Palestine Polytechnic University

College of Applied Sciences
Applied Electronics

Graduation Project

Control of Dot Matrix Display (CDMD)

Project Team

Jalal Hamdi Dweik

Zedan Naser Salaimah

Project Supervisor

Eng. Luay Shahin

Hebron – Palestine

Palestine Polytechnic University

Hebron – Palestine

Collage of Applied Sciences

Applied Electronics

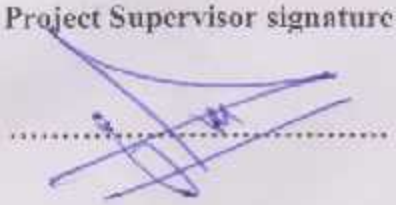
Control of dot Matrix Display (CDMD)

Project Team

Jalal Hamdi Dweik
Zedan Naser Salaimah

According to the orientations of the supervisor on the project and the examined committee is by the agreement of a staffers all, sending in this project to the applied electronics are in the Collage of Applied Sciences by the requirements of the department for the step of the bachelor's degree.

Project Supervisor signature



.....

Committee signature

....

Department head signature

.....

Abstract

The aim of this project is to control the displaying operation of a dot matrix display in order to show a clear short moving message in proper viewing time.

The control process will be achieved through two approaches, first using PC (Personal Computer) control, second using PIC (Microcontroller), both methods will be able to scan the desired massage on the display with proper software and hardware designs.

Index

Chapter one: Introduction *Page*

Chapter Two: Theoretical Background *Page*

Chapter three: Design concept *Page*

Chapter four: Implementation and testing *Page*

Chapter five: Conclusion and future work *Page*

List of Tables

Table *Page*

List of Figures

Figure *Page*

Chapter one: Introduction

1.1 Introduction

1.2 Objectives

1.3 literature review

1.4 Time plan

1.5 Project budget

1.6 Project content

1.1: Introduction

The world that we live in today is developing in rapid steps, one field that is affected by such developments and also affecting the daily life of every one of us is the advertisement world.

A dot matrix display is a common way that is used in the advertisement field. Unlike other costly and bulky displays found in the market, our project is a low cost, robust and optimized for short moving messages and announcements.

The control used in our project achieved using two approaches, to optimize the display work to the maximum.

Each of these two control methods is designed with different soft ware and hard ware requirements.

1.2: Objectives

Our main aim in this project is to build display using the dot matrix led and control it by two methods, by PC control and with simple PIC microcontroller approach and compare them.

For that, we use scanning method that is similar used in to one television.

The display should be capable of showing a clear short message to the viewer in reasonable time duration.

Our final design will be using a scanning line technique that is consuming a minimum power requirement.

1.3 Literature Review

Related to our project is a previous work that used the microprocessor system to display on LCD and displaying Dot matrix such as:

a- "Microprocessor controlled displaying board" for Yahya.Z.Daife in the " Al-Quds University using the 8086 Microprocessor.

b- " Displaying on the LCD by Microcomputer" for Liana Jallal Tammimi and Mohammad Kabajh in PPU use the serial port and the 8085 Microprocessor.

e- " Remotely interactive dot matrix" by internet for language java for Sami Salamin and Waseem Azzam and Muneer Atawneh, the project is problem on display.
From internet resources on PIC & PC Controlled.

1.4 Time Plan

The time planning includes two time schedules; the first one show what is done in the first semester and the second shows the tasks scheduling for the second semester.

Table 1-1 Time Planning (first semester)

Task / Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Literature Review																
Data Collection																
Design and Analysis																
Documentation																

Table 1-2 Time planning (second semester).

Task/Week	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Implementation																
System Testing																
Documentation																

1.5 Project Budget

This section lists the overall cost of the system, cost divided into software cost, hardware cost, and human recourses cost of the components that will be used to implement the system.

Table 1-3 Project Budget

No.	Part Name		Qut.	Cost (NIS)	Total
1	DOTMATRIX	LTP2558CA	32	15	480
2	PC	PENTUM 2	1	700	700
3	PIC	16F84	1	50	50
4	CONTER	74393	1	10	10
5	DECODER	47154	12	10	120
6	TRANSISTOR	2N3904	8	1	8
7	TRANSISTOR	2N3906	5	1	5
8	Copper board		1	100	100
9	Power supply	5v	1	20	20
10	Printing		4	50	200
			Net Total (NIS)		**1693**

1.6 Report Contents

This document consists of three chapters describing the logical and physical parts of the system.

- **Chapter One: Introduction**

  This chapter demonstrates an objective about the system, literature review, time planning, and project budget.

- **Chapter Two: Theoretical Background**

  This chapter focuses on theories and materials that are related to our system operation and behavior. It is mainly talking about Dot Matrix Display, PIC microcontroller, PC parallel port, Counter, Decoder, and buffer circuit.

- **Chapter Three: Design Concepts**

  This chapter describes the system in its abstract formula, a general block diagram and how the system works.

- **Chapter four : System Implementation and testing**

  This chapter includes the implementation phases with testing of these phases, general hardware and software components are tested and shown in this chapter.

- **Chapter five : Conclusions and Future Work**

  This chapter will provide our conclusions and suggestions for future work.

Chapter Two: Theoretical Background

2.1 Introductions

2.2 Dot matrixes

2.3 PIC microcontroller

2.4 Counter

2.5 IBM-PC Parallel Port

2.6 Decoder

2.7 The Opto-couplers

2.1 Introductions

This chapter will describe the main hardware components that our project will be using it in the implementation process to achieve the needed functionality of the system, and the components will be described as following.

2.2 Dot matrix display

Description:

The devices are available in either common row anode or common row cathode configurations, these displays have 5X8 single hetero junction red dot matrix display it has a grey face with neutral segment color.

The displays come in only black face paint. This bi-color display consists of gap red and gap green colors. As shown in figure (2.1) and table (2.1).

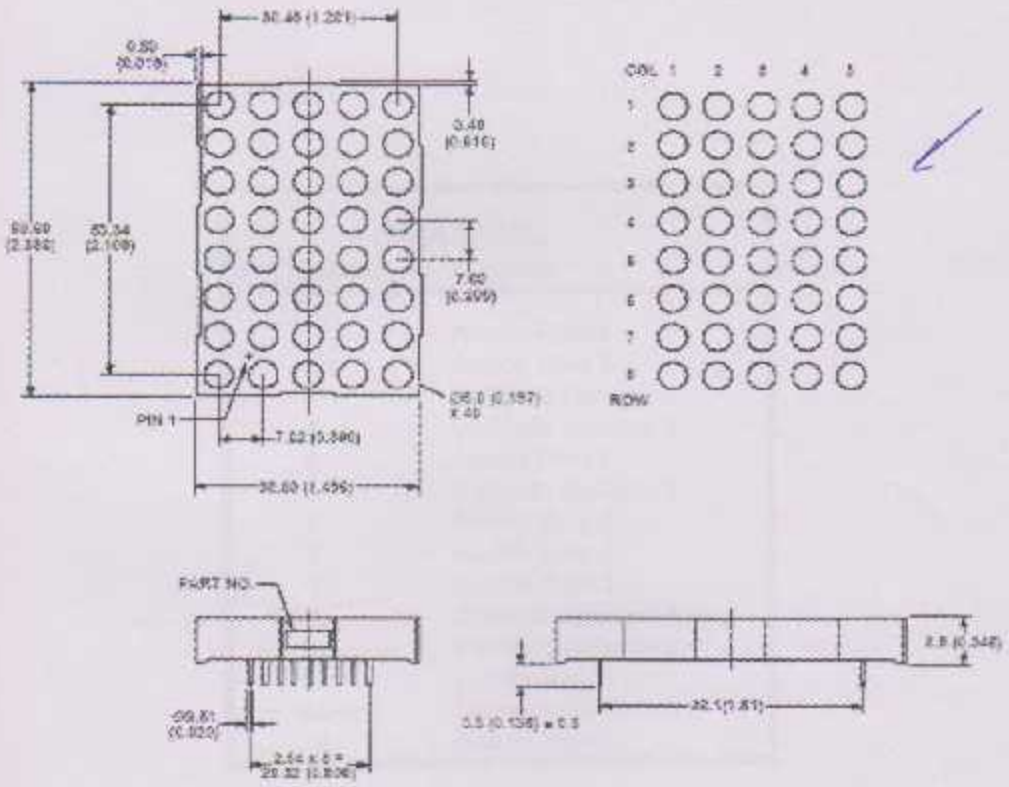


Figure 2.1 the package dimension for (5X8) dot matrix

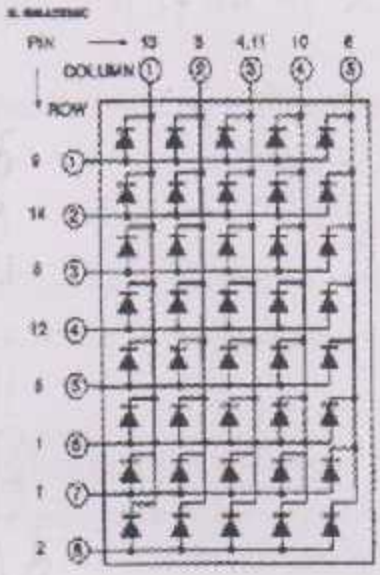


Figure 2.2 the schematic of dot matrix

Table (2.1) pin connection of dot matrix

GMA2285C	
Pin Number	Function
1	Anode Row 6
2	Anode Row 8
3	Cathode Column 2
4	Cathode Column 3
5	Anode Row 5
6	Cathode Column 5
7	Anode Row 7
8	Anode Row 3
9	Anode Row 1
10	Cathode Column 4
11	Cathode Column 3
12	Anode Row 4
13	Cathode Column 1
14	Anode Row 2

2.3 PIC microcontroller

PIC (peripheral interface controller) is a microcontroller that was developed to control peripheral devices; PIC chip is shown in figure (2.3) with its pins designations.

