAY
MOV A, #"l"
LCALL DISPLAY
MOV A, #"o"
LCALL DISPLAY
MOV A, #"g"
LCALL DISPLAY
MOV A, #"i"
LCALL DISPLAY
MOV A, #"e"
LCALL DISPLAY
MOV A, #"s"
LCALL DISPLAY
---------------------------------------------------------------------------------------------------------
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LJMP START
---------------------------------------------------------------------------------------------------------
DELAY_1S:
MOV R2, #11
LOOP3: MOV R3, #98
LOOP2: MOV R4, #106
LOOP1: NOP
NOP
NOP
NOP
NOP
```

```
NOP
DJNZ R4, LOOP1
DJNZ R3, LOOP2
DJNZ R2, LOOP3
RET
-----------------------------------------------------------------------------------------------------------------
LCDINIT: MOV A, #38H
LCALL COMMAND
MOV A, #0CH
LCALL COMMAND


MOV A, #01H
LCALL COMMAND
MOV A, #06H
LCALL COMMAND
RET
-----------------------------------------------------------------------------------------------------------------
COMMAND:
ACALL READY
MOV P0, A
CLR RS_LCD
CLR RW_LCD
SETB E_LCD
CLR E_LCD
RET
-----------------------------------------------------------------------------------------------------------------
DISPLAY:
ACALL READY
MOV P0, A
SETB RS_LCD
CLR RW_LCD
SETB E_LCD
CLR E_LCD
RET
-----------------------------------------------------------------------------------------------------------------
READY: SETB P0.7
CLR RS_LCD
SETB RW_LCD
WAIT: CLR E_LCD
SETB E_LCD
JB P0.7, WAIT
RET
-----------------------------------------------------------------------------------------------------------------
END
```

PROCEDURE:

1. Insert AT89C52 Microcontroller in Programmer unit (in NV5001).
2. Connect serial cable between computer serial port and programmer unit serial port female connector
(in NV5001).
3. Switch 'On' the programmer switch in programmer unit (in NV5001) and switch on the power supply.
4. Program LCD interface module.hex file (Via CD - NV5001/ Modules programs\MC-04 Display module
\LCD module) in AT89C52 Microcontroller via programmer.
5. Switch 'Off' the power supply and remove the programmed controller from programmer ZIF socket.
6. Switch 'Off' the programmer switch in Programmer unit (in NV5001).
7. Insert programmed Microcontroller to microcontroller unit ZIF socket.
8. Connect 20 Pin FRC cable to LCD Interface block left side socket/connector (MC04) to Port P3 in
NV5001 Trainer.
9. Connect 20 Pin FRC cable to LCD Interface block right side socket/connector (MC04) to Port P0 in NV5001 Trainer.

10. Turn contrast control potentiometer in LCD interface block to clockwise position (in MC04).
11. Turn Backlight control potentiometer in LCD interface block to anticlockwise position (in MC04).
12. Switch 'On' the power supply.
13. Observe "NVIS'' is coming on LCD and after some delay "Technologies".
14. Observe the output waveform between RS pin of LCD (P3.5) and ground, on oscilloscope.
15. Observe the output waveform between R/W pin of LCD (P3.6) and ground, on oscilloscope
16. Observe the output waveform between E pin of LCD (P3.7) and ground, on oscilloscope.
17. Turn contrast control potentiometer and observe the contrast change on LCD.
18. Turn Backlight control potentiometer and observe the change on backlight of LCD.

RESULT: 'NVIS Technologies' on displayed on 16x2 LCD.

Question & Answer:

1. The end-of-conversion on the ADC0804 is done by which line?
Ans. **EOC**
2. What is the difference between the 8031 and the 8051?
Ans. **The 8031 is ROM-less**.

3. Which I/O port that does not have a dual-purpose role is?
Ans. **Port 1**
4. The ADC0804 has resolution of?
Ans. **8 Bit**
5. A HIGH on which pin resets the 8051 microcontroller?
Ans. **RST**
6. An alternate function of port pin P3.1 in the 8051 is:
Ans. **Serial Port Output**
7. An alternate function of port pin P3.0 (RXD) in the 8051 is:
Ans. **Serial Port Input**
8. Magnetic tape is a
Ans. **Direct Access Storage Device**
9. Which parts of the computer perform arithmetic calculation
Ans. **ALU**
10. Vacuum tube based electronic computers are:
Ans. **Hoover generation**

EXPERIMENT NO 5

AIM: Study of implementation, analysis and interfacing of seven segment display.

APPARATUS: NV5001 Microcontroller development Board, MC-04 Kit, power cord and connecting leads.

PROGRAM: for displaying "1234" on seven segment

```
SEG_A EQU P1.0
SEG_B EQU P1.1
SEG_C EQU P1.2
SEG_D EQU P1.3
-----------------------------------------------------------------------------------------------------------------
---
ORG 0000H
JMP START
-----------------------------------------------------------------------------------------------------------------
---
ORG 0200H
START: MOV P0, #00H
MOV A, #00H
CLR C
LOOP: MOV P2, #0FFH
CLR SEG_A
CLR SEG_B
CLR SEG_C
CLR SEG_D
SETB SEG_A
MOV P2, #00000110B
LCALL DELAY_1S
CLR SEG_A
CLR SEG_B
CLR SEG_C
CLR SEG_D
MOV P2, #00H
SETB SEG_B
MOV P2, #01011011B
LCALL DELAY_1S
CLR SEG_B
CLR SEG_A
CLR SEG_C
CLR SEG_D
```

```
MOV P2, #00H
SETB SEG_C
MOV P2, #01001111B
LCALL DELAY_1S
CLR SEG_C
CLR SEG_B
CLR SEG_A

CLR SEG_D
MOV P2, #00H
SETB SEG_D
MOV P2, #01100110B
LCALL DELAY_1S
CLR SEG_D
CLR SEG_B
CLR SEG_A
CLR SEG_C
MOV P2, #00H
LJMP LOOP
------------------------------------------------------------------------------------------------------------
-----
DELAY_1S: MOV R2, #06
DO3: MOV R3, #10
DHERE1: MOV R4, #10
DAGAIN: NOP
NOP
NOP
NOP
DJNZ R4, DAGAIN
DJNZ R3, DHERE1
DJNZ R2, DO3
RET
------------------------------------------------------------------------------------------------------------
-----
END
```

PROCEDURE:

1. Insert AT89C52 Microcontroller in Programmer unit (in NV5001).
2. Connect serial cable between computer serial port and programmer unit serial port female connector (in NV5001).
3. Switch 'On' the programmer switch in programmer unit (in NV5001) and switch on the power supply.
4. Program seven segment display.hex file (Via CD - NV5001/ Modules programs\MC04 Display module\Seven segment module) in AT89C52 Microcontroller via programmer.

5. Switch ‘Off’ the power supply and remove the programmed controller from programmer ZIF socket
6. Switch ‘Off’ the programmer switch in Programmer unit (in NV5001).
7. Insert programmed Microcontroller to microcontroller unit ZIF socket.
8. Connect 20 Pin FRC cable to seven segment display Interface block left side socket/connector (MC04) to Port P2 in NV5001 Trainer.
9. Connect 20 Pin FRC cable to seven segment Interface block right side socket/connector (MC04) to Port P1 in NV5001 Trainer.
10. Switch ‘On’ the power supply.
11. Observe “1234’’ is coming on seven segment.
12. Observe the status of segment selection pins on tp28, tp29, tp30 & tp31.output waveform between RS pin of LCD (P3.5) and ground, on oscilloscope.
13. Observe the status of data bits on tp17 to tp24.

RESULT: “1234” displayed on seven segment display.

Question & Answer:

1. Explain how Timer2 works in a STANDARD 8051 device:
   Ans.**TIMER2 is NOT implemented in standard 8051 device. It is implemented only in 8052 devices.**
2. Give an example haw to write code memory using MOVC instruction.
   Ans. **MOVC instruction can only used to READ code memory. It cannot be used to write (use an dedicated programmer to write in code memory)**
3. What is asynchronous data transfer scheme?
   Ans.**In asynchronous data transfer scheme, first the processor sends a request to the device for read/write operation. Then the processor keeps on polling the status of the device. Once the device is ready, the processor executes a data transfer instruction to complete the process.**
4. What are the internal devices of 8255?
   Ans.**The internal devices of 8255 are port-A, port-B, port-C. The ports can be programmed for either input or output function in different operating modes.**
5. What is USART?
   Ans.The device which can be programmed to perform Synchronous or Asynchronous serial communication is called USART (Universal Synchronous Asynchronous Receiver Transmitter). Eg: INTEL 8251
6. What is scanning in keyboard and what is scan time?
   Ans.**The process of sending a zero to each row of a keyboard matrix and reading the columns for key actuation is called scanning. The scan time is the time taken by the processor to scan all the rows one by one starting from first row and coming back to the first row again.**
7. What is programmable peripheral device**?**
   Ans.**If the function performed by the peripheral device can be altered or changed by a program instruction then the peripheral device is called programmable device.**

It have control register. The device can be programmed by sending control word in the prescribed format to the control register.

8. What is baud rate?

Ans.**The baud rate is the rate at which the serial data are transmitted. Baud rate is defined as (The time for a bit cell). In some systems one bit cell has one data bit, then the baud rate and bits/sec are same.**

9. What is a port?

Ans.**The port is a buffered I/O, which is used to hold the data transmitted from the microprocessor to I/O devices and vice versa.**

10. What is the need for interrupt controller?

Ans. **The interrupt controller is employed to expand the interrupt inputs. It can handle the interrupt request from various devices and allow one by one to the processor.**

EXPERIMENT NO 6

AIM: Study of implementation of stepper motor angle control.

APPARATUS: NV5001 Microcontroller development Board, MC-05 Kit, power cord and connecting leads.

DESCRIPTION: The experiment has been designed to have a clear understanding of how motors are interfaced and controlled with microcontroller. The Motor drive module is made in such a way that student can understand the whole concepts of steeper motor, DC motor and Servo system. The object is to connect and program a microcontroller to do any operation with motors. It has input and output terminals for connection of external real world applications.

PROGRAM: To monitor the status of switch and rotate the stepper motor.

