8 I/O Device Interface to 8086 Microprocessor
Stepping Motor Control and FND Display

8.1 Objective

The objectives of this experiment are

- Familiarization with a programmable peripheral interface device, 8255
- Use of 8255 PPI for controlling external devices
- Controlling a stepper motor through peripheral devices connected to 8086 microprocessor
- Controlling a seven segment LED display unit

8.2 Learning Outcome

At the end of the experiment the students will be enable to

- Configure 8255 PPI for controlling its I/O ports
- Learn how to write assembly program related to accessing I/O devices and control of actuators
- Control a stepper motor in various modes using a microprocessor trainer
- Interface seven segment LED display

8.3 Introduction

The stepping motor is a device which can transfer the incoming pulses to stepping motion of a predetermined angular displacement. By using suitable control circuitry, the angular displacement can be made proportional to the number of pulses. Using microcomputer, one can have better control of the angular displacement resolution and angular speed of a stepping motor. Stepping motors are suitable for translating digital inputs into mechanical motion. In general, there are three types of stepping motor:

VR (Variable Reluctance) stepping motors

Hybrid stepping motors

PM (Permanent Magnet) stepping motors

8.4 Theory of operation

Stepper motors operate differently from normal DC motors, which simply spin when voltage is applied to their terminals. Stepper motors, on the other hand, effectively have multiple "toothed" electromagnets arranged around a central metal gear, as shown in Fig. 8.1.

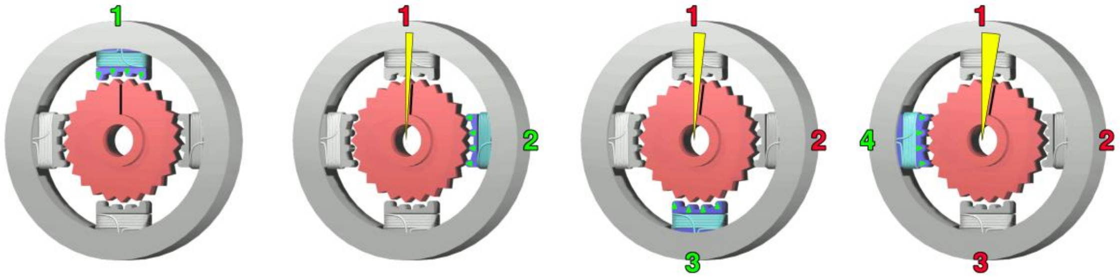

Fig. 8.1 Toothed stepper motor and state of motion at different coil excitation.

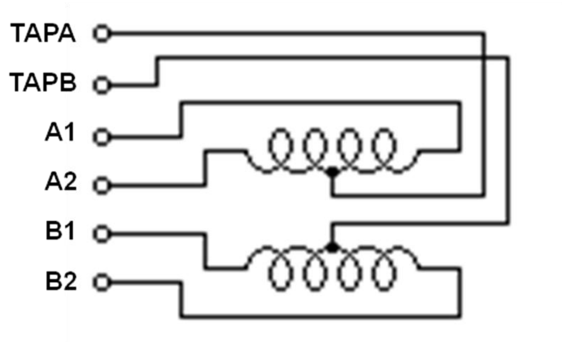

Fig. 8.2 Unipolar 4-Phase (or bipolar 2 phase) stepper motor having six terminals

To make the motor shaft turn, first one electromagnet is given power, which makes the gear's teeth magnetically attracted to the electromagnet's teeth. When the gear's teeth are thus aligned to the first electromagnet, they are slightly offset from the next electromagnet. So, when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one, and from there the process is repeated. Each of those slight rotations is called a "step." In that way, the motor can be turned a precise angle. There are two basic arrangements for the electromagnetic coils: bipolar and unipolar. We will experiment on a unipolar 4-phase six wire stepper motor.

The step angle for each step depends on the number of teethes on the rotor and pole faces. Stepper motors are mostly Hybrid type. In this experiment we will use a hybrid stepper motor with full step angle of 1.8^0 and half step angle of 0.9^0.