The PIC, like the CPU, has calculation functions and memory, and is controlled by the software. However, the throughput and the memory capacity are low.

Depending on the kind of PIC, the maximum clock operating frequency is about 20 MHz and the memory capacity (to write the program) is about 1k to 4k words.

The clock frequency determines the speed at which a program is read and an instruction is executed. The throughput cannot be judged with the clock frequency alone, it changes with the processor architecture.

However, within the same architecture, the one with the highest clock frequency has the highest throughput. We use a 14-bit word for program memory capacity.

An instruction is a word long. The instruction word of the PIC16f84a is composed of 14 bits. 1k words are equal to 1 x 1,024 x 14 = 14,336 bits. To convert this to bytes divide it by 8 x 1024, as shown in figure (2.4), as shown in table (2.3).

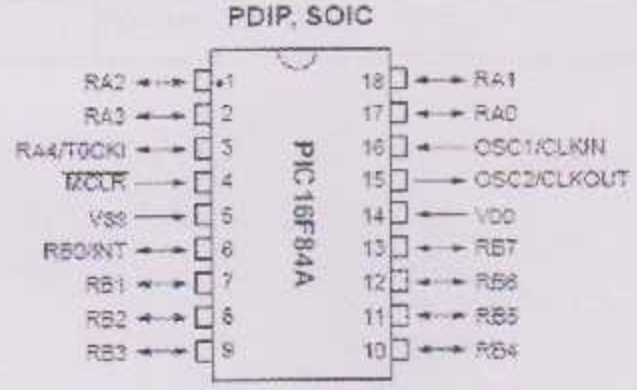


Figure 2.3 pin diagram of PIC

Table (2.3) pin Assignments of PIC

Osc1/clkin	Oscillator crystal input. External clock source input.
Osc2/clkout	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode.
Mclr(inv)	Master clear(reset)input. Programming voltage input. This pin is an active low reset to the device.
Ra0 - ra3	Bi-directional i/o port.
Ra4/t0cki	Bi-directional i/o port. Clock input to the tmr0 timer/counter.
Rb0/int	Bi-directional i/o port. External interrupt pin.
Rb1 - rb7	Bi-directional i/o port.
V_{ss}	Ground
V_{dd}	Positive supply(+2.0v to +5.5v)

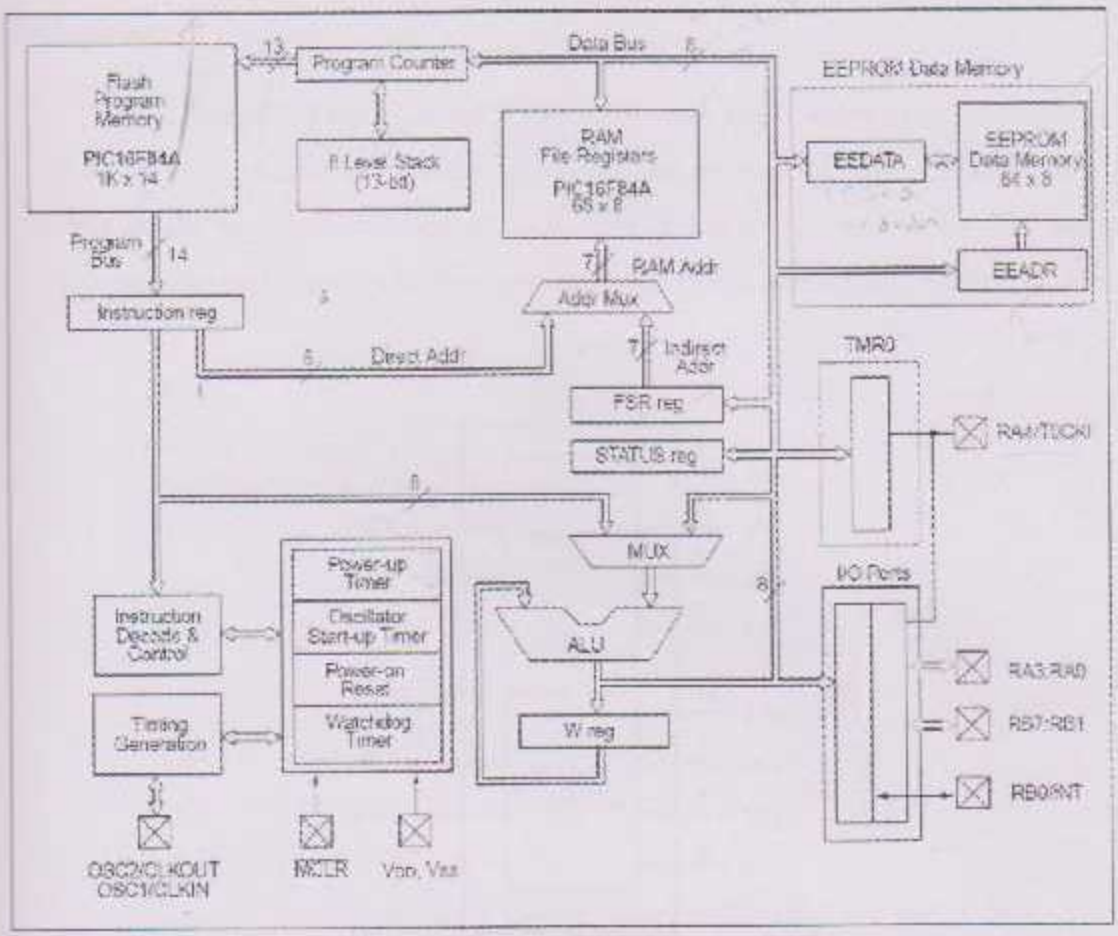


Figure 2.4 PIC Architecture

2.4 Counter

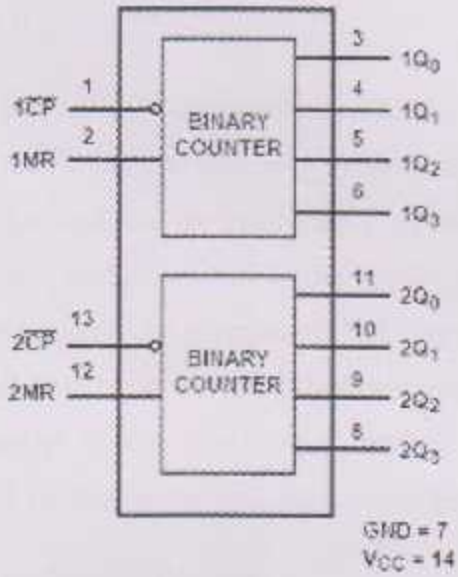
A counter is essentially a register that goes through a predetermined sequence of states. The gates in the counter are connected in such a way as to produce the prescribed sequence of binary states. Although counters are a special type of a register, it is common to differentiate them by giving them a different name.

In our project, the counter will be used for scanning operation that will transfer the selection of the dot matrix piece automatically.

In the figure (2.5), that shows the 4-bit synchronous binary counter (74hc).

Description

The CD74HC393 and CD74HCT393 are 4-stage ripple-carry binary counters. Al counter stages are master slave flip-flops. The state of the stage advances one count on the negative transition of each clock pulse; a high voltage level on the MR line resets all counters to their zero state. All inputs and outputs are buffered.



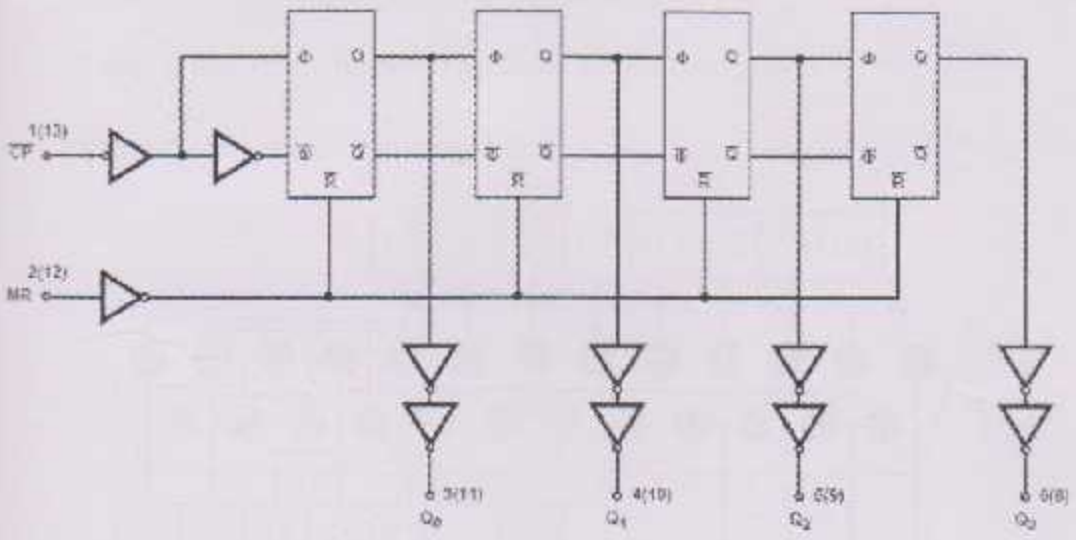


Figure 2.5 dual 4-bit binary counters

2.5 IBM-PC Parallel Port

IBM originally supplied three adapters that included a parallel printer port for its PC/XT/AT range of microcomputers. Depending on which were installed, each available parallel port's base address in the processor's I/O space would be one of 278, 378 and 3BC (all Hex).