```
ANG_SW EQU P2.5
D_ST EQU P2.3
C_ST EQU P2.2
B_ST EQU P2.1
A_ST EQU P2.0
------------------------------------------------------------------------------------------------------------------
ORG 0000H
JMP START
------------------------------------------------------------------------------------------------------------------
ORG 0200H
START : MOV A, #00H
BACK : LCALL DELAY
SETB ANG_SW
SETB A_ST
SETB B_ST
CLR C_ST
CLR D_ST
JNB ANG_SW, SEC_STATE
SJMP BACK
SEC_STATE : LCALL DELAY
SETB ANG_SW
CLR A_ST
SETB B_ST
SETB C_ST
CLR D_ST
JNB ANG_SW, THI_STATE
SJMP SEC_STATE
THI_STATE : LCALL DELAY
SETB ANG_SW
```

```
CLR A_ST
CLR B_ST
SETB C_ST
SETB D_ST


JNB ANG_SW, FOU_STATE
SJMP THI_STATE
FOU_STATE : LCALL DELAY
SETB ANG_SW
SETB A_ST
CLR B_ST
CLR C_ST
SETB D_ST
JNB ANG_SW, BACK
SJMP FOU_STATE
-------------------------------------------------------------------------------------------------------------------
--
DELAY : MOV R2, #20
DO3 : MOV R3, #30
DHERE1 : MOV R4, #70
DAGAIN : NOP
NOP
NOP
DJNZ R4, DAGAIN
DJNZ R3, DHERE1
DJNZ R2, DO3
RET
-------------------------------------------------------------------------------------------------------------------
--

END
```

PROCEDURE:

1. Insert AT89C52 Microcontroller in Programmer unit (in NV5001).
2. Connect serial cable between computer serial port and programmer unit serial port female connector (in NV5001).
3. Switch 'On' the programmer switch in programmer unit (in NV5001) and switch 'On' the power supply.
4. Program angle control.hex file (Via CD - NV5001/ \Modules programs**MC05 Drive module**\Stepper motor interface module\Angle control program) in AT89C52 Microcontroller via programmer.
5. Switch 'Off' the power supply and remove the programmed controller from programmer ZIF socket.

6. Switch 'Off' the programmer switch in Programmer unit (in NV5001).
7. Insert programmed Microcontroller to microcontroller unit ZIF socket.
8. Connect 20 Pin FRC cable to Stepper motor interface block socket (MC05) to Port P2 in NV5001 Trainer.
9. Switch 'On' the power supply.
10. Check the status of port pins on tp1 to tp 4 i.e., A to D pins.
11. Observe the status of Angle control switch 'On' SW2 line or tp 6.
12. Press Angle control switch and observe the direction of rotation of stepper motor.
13. Check the status of port pins on tp1 to tp 4 i.e., A to D pins.
14. Repeat steps 12 and 13 three times and observe the angle change of stepper motor.

RESULT: Stepper motor angle is observed.

Question & Answer:

1. How a keyboard matrix is formed in keyboard interface using 8279?
   Ans.**The return lines, RL0 toRL7 of 8279 are used to form the columns of keyboard matrix. In decoded scan lines SL0 t0SL3 of 8279 are used to form the rows of keyboard matrix. In encoded scan mode, the output lines of external decoder are used as rows of keyboard matrix.**
2. What is GPIB?
   Ans.**GPIB is the General Purpose interface Bus. It is used to interface the test instruments to the system controller.**
3. Advantages of differential data transfer?
   Ans.**1. Communication at high data rate in real world environment. 2. Differential data transmission offers superior performance. 3. Differential signals can help induced noise signals.**
4. Features of INTEL 8259**?**
   Ans.**1. It manages 8 interrupt request. 2. The interrupt vector addresses are programmable. 3. The priorities of interrupts are programmable. 4. The interrupt can be masked or unmasked individually.**
5. What is meant by micro controller?
   Ans.**A device which contains the microprocessor with integrated peripherals like memory, serial ports, parallel ports, timer/counter, interrupt controller, data acquisition interfaces like ADC, DAC is called micro controller**.
6. List the features of 8051 micro controllers**?**
   Ans.**Single supply +5v operation using HMOS technology.**
   · 4096 bytes program memory on-chip. · 128 data memory on chip. · 4 register banks

· 2 multiple modes, 16 bit timer/counter · Extensive Boolean processing capabilities. · 64KB external RAM size. · 32 bi-directional I/O lines.

7. Explain the operating mode 0 of 8051 serial port**?**
   Ans.**In this mode serial data enters and exists through RXD, TXD outputs the shift clock. 8-bits are transmitted or received:8-data bits(LSB first). The baud rate is fixed at 1/12 the oscillator frequency.**
8. Explain the operating mode 2 of 8051 serial port**?**
   Ans. **In this mode 11 bits are transmitted (through TXD) or received (through (RXD): a start bit(0), 8 data bits(LSB first), a programmable 9th data bit and a stop bit(1). On transmit, the 9th data bit can be assigned the value 0 or 1. On receive, the 9th data bit go into the RB8 in special function register SCON, while the stop bit is ignored. The baud rate is programmable to either 1/32 or 1/64 the oscillator frequency.**
9. Explain the mode 3 of 8051 serial port**?**
   Ans.**In this mode, 11 bits are transmitted (through TXD) or (received (through RXD): a start bit(0), 8 data bits(LSB first), a programmable 9th data bit and a stop bit(1).It is same as mode 2 except the baud rate. The baud rate in mode 3 is variable.**
10. Explain the interrupts of 8051 micro controller? ·
    Ans**. External interrupt 0 (IE0) – Highest priority**
    · Timer interrupt 0 (TF0) · External interrupt 1 (IE1) · Timer interrupt 1 (TF1) · Serial port Interrupt Receive interrupt (RI) – lowest priority Transmit interrupt (TI)

EXPERIMENT NO 7

AIM: Study of implementation of Motor control using PWM method.

APPARATUS: NV5001 Microcontroller development Board, MC-05 Kit, power cord and connecting leads.

PROGRAM: To monitor the PWM status and control the speed of DC motor in 100% and 25% duty cycle pulse.

```
PWM_SW EQU P2.4
INPUT2 EQU P2.2
INPUT1 EQU P2.1
PWM_INPUT EQU P2.0
-----------------------------------------------------------------------------------------------------------
ORG 0000H
JMP START
-----------------------------------------------------------------------------------------------------------
ORG 0200H
START : MOV A, #00H
CLR C
SETB PWM_SW
SETB PWM_INPUT
SETB INPUT1
CLR INPUT2
JNB PWM_SW, FIR_ROU
SJMP START
-----------------------------------------------------------------------------------------------------------
FIR_ROU : SETB PWM_SW
CLR PWM_INPUT
LCALL DELAY_1S_2
LCALL DELAY_1S_2
LCALL DELAY_1S_2
LCALL DELAY_1S_2
SETB PWM_INPUT
-----------------------------------------------------------------------------------------------------------
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S


LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
SJMP FIR_ROU
-----------------------------------------------------------------------------------------------------------------
DELAY_1S_2 : MOV R2, #50
DHERE1_1_1 : MOV R3, #100
DHERE1_2 : NOP
DJNZ R3, DHERE1_2
DJNZ R2, DHERE1_1_1
RET
-----------------------------------------------------------------------------------------------------------------
DELAY_1S : MOV R2,#100
DO3_3 : DEC R2
DJNZ R2, DO3_3
RET
-------------------------------------------------------------------------------------------------------------------
END
```

PROCEDURE:

1. Insert AT89C52 Microcontroller in Programmer unit (in NV5001).
2. Connect serial cable between computer serial port and programmer unit serial port female connector (in NV5001).
3. Switch 'On' the programmer switch in programmer unit (in NV5001) and switch 'On' the power supply.
4. Program PWM interface.hex file (Via CD - NV5001/ \Modules programs\MC05 Drive module \DC motor interface module\PWM Interface program) in AT89C52 Microcontroller via programmer.
5. Switch 'Off' the power supply and remove the programmed controller from programmer ZIF socket
6. Switch 'Off' the programmer switch in Programmer unit (in NV5001).
7. Insert programmed Microcontroller to microcontroller unit ZIF socket.
8. Connect 20 Pin FRC cable to DC motor /PWM interface block socket (MC05) to Port P2 in NV5001 Trainer.
9. Connect 2 mm patch cord between +12V DC block socket (in NV5001) to +12V DC socket in DC motor /PWM interface block (in MC05).
10. Switch 'On' the power supply.
11. Check the status of port pins on tp7 to tp11

12. Observe the status of PWM switch at tp11.
13. Observe the rotation speed of DC Motor.
14. Press PWM switch and repeat steps 11 to 13 one time and observe the speed change of DC motor.

RESULT: Wave observed on CRO and duty cycle is measured.

Question & Answer:

1. How many bytes of internal RAM and ROM supported by 8051 micro controller?
   Ans.**128 bytes of internal RAM and 4 bytes of ROM.**
2. Define machine cycle of 8051?
   Ans.**8051 machine cycle consists of 6 states, S1 through S7. One state is made up of 2 clock pulses. Thus 12 clock period constitute one machine cycle. Two clock periods in a state is termed as phase 1 and phase 2.**
3. What are the special function of port 0 of 8051**?**
   Ans.**Port 0 is used as a multiplexed low order address/data bus during the external memory access. When ALE is enabled, the addresses on port 0 pins are latched and bus is ready to act as a data bus when ALE is low.**
4. What are the alternative function of port 3 of 8051?
   Ans.**Serial data input (P3.0), serial data output (P3.1), external interrupt 0 (P3.2), external interrupt 1 (P3.3), external input for timer 0(P3.4), external input for timer 1 (P3.5), external memory write pulse (P3.6), external memory read (P3.7) are the alternative functions of port 3.**
5. What are the use of scratch pad area of internal RAM of 8051**?**
   Ans.**In internal RAM 80 bytes constitutes the scratch pad area. The scratch pad bytes can be programmed as a general purpose registers.**
6. What are the flags supported by 8051 controller? –
   Ans. **Carry flag · Auxiliary carry flag · Over flow flag · General purpose user flag · Register bank select bit one · Register bank select bit zero · Parity flag**
7. What is meant by Power-on- Reset in 8051 controller?
   Ans.**When RESET pin is activated, the 8051 jumps to address location 0000H. This is called as Power-on-Reset. Reset pin is considered as a sixth interrupt source of 8051.**
8. What are the significance of SFRs**?**
   Ans. **All the controller registers such as port latches, timer register, peripheral control register, accumulator, PC and DPTR all are available in SFR region.**
9. What are the different group of instructions supported by 8051**?** ·
   Ans.**Data Transfer Group · Arithmetic Logical Branching Bit manipulation**

10. Write a program to mask the 0th and 7th bit using 8051?

Ans:

```
MOV A,#data
ANL A,#81
MOV DPTR,#4500
MOVX @DPTR,A
LOOP:  SJMP LOOP
```

EXPERIMENT NO 8

AIM: Study and observation of Position control of Servo Motor.

APPARATUS: NV5001 Microcontroller development Board, MC-05 Kit, power cord and
connecting leads.

PROGRAM: To monitor the status of position control switch and control the angle of servo motor.