Table 8.1 Comparison stepping motor characteristics

Characteristics	Motor type		
	PM	VR	Hybrid
Efficiency	High	Low	High
Rotor Inertia	High	Low	Low
Speed	High	High	Low
Torque	Fair	Low	High
Power O/P	High	Low	Low
Damping	Good	Poor	Poor
Typical Step Angle	1.8° 15° 30°	7.5° 15° 30°	0.18° 0.45°

Commercial stepping motor uses multimotor rotor, the rotor features two bearlike PM cylinders that are turned one-half of tooth spacing. One gear is south pole, the other gear is north pole. If a 50-tooth rotor gear is used, the following movement sequences will proceed.

8.4.1 Single-phase excitation

The stepping position will be 0^0,1.8^0, 3.6^0, ... 358.2^0, total 200 steps in one round.

Table 8.2 Full step truth table (One coil excitation)

Full Step Motion → Single coil excitation							
STEP	B2 (Coil 4)	A2 (Coil 3)	B1 (Coil 2)	A1 (Coil 1)	Byte	Forward	Reverse
1X	0	0	0	1	1	↓	↑
2X	0	0	1	0	2		
3X	0	1	0	0	4		
4X	1	0	0	0	8		

8.4.2 Two-phase excitation

The stepping positions will be 0.9^0, 2.7^0, 4.5^0, ... 359.1^0, total 200 steps in one round.

Table 8.3 Full step truth table (two coil excitation)

Full Step Motion → Two coil excitation							
STEP	B2 (Coil 4)	A2 (Coil 3)	B1 (Coil 2)	A1 (Coil 1)	Byte	Forward	Reverse
1Y	0	0	1	1	3	↓	↑
2Y	0	1	1	0	6		
3Y	1	1	0	0	12		
4Y	1	0	0	1	9		

8.4.3 Single-phase and two-phase excitations combined

The stepping positions will be 0^0, 0.9^0, 1.8^0, 2.7^0, 3.6^0, 4.5^0, ...358.2^0, 359.1^0, total 400 steps in one round. Since stepping motor makes step-by-step movement and each step is equidistant, the rotor and stator magnetic field must be synchronous. During start-up and stopping, the two fields may not be synchronous, so it is suggested to slowly accelerate and decelerate the stepping motor during the start-up or stopping period.

Table 8.4 Truth table for operating a stepper motor in wave motion (Half step operation)

Wave (half Step) Motion → One coil excitation followed by Two coil excitation							
STEP	B2 (Coil 4)	A2 (Coil 3)	B1 (Coil 2)	A1 (Coil 1)	Byte	Forward	Reverse
1X	0	0	0	1	1	↓	↑
1Y	0	0	1	1	3		
2X	0	0	1	0	2		
2Y	0	1	1	0	6		
3X	0	1	0	0	4		
3Y	1	1	0	0	12		
4X	1	0	0	0	8		
4Y	1	0	0	1	9		

8.4.4 Hardware Interface

To run a stepper motor from a microprocessor trainer (microcomputer), we need a parallel port interface. Here we will be using Intel 8255 PPI (programmable peripheral interface) that has three 8-bit ports configurable in few modes. The 8255 has one control port where one can send the control word for configuring the 8255 ports (For details see 8255 data sheet). In this application, one port of 8255 should be configured as output port for stepper motor control signals to pass through. In MDA8086 trainer, the stepper motor interface is built in the motherboard, as shown in Figs. 8-1, 8-2. Upper 4-bits of Port B of the odd addressed 8255 are connected to the stepper motor circuitry. The signal mappings of the port lines are as follows:

Table 8.5 Stepper motor signals and power interface connector details.

Port Bit	Phase	Terminal Connector P10
PB4	Coil A1	1
PB5	Coil B1	4
PB6	Coil A2	3
PB7	Coil B2	6
-	TAPA	2
-	TAPB	5