All contemporary PCs, shipped with a single parallel printer port, seem to have the base address at 378 Hex.

The PC parallel port adapter is specifically designed to attach printers with a parallel port interface, but it can be used as a general input/output port for any device or application that matches its input/output capabilities. It has 12 TTL-buffer output points, which are latched and can be written and read under program control using the processor In or Out instruction. The adapter also has five steady-state input points that may be read using the processor's in instruction.The original IBM-PC's Parallel Printer Port had a total of 12 digital outputs and 5 digital inputs accessed via 3 consecutive 8-bit ports in the processor's I/O space.

- 8 output pins accessed via the data port
- 5 input pins (one inverted) accessed via the status port
- 4 output pins (three inverted) accessed via the control port
- The remaining 8 pins are grounded

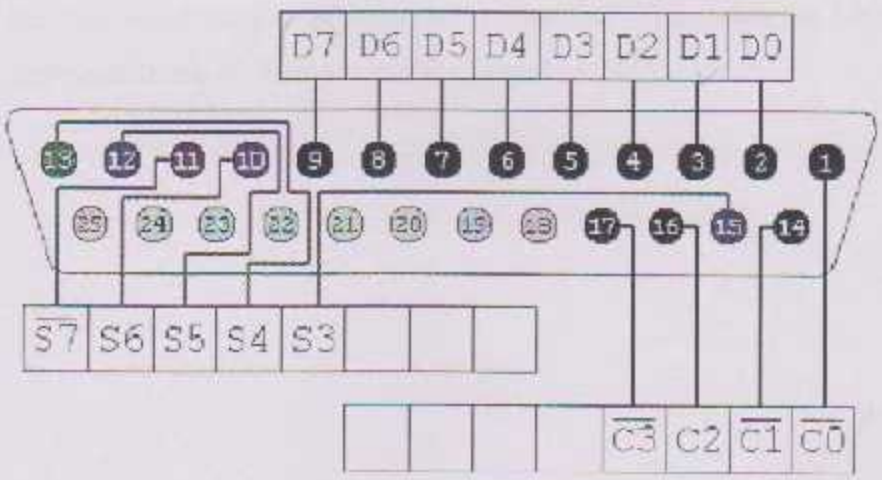


Figure 2.6 : 25-way Female D-Type Connector

2.6 Decoder

A 4 binary –coded decoder input into of 16 mutually excusive output, the number 74154 is ideal for high-performance memory decoding, performs the demultiplixing function by distribution data from one input line to any one of output, input clamping diodes simplify system design as shown the figure (2.7).

Figure 2.7 Decoder (4 X16)

Description:

Each of these monolithic, 4 lines to 16 line decoder utilizes circuitry to decode four binary –coded input into one of sixteen mutually exclusive output when both the strobe input . G1 and G2, are low, the demultiplixing function is performed by using. the four input lines to address the output line. The following table (2.2) shows correspondence of the input and the output of 74154.

Table 2.2 Correspondence of the input and the output of 74HC154

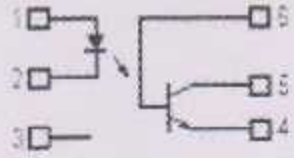
Input				Output															
D	C	B	A	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L

2.7 Opto-couplers

Optocouplers incorporate two electronic components in a single device — an input infrared (IR) light emitting diode (LED) and an output photo detector. Solid State Relays, on the other hand, consist of an infrared LED optically coupled to a photovoltaic generator which drives a power MOSFET detector.

Two characteristics are critical to Optocoupler performance — Voltage Isolation and Current Transfer Ratio. Voltage isolation (VISO), between the IR LED and the photodetector, is controlled by materials in the light path and by the physical separation between the two components. Current transfer ratio (CTR), is defined as the ratio of detector current to emitter current Fairchild Semiconductor's product offering allows customers the broadest choice of Voltage Isolation and CTR options. For the latest information and datasheets on our optocoupler and SSR families, as shown in figure (2.8), and table (2.3).

SCHEMATIC



PIN 1. LED ANODE
2. LED CATHODE
3. N.C.
4. EMITTER
5. COLLECTOR
6. BASE

Figure 2.8 Pin diagram of Opto-couplers

Table 2.3 Correspondence of the input and the output of Optocouplers

INPUT LED

Reverse Voltage	V_R	3	Volts
Forward Current — Continuous	I_F	60	mA
LED Power Dissipation @ T_A = 25°C with Negligible Power in Output Detector Derate above 25°C	P_D	120 1.41	mW mW/°C

OUTPUT TRANSISTOR

Collector–Emitter Voltage	V_{CEO}	30	Volts
Emitter–Collector Voltage	V_{ECO}	7	Volts
Collector–Base Voltage	V_{CBO}	70	Volts
Collector Current — Continuous	I_C	150	mA
Detector Power Dissipation @ T_A = 25°C with Negligible Power in Input LED Derate above 25°C	P_D	150 1.76	mW mW/°C

Chapter Three: Design concept

3.1 Introduction

3.2 General block diagram of PC

3.3 The schematic design of PC

3.4 The Overall Schematic Diagram OF PC

3.5 General block diagram of PIC

3.6 The schematic design Of PIC

3.7 The Overall Schematic Diagram OF PIC

3.8 The LED Driving Circuit

3.9 Flow chart

3.1 Introduction

In this chapter we will describe the systematic and general block diagrams.

3.2 General block diagram of PC

Our system Block diagram representation is shown in figure 3.1

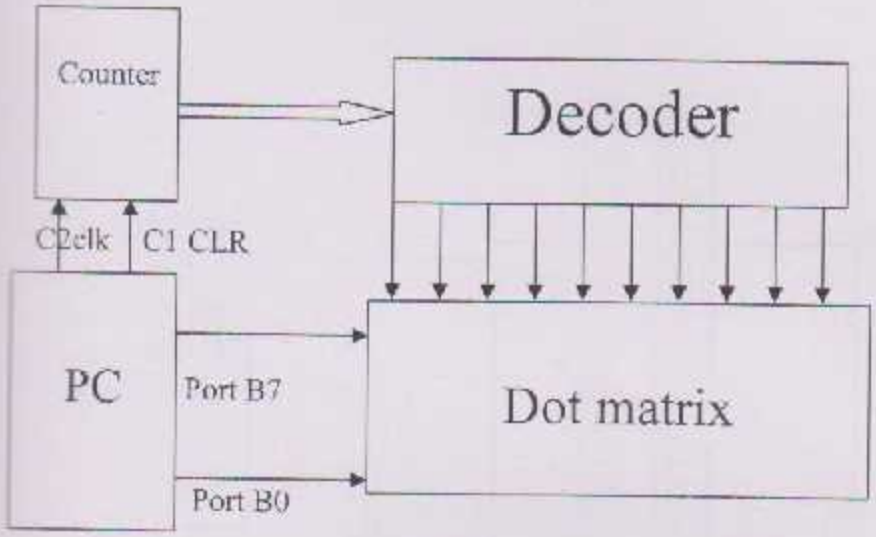


Figure 3.1 The block diagram of PC control

The 5X8 Dot matrix display can be controlled by selecting one of the five columns, and sending the desired 8 data bits to the assigned dots.

Hence, the display operation can be done in two steps:

1) Select the column.
2) Send data that will be displayed

Selecting the column can be done through a decoder circuit, then sending data from PC throw the parallel port to target dots , the interfacing and decoding circuit task is to manage the dot matrix display position and state of ON/OFF leds using the scanning technique.

3.3 The schematic design Of PC

The data will be send from PC to the designed system that will be used to display one column from each piece of the dot matrix display screen, this operation will continue to make it for each columns in each piece, after finishing the first dot matrix piece the selected circuit will choose the next dot matrix piece to display on it and so on, as shown the figure 3.2, since of display consist of several number of single 5x8 dot matrix, arranged in 2 rows & 8 columns.