```
SERVO_PIN EQU P2.0
SW_PIN EQU P2.1
----------------------------------------------------------------------------------------------------
-------
ORG 0000H
JMP START
----------------------------------------------------------------------------------------------------
------
ORG 0200H
START : CLR SERVO_PIN
SETB SW_PIN
LOOP_S : LCALL DELAY
SETB SW_PIN
SETB SERVO_PIN
LCALL DELAY_15MS_P
CLR SERVO_PIN
LCALL DELAY_16MS
JNB SW_PIN, SW_1_1
LCALL DELAY
SJMP LOOP_S
-----------------------------------------------------------------------------------------------------------------
SW_1_1 : LCALL DELAY
SETB SW_PIN
SETB SERVO_PIN
LCALL DELAY_25MS_P
CLR SERVO_PIN
LCALL DELAY_16MS
LCALL DELAY
SJMP SW_1_1
-----------------------------------------------------------------------------------------------------------------
DELAY_16MS : MOV R2, #150
DHERE1_16 : MOV R3, #32
DAGAIN_16 : NOP
```

```
DJNZ R3, DAGAIN_16
DJNZ R2, DHERE1_16
RET
-----------------------------------------------------------------------------------------------------------------



DELAY_25MS_P : MOV R2, #20
DHERE1_25_P : MOV R3, #37
DAGAIN_25_P : NOP
DJNZ R3, DAGAIN_25_P
DJNZ R2, DHERE1_25_P
RET
-----------------------------------------------------------------------------------------------------------------
DELAY_15MS_P : MOV R2, #20
DHERE1_15_P : MOV R3, #20
DAGAIN_15_P : NOP
DJNZ R3, DAGAIN_15_P
DJNZ R2, DHERE1_15_P
RET
DELAY : MOV R5, #250
DHERE1 : MOV R4, #220
DAGAIN : NOP
NOP
DJNZ R4, DAGAIN
DJNZ R5, DHERE1
RET
------------------------------------------------------------------------------------------------------------------
END
```

PROCEDURE:

1. Insert AT89C52 Microcontroller in Programmer unit (in NV5001).
2. Connect serial cable between computer serial port and programmer unit serial port female connector (in NV5001).
3. Switch ‘On’ the programmer switch in programmer unit (in NV5001) and switch ‘On’ the power supply.
4. Program servo motor module.hex file (Via CD - NV5001/ \Modules programs\MC05 Drive module \DC motor interface module\Servo motor module) in AT89C52 Microcontroller via programmer.
5. Switch ‘Off’ the power supply and remove the programmed controller from programmer ZIF socket
6. Switch ‘Off’ the programmer switch in Programmer unit (in NV5001).

7. Insert programmed Microcontroller to microcontroller unit ZIF socket.
8. Connect 20 Pin FRC cable to servo motor interface block socket (MC05) to Port P2 in NV5001 Trainer.
9. Switch 'On' the power supply.
10. Check the status of port pins on tp12 to tp13.
11. Observe servo motor rotates and stop in the centre position or in 90 degree angle.
12. Press position control switch and repeat steps 10.
13. Observe servo motor rotates and stop in l80 degree angle or in a left side position.

RESULT: Angle rotation of the servo motor observed.

Question & Answer:

1. What is a Data pointer register?
   Ans.**The data pointer register (DPTR) consists of a high byte(DPH) and a low byte (DPL) functions to hold 16 bit address. It may be manipulated as a 16-bit data register or as independent 8-bit registers. It serves as a base register in indirect jumps, look up table instructions and external data transfer.**
2. What are the operating modes of 8279?
   Ans.**1. Input modes · Scanned keyboard · Scanned sensor matrix · Strobed input 2. Display modes · Left entry (Type writer mode) · Right entry (Calculator mode)**
3. What are the different functional units in 8279?
   Ans.**CPU interface section, Keyboard section, Display section, Scan section**
4. What are the priority modes in 8259?
   Ans.**a. Fully nested mode b. Special fully nested mode c. Rotating Priority mode d. Special Masked mode e. Polled mode**
5. What is IMR(Interrupt mask register)?
   Ans.**IMR stores the masking bits of the interrupt lines to be masked. This register can be programmed by an operation command word (OCW).**
6. What is priority resolver?
   Ans. **It determines the priorities of the bits set in the Interrupt request register (IRR).The bit corresponding to the highest priority interrupt input is set in the ISR during INTA input.**
7. What is the use of IRR?
   Ans.**The interrupt request register is used to store all the interrupt levels which are requesting the service. The eight interrupt inputs sets corresponding bits of the Interrupt Request Register upon the service request.**
8. What is Interrupt service register(ISR)?

Ans.**The interrupt service register stores all the levels that are currently being serviced.**

9. What is the difference between SHLD and LHLD?
Ans.**SHLD- Store HL register pair in memory.**
This instruction is used to store the contents of H and L register directly in to memory.
LHLD- Load HL register pair from memory. This instruction copies the contents of memory location given with in the instruction in to the L register and the contents of next memory location in to the H register.

10. What is the difference between STAX and LDAX?
Ans.**STAX rp – Store the contents of Accumulator register (A) in memory location whose address is specified by BC or DE register pair.**
LDAX rp – Load Accumulator register (A) with the contents of memory location whose address is specified by BC or DE register pair.

EXPERIMENT NO 9 (a)

AIM: Study of Programming and Transmission of data through serial port.

APPARATUS: NV5001 Microcontroller development Board, MC-03 Kit, power cord and connecting leads.

DESCRIPTION: Computer interface module for Microcontroller development board with programmer trainer is an Extension module. The module has been designed to have a clear understanding of how serial port and parallel port interfaced devices are controlled and interface with microcontroller. The apparatus is connected with microcontroller unit and PC. The computer interface trainer is made in such a way that student can understand the whole concepts of serial and parallel port and how they are interfaced with microcontroller.

INSTALLATION:

If your computer doesn't have Hyper Terminal software please follow the below procedure.
1. First go to control panel and click Add/Remove Program.
2. Click Windows Setup.
3. Select Hyperterminal then press OK and click Apply.
4. Select Hyperterminal then press OK and click Apply.
5. Computer will ask for Windows CD for installing Hyperterminal.
6. After installing hyperterminal click cursor on
<start> <programs><Accessories><communications> <Hyperterminal> then open it.
7. Click on < Hyperterminal> icon give ABC in name block and click OK.
8. Select Communication port < Direct to COM1>.
9. For Trainer kit select following COM 1 properties :
Bits per second : 9600
Data bits : 8
Parity : None
Stop bits : 1
Flow control : None

PROGRAM: To transfer the message "YES" serially at 9600 baud, 8 bit data and 1stop bit continuously.

```
ORG 0000H
JMP START
--------------------------------------------------------------------------------------------------------------
ORG 0200H
START: MOV A, #00H
MOV PSW, #00H
MOV SBUF, #00H
```

```
CLR C
SERI_TRANSMIT: MOV P3, #0FFH
MOV TMOD, #20H
MOV TH1, #0FDH
MOV TL1, #0FDH

MOV SCON, #50H
SETB TR1
CLR TI
CLR RI
MOV A, #"Y"
LCALL TRANSMIT
MOV A, #"E"
LCALL TRANSMIT
MOV A, #"S"
LCALL TRANSMIT
MOV A, #" "
LCALL TRANSMIT
LJMP SERI_TRANSMIT
TRANSMIT: MOV SBUF, A
MOV B, A
HERE_S: JNB TI, HERE_S
CLR TI
RET
-----------------------------------------------------------------------------------------------------------------
END
```

PROCEDURE:

1. Insert AT89C52 Microcontroller in Programmer unit (in NV5001).
2. Connect serial cable between computer serial port and programmer unit serial port female connector (in NV5001).
3. Switch 'On' the programmer switch in programmer unit (in NV5001) and switch 'On' the power supply.
4. Program Serial transmit.hex file (Via CD - NV5001/ Modules programs\ MC03 Computer Interface module\Serial port interface module\Serial transmit) in AT89C52 Microcontroller via programmer.
5. Switch 'Off' the power supply and remove the programmed controller from programmer ZIF socket.
6. Switch 'Off' the programmer switch in Programmer unit (in NV5001).
7. Insert programmed Microcontroller to microcontroller unit ZIF socket.
8. Connect 20 Pin FRC cable to Serial port Interface block socket/connector (MC03) to Port P3 in NV5001 Trainer.
9. Unplug the serial cable which is connected between programmer unit and PC.
10. Connect serial cable between computer serial port and serial port interface block

female connector (inMC03).
11. Click on < Hyperterminal> icon in your Pc and give ABC in name block then click OK.
12. Select Communication port < Direct to COM1 > (If you connect cable to COM1)
13. For Module select following COM 1 properties:
Bits per second : 9600
Data bits : 8
Parity : None
Stop bits : 1
Flow control : None
14. Switch 'On' the power supply.

15. Observe transmitted data on Hyperterminal window (message "YES ").This Y, E and S
character transmitted through microcontroller and received by PC through serial cable.
16. To see transmitted data connect CRO probe between tp1, tp2, tp3 & tp4 test points and
observe it.

RESULT: Data 'yes' is checked on Hyper Terminal transferred by serial port.

EXPERIMENT 9 (b)

AIM: Study of Programming and Reception of data through serial port.

APPARATUS: NV5001 Microcontroller development Board, MC-03 Kit, power cord and connecting leads.

PROGRAM: To receive bytes of data serially and put them in P0. Set the baud rate at 9600, 8- bit data and 1 stop bit.

```
RS_LCD EQU P2.5
RW_LCD EQU P2.6
E_LCD EQU P2.7
ORG 0000H
JMP START
------------------------------------------------------------------------------------------------------------------
ORG 0200H
START: MOV PSW, #00H
CLR C
MOV P0, #00H
MOV A, #00H
LCALL LCDINIT
MOV A, #084H
LCALL COMMAND
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #"N"
LCALL DISPLAY
MOV A, #"v"
LCALL DISPLAY
MOV A, #"i"
LCALL DISPLAY
MOV A, #"s"
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
LOOP_RECEIVE: MOV SBUF, #00H
-------------------------------------------------------------------------------------------------------------------
MOV A, #00H
MOV B, #00H
MOV P0, #00H
MOV P3, #0FFH


MOV TMOD, #20H
MOV TH1, #0FDH
MOV TL1, #0FDH
MOV SCON, #50H
SETB TR1
CLR RI
-----------------------------------------------------------------------------------------------------------------------
HERE_RECEIVE: JNB RI, HERE_RECEIVE
MOV A, SBUF
MOV B, A
MOV A, #084H
LCALL COMMAND
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #" "
LCALL DISPLAY
MOV A, #084H
LCALL COMMAND
-----------------------------------------------------------------------------------------------------------------------
MOV A, B
LCALL DISPLAY
CLR RI
LJMP LOOP_RECEIVE
LCDINIT: MOV A, #38H
LCALL COMMAND
MOV A, #0CH
```

```
LCALL COMMAND
MOV A, #01H
LCALL COMMAND
MOV A, #06H
LCALL COMMAND
RET
-----------------------------------------------------------------------------------------------------------------
COMMAND:
ACALL READY
MOV P0, A