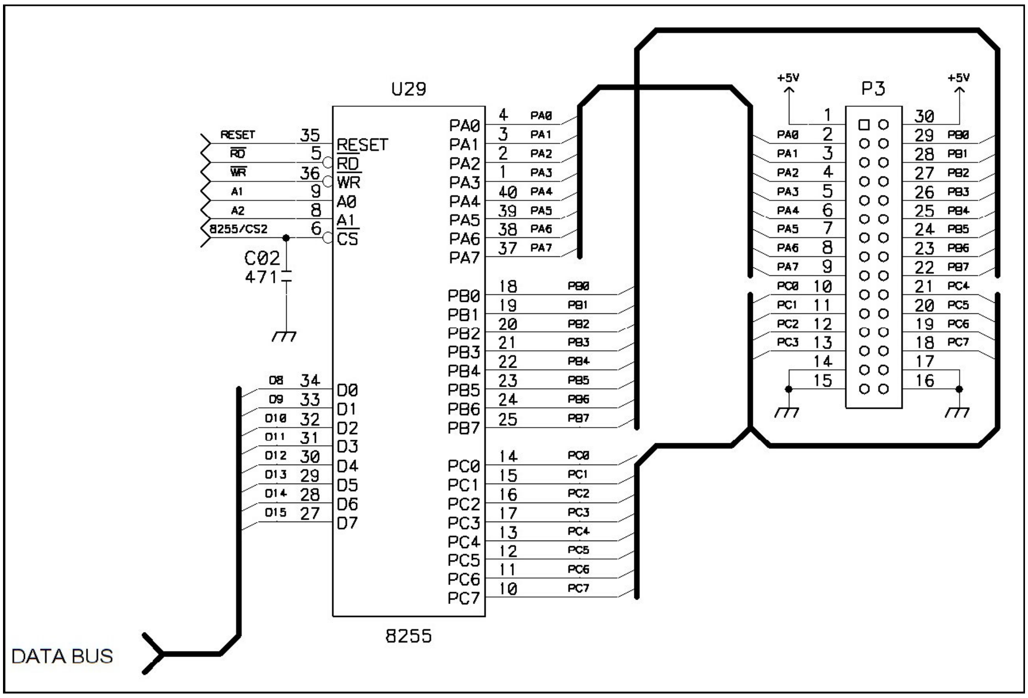

Fig. 8.3 Stepping motor interface through odd addressed 8255 in MDA8086.

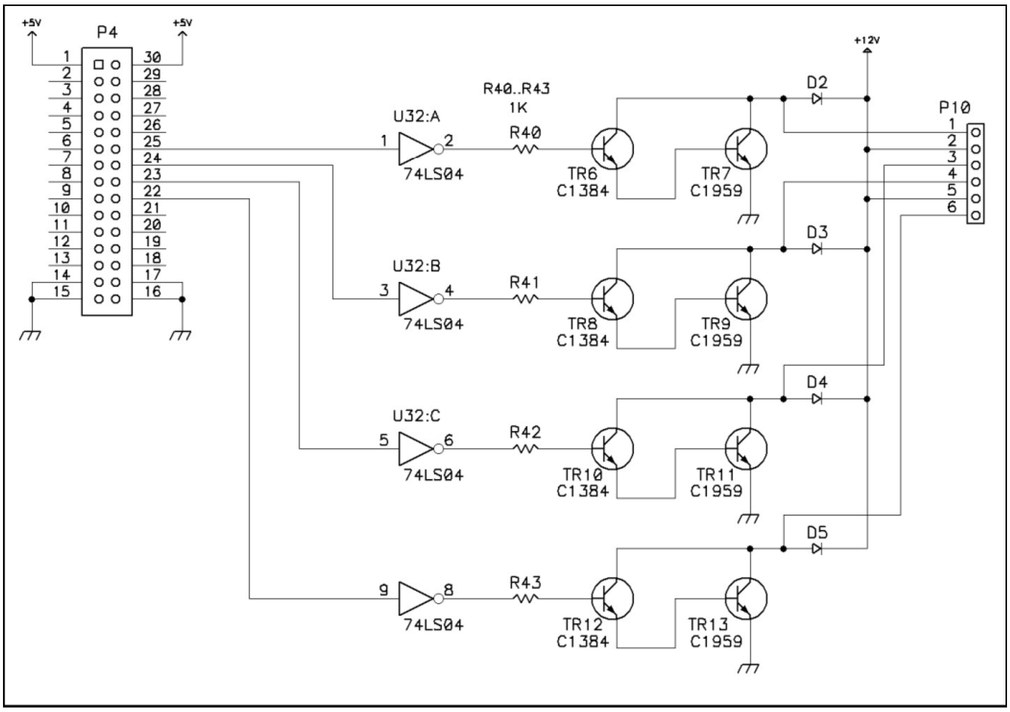

Fig. 8.4 Stepping motor power interface in MDA8086.