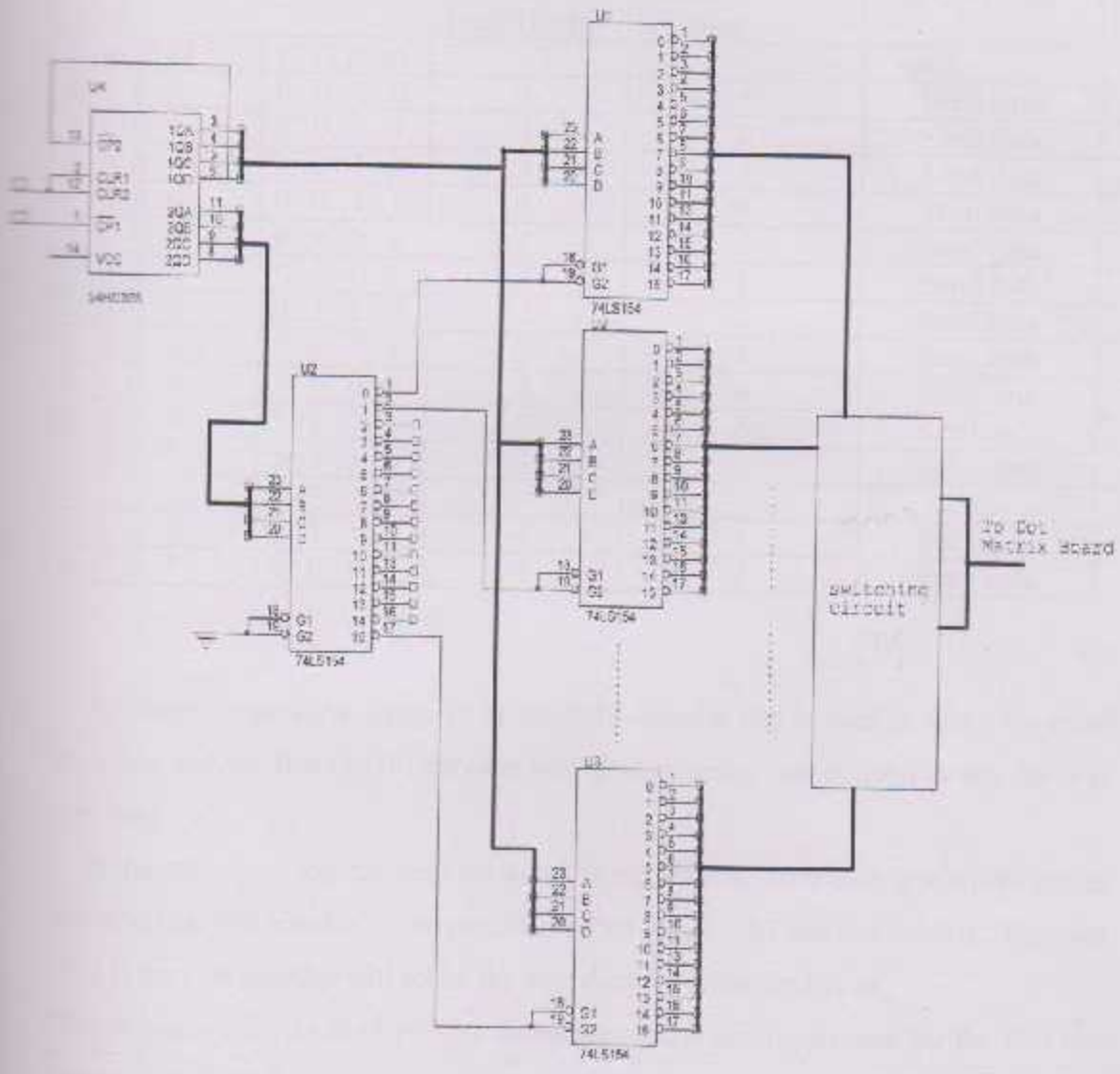


Figure 3.2 Schematic design of PC Control

The scanning and selection operation can be achieved by using a simple (8 bit) counter.

The counter will take the responsibility of selecting the needed dot matrix column in each piece; this operation can be implemented using 5 decoders.

The following table shows the first select and scanning process of the first two pieces (decoders) with the dot matrix column.

Table 3.1 the selection technique using the counter.

Counter Output		Selected Piece (DM)	Selected Column	SW Operation	Selected decoder
s7 s6 s5 s4	s3 s2 s1 s0				
0 0 0 0	0 0 0 0	1	1	Send data	1
0 0 0 0	0 0 0 1	1	2	Send data	1
0 0 0 0	0 0 1 0	1	3	Send data	1
0 0 0 0	0 0 1 1	1	4	Send data	1
0 0 0 0	0 1 0 0	1	5	Send data	1
0 0 0 0	0 1 0 1	2	1	Send data	1
0 0 0 0	0 1 1 0	2	2	Send data	1
0 0 0 0	0 1 1 1	2	3	Send data	1
0 0 0 0	1 0 0 0	2	4	Send data	1
0 0 0 0	1 0 0 1	2	5	Send data	1
0 0 0 0	1 0 1 0	3	1	Send data	1
.....			.....	.....	1
0 0 0 0	1 1 1 1	4	1	Send data	1
0 0 0 1	0 0 0 0	4	2	Send data	2

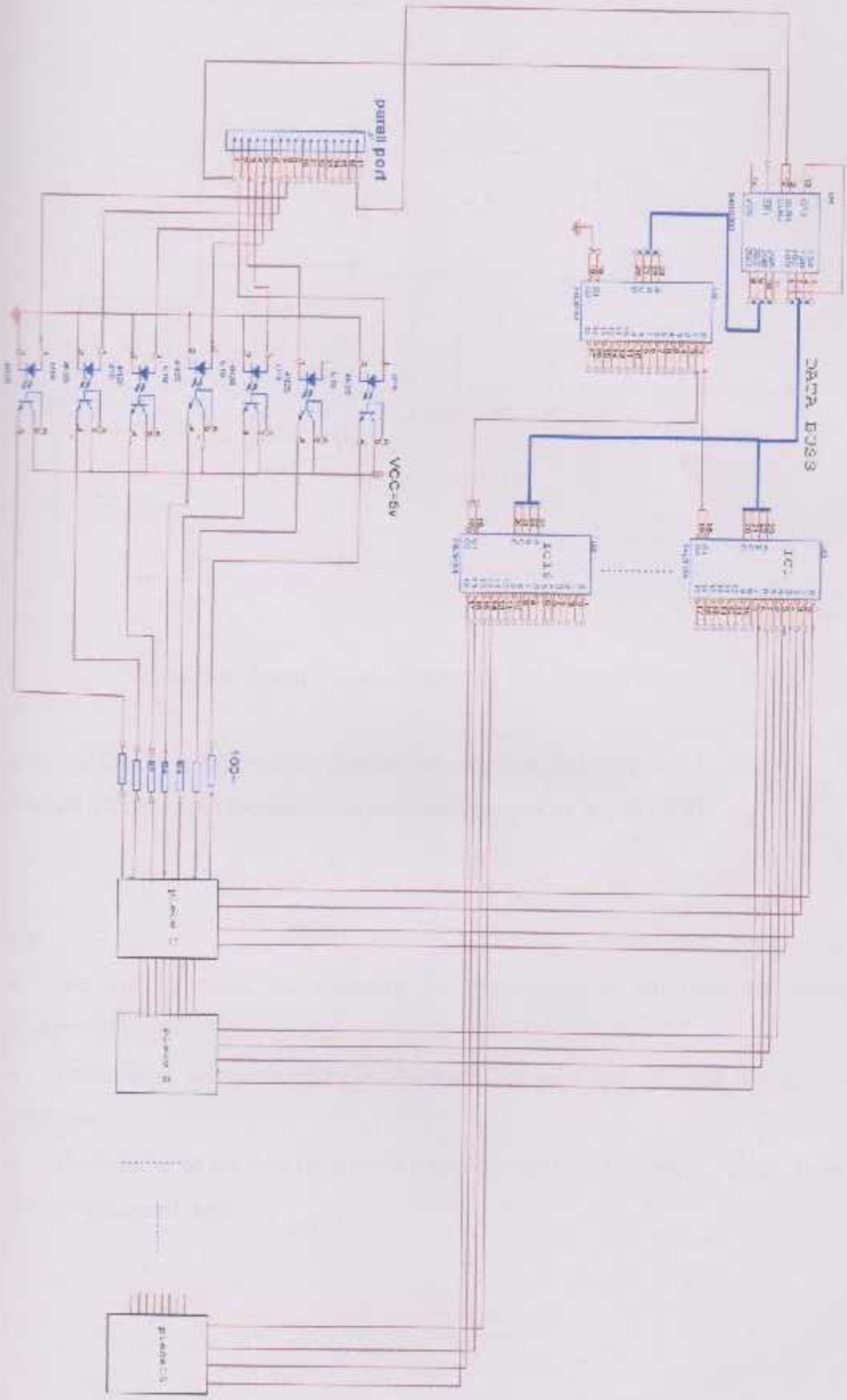
As shown in previous figure (3.2) the 4x16 decoder that is used to select the other decoders and the five (4x16) decoder will used to select one column in any piece at any time.

In the next count step the next column will be selected, after each five clocks pulses the selection will transfer to the next dot matrix piece, after the first 4-bit (LSB) reach (1111) the first decoder will select the next decoder piece, and so on.

The previous table (3.1) shows the first select and scanning process for the first two pieces (decoders) with the dot matrix column.

3.4 The Overall Schematic Diagram OF PC

In this section we will consider the overall electrical circuit for the PC method, as shown in figure (3.3)


parall port
VCC-5v
DATA BUS

3.5 General block diagram of PIC

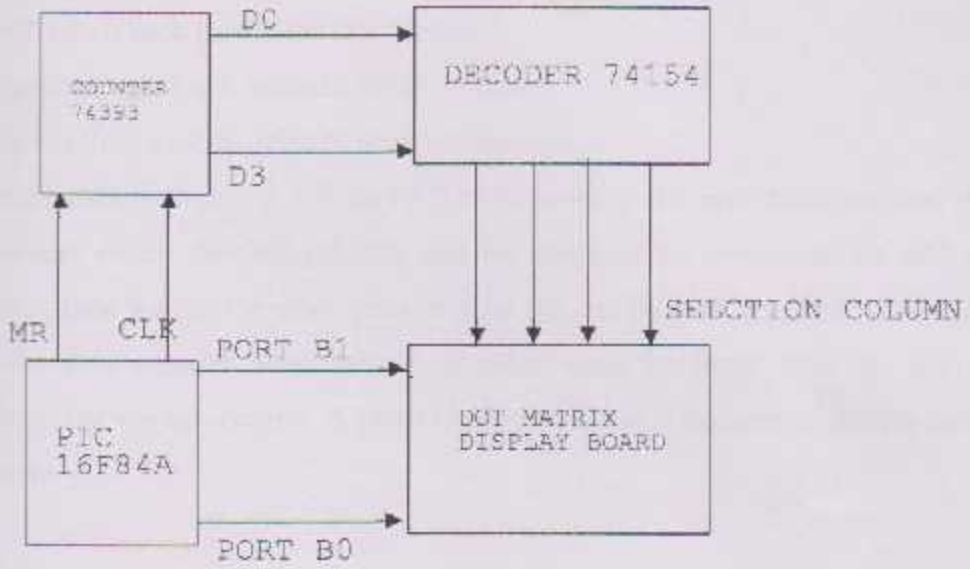


Figure 3.4 General block diagram of PIC Control Circuit

Controlling the dot matrix will be the same as we explained in section 3.2 but the difference is that we get the data from the memory by port B in the PIC.