CLR RS_LCD
CLR RW_LCD
SETB E_LCD
CLR E_LCD
RET
-----------------------------------------------------------------------------------------------------------------
DISPLAY:
ACALL READY
MOV P0, A
SETB RS_LCD
CLR RW_LCD
SETB E_LCD
CLR E_LCD
RET
READY: SETB P0.7
CLR RS_LCD
SETB RW_LCD
WAIT: CLR E_LCD
SETB E_LCD
JB P0.7, WAIT
RET
-----------------------------------------------------------------------------------------------------------------
DELAY_1S:
MOV R2, #11
DO3: MOV R3, #98
DHERE1: MOV R4, #106
DAGAIN: NOP
NOP
NOP
NOP
NOP
NOP
DJNZ R4, DAGAIN
DJNZ R3, DHERE1
DJNZ R2, DO3
RET
-----------------------------------------------------------------------------------------------------------
END
```

PROCEDURE:

1. Insert AT89C52 Microcontroller in Programmer unit (in NV5001).
2. Connect serial cable between computer serial port and programmer unit serial port female connector (in NV5001).
3. Switch 'On' the programmer switch in programmer unit (in NV5001) and switch 'On' the power supply.
4. Program Serial receive.hex file (Via CD - NV5001/ Modules programs\ MC03 Computer Interface module\Serial port interface module\Serial receive) in AT89C52 Microcontroller via programmer.

5. Switch 'Off' the power supply and remove the programmed controller from programmer ZIF socket.
6. Switch 'Off' the programmer switch in Programmer unit (in NV5001).
7. Insert programmed microcontroller to microcontroller unit ZIF socket.
8. Connect 20 Pin FRC cable to Serial port Interface block socket/connector (MC03) to Port P3 in NV5001 Trainer.
9. Connect 20 Pin FRC cable to LCD Interface block left side socket/connector (MC04) to Port P2 in NV5001 Trainer.
10. Connect 20 Pin FRC cable to LCD Interface block right side socket/connector (MC04) to Port P0 in NV5001 Trainer.
11. Unplug the serial cable which is connected between programmer unit and PC.
12. Connect serial cable between computer serial port and serial port interface block female connector (inMC03).
13. Click on < Hyperterminal> icon in your PC and give ABC in name block then click OK.
14. Select Communication port < Direct to COM1 > (If you connect cable to COM1).
15. For Module select following COM 1 properties:
Bits per second : 9600
Data bits : 8
Parity : None
Stop bits : 1
Flow control : None
16. Switch 'On' the power supply.
17. "Nvis" is displayed on LCD screen.
18. Now press any key from keyboard (Keep cursor on hyper window).The character you pressed is displayed on LCD. The character is received by microcontroller via serial cable and displayed on LCD.
19. To see received data connect CRO probe between tp1 & tp3 test point and ground.

20. To see received data connect CRO probe between tp3 & tp4 test point and ground.

RESULT: Serial data is observed on port 0.

Question & Answer:

1. Translator for low level programming language were termed as
   Ans.**assembler.**
2. Load address for the first word of the program is called
   Ans. **load address origin.**
3. Shell is the exclusive feature of
   Ans. **UNIX.**
4. A program in execution is called
   Ans. **process.**
5. A scheduler which selects processes from secondary storage device is called
   Ans.**medium term scheduler.**
6. The term 'page traffic' describes
   Ans.**the movement of pages in and out of memory.**
7. Program 'preemption' is
   Ans. **forced de allocation of the CPU from a program which is executing on the CPU.**
8. 'LRU' page replacement policy is
   Ans.**least recently used.**
9. "Throughput" of a system is
   Ans.**number of programs processed by it per unit time.**
10. Nested Macro calls are expanded using the
    Ans.**LIFO (Last in First out).**

EXPERIMENT NO 10

AIM: To study implementation and programming of Pressure measurement.

APPARATUS: NV5001 Microcontroller development Board, MC-15 Kit, power cord and connecting leads.

DESCRIPTION: Sensor module, MC15 has input and output terminals for connection of External real world applications. Pressure Sensor and Temperature Sensor Interface Module MC15 is generally used in the applications such as Monitoring and Controlling in Industries and many more.

PROGRAM:

```
#include <LPC214x.H>                                    /* LPC214x definitions */
#include <stdio.h>
/************************************************************************
*  Function Prototypes
************************************************************************/
void LCD_Init(void);                                    /* LCD Init Function  */
void LCD_Delay(unsigned int);                          /* LCD Delay Function  */
void LCD_Cmd(unsigned long);                         /* LCD Command Function */
void LCD_Data(unsigned long);                           /* LCD Data Function  */
void LCD_Disp(unsigned char, char *);                 /* LCD Display Function */
void display_lcd1 (unsigned char location, char *d);
void ADC_Init(void);                                     /* ADC Init Function */
void ADC_Convert(unsigned int);                       /* ADC Display Function */
void ADC_CALL(void);
/*************************************************************************
*  LCD Pin Out Description
*************************************************************************
/
#define RS_Set IO1SET = 0x20000000;
#define RS_Clr IO1CLR = 0x20000000;
#define EN_Set IO1SET = 0x80000000;
#define EN_Clr IO1CLR = 0x80000000;
unsigned int ADC_Val;                                       /* ADC Result (HEX)
*/
unsigned int FirstBit,SecondBit,ThrdBit,FourthBit;
/*************************************************************************

*Delay
*Description : This function provide Delay in Mili Sec.
*************************************************************************
/
```

```
void LCD_Delay(unsigned int Time)
{
   unsigned int i,j;
   for(i=0;i<=Time;i++)
   for(j=0;j<110;j++);
}




/***************************************************************************
*
* ADC initialization
* Description: This function initializes the LCD module by the following steps:
*1. Select ADC Channel
*2. Set A/D: 10-bit AIN0 @ 12MHz
* Note: This function should be called once before any of the other functions of ADC.
****************************************************************************
/
void ADC_Init()
{
PINSEL0 |= 0x00003000;                                          /* channel
AD1.0*/
AD1CR    = 0x01210400;                            /* Setup A/D: 10-bit AIN0 @ 3MHz
*/
}
/***************************************************************************
*
* ADC Conversion
* Description: This function convert ADC data into ASCII by the following steps:
* 1. Convert each byte into ASCII
* 2. Each Value will be used for LCD Display
* Arguments   : 'RADC_Value'  is the Value which needs to convert in ASCII.
****************************************************************************
/
void ADC_Convert(unsigned int ADC_Value)
{
 unsigned int X,Y,Z;                                        /* Intermidiate Variables
*/
 FirstBit=0,SecondBit=0,ThrdBit=0,FourthBit=0;
 X = ADC_Value/10;
 FirstBit = ADC_Value%10;
 FirstBit = 0x30|FirstBit;                                       /* First Byte(LSB)
*/
 Y = X/10;
 SecondBit = X % 10;
 SecondBit = 0x30|SecondBit;                                          /* Second Byte
*/
 Z = Y/10;
 ThrdBit = Y % 10;
 ThrdBit = 0x30|ThrdBit;                                                /* Third Byte */
 FourthBit = Z;                                                   /* Last Byte(MSB) */
 FourthBit=0x30|FourthBit;
}
/***********************************************************************
*
* LCD initialization
* Description : This function initializes the LCD module by the following steps:
* 1. Set 8bit : 2 Line 5x7 Dots (0x38)
* 2. Display On curser Off (0x0C)
* 3. Clear Display (0x01)
*4. Entry Mode   (0x06)
* Note: This function should be called once before any of the other functions
***********************************************************************/
void LCD_Init(void)
{
LCD_Cmd(0x38);                               /* Function Set 8bit : 2 Line 5x7 Dots */
LCD_Cmd(0x0C);                                          /* Display On curser Off */
LCD_Cmd(0x01);                                                  /* Clear Display */


LCD_Cmd(0X06);                                                     /* Entry Mode */
}
/***********************************************************************
*
* LCD Command (Shifting is Done here)
* Description : This function initializes the LCD module by the following steps:
* Note : Here we have selected Pin P1.16 to P1.23 as LCD data line that's
* why we need to shift data by 16.
************************************************************************
/
void LCD_Cmd(unsigned long Cmd)
{
  unsigned long Shifted_Cmd;
  Shifted_Cmd = Cmd << 16;    /* because We have selected P1.16 to P1.23 as LCD data line
*/

  RS_Clr;                                                             /* RS Pin
Clear */
  LCD_Delay(10);                                                    /* Delay for
```

```
Clock*/
 EN_Set;                                                        /* Enable Pin
SET */
 LCD_Delay(10);
 IO1SET = Shifted_Cmd;                         /* Write Command Value to LCD Data
Pin */
 LCD_Delay(20);                                            /* Delay for enable
Clock*/
 EN_Clr;                                                        /* Enable Pin
Clear */
 LCD_Delay(10);
 IO1CLR = 0x00FF0000;                                       /* Clear All pins of
LCD */
}

/*************************************************************************
*
* LCD Data
* Description: This function initializes the LCD module by the following steps:
* 1. Set Register select(RS) to High
* 2. Set enable pin high to low
* 3. Send data to LCd data pin
* Note: Here we have selected Port 1 Pin P1.16 to P1.23 as LCD data line
* thats why we need to shift data by 16.
**************************************************************************
/
void LCD_Data(unsigned long Data)
{
 RS_Set;                                                        /* RS Pin Clear
*/
 LCD_Delay(10);                                                  /* Delay for
Clock*/
 EN_Set;                                                      /* Enable Pin SET
*/
 LCD_Delay(10);
 IO1SET = Data << 16;                         /* Write Command Value to LCD Data Pin
*/
 LCD_Delay(20);                                           /* Delay for enable
Clock*/
 EN_Clr;                                                     /* Enable Pin Clear
*/
 LCD_Delay(10);
 IO1CLR = 0x00FF0000;                                      /* Clear All pins of
LCD */
}
```

```
/***************************************************************************
*
* LCD Display
* Description : This function initializes the LCD module by the following steps:
* 1. Send Loc from where data needs to write


* 2. Display string on LCD
****************************************************************************
/
void LCD_Disp(unsigned char Loc,  char *String)
{
 LCD_Cmd(Loc);                                              /* Send Command to LCD
*/
 while(*String)                                          /*    wait untill Null char come
*/
{
          LCD_Data(*String++);                                      /* Write data to LCD
*/
          IO1CLR = 0x00FF0000;                                  /* Clear All pins of LCD
*/
         }
}

void display_lcd1 (unsigned char location,  char *d)
{
 unsigned long shift_data;
 shift_data= 0x80 | location;
 shift_data= shift_data<<16;
 LCD_Cmd(shift_data);
 LCD_Delay(100);
 while(*d)
        {
          LCD_Data(*d++);
          LCD_Delay(100);
          IO1CLR = 0x00FF0000;                     // ADDED NEW
         }
}

void ADC_CALL(void)
{
  AD1CR |= 0x01000000;                                     /* start of ADC conversion
*/
  do{
        ADC_Val = AD1DR0;                                             /* 10 bit value
*/
       }while ((ADC_Val & 0x80000000) == 0);           /* Wait ADC Conversion Complete
*/

   AD1CR &= ~0x01000000;                                         /* Again start ADC
*/
   ADC_Val = (ADC_Val >> 6) & 0x03FE;
}