8.5 Experimentation with a Stepper Motor

A stepper motor can be operated in Full-step or Half-step mode in Forward/Reverse direction as shown in Tables 8.2, 8.3 and 8.4. In any case the next move depends on the present state (position) of the motor. The basic connection scheme of a stepper motor to a drive circuit is shown in Fig. 8.5. In MDA8086 trainer the drive circuitry is built in the motherboard along with the anti-parallel diodes. Only the motor coils are to be connected to the trainer terminal.

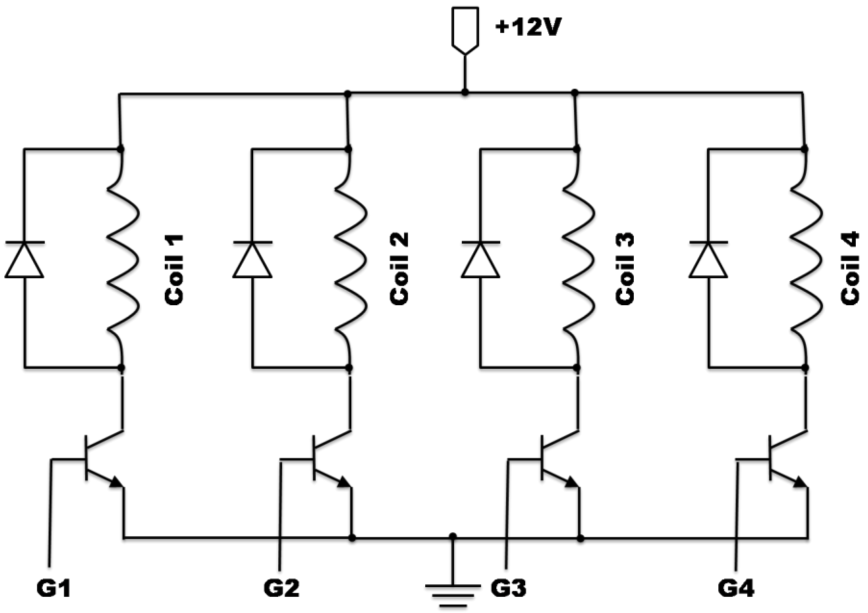

Fig. 8.5 Basic stepper motor drive circuit

8.5.1 Equipment List

- 4-Phase, 6-wire unipolar stepper motor, 12V 1 no.

- MDA8086 Trainer 1 no.
- Personal Computer with a COM port 1 no.

8.5.2 Experimental Procedures

8.5.2.1 Test RUN #1

Step 1. Plug in the stepper motor into the socket of MDA8086 trainer

Step 2. Run WINDOWS editor "EDIT.COM"

Step 3. Type the assembly program given in section 8.5.3

Step 4. Save it using a file name STEPMO.ASM

Step 5. Close the text editor.

Step 6. Go to the command prompt C:\ (*start →All Programs → Accessories → Command Prompt*)

Step 7. Assemble the STEPMO.ASM file (*C:\UP_LAB\MASM STEPMO STEPMO.OBJ STEPMO.LST NULL.CRF←*)

Step 8. Run the loader program to make the HEX (ABS) file (*C:\UP_LAB\LOD186 STEPMO.OBJ STEPMO.ABS NULL.MAP←*)

Step 9. To save time steps 8 and 9 can be avoided by running the A.BAT file (*C:\ A STEPMO←*). This will make the STEPMO.ABS file like this

```
:14100000B800008ED8B080E61FB0FFE619B000E61DB0EEE6A4
:121014001BE80400D0C0EBF7B9000090909090E2FAC3B9
:00000001FF
```

Step 10. Run "WinComm" from WINDOWS

Step 11. Push the RESET button of MDA8086 Trainer

Step 12. Type L in the "WinComm" command window and press ENTER button of PC keyboard

Step 13. Press "PgUp" button on the PC keyboard

Step 14. Browse and select the STEPMO.ABS file from the C:\UP_LAB folder and press ENTER

Step 15. The STEPMO.ABS will be loaded into the MDA8086 Trainer kit at location 0000:1000H

Step 16. Type G in the "WinComm" window command prompt and press ENTER button

Step 17. The stepper will start running

Step 18. Note down the stepping angle, step time and step directions.