Notes:

- We will be using the scanning line techniques to minimize the power dissipation.
- There must be synchronization between the data transfer and the column selection.
- The transfer of the data must be in a speed more than the eye response to not see the change of data.

3.6 The schematic design Of PIC

PIC control circuit operation:

RB6-RB0 ports are used for the input of the optocoupler that to amplify the current that will reach each row in the Dot Matrix.

RA0 generating a clock pulse to set the counter.

RA1 generating a clock pulse to reset the counter.

we send data from port A (of the PIC) to all rows, at the same time we must choose by decoder circuit the first column, and we continue the process to the end of the columns, then we run the other decoder after the end the first one by the control unit. And the data transfer from column to other must be faster than the eye speed response, but we can control 8 Dot Matrix pieces only. Because of the limitation of its memory.

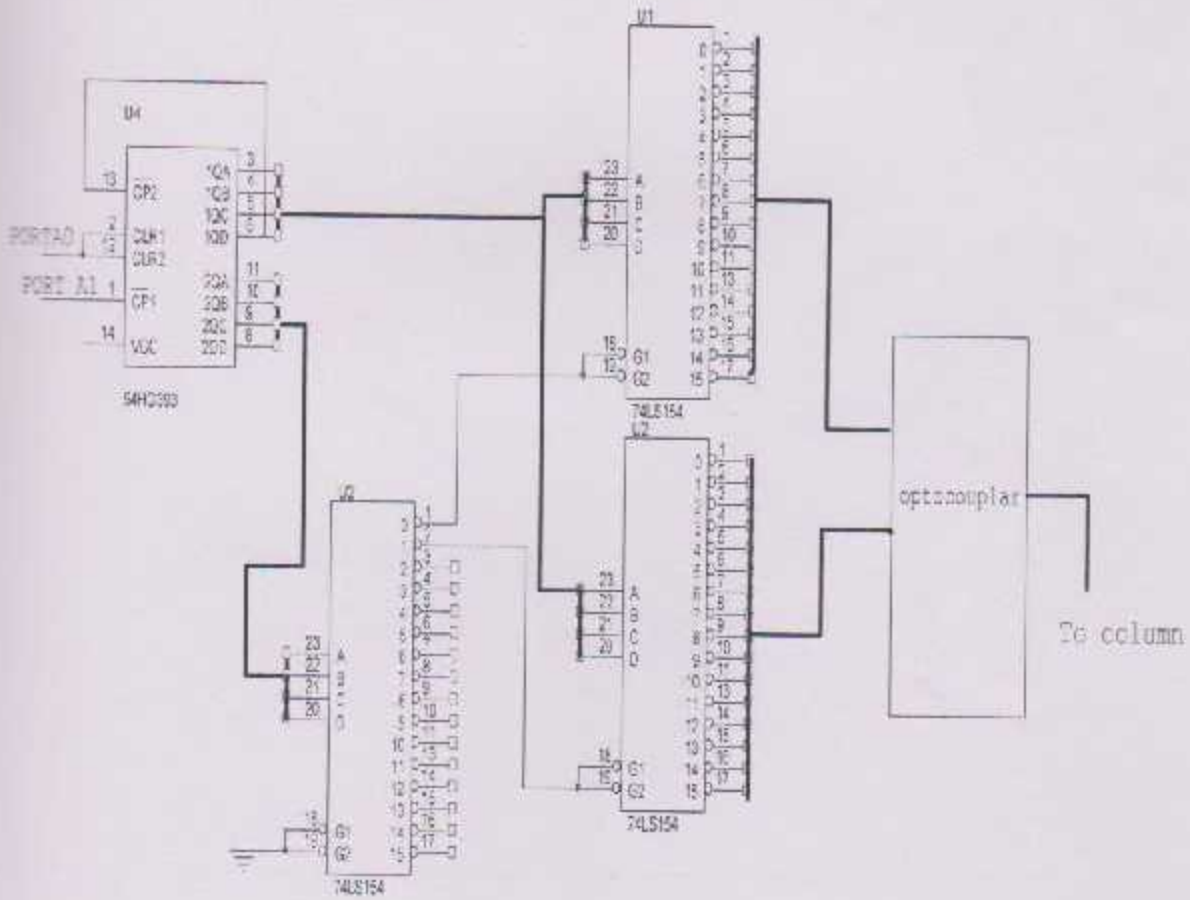
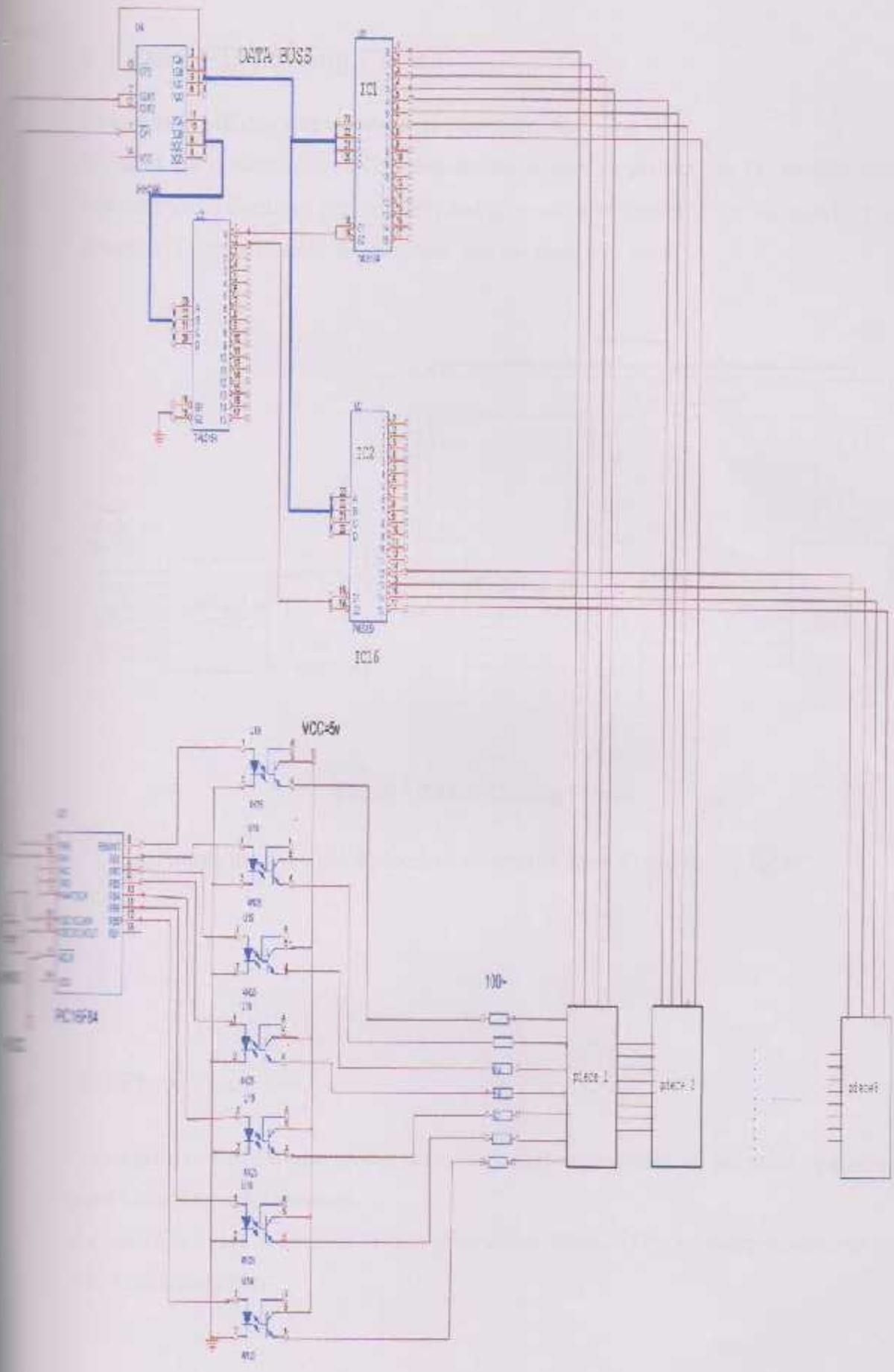


Figure 3.5 Schematic design of PIC control

3.7 The Overall Schematic Diagram OF PIC

In this section we will consider the overall electrical circuit for the PIC method, as shown in figure (3.6)


DATA BUS8
IC1
IC2
IC1.6
VCC=5v
100-

3.8 The LED Driving Circuit

This section will describe the electrical drive circuit

We used the optocoubler (4N25) this device is used to protect the PC parallel port from any short circuit or high current, and give suitable current to the dot matrix, this device is connected before the resistors and we used 8 of them.