/**************************************************************************
*
* Main: Initialize and start RTX Kernel
* Description : This function Initialize the LCD,RTC and ADC and RTX Kernal
***************************************************************************
/
int main (void)         /* program exec. starts here  */
{
 float volt[20],dispvolt;
char k[5];int i;
IO1DIR = 0xFFFF0000;                    /* Define Port pin P1.16 to P1.31 as Output for LCD
*/
LCD_Init();                                                            /* LCD Initialize
*/
ADC_Init();                                                             * ADC Initialize
*/

display_lcd1(0x80,"     Voltage");
while(1)
            {
              for(i=0;i<=19;i++)
              {
               ADC_CALL();
               volt[i] = (ADC_Val * 3.3) / 1023.0;
              }
dispvolt =
(volt[0]+volt[1]+volt[2]+volt[3]+volt[4]+volt[5]+volt[6]+volt[7]+volt[8]+volt[9]+volt[10]+
volt[11] +volt[12]+volt[13]+volt[14]+volt[15]+volt[16]+volt[17]+volt[18]+volt[19])/20;
sprintf(k,"%f",dispvolt + 0.01);
LCD_Cmd(0xC6);
LCD_Data(k[0]);
LCD_Cmd(0xC7);
LCD_Data(k[1]);
LCD_Cmd(0xC8);
LCD_Data(k[2]);
```

```
LCD_Cmd(0xC9);
LCD_Data(k[3]);
LCD_Cmd(0xCA);
LCD_Data(k[4]);
LCD_Cmd(0xCC);
LCD_Data('V');
LCD_Delay(200000);
}
}
/*---------------------------------------------------------------------------
* end of file
*--------------------------------------------------------------------------*/
```

Pressure Sensor Module MC-15 with NV500X series

PROCEDURE:

1. Connect 20 Pin FRC cable between Sensor Module (MC15) & Port of NV500Xseries Trainer as mentioned in respective programs.
2. Connect LCD Module compatible with NV500X series Trainer LCD Data and Control Pins to Ports of NV500X series trainers as mentioned in respective programs (Optional).
3. Make the connection of Sensor Module with NV500X series as given in the respective program.
4. Observer the voltage that is being displayed on the LCD and note down the readings.

RESULT: Pressure measured and corresponding digital value observed on the LCD.

Question & Answer:

1. A UNIX device driver is
   Ans.**structured into two halves called top half and bottom half**.
2. Before proceeding with its execution, each process must acquire all the resources it needs is called
   Ans. **hold and wait.**
3. Resolution of externally defined symbols is performed
   Ans. **by linker.**
4. Access time is faster for
   Ans. **SRAM.**
5. For the most static RAM, the write pulse width should be at least
   Ans.**60 ns**.
6. For the most static RAM, the maximum access time is about
   Ans.**100ns**.
7. An embedded microcontroller means

Ans. **a microcontroller with external memories storing the embedded software.**

8. The translator which perform macro expansion is called a
   Ans. **Macro pre-processor**.
9. Symbolic names can be associated with
   Ans.**data or instruction**.
10. The scheduling in which CPU is allocated to the process with least CPU-burst time is called Ans.**Shortest job first Scheduling**.

EXPERIMENT NO 11

AIM: To study implementation and programming of Temperature measurement.

APPARATUS: NV5001 Microcontroller development Board, MC-15 Kit, power cord and connecting leads.

PROGRAM:

```
#include <LPC214x.H>                                    /* LPC214x definitions
*/
#include <stdio.h>
/**************************************************************************
*
 Function Prototypes
void LCD_Init(void);                                      /* LCD Init Function
*/
void LCD_Delay(unsigned int);                            /* LCD Delay Function
*/
void LCD_Cmd(unsigned long);                          /* LCD Command Function
*/ void LCD_Data(unsigned long);                          /* LCD Data Function
*/    void LCD_Disp(unsigned char,unsigned char *);             /* LCD Display
Function */
void ADC_CALL(void);
void ADC_Init(void);                                      /* ADC Init Function
*/
void ADC_Convert(unsigned int);                        /* ADC Display Function
*/
/**************************************************************************
*
* LCD Pin Out Discription
#define RS_Set IO1SET = 0x20000000;
#define RS_Clr IO1CLR = 0x20000000;

#define RW_Set IO1SET = 0x40000000;
#define RW_Clr IO1CLR = 0x40000000;

#define EN_Set IO1SET = 0x80000000;
#define EN_Clr IO1CLR = 0x80000000;

unsigned int ADC_Val;                                    /* ADC Result (HEX)
*/
unsigned int FirstBit,SecondBit,ThrdBit,FourthBit;
```

```
/*****************************************************************************
**                                                                          *]
* Description : This function provide Delay in Mili Sec.
*****************************************************************************
*/
void Delay(unsigned int Time)
{
  unsigned int i,j;
  for(i=0;i<=Time;i++)
  for(j=0;j<1275;j++);
}




/*****************************************************************************
*
* LCD_Delay
* Description : This function provide Delay in Mili Sec.
*****************************************************************************
*/
void LCD_Delay(unsigned int Time)
{
unsigned int i,j;
for(i=0;i<=Time;i++)
 for(j=0;j<1005;j++);
}

/*****************************************************************************
***
*  ADC initialization
* Description : This function initializes the LCD module by the following steps:
* 1. Select ADC Channel
* 2. Set A/D: 10-bit AIN0 @ 12MHz
* Note: This function should be called once before any of the other functions of ADC.
*****************************************************************************
***/
void ADC_Init()
{
 PINSEL0 |= 0x00003000;                                          /* channel
AD1.0*/
AD1CR |= 0x01210400;
}
```

```
/*****************************************************************************
**
* ADC Conversion
* Description : This function convert ADC data into ASCII by the following steps:*
* 1. Conver each byte into ASCII
* 2. Each Value will be used for LCD Display
* Arguments   : 'RADC_Value'  is the Value which needs to convert in ASCII.
******************************************************************************
**/
void ADC_Convert(unsigned int ADC_Value)
{
 unsigned int X,Y,Z;                                          /* Intermidiate
Variables */
 FirstBit=0,SecondBit=0,ThrdBit=0,FourthBit=0;
 X = ADC_Value/10;
 FirstBit = ADC_Value%10;
 FirstBit = 0x30|FirstBit;                                         /* First Byte
(LSB) */
   Y = X/10;
SecondBit = X % 10;
SecondBit = 0x30|SecondBit;                                          /* Second
Byte */

 Z = Y/10;
ThrdBit = Y % 10;
ThrdBit = 0x30|ThrdBit;                                                 /* Third
Byte */
 FourthBit = Z;                                                     /* Last
Byte(MSB) */
 FourthBit=0x30|FourthBit;
}



/*****************************************************************************
*
* LCD initialization
* Description : This function initializes the LCD module by the following steps:
* 1. Set 8bit : 2 Line 5x7 Dots (0x38)
* 2. Display On curser Off   (0x0C)
* 3. Clear Display   (0x01)
* 4. Entry Mode   (0x06)
* Note : This function should be called once before any of the other functions
******************************************************************************
**/
```

```
void LCD_Init(void)
{
 LCD_Cmd(0x38);                                    /* Function Set 8bit : 2 Line 5x7
Dots */
 LCD_Cmd(0x0C);                                              /* Display On curser
Off */
 LCD_Cmd(0x01);                                                     /* Clear
Display */
 LCD_Cmd(0X06);                                                      /* Entry
Mode */
}
/****************************************************************************
**                                                                                  *]
*  Description : This function initializes the LCD module by the following steps:
* Note : Here we have selected Pin P1.16 to P1.23 as LCD data line thats why we need to
 * shift data by 16.
*****************************************************************************
**/
void LCD_Cmd(unsigned long Cmd)
{
unsigned long Shifted_Cmd;
Shifted_Cmd = Cmd << 16;          /* because we have selected P1.16 to P1.23 as LCD data line
*/
// RW_Clr;                                                          /* RW Pin Clear
*/

RS_Clr;                                                             /* RS Pin Clear
*/
LCD_Delay(5);                                                         /* Delay for
Clock*/
EN_Set;                                                           /* Enable Pin SET
*/
LCD_Delay(5);
IO1SET = Shifted_Cmd;                              /* Write Command Value to LCD Data Pin
*/
LCD_Delay(10);                                              /* Delay for enable
Clock*/
EN_Clr;                                                           /* Enable Pin Clear
*/
LCD_Delay(5);
IO1CLR = 0x00FF0000;                                          /* Clear All pins of LCD
*/
}

/****************************************************************************
```

```
* LCD Data
* Description : This function initializes the LCD module by the following steps:
* 1. Set Register select(RS) to High
* 2. Set enable pin high to low
* 3. Send data to LCd data pin
* Note: Here we have selected Port 1 Pin P1.16 to P1.23 as LCD data line thats why we need to
shift
          data by 16.
******************************************************************************
**/



void LCD_Data(unsigned long Data)
{

RS_Set;                                                                    /* RS Pin
Clear */
LCD_Delay(5);                                                             /* Delay for
Clock*/
EN_Set;                                                                  /* Enable Pin
SET */
LCD_Delay(5);
IO1SET = Data << 16;                                 /* Write Command Value to LCD Data
Pin */
LCD_Delay(10);                                                     /* Delay for enable
Clock*/
EN_Clr;                                                                  /* Enable Pin
Clear */
LCD_Delay(5);
IO1CLR = 0x00FF0000;                                                /* Clear All pins of
LCD */
}

/*****************************************************************************
**
* LCD Display
* Description : This function initializes the LCD module by the following steps:
* 1. Send Loc from where data needs to write
* 2. Display string on LCD
******************************************************************************
*/

void LCD_Disp(unsigned char Loc, unsigned char *String)
{
  LCD_Cmd(Loc);                                                    /* Send Command to
LCD */
  while(*String != '\n')                                          /* Wait untill Null char
come */
{
LCD_Data(*String++);                                                    /* Write data to
LCD */
LCD_Delay(10);
CLR = 0x00FF0000;                                                /* Clear All pins of LCD
*/
 }
}
void ADC_CALL(void)
{
 do{
       ADC_Val = AD1DR0;                                                        /* 10 bit
value */
     }while ((ADC_Val & 0x80000000) == 0);                 /* Wait ADC Conversion
Complete */

AD1CR &= ~0x01000000;                                                     /* Again start
ADC */
ADC_Val = (ADC_Val >> 6) & 0x03FE;
}
```

```
/****************************************************************************
**
* Main: Initialize and start RTX Kernel
* Description : This function Initialize the LCD,RTC and ADC and RTX Kernal
*****************************************************************************
*/
int main (void)                                                 /* program exec. starts here
*/
{
int i;
```