8.5.2.2 Test RUN #2

Step 1. Open the text editor.

Step 2. Change the delay time in PROG1, set the counter CX to 1000H, (MOV CX, 1000H) and save the assembly program.

Step 3. Following the procedures given in section 8.5.2.1, run the stepper motor.

Step 4. Adjust the CX value so that the stepper motor takes 10 seconds to have a complete 360^0 rotation.

Step 5. Replace the instruction "ROL AL, 1" with "ROR AL, 1" and RUN the program.

Step 6. Note down the motion direction.

8.5.2.3 Test RUN #3

Step 1. Open the text editor.

Step 2. Write the assembly program PROG2 given in section 8.5.3.2 and save it with a filename STEPMO1.ASM.

Step 3. Make STEPMO1.ABS file using the MASM and LOD186 utilities.

Step 4. Download the STEPMO1.ABS file into the MDA8086 trainer and run the program.

Step 5. Note down the step sizes, direction and time taken to complete 360^0 rotation.

Step 6. Adjust the count value in CX of PROG2 so that the motor takes 10 seconds for a complete 360^0 rotation.

Step 7. Modify the assembly program to change the direction of rotation (Change the following lines

MOV BL, 11001100B → MOV BL, 01100110B

ROL AL,1 → ROR AL,1

Step 8. Note down the time taken by the motor to complete 360^0 rotation.

8.5.3 Assembly Program

8.5.3.1 Running a stepper motor in Full-Steps

```
;*****************************************************************
;*****************************************************************
        ; PROG1
        ; Odd addressed 8255 used
        ; Port B (upper 4-lines, PB4-PB7) is connected to the Stepping Motor Interface
        ; PB4 → Coil 1 (A)
        ; PB5 → Coil 1 (B)
        ; PB6 → Coil 2 (A)
        ; PB7 → Coil 2 (B)
        ; To energize a coil the corresponding bit on PB should be active LOW
;*****************************************************************
CODE    SEGMENT
        ASSUME      CS:CODE,DS:CODE,ES:CODE,SS:CODE
PORT_CON        EQU     1FH             ; Control Port 8-bit Address
PORTC           EQU     1DH             ; Port C 8-bit Address
PORTB           EQU     1BH             ; Port B 8-bit Address
PORTA           EQU     19H             ; Port A 8-bit Address
        ;
        ORG     1000H                   ; Program Effective Address, IP = 1000H
        ;
        MOV     AX, 0
        MOV     DS, AX                  ; Initialize Data Segment register DS to 0000H
        ;
        MOV     AL, 10000000B           ; Configure all ports of 8255 as output
        OUT     PORT_CON, AL
        ;
        MOV     AL, 11111111B           ; Can write 0FFH as well
        OUT     PORTA, AL               ; All pins of Port A to HIGH
        MOV     AL, 00000000B
        OUT     PORTC, AL               ; All pins of Port C to LOW
```

```
        ;
        MOV     AL, 11101110B       ; Only one coil to be energized at a time
L1:     OUT     PORTB, AL
        CALL    DELAY               ; Call DELAY subroutine
        ROL     AL, 1               ; Rotate AL left by 1 bit
        JMP     L1
;
; Subroutine of DELAY
DELAY:          MOV     CX, 0       ; Similar to loading CX by FFFFH
AGAIN:  NOP
        NOP                         ; Dummy instructions to cause time delay
        NOP
        NOP
        LOOP    AGAIN
        RET                         ; Return from subroutine call
        ;
CODE    ENDS                        ; End of Subroutine DELAY
        END                         ; End of Assembly Program
;*****************************************************************************************
```