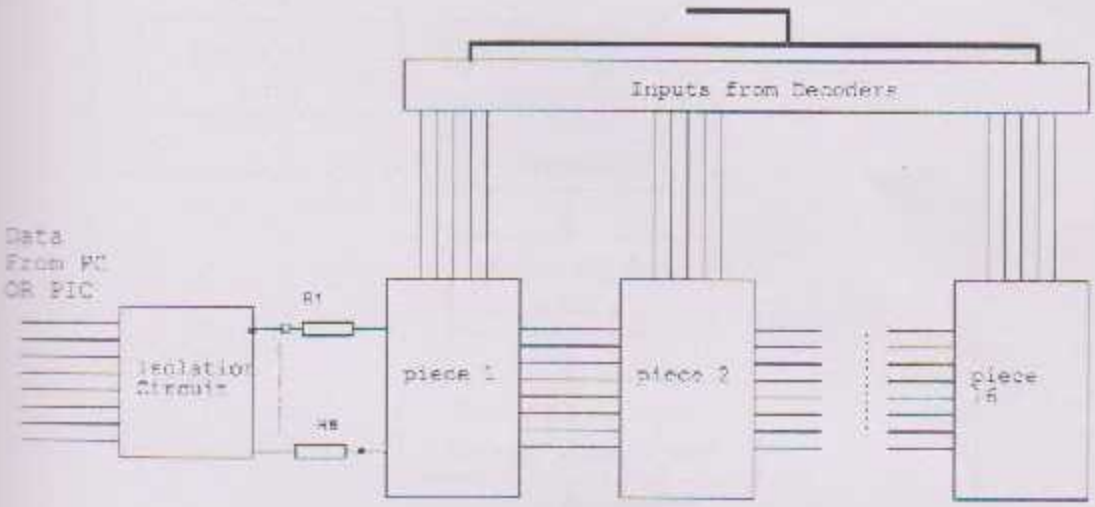


Figure 3.7 LED Driving Circuit

Noting that we used 8 pieces of dot matrix for PIC and the 16 for PC

3.9 Flow chart

As shown in fig (3.6) the sender work is to send the message on pc, other operation used from the sender program.

As shown in the flow chart (3.7), this showed the outline of the software processing of PIC microcontroller.

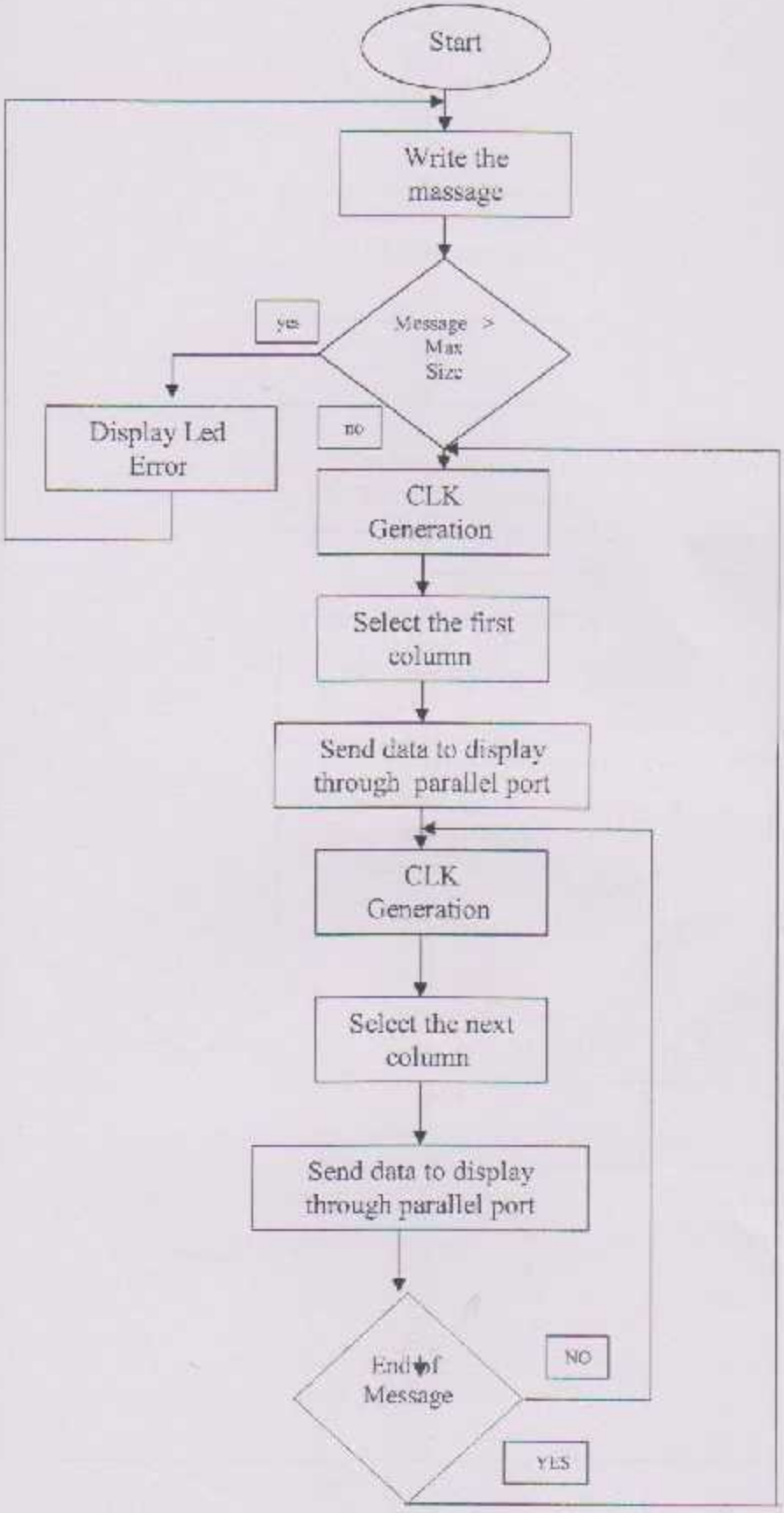


Figure 3.8 Send the message on PC

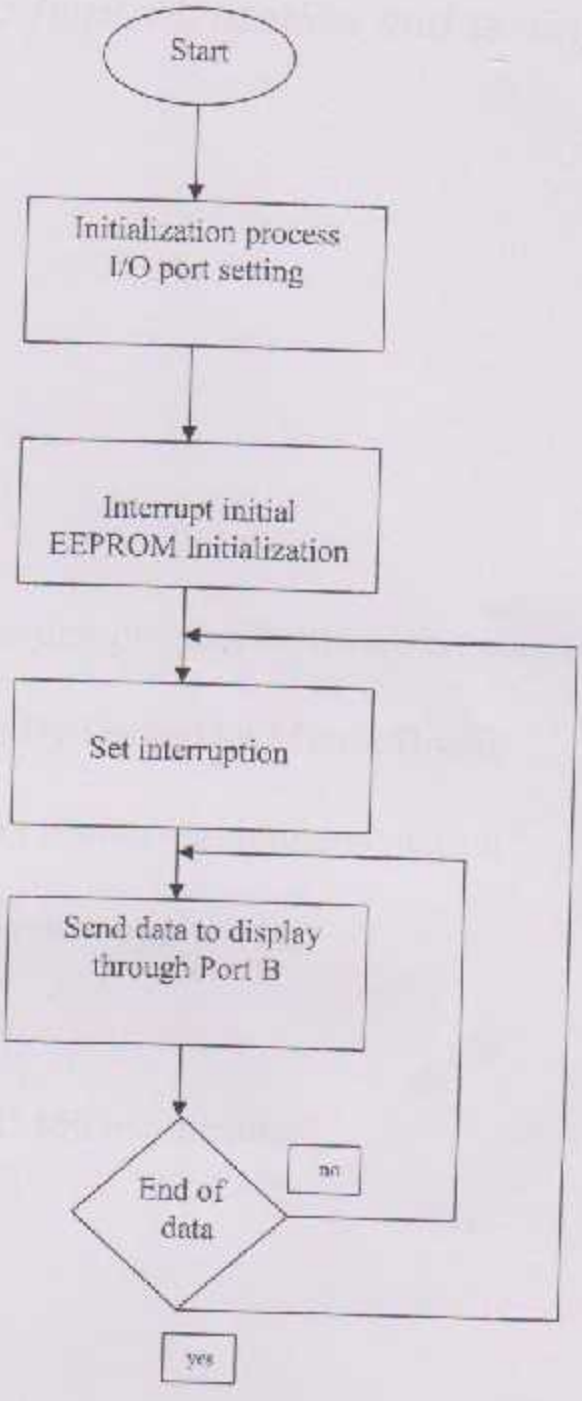


Figure 3.9 Send message using PIC

Chapter four: Implementation and testing

4.1 Introduction

4.2 Implementation

4.3 Implementing and testing projects hardware components

4.4 Testing the Program By Using Dot Matrix Board

4.5 Problem faced during testing and implementation

4.6 Clock speed and power supply

4.6.1 Clock Speed of PC

4.6.2 Clock Speed of PIC Microcontroller

4.6.3 Power Used

4.1 Introduction

In this chapter we show the implementation and the testing process for our system.

The implementation and testing was done using the following tools and components:

- Connectors with different colors
- IC stands
- 10*50 cm bread board
- All the ICs that are depicted in the design chapter (see chapter three)
- Wrapper tool for wrapping the connectors on the ICs stands
- A digital Multimeter for testing

4.2 Implementation

In this section we are going to talk about the implementation process. The implementation and testing process are done in synchronization, individual design element are assembled and tested before connecting it together.

4.3 Implementing and testing of projects hardware components

As we mentioned previously after completion of individual circuits that we described it in chapter three, we used the C language and the probing tester circuit to test each subsystem and test every input and output with different cases and speed.

