** Daffodil International University (DIU)**

**Department of Electrical and Electronic Engineering**

**EEE 324: Microprocessors and Interfacing Laboratory**

**Experiment No: 12**

**Name of the Experiment:   
WRITE A PROGRAM TO DISPLAY THE DIGITS IN DECIMAL, FROM 0-7 INTO 7 SEGMENT DISPLAY**

**Objective:**

1. Program the 8255 to show digits on a 7-segment LED.
2. Program both the 8255 and 8279 in order to accept keypad inputs from 0 to F and display it on a 7-segment LED.

# Part 1: Displaying a 7-Semgent LED from keypad inputs

The 7-segment LED display can be found in many displays such as microwave or fancy toaster ovens and occasionally in non-cooking devices. It is composed of 7 LEDs that are fabricated in one case to make a convenient device for displaying numbers and some letters. Figure L10.1 shows a typical 7-segment LED display with its pin layout.

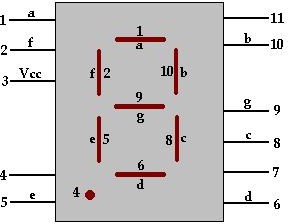
The 7-segment LED shown in Figure L10.1 is a common anode LED display. This means that the positive leg of each LED is connected to a common point which is pin 3 in this case. Each LED has a negative leg that is connected to one of the pins of the device. To make it work, you need to connect pin 3 to +5 volts. Then to make each segment light up, connect the ground pin to ground. A resistor is required to limit the current. Rather than using a resistor from each LED to ground, you can just use one resistor from Vcc to pin 3 to limit the current.

Figure L10.1 A 7-segment LED display and its pin layout.

Table L10.1 provides the information that is needed to display digits from 0 to 9 and the letters A, b, C, d, E, and F. '0' means that pin is connected to ground and '1' means that pin is connected to Vcc.

Table L10.1 Logic states of 7-segment LED to display different digits or letters.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | a (Pin 1) | b (Pin 10) | c (Pin 8) | d (Pin 6) | e (Pin 5) | f (Pin 2) | g (Pin 9) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 4 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| A | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| b | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| d | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| E | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| F | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

Figure L10.2 shows the 7-segment LED decoder circuit. With this information we can design assembly instructions to display different digits or letters on the 7-segment LED. For example, the instruction to display ‘1’ on the 7-segment LED is:

OUT 3FF0, F9

To avoid showing the consecutive digits from 0 to F too quickly, we need to insert time delay before outputting the next digit on the 7-segment LED. This can be accomplished by writing a time delay loop as follows:

MOV CX, 0A000H LOOP $

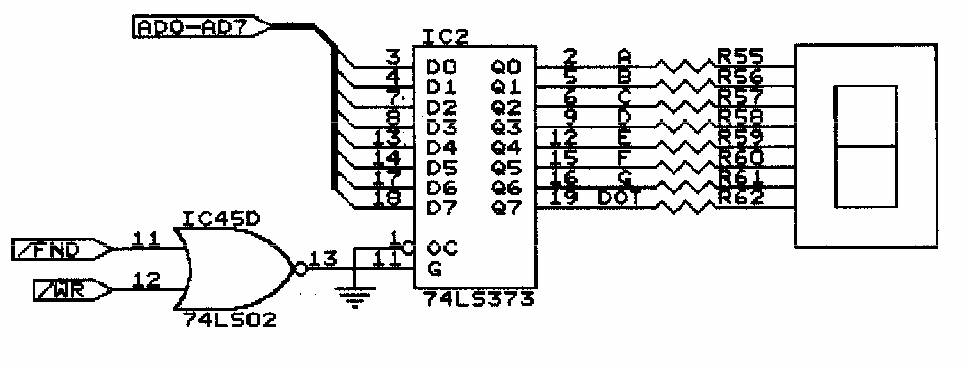


Figure L10.2 The 7-segment LED decoder circuit

Here we will begin our practice of displaying digits or letters on a 7-segment LED with a simple program. We will analyze, run, and modify the program. *Check off each step as it is completed.*

Part 2: **Code** *DATA SEGMENT*

PORTA EQU 120H

PORTB EQU 121H

PORTC EQU 122H

CWRD EQU 123H

TABLE DB 8CH, 0C7H, 86H, 89H

DATA ENDS

*CODE SEGMENT*

ASSUME CS:CODE, DS: DATA

   START: **MOV** AX, DATA ;intialise data segment

**MOV** DS, AX

**MOV** AL, 80H ;initialise 8255 portb and portc as o/p

**MOV** DX, CWRD

**OUT** DX, AL

**MOV** BH, 04 ; BH = no of digitsto be displayed

**LEA** SI, TABLE ; SI = starting address of lookup table

NEXTDIGIT: **MOV** CL,08 ; CL = no of segments = 08

**MOV** AL, [SI]

NEXTBIT: **ROL** AL, 01

**MOV** CH, AL ;save al

**MOV** DX, PORTB ;one bit is sent out on portb

**OUT** DX, AL

**MOV** AL, 01

**MOV** DX, PORTC ;one clock pulse sent on pc0

**OUT** DX, AL

**DEC** AL

**MOV** DX, PORTC

**OUT** DX, AL

**MOV** AL, CH ; get the sevensegment code back in al

**DEC** CL ;send all 8 bits,thus one digit is displayed

**JNZ** NEXTBIT

**DEC** BH

**INC** SI ;display all the four digits

**JNZ** NEXTDIGIT

**MOV** AH, 4CH ;exit to dos

**INT** 21H

CODE ENDS

END START