8.5.3.2 Running a stepper motor in Half-Steps

```
;*****************************************************
;*****************************************************
        ; PROG2
        ; Odd addressed 8255 used
        ; Port B (upper 4-lines, PB4-PB7) is connected to the Stepping Motor Interface
        ; PB4 → Coil 1 (A)
        ; PB5 → Coil 1 (B)
        ; PB6 → Coil 2 (A)
        ; PB7 → Coil 2 (B)
        ; To energize a coil the corresponding bit on PB should be active LOW
;*****************************************************
CODE    SEGMENT
        ASSUME          CS:CODE,DS:CODE,ES:CODE,SS:CODE
PORT_CON        EQU     1FH         ; Control Port 8-bit Address
PORTC           EQU     1DH         ; Port C 8-bit Address
PORTB           EQU     1BH         ; Port B 8-bit Address
PORTA           EQU     19H         ; Port A 8-bit Address
        ;
        ORG     1000H               ; Program Effective Address, IP = 1000H
        ;
        MOV     AX, 0
        MOV     DS, AX              ; Initialize Data Segment register DS to 0000H
        ;
        MOV     AL, 10000000B       ; Configure all ports of 8255 as output
        OUT     PORT_CON, AL
        ;
        MOV     AL, 11111111B       ; Can write 0FFH as well
        OUT     PORTA, AL           ; All pins of Port A to HIGH
        MOV     AL, 00000000B
        OUT     PORTC, AL           ; All pins of Port C to LOW
        ;
        MOV     AL, 11101110B       ; One phase energized at a time
        MOV     BL, 11001100B       ; Two phases energized at a time
        ;
L1:     OUT     PORTB, AL           ; Send signal to port
        CALL    DELAY               ; Call DELAY subroutine
        ROL     AL, 1               ; Rotate AL left by 1 bit
        XCHG    AL, BL              ; Exchange AL and BL contents
```

```
        JMP     L1
;
; Subroutine of DELAY
DELAY:          MOV     CX, 2000H       ; Similar to loading CX by FFFFH
AGAIN:  NOP
        NOP                             ; Dummy instructions to cause time delay
        NOP
        NOP
        LOOP    AGAIN
        RET                             ; Return from subroutine call
        ;
CODE    ENDS                            ; End of Subroutine DELAY
        END                             ; End of Assembly Program
;*********************************************************************************************
```

8.6 Experimentation with Seven Segment Display (FND)

It is often required to interface seven segment displays with a microprocessor system. In MDA8086 trainer there is a single seven segment LED display interface called FND. The FND display is driven from Port A of odd addressed 8255. The segment assignments to the FND display are PA0-PA6 → a-g, and PA7 → dot point p.