The testing operation is done in more than one phase as following:

Phase 1

The dot matrix board is made and tests each piece using external circuit and digital multi meter.

Phase 2

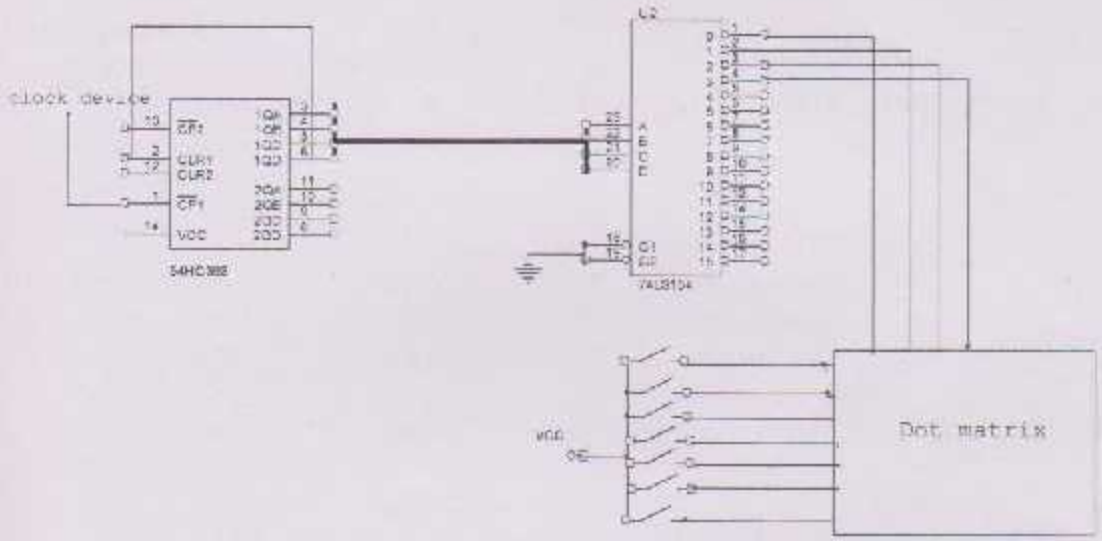
Testing of the counter circuit synchronization by using external a clock device, and its speed of response.

Phase 3

Testing of the decoder IC and counter circuit synchronization, connecting external LEDs to the output and voltage sources to the input.

Phase 4

Connecting the counter circuit and decoder output to a single 5X8 dot matrix display, and then entering the data on the rows by using external voltage source, and we used several clock frequencies to get to have a fixed data on the dot matrix display, as show the figure (4.1).



The figure (4.1) Testing of dot matrix

Phase 5

In this case we changed the function generator and the external voltage source by using the parallel port from the pc, and we face many difficulties to get the suitable clock speed and data from it.

Phase 6

After we succeed in getting fixed data on a single 5X8 dot matrix we built all the board design as shown in fig (3.3).

Phase 7

Writing program code in C language and running it on the board.

Phase 8

And in similar way we test the PIC microcontroller but in different in the last phases that we test the PIC microcontroller by using simple program as shown

```
LIST p=16f84a
     #include p16f84a.inc
     __CONFIG  XT OSC &  CP OFF & _WDT_OFF & _PWRTE_OFF



V1        EQU       0X0C              ;counter
V2        EQU       0X0D              ;counter
V3        EQU       0X0E              ;counter

          ORG       0X00 ; start origin 00 in memmory

          BSF       STATUS,5
          CLRF TRISB                  ; define PB output
          CLRF TRISA
          BSF       TRISA,2           ;  define PA2 input
          BCF       STATUS,5
          BSF       PORTA,0           ; define PA0 input
          BCF       PORTA,1           ;define PA1 input
LOPTI
          ;;;;;;;;;;;;;;;;;;;;;;    ; Display chara P
          ;;;;;;;;;;;;;;;;;;;;;;
          MOVLW     0X7F        ; load 7f in rigestar w
          MOVWF     PORTB       ; send data on PB
          CALL DELY0            ; delay
          CALL CLK              ; send clk
          MOVLW     0X09        ;load 09 in rigestar w
          MOVWF     PORTB       ; send data on PB
          CALL DELY0            ; delay
          CALL CLK              ; send clk
          MOVLW     0X09        ;load 09 in rigestar w
```

```
            MOVWF     PORTB       ; send data on PB
            CALL DELY0            ; delay
            CALL CLK              ; send clk
            MOVLW     0X06        ;load 06 in rigestar w
            MOVWF     PORTB       ; send data on PB

CLK         BCF       PORTA,0           ; send rest of PA0
            NOP
            BSF       PORTA,0           ; send clk of PA0
            RETURN

            ;;;  ; delay between two succissive data

DELY0       MOVLW     D'15'
            MOVWF     V2
LOOP1       MOVWF     CXFF
            MOVWF     V3
LOOP1       DECFSZ    V3,1
            GOTO LOOP1
            DECFSZ    V2,1
            GOTO LOOP1
            RETURN

            END
```

Phase 10

Writing the final program of PC and PIC microcontroller.

Phase 11

End the project and present it in the final version.

4.4 Testing the Program on the Dot Matrix Board

We used the C language to testing of dot matrix alone and the C program is.

```
#include<conio.h>
#include<dos.h>
#define cp 0x37A
#define dp 0x378
main()
{
outport(cp,0);
while(1)
{

for (j=0;j<1;++j)                    //  send  char
     {

     for (int i=0;i<400;++i)          // delay
     outport(dp,0x7e);                // send data of coulmn 1
     outport(cp,7);                   // send clk
     outport(cp,3);                   // send rest

     for (i=0;i<400;++i)              // delay
     outport(dp,0x11);                // send data of coulmn 2

     outport(cp,7);                   // send clk
     outport(cp,3);                   // send rest


     for (i=0;i<400;++i)              // delay
     outport(dp,0x11);                // send data of coulmn 3
     outport(cp,7);                   // send clk
     outport(cp,3);                   // send rest
     for (i=0;i<400;++i)              // delay
     outport(dp,0x7e);                // send data of coulmn 4
     outport(cp,7);                   // send clk
     outport(cp,3);                   // send rest
     }


outport(cp,0);

}
}
```

The final version of the program is included in the appendices and the complete program will reformat the message and then send the reformatted message sequentially - byte by byte- to the dot matrix board using the parallel port.

4.5 Problem faced during testing and implementation

- In phase 5, we changed the function generator and the external voltage source by using the parallel port of the pc, and we faced many difficulties to get the suitable clock speed and data from it, then we solve this problem by using simple program on C language to show data on one item of dot matrix, as shown in the program (4-5).

- In phase 7 we noted that the lighting is not sufficient , on other hand the board response to the data is too weak, we use many types of transistors to increase the lighting intensity and to have good response, but all the transistors types give good current but its too weak in response to the clock speed, then we used the opto_coupler IC and its give us good current and its response for data is high too.

- Some chips unavailable or cannot be obtained easily.

4.6 Clock speed and power supply

4.6.1 Clock Speed of PC

We use the PC in our project to control the Dot Matrix Display Board and we use Pentium three exactly , and by using another type of PC the delay of the program will different to the new PC speed that chosen and the following figure show the PC type and its speed.

Here the way that we calculate the time duration

We get the frequency of PC that out from the parallel port by the oscilloscope

And it where 180 Hz

4.6.2 Clock Speed of PIC Microcontroller

We use the crystal IC as pulse generator, that it's generating a pulse.

And we use four mega hertz crystal that it's the suitable speed that we need

Here is the way that we calculate the time duration

Machine cycle =5980*1 μs = 6 ms

Frequency =1/machine cycle = 1/6 ms= 170 Hz

4.6.3 Power Used

In our project we used an uninterrupted power supply taken from a PC, which gives a multi voltage outputs. We use 5 volt line with 0.5 A. used for optocouplers, and other 5 volt line with 0.3A.for other ICS.

The advantage of uninterrupted power supply is that out put current is high other wise its can used for long period of time without any effects on output.

Chapter five: Conclusion and future work

5.1 Introduction

5.2 Conclusion

5.3 Future Work

5.3.1 Multi Color Display

5.3.2 USB Interfacing

5.3.3 Multi Function Display

5.3.4 Wireless data board

5.1 Introduction

By the end of this work, it's very important to mention our conclusions and to put some notes for future work in order to develop this project.

5.2 Conclusion

Finally, and after all the work that we have done , we hope that our project is the first step for others to make good projects , and we also hope to present our hard work as an applied work to our study and skills in our college of applied science and a real application in applied electronics .

Lastly, and from our work in the project we note some differences between the two methods of controlling the dot matrix, these points can be summarized as following.

- Speed: the scanning speed on the dot matrix board will be the same, but the different speed of PC and PIC that in PC method we use the PC speed itself, and in our project we use the Pentium three family which have 600 mega hertz in PIC method we use the crystal IC'S which have 4 mega hertz only and we use delay method to get suitable scanning speed.

- Memory size : as we know that the PC have a hard disk in high size, and you can write whatever you want ,and use the best program without limitations on size, but in PIC microcontroller that we use the memory size that have its only one kilo byte ,so you will face a lot of limitation in writing the program ,so the data that you can out so limit and it must be under the one kilo that the PIC have ,the reason its to not forget the instruction of PIC and its size in memory .

- Operation: the PC method is better tenth times than the PIC 16f84A that we use, that in PC you can handle with big programs which means more advanced operations and large controlling, and you can get multi operation by using the

PC method, on the other hand the PIC microcontroller can give an one uniquely operation with limit control but really in less facing of problems.