```
 float temp[10],
disptemp;IO1DIR = 0xF8FF0000;              /* Define Port pin P1.16 to P1.31 as Output for
LCD */
LCD_Init();                                                          /* LCD
Initialize */
ADC_Init();                                                          /* ADC
Initialize*/
LCD_Cmd(0x01);                                                       /* Clear
Display */
while(1)                                                             /* Loop
Continue*/
 {
LCD_Cmd(0x83);                                                   /* Set Curser at
'0x83' */
LCD_Data('T');                                            /* Display ADC Value on
LCD */
LCD_Cmd(0x84);
LCD_Data('e');
LCD_Cmd(0x85);
LCD_Data('m');
LCD_Cmd(0x86);
LCD_Data('p');
LCD_Cmd(0x87);
LCD_Data('e');
LCD_Cmd(0x88);
LCD_Data('r');
LCD_Cmd(0x89);
LCD_Data('a');
LCD_Cmd(0x8A);
LCD_Data('t');
LCD_Cmd(0x8B);
LCD_Data('u');
LCD_Cmd(0x8C);
LCD_Data('r');
LCD_Cmd(0x8D);
LCD_Data('e');
for(i=0;i<=9;i++)
{
ADC_CALL();
temp[i] =  ((ADC_Val * 10) /33);
}
   disptemp =
(temp[0]+temp[1]+temp[2]+temp[3]+temp[4]+temp[5]+temp[6]+temp[7]+temp[8]+temp[9])/10
;
```

```
 //    ADC_Val = ADC_Val /4;
//ADC_Val = ((ADC_Val * 10) /33);
ADC_Convert(disptemp + 4);




 /* Convert ADC value into ASCII for LCD Display */

LCD_Cmd(0xC5);                                        /* Display ADC Value on LCD
*/
LCD_Data(ThrdBit);
LCD_Cmd(0xC6);
LCD_Data(SecondBit);
LCD_Cmd(0xC7);
LCD_Data(FirstBit) ;
LCD_Cmd(0xC8);
LCD_Data(0xDF) ;
LCD_Cmd(0xC9);
LCD_Data('C') ;

Delay(10000);                                                        /* Delay */
}
}
/*-----------------------------------------------------------------------------------------------------------------
* end of file
*-----------------------------------------------------------------------------------------------------------------
-*/
```

Temperature Sensor Module MC-15 with NV500X series

PROCEDURE:

1. Connect 20 Pin FRC cable between Sensor Module (MC15) & Port of NV500Xseries Trainer as mentioned in respective programs.
2. Connect LCD Module compatible with NV500X series Trainer LCD Data and Control Pins to Ports of NV500X series trainers as mentioned in respective programs (Optional).
3. Make the connection of Sensor Module with NV500X series as given in the respective program.
4. Connect +12V DC supply from respective NV500X series trainer.
5. Observer the voltage that is being displayed on the LCD and note down the readings.

RESULT: Temperature varying digital readings observed.

Question & Answer:

1. The term 'page traffic' describes
   Ans.**the movement of pages in and out of memory.**
2. An imperative statement indicates
   Ans.**an action to be performed during execution of assembled program.**
3. A linker program links
   Ans.**the program with other programs needed for its execution.**
4. A critical section is a program segment
   \ Ans.**where shared resources are accessed.**
5. The syntax of the assembler directive EQU is
   Ans.**<symbol>EQU<address space>.**
6. The translator which perform macro expansion is called a
   Ans. **Macro pre-processor**.

7. Interval between the time of submission and completion of the job is called
   Ans.**Turnaround time.**
8. The "turn-around" time of a user job is
   Ans. **the total time taken to execute the job.**
9. Memory utilization factor shall be computed as follows
   Ans. **memory in use/total memory connected.**
10. Program generation activity aims
    Ans. **at automatic generation of program.**

PIC MICROCONTROLLER

EXPERIMENT NO.12

AIM: Study and analysis of interfacing of graphical LCD using PIC Controller.

APPARATUS: NV5002 Trainer, Graphical Display module, Two 20 Pin FRC Cable

THEORY: This module provides an interface to JHD12864J Graphical LCD module of size 128x64 dots Display controller chip is KS0107/8.

LCD pinout function summarized as below
* pin 1 Vss - 0V
* pin 2 Vdd - 5.0V
* pin 3 Vo - LCD Drive Voltage (adjust contrast, trimmer to Vee in our case)
* pin 4 D/I- GLCD_RS (H:Data, L:Command (data or command control))
* pin 5 R/W - GLCD_RW (H:Read, L:Write (read status or write data/command))
* pin 6 E-GLCD_EN (H/H->L (Strobe signal, H->L effective))
* pin 7:14 DB0:DB7- LCD_DATA
* pin 15 CS1- GLCD_CS1 (left, high effective)
* pin 16 CS2- GLCD_CS2 (right, high effective)
* pin 17 RST- GLCD_RST (reset pin, low effective)
* pin 18 Vee- -10.0V, LCD driving voltage. There is a Vee generator built-in LCD module, thus
* there is no need to use external generator IC
* pin 19 EL1 / A - LCD_BL (Depends on the backlight option.If it's a LED backlight, it's an anode.
* If it is EL type, this is EL controller input)
* pin 20 EL2 / K - GND (Cathode for LED-backlight. EL input (OR NC) for EL-backlight model)

PROGRAM:

```
LIST  P=16F877

INCLUDE "P16F877.INC"
ORG     0000H

REG2:   EQU     20H
DLY2:   EQU     21H
DLY3:   EQU     22H
PTR:    EQU     23H
TEMP1:  EQU     24H
TEMP2:  EQU     25H
```

```
ORG     0000H
GOTO    START

ORG     0004H
RETFIE



ORG     0005H
START:  BSF     STATUS,RP0
        BCF     STATUS,RP1
        MOVLW   00H
        MOVWF   TRISC               ;PORTC,PORTD AS OUTPUT
        MOVWF TRISD
        BCF     STATUS,RP0
        CLRF    PIR1
        CLRF    CCP1CON

  ; INITIALIZATION OF LCD

INIT:   BCF     PORTD,0
        BCF     PORTD,1
        BCF     PORTD,2             ;MAKE RS,R/W,EN LOW
        MOVLW   30H                 ;OUTPUT 30H ON DATA BUS
        MOVWF   PORTC
        CALL    WRITE1              ;CALL WRITE CYCLE
        CALL    DELAY3              ;CALL DELAY ROUTINE FOR LCD
        MOVLW   30H
        MOVWF   PORTC               ;OUTPUT 30H ON DATA BUS
        CALL    WRITE1              ;CALL WRITE CYCLE
        CALL    DELAY1
        MOVLW   38H
        MOVWF   PORTC               ;OUTPUT 38H ON DATA BUS
        CALL    WRITE1
        CALL    DELAY1
        MOVLW   08H                 ;OUTPUT 08H ON DATA BUS
        MOVWF   PORTC
        CALL    WRITE1
        CALL    DELAY1
        MOVLW   01                  ;OUTPUT 01H ON DATA BUS
        MOVWF   PORTC
        CALL    WRITE1
        CALL    DELAY1
        MOVLW   06H                 ;OUTPUT 06H ON DATA BUS
        MOVWF   PORTC
```

```
           CALL    WRITE1
           CALL    DELAY1
           MOVLW   0CH                  ;OUTPUT 0CH ON DATA BUS
           MOVWF   PORTC
           CALL    WRITE1
           CALL    DELAY1
           MOVLW   080H                 ;OUTPUT 80H ON DATA BUS
           MOVWF   PORTC
           CALL    WRITE1
           BCF PORTD,0
           CALL    DISPLAY1             ;DISPLAY



        CALL DELAY1
        MOVLW   0C0H          ;OUTPUT 0C0H ON DATA BUS
        MOVWF   PORTC
        CALL    WRITE1
        MOVLW   10
        MOVWF   TEMP2
        CALL    LOP1                ;DISPLAY
SLF1:   GOTO    SLF1

WRITE1: BCF     PORTD,0          ;
        BCF     PORTD,1         ; RS,R/W,EN low
        BCF     PORTD,2         ;
        NOP
        NOP
        NOP
        BSF     PORTD,2         ; EN high
        CALL    DELAY1
        BCF     PORTD,2         ; EN low
        NOP
        NOP
        NOP
        NOP
        NOP
        BSF PORTD,0                ;RW HIGH
        NOP
        NOP
        NOP
        NOP
        BCF     PORTD,0         ; R/W low
        RETURN
```

```
DELAY1: MOVLW   0FFH
MOVWF   DLY2
LOOP3:  NOP
           NOP
           NOP
           NOP
           DECFSZ  DLY2,1
           GOTO    LOOP3
           RETURN
DELAY3: MOVLW   10H
            MOVWF   DLY3
LOOP2:  MOVLW   0FFH
           MOVWF   DLY2
LOOP1:  DECFSZ  DLY2,1
           GOTO    LOOP1





DECFSZ  DLY3,1
GOTO    LOOP2
RETURN

DISPLAY2:MOVF   REG2,0
        MOVWF  PORTC
        BSF    PORTD,1         ; RS high
        BSF    PORTD,2         ; EN high
        CALL   DELAY1
        BCF    PORTD,2         ; EN low
        BSF    PORTD,0          ; RW HIGH
        BCF    PORTD,1
        BCF    PORTD,0         ;R/W LOW
        RETURN

DISPLAY1: MOVLW   10
        MOVWF   TEMP2
        MOVLW   0
        MOVWF   TEMP1
LOP1:   CALL    CODE2
        MOVWF   REG2
        CALL    DISPLAY2
        INCF    TEMP1,1
        DECFSZ  TEMP2,1
```

```
         GOTO    LOP1
         BCF    PORTD,0          ; R/W LOW
         RETURN

CODE2:  MOVF    TEMP1,0
        ADDWF   PCL,1
        RETLW   57H    ;'W'
        RETLW   45H    ;'E'
        RETLW   4CH    ;'L'
        RETLW   43H    ;'C '
        RETLW   4FH    ;'O'
        RETLW   4DH    ; M
        RETLW   45H    ;'E'
        RETLW   20H    ;
        RETLW   54H    ;'T'
        RETLW   4FH    ;'O'
        RETLW   20H    ;
        RETLW   4BH    ;'K'
        RETLW   4EH    ;'N'
        RETLW   4FH    ;'O'
        RETLW   57H    ;'W'
        RETLW   4CH    ;'L'