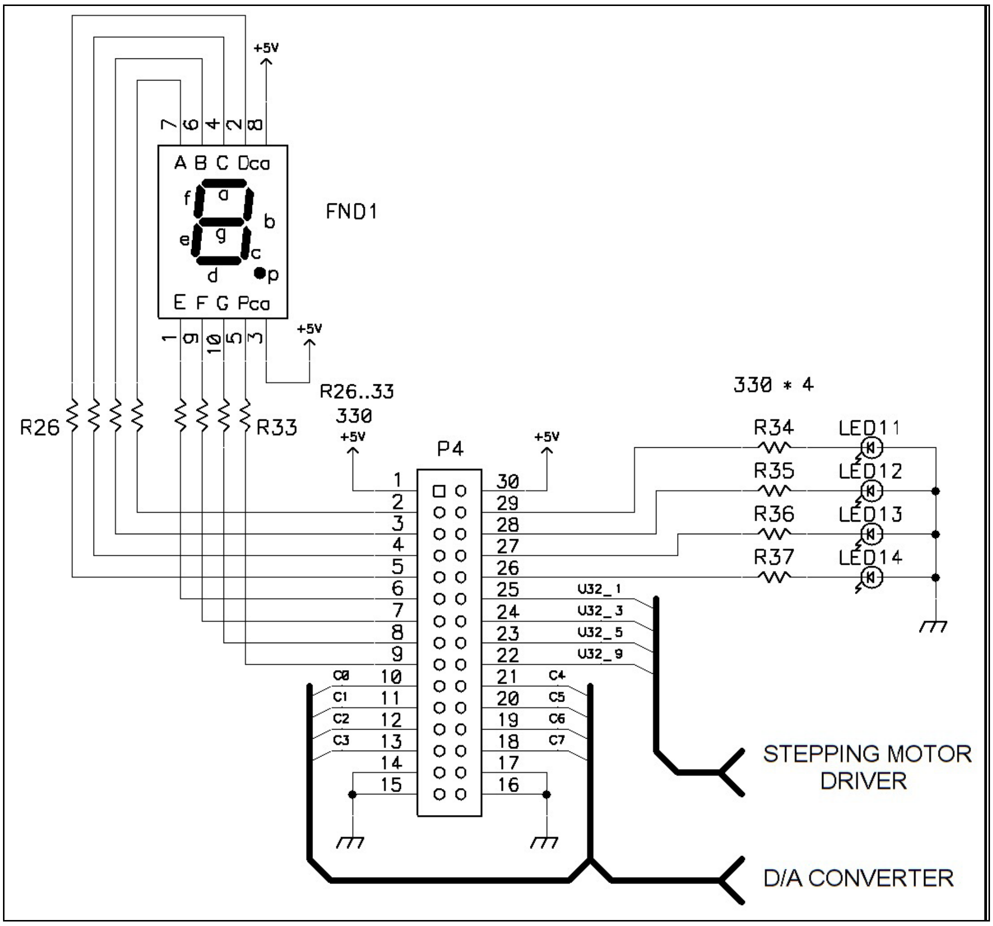

Fig. 8.6 FND interface to MDA8086 trainer.

8.6.1 Experiment Procedure

Step 1. Open the text editor and write the assembly program PROG3 given in section 8.6.2.

Step 2. Make the ABS file, download it to MDA8086 trainer and run it.

Step 3. Note down the observations in the FND display.

Step 4. Change the delay time appropriately so that the display changes ever 1 second.

8.6.2 Assembly Program for FND

```
;*********************************************************
;*********************************************************
        ; PROG3
        ; Odd addressed 8255 used
        ; Port A lines, PA0-PA7 are connected to the FND display
        ; To energize an FND segment the corresponding bit on PA should be active LOW
;*********************************************************
CODE    SEGMENT
        ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
PORT_CON        EQU     1FH             ; Control Port 8-bit Address
PORTC           EQU     1DH             ; Port C 8-bit Address
PORTB           EQU     1BH             ; Port B 8-bit Address
PORTA           EQU     19H             ; Port A 8-bit Address
        ;
        ORG     1000H                   ; Program Effective Address, IP = 1000H
        ;
        MOV     AX, CS
        MOV     DS, AX                  ; CS = DS
        ;
        MOV     AL, 10000000B           ; Configure all ports of 8255 as output
        OUT     PORT_CON, AL
        ;
L1:     MOV     BL, 16                  ; Setup number
        MOV     SI, OFFSET FONT         ; Setup address of font
L2:     MOV     AL, [SI]                ; Transfer font data
        OUT     PORTA, AL               ; Output data
        MOV     CX, 0B000H              ; Delay
        LOOP    $
        INC     SI                      ; Font address + 1
        DEC     BL                      ; Next digit
        JNZ     L2
        JMP     L1

        ;       Dgfedcba        ; Segment Display
FONT    DB      11000000B       ; 0
        DB      11111001B       ; 1
        DB      10100100B       ; 2
        DB      10110000B       ; 3
        DB      10011001B       ; 4
        DB      10010010B       ; 5
        DB      10000010B       ; 6
        DB      11011000B       ; 7
        DB      10000000B       ; 8
        DB      10010000B       ; 9
        DB      100010000B      ; A
        DB      10000011B       ; B
        DB      11000110B       ; C
        DB      10100001B       ; D
        DB      10000110B       ; E
        DB      11000000B       ; F
        ;
CODE    ENDS
        END                             ; End of Assembly Program
```

;**

8.7 Report

1. In the report, put all results obtained during the experiment in a section titled "EXPERIMENTAL RESULTS".
2. In the report, put a section titled "PROGRAMMING HOME TASK". Under the PROGRAMMING HOME TASK section, write an assembly program that will run a stepper motor from MTS86C trainer with the following controls:

   If Button 1 is pushed, Motor will run forward in Full-Steps (One phase operation)

   If Button 2 is pushed, Motor will run reverse in Full-Steps (One phase operation)

   If Button 3 is pushed, Motor will run forward in Full-Steps (Two phase operation)

   If Button 4 is pushed, Motor will run reverse in Full-Steps (Two phase operation)

   If Button 5 is pushed, Motor will run forward in Half-Steps

   If Button 6 is pushed, Motor will run reverse in Half-Steps

   If Button 7 is pushed, Motor will stop running
3. Write a conclusion on this experiment