CC2:   RETLW   45H    ;'E'
          RETLW   4DH    ; M
          RETLW   42H    ;'B'
          RETLW   45H    ;'E'
          RETLW   44H    ;'D'
          RETLW   44H    ; D
          RETLW   45H    ; E
          RETLW   44H    ; D
          RETLW   20H
        RETLW   54H    ;'T'
        RETLW   52H    ;'R'
        RETLW   41H    ;'A'
        RETLW   49H    ;'I'
        RETLW   4EH    ;'N'
        RETLW   45H    ; E
        RETLW   52H    ;'R'
        RETLW   20H    ;
        END
```

PROCEDURE:

1. Insert PIC16F877 Microcontroller in Programmer unit (in NV5002).
2. Connect serial cable between computer serial port and programmer unit serial port female connector (in NV5002).
3. Switch 'On' the programmer switch in programmer unit (in NV5001) and switch on the power supply.
4. Program LCD interface Graphical_LCD.hex file (Via CD - NV5002/ Modules programs\MC-09 Display module\ Graphical Display module).
5. Switch 'Off' the power supply and remove the programmed controller from programmer ZIF socket
6. Switch 'Off' the programmer switch in Programmer unit (in NV5002).
7. Insert programmed Microcontroller to microcontroller unit ZIF socket.
8. Connect 20 Pin FRC cable to graphical display module Interface block left side socket/connector (MC09) to Port P1 in NV5002 Trainer.
9. Connect 20 Pin FRC cable to graphical display module Interface block right side Socket/connector (MC09) to Port P3 in NV5002 Trainer.
10. Turn contrast control potentiometer in graphical display module interface block to clockwise position (in MC09).
11. Turn Backlight control potentiometer in graphical display module interface block to anticlockwise position (in MC09).
12. Switch 'On' the power supply.
13. Observe "------ABBBBBBB---------------------------" is coming on graphical display module and after some delay below mentioned picture will displayMC09 .
14. Observe the output waveform between R/W pin of graphical LCD (P1.1) and ground, on oscilloscope.
15. Turn contrast control potentiometer and observe the contrast change on LCD.
16. Turn Backlight control potentiometer and observe the change on backlight of LCD.MC09

RESULT: Characters displayed on the LCD screen.

Question & Answer(Instruction Set)

1. SWAP A performs Alsn ←→Amsn; 1 byte instruction with 1 cycle.
2. LCALL addr 16 performs PC+3 → [SP]; addr 16 → PC 3 bytes instruction with 2 cycle.
3. RET performs (SP) → PC; 1 byte instruction with 2 cycle.
4. SJMP rel performs PC+2+rel → PC; 2 byte instruction with 2 cycle.
5. DAA performs Abin → Adec; 1 byte instruction with 1 cycle.
6. ANL C,bit performs C AND bit → C; 2 byte instruction with 1 cycle.
7. NOP performs PC+1→PC ; 1 byte instruction with 1 cycle.
8. ladd long address of 16 bits from 0000h to FFFFh.
9. add address of the internal RAM from 00h to FFh.
10. radd relative address, a signed number from -128d to +127d.

PIC MICROCONTROLLER

EXPERIMENT NO.13

AIM: To interface PWM based voltage regulator using PIC microcontroller.

APPARATUS: NV5002 Trainer, 2mm Patch Cords 8, Learning Material CD, MC-16.

THEORY: **(**MC16) PWM based Voltage Regulator for Microcontroller development board with programmer trainer is an Extension module. The module has been designed to have a clear understanding of how a PWM is Converted into Voltage .PWM based Voltage Regulator module enable students and practicing engineers to gain invaluable practical experience of voltage regulation using Pulse Width Modulation (PWM). The object is to have a clear understanding of how PWM is generated using microcontroller to use in various applications like Servo Motor speed control etc. The object is to connect and program a microcontroller to generate PWM of any duty cycle and convert it into voltage. It has input and output terminals for connection of external real world applications.

PROGRAM:

Experiment (a) Generating PWM from Microcontroller

Programming Steps:
To generate the PWM of varying duty cycle 0 to100% and than from 100 to 0% continuously.

```
#include<pic.h> /* include controller specific header file */
/*****************************************************************
* Description: Configuration Bit Setting.
*****************************************************************/
__CONFIG (WDTDIS & LVPDIS &  PWRTEN &  XT);
/*****************************************************************
*Delay_ms
* Description: This function provides Delay in Milisec.
*****************************************************************/
void Delay_ms(unsigned int rTime)
{
unsigned int x,y;
```

```
for(x=0;x<rTime;x++)
for(y=0;y<500;y++);
}
}

void main()
{
unsigned char  Counter ;

   TRISC = 0 ;         /* Set PORTC as output  */
                       /* Ensure that your PWM pin is configured as digital output*/
   PORTC = 0 ;         /* Clear PORTC */
* PWM registers configuration
* Fosc = 4000000 Hz
* Fpwm = 40000.00 Hz
* Duty Cycle = 100 %
* PR2 = 0b01111100 ; /* TIMER2 Period Register */
T2CON = 0b00000111 ; /* TIMER2 Control Register */
CCP1CON = 0b00111100 ; /* PWM1 Control Register */
CCP2CON = 0b00111100 ; /* PWM2 Control Register */
CCPR1L = 0b01111100 ; /* PWM1 Data Register*/
CCPR2L = 0b01111100 ; /* PWM2 Data Register*/
while(1)
   {
         /*  Forever*/
for(Counter = 0 ; Counter < 128 ; Counter++)
       {
CCPR1L = Counter ;                    /* PWM1 Data Register  */
         CCPR2L = 128 - Counter ;            /* PWM1 Data Register  */
     Delay_ms(10) ;
       }
       for (Counter = 127 ; Counter > 0 ; Counter--)
       {
CCPR1L = Counter ;
```

```
            CCPR2L = 128 - Counter ;
            Delay_ms(10) ;
        }
       }
    }
```

PROCEDURE:

1. Insert PIC 18F77A Microcontroller in Programmer unit (in NV5002).
2. Connect serial cable between computer serial port and programmer unit serial port female connector (in NV5002).
3. Switch '**On**' the programmer switch in programmer unit (in NV5002) and switch '**On**' the power supply.
4. Program **PWM Interface. hex** file (Via CD -NV5002/Modules Programs\MC16).
5. Remove the programmed controller from programmer ZIF socket.
6. Insert programmed microcontroller to "PIC Controller" unit ZIF socket.
7. Connect 20 Pin FRC cable to PWM Based Voltage Regulator socket (MC16) to Port RC in NV5002 Trainer.
8. Press Reset Switch on the NV5002 Trainer.
9. Check the PWM signal on PWM1 and PWM2 using CRO or DSO on MC16.

RESULT: PWM signal observed on the CRO.

EXPERIMENT No 13 (b)

AIM: Converting PWM to DC Voltage.

PROCEDURE:

1. Burn Hex file into PIC16F877A by following Experiment 1 and insert programmed Microcontroller to microcontroller unit ZIF socket.
2. Connect 20 Pin FRC cable to PWM Based Voltage Regulator socket (MC16) to Port RC in NV5002 Trainer.
3. Connect a patch cord from **PWM1** to **PWM Input** socket.
4. Switch 'On' the Power Supply (NV5002).
5. Press Reset Switch from NV5002 Trainer.
6. Now set a multimeter to check "DC Voltage" mode
7. Now check voltage at **Vout** using multimeter with respect to ground.
8. Voltage vary from 0 to 5V and than decrease from 5 to 0V continuously.
9. Follow the same process for PWM2 socket also.

RESULT: Corresponding value of observed on multimeter.

EXPERIMENT No13 (c)

AIM: Amplifying PWM generated voltage to gain of 2 using Amplifier.

PROCEDURE:

1. Burn Hex file into PIC16F877A by following Experiment 1 and insert programmed Microcontroller to microcontroller unit ZIF socket.
2. Connect 20 Pin FRC cable to PWM Based Voltage Regulator socket (MC16) to Port RC in NV5002 Trainer.
3. Connect a patch cord from **PWM1** to **PWM Input** socket.
4. Connect a patch cord from **Vout** to **Input** socket of Amplifier block.
5. Connect a patch cord between +12V DC Supply (on **NV5002**) & +12V socket of Amplifier block (on **MC16**).
6. Switch 'On' the Power Supply (NV5002).
7. Press Reset Switch from NV5002 Trainer.
8. Now take two separate multimeters.
9. Set multimeters to "DC Voltage" Mode.
10. Now connect one multimeter to **Vout** socket with respect to ground.
11. Connect another multimeter to **Output** socket of Amplifier bolck with respect to ground.
12. Now observe voltages on **Vout** and **Output** of Amplifier Bolck.
13. Now user can vary gain of Amplifier by varying Amplifier potentiometer.
14. Set pot of Amplifier to Unity Gain by rotating it fully clockwise direction and observe voltage. Both multimeres will display approximately same voltage level.
15. Now set pot of Amplifier to Gain of 2 by rotating it fully counter clockwise direction and observe voltage. The voltage level of **Output** terminal of Amplifier is approximately double of **Vout** terminal.
16. Follow the same process for PWM2 terminal also.

RESULT: Amplified value observed on Multimeter.

Question & Answer(Istruction SET)

1. bcf f,b performs clear bit b of register f.
2. bsf f,b performs set b of register f.
3. movlw k instruction move literal k to W register.
4. Swap f,F(W) instruction swapnibbles of f; putting result in F or W.
5. Iorlw k instruction performs inclusive OR literal value into W.
6. Iorwf f,F(W) instruction performs inclusive OR W with f and put the result in F or W.
7. clrwdt instruction reset watchdog timer if enabled.
8. sleep instruction stop clock, reduce power and wait for watchdog timer or external signal to begin program execution again.
9. nop instruction does nothing and waits for a cycle.
10. Retfie instruction reenables interrupts.

PIC MICROCONTROLLER

EXPERIMENT NO.14

AIM: Study and interface of IR (RC5 Protocol) and RF Communication using PIC Controller.

APPARATUS: RS-232 Cable, NV500X Trainer (2), MC12 Module (1), MC10 Module (2), Patch Cords (3), 20 Pin FRC Cable (8)

THEORY: Infrared Module MC12 is an Extension module. The object is to have a clear understanding of how infrared transmitter and receiver are interfaced and controlled with microcontroller. The object is to connect and program a microcontroller to know about basic wireless technology infrared. Infrared Module MC12 has input and output terminals for connection of external real world applications

PROGRAM:

```
******************************************************************************
* File Description: IR Module Demo Code
* Steps for Operation:
* Step 1:-
* Swith ON NV5001
* Burn the Program for IR Demo.hex file in 8051/52 MicroController using NV5001.
* Step 2:-
*       Connections : Connect 20 Pin FRC cable in following manner:
*       NV5001 Port P0 wuth "LCD_Data" Connector of MC10.
*       NV5001 Port P1 with "Hex Keypad Interface Block" Connector of MC10.
*       NV5001 Port P2 with "LCD_Control"       Connector of MC10.
*       NV5001 Port P3 with "CN1" of MC12.
* Connection On MC12 Module:
* Connect one patch cord from CN1 - '1' to IR Rx.
* Connect one patch cord from CN1 - '2' to Data In.
* Connect one patch cord from Data Out to IR Tx.
* Step 3:-
* Place MicroController on ZIF Socket and Press Reset Switch.
* Step 4:-
*       OutPut: Initially On MC10 LCD "Sent Data: " in first line
* And "Receive Data: 0" in second line display.
* Now, By Using Keypad you can send any character
* which will display on "Send Data : "and the same will  be display on "Receive data :".
* For example if you will Send '1' from keypad then On LCD
* Send Data     :1
* Recieve Data:1
```

```
******************************************************************************/

#include <regx52.h>                           /* Include controller library */
#include <delay.h>
#include <stdio.h>
#include <uart.h>

/****************************************************************************
* Function Prototypes Definition
*****************************************************************************/
void LCD_Init();
void LCD_Busy();
void LCD_Command(unsigned char);
void LCD_Data(unsigned char);
void LCD_WriteS(unsigned char*);
/******************************************************************************
* Pin allocation for LCD
*******************************************************************************/
#define LCD_Port P0                          /* LCD data pin(D0-D7) to ort P0 */
#define LCD_Flag P0_7                               /* LCD Busy Pin*/
#define LCD_rs P2_5                          /* LCD RS (Register select) pin */
#define LCD_rw P2_6                          /* LCD R/W (Read/ Write) pin*/
#define LCD_en P2_7                          /* LCD Enable pin   */
/******************************************************************************
* Pin allocation for Keypad
******************************************************************************/
#define ROW1  P1_4
#define ROW2  P1_5
#define ROW3  P1_6
#define ROW4  P1_7
#define COL1   P1_0
#define COL2   P1_1
#define COL3   P1_2
#define COL4   P1_3
/******************************************************************************
* Global variable Declaration
******************************************************************************/
static unsigned char code *LCDArry[]={"  Nvis","  Technologies  "};
unsigned char KeyVal,Serial_buf[20];
/*******************************************************************************
* KeyPad Function Definition
*******************************************************************************/
unsigned char Keypad(void)
{
  KeyVal = 100;
  P1 = 0xFF;               // All rows & columns are hi
```

```
while(1)
{
// Row1 Scan
  ROW1 = 0;
  if(!COL1)
  {
        delay_msec(50);
        if(!COL1)
        KeyVal='1';
        return KeyVal;
  }
  if(!COL2)
  {
        delay_msec(50);
        if(!COL2)
        KeyVal='2';
        return KeyVal;
  }
  if(!COL3)
  {
        delay_msec(50);
        if(!COL3)
        KeyVal='3';
        return KeyVal;
  }
  if(!COL4)
  {
        delay_msec(50);
        if(!COL4)
        KeyVal='F';
        return KeyVal;
  }
  ROW1 = 1;
// Row2 Scan
  ROW2 = 0;
  if(!COL1)
  {
        delay_msec(50);
        if(!COL1)
        KeyVal='4';
        return KeyVal;
  }
  if(!COL2)
  {
          delay_msec(50);
          if(!COL2)
          KeyVal='5';
          return KeyVal;
  }

  if(!COL3)
  {
          delay_msec(50);
          if(!COL3)
          KeyVal='6';
          return KeyVal;
  }
  if(!COL4)
  {
          delay_msec(50);
          if(!COL4)
          KeyVal='E';
          return KeyVal;
  }
  ROW2 = 1;
// Row3 Scan
  ROW3 = 0;
  if(!COL1)
  {
          delay_msec(50);
          if(!COL1)
          KeyVal='7';
          return KeyVal;
  }
  if(!COL2)
  {
          delay_msec(50);
          if(!COL2)
          KeyVal='8';
          return KeyVal;
  }
  if(!COL3)
  {
          delay_msec(50);
          if(!COL3)
          KeyVal='9';
          return KeyVal;
  }
  if(!COL4)
  {
          delay_msec(50);
          if(!COL4)
          KeyVal='D';
          return KeyVal;
  }
  ROW3 = 1;
// Row4 Scan
  ROW4 = 0;
  if(!COL1)
  {
          delay_msec(50);
          if(!COL1)
          KeyVal='0';
          return KeyVal;
  }
  if(!COL2)
  {
          delay_msec(50);
          if(!COL2)
          KeyVal='A';
          return KeyVal;
  }
  if(!COL3)
  {
          delay_msec(50);
          if(!COL3)
          KeyVal='B';
          return KeyVal;
  }
  if(!COL4)
  {
          delay_msec(50);
          if(!COL4)
          KeyVal='C';
          return KeyVal;
  }
  ROW4 = 1;
  }
}
/******************************************************************************
* PrintNum Function Definition
* Here, we will display the number on LCD and same number will be send to IR.
******************************************************************************/
void PrintNum(unsigned char printval)
{
  LCD_Command(0x8D);                              /* Set LCD cursor at (1,13) */
  switch(printval)
  {
          case '1':
                  LCD_Data('1');
                  delay_msec(50);
                  putchar('1');
                  break;
case '2':
                  LCD_Data('2');
                  delay_msec(50);
                  putchar('2');
                  break;

case '3':
                  LCD_Data('3');
                  delay_msec(50);
                  putchar('3');
                  break;

case '4':
                  LCD_Data('4');
                  delay_msec(50);
                  putchar('4');
                  break;

case '5':
                  LCD_Data('5');
                  delay_msec(50);
                  putchar('5');
                  break;

case '6':
                  LCD_Data('6');
                  delay_msec(50);
                  putchar('6');
                  break;

case '7':
                  LCD_Data('7');
                  delay_msec(50);
                  putchar('7');
                  break;

case '8':
                  LCD_Data('8');
```

```
                        delay_msec(50);
                        putchar('8');
                        break;

        case '9':
                        LCD_Data('9');
                        delay_msec(50);
                        putchar('9');
                        break;

        case '0':
                        LCD_Data('0');
                        delay_msec(50);
                        putchar('0');
                        break;
        case 'A':
                        LCD_Data('A');
                        delay_msec(50);
                        putchar('A');
                        break;
        case 'B':
                        LCD_Data('B');
                        delay_msec(50);
                        putchar('B');
                        break;
        case 'C':
                        LCD_Data('C');
                        delay_msec(50);
                        putchar('C');
                        break;
        case 'D':
                        LCD_Data('D');
                        delay_msec(50);
                        putchar('D');
                        break;

        case 'E':
                        LCD_Data('E');
                        delay_msec(50);
                        putchar('E');
                        break;
        case 'F':
                        LCD_Data('F');
                        delay_msec(50);
                        putchar('F');
                        break;
```

```
  }
}

/**************************************************************************
* Decode code for pressed Key
**************************************************************************/
void DecodeKey()
{
        switch(Serial_buf[0])
  {
        case '0'-'9':
                Serial_buf[0] = Serial_buf[0]|0x30;
                break;
        case 'A'-'F':
                Serial_buf[0] = Serial_buf[0]|0x40;
                break;
        default         :
                break;
  }
}
/**************************************************************************
* ISR definition for UART
* Here, whatever data we are sending from IR same will be recieved here.
**************************************************************************/
static void ComISR(void) interrupt 4
{
  ES = 0;                                         /* disables intr4*/
  if(RI)                                          /* RI=Rcv intr flag*/
  {
        Serial_buf[0]= SBUF;
        RI = 0;
  }
  ES = 1 ;
}
/**************************************************************************
* Main Function Definition
**************************************************************************/
void main(void)
{
  LCD_Init();                                /* Initialize LCD */
  UART_Init();                               /* Initialize UART */
  P2      = 0x00;                            /* Clear Port 2 */
  LCD_Command(0x01);                         /* Clear Display */
  LCD_Command(0x84);                     /* Set LCD cursor at (1,4)  */
  LCD_WriteS(LCDArry[0]);             /* Send String Data to LCD  */
  LCD_Command(0xC0);                     /* Set LCD cursor at (2,0)  */
  LCD_WriteS(LCDArry[1]);                    /* Send String Data to LCD  */
  delay_sec(1);
  LCD_Command(0x01);                               /* Clear LCD Display            */
  LCD_Command(0x80);                               /* Set LCD cursor at (1,0)      */
  LCD_WriteS("Send Data   :");                  /* Send String Data to LCD      */
  LCD_Command(0xC0);                               /* Set LCD cursor at (2,0)      */
  LCD_WriteS("Recieve Data:");               /* Send String Data to LCD          */


        P2= 0x00;
  Serial_buf[0] = '0';
  KeyVal = 100;
  EA = 1;                                                   /* Enable global Interrupt       */
  while(1)
            {
            KeyVal = Keypad();                            /* Check if any key pressed          */
            PrintNum(KeyVal);                       /* Display and send data if any key pressed
  */
            delay_msec(100);
            LCD_Command(0xCD);                       /* Set LCD cursor at (2,13)    */
            DecodeKey();                          /* Decode the pressed Key     */
            LCD_Data(Serial_buf[0]);                /* Send String Data to LCD         */
  }
}

/********** Initialize LCD Routine *****************/

void LCD_Init()
{
   LCD_Command(0x38);                              /* Function Set 8bit : 2 Line 5x7 Dots
  */
   LCD_Command(0x0C);                              /* Display On curser Off
            */
   LCD_Command(0x01);                              /* Clear Display
                      */
   LCD_Command(0x06);                              /* Entry Mode
                                */
}

/**************** LCD Commands Routine **************/

void LCD_Command(unsigned char Cmd)
{
   LCD_Busy();

  LCD_Port = Cmd;                                   /* Write Command to LCD */
  LCD_rs = 0;                                    /* Clear bit P3.5 */
  LCD_rw = 0;                                    /* Clear bit P3.6 */
  LCD_en = 1;                                    /* Set bit P3.7 */
  LCD_en = 0;
}

/**************** LCD Busy Routine ***************/
void LCD_Busy()
{
  LCD_Flag=1;
  LCD_rs=0;
  LCD_rw=1;
  while(LCD_Flag!=0)

{
            LCD_en=0;
            LCD_en=1;
  }
}
/*************** Write Srting to LCD ***************/
void LCD_WriteS(unsigned char *lcd_string)
{
  while (*lcd_string)
  {
            LCD_Data(*lcd_string++);    /* write data to LCD */
  }
}
/*************** Write character to LCD ***************/
void LCD_Data(unsigned char ASCII)
{
  LCD_Busy();
  LCD_Port=ASCII;                                        /* Write data to LCD in ascii value*/
  LCD_rs=1;
  LCD_rw=0;
  LCD_en=1;
  LCD_en=0;
}
```

PROCEDURE:

1. Connect 20 Pin FRC cable form one NV500X trainer (Transmitter) to CN1 of MC12 and other NV500X trainer (Receiver) to CN2.
2. Connect one patch cord from any one of “1-8 pin” of **CN1** (as mentioned in respective program) to “**Data In**” and another patch cord from any one of “1-8 pin” of **CN2** (as mentioned in respective program) to “**IR Rx**”.
3. Connect both MC10 modules as per mentioned in program.

4. Connect Patch Cord (Red Line) as shown in above figure.
5. On MC10 module which is connected with NV500X transmitter trainer below message will display, you can send data using keypad.

RESULT: Reception and Transmission is checked by the **IR transceiver.**

Question & Answer:

1. PIC is
   Ans.**RISC Processor and use Harvard Architecture.**
2. PIC machine cycle consists of
   Ans. **4 cycles**.
3. Execute instruction in
   Ans.**0.2 µs.**
4. interrupt sources are available in PIC16 series.
   Ans.12
5. Register available in PIC microcontroller are
   Ans.**Status, Indirect, File select and Working register.**
6. Compare Mode can
   Ans. **drive a pin H or L at a precisely controlled time**.
7. Capture Mode permits
   Ans.**PIC to be used to determine the time of occurrence of an i/p edge.**
8. movwf PORTA performs
   Ans. **move the working register into the file register named PORTA**.
9. An imperative statement
   Ans.**indicates an action to be performed during execution of assembled program.**
10. The "blocking factor" of a file is
    Ans.**the number of logical records in one physical record**.