- Coding: the way of writing the program in the PC method by using the C language is difficult but we use functions way to make it easy and applicable, but the way that we writing the program on PIC method by PIC programmer we must use the way of writing its instruction and to compile it to hex cods that PIC can handle with it, and its waste time and take more work.

- Cost: PC device price that we use in our project reach to 500$, but the PIC microcontroller price maximum reach 10$.

- Limitation: our project is designed to use it in halls, and in covered places, so the thermal protection is come from the place itself, that the components which we use have thermal protect and limitations suitable to the covered places and you can see it in the data sheet of every item.

And from the previous points we conclude that the PC method is more advanced and technical way for changeable advertising.

The PIC method is good for fixed advertising and data, the difficulty of use comes from the complexity of its program and instructions, and the main disadvantage of it is the limitation of memory size.

Finally, we achieved all our project aims that we explain in chapter one.

5.3 Future improvement

5.3.1 Multi Color Display

As maintained before the dot matrix that we used is a bi-color that can display Red and Green character, and there another type of dot matrix that can display a 6 color we assume the future work, make a multi-color display board.

5.3.2 USB Interfacing

In our project we used the parallel port and we assume the future work make an USB interfacing for this display board.

5.3.3 Multi Function Display

In our project we use the display board to present letters and numbers, but we hope its can be developed to present time, date, temperature degree, counting entering persons, and drawings.

5.3.4 Wireless data board

In our project we get the data from the parallel port and the PIC by using wires, but we hope that to develop it, by sending data by wireless method.

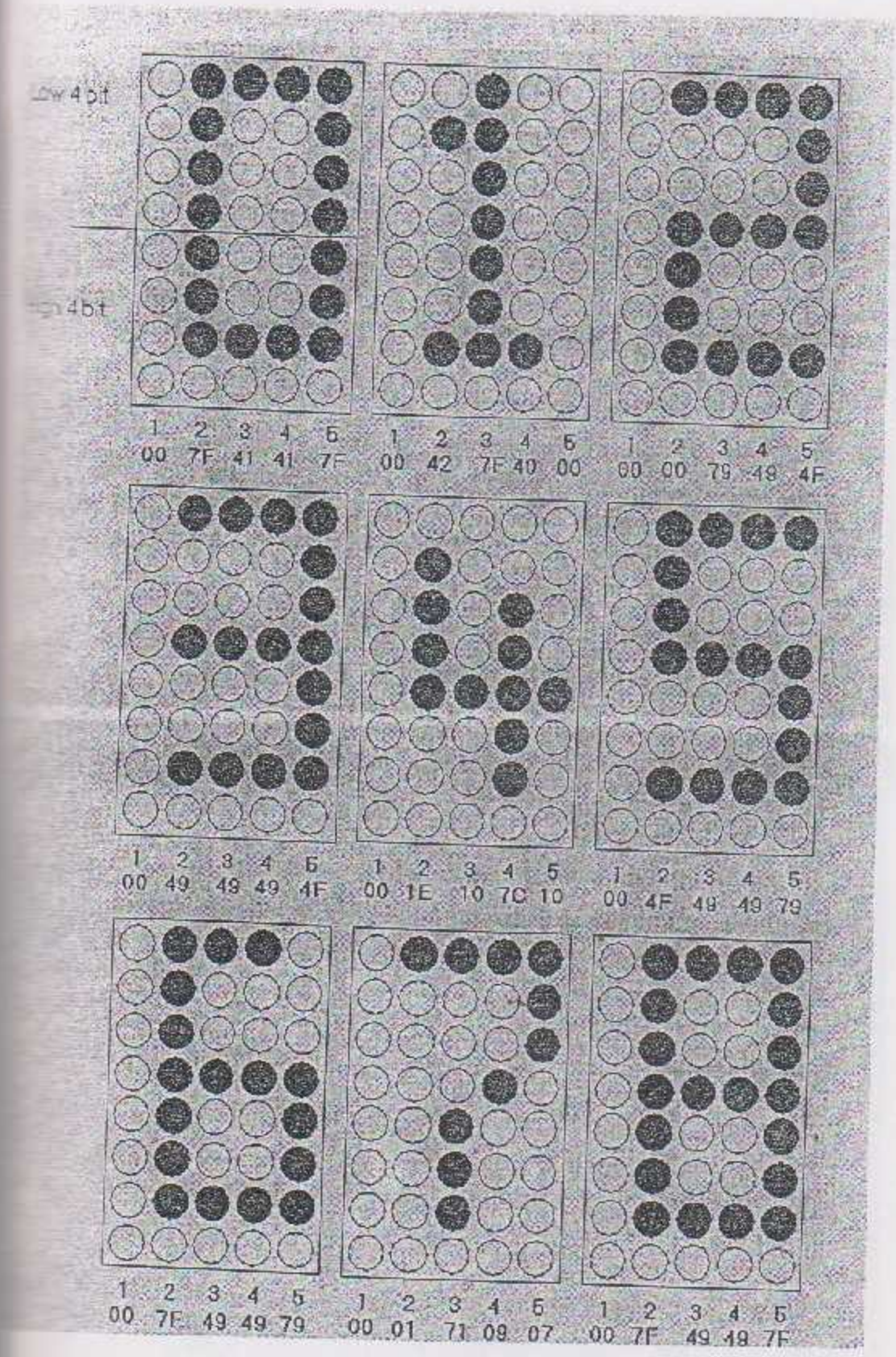
References

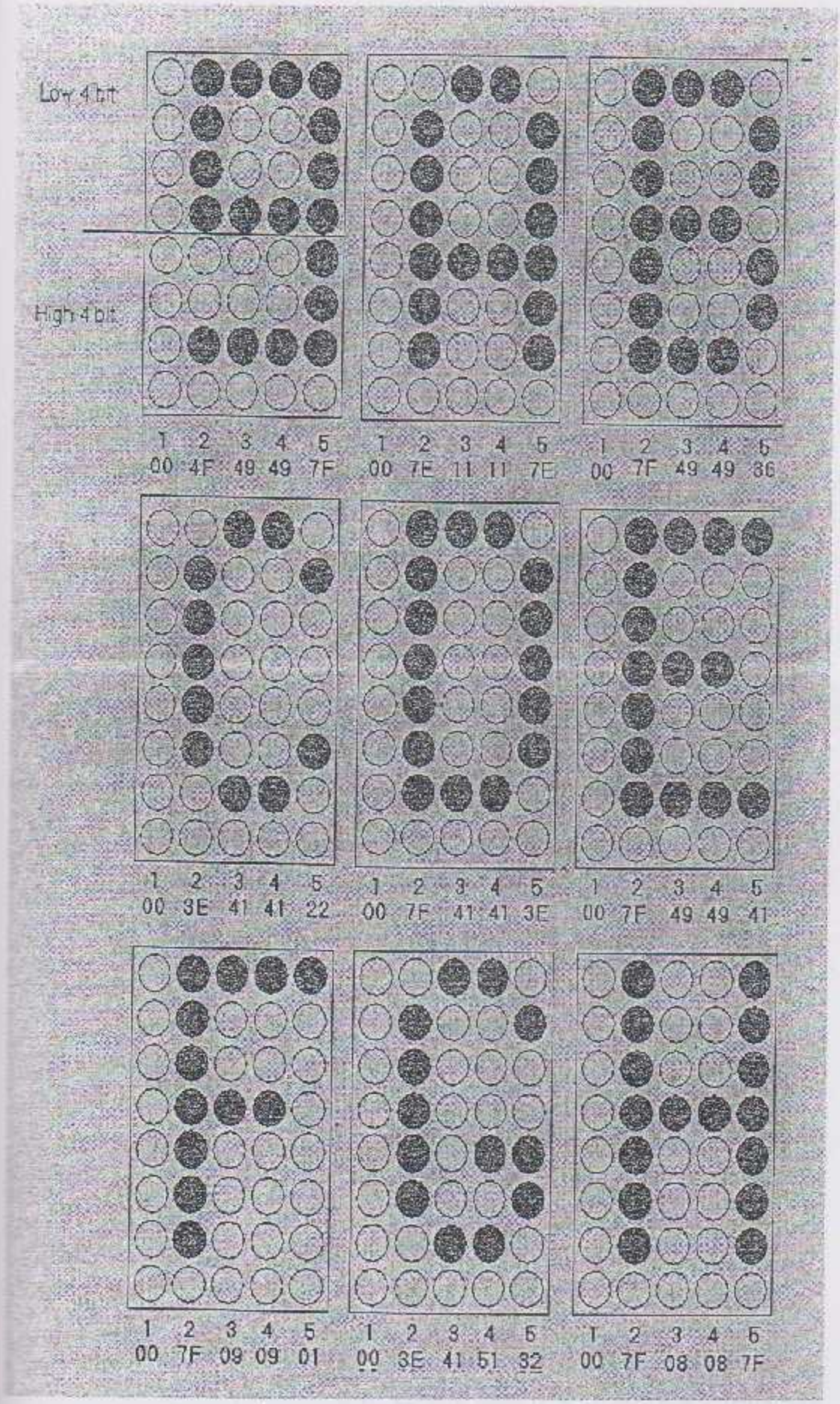
1. Thomas L. Floyd, Electronic Devices, (5th Ed.).Prentice-Hall, Inc. Upper saddle River New jersey 07458 USA, 1999.
2. M. Morris Mano, Digital Design, (3rd Ed.). Prentice-Hall (Tom Robbins). Inc. Upper saddle River New jersey 07458 USA, 2002.
3. John lovine, PIC microcontroller project book (2rd ED). Mcgraw-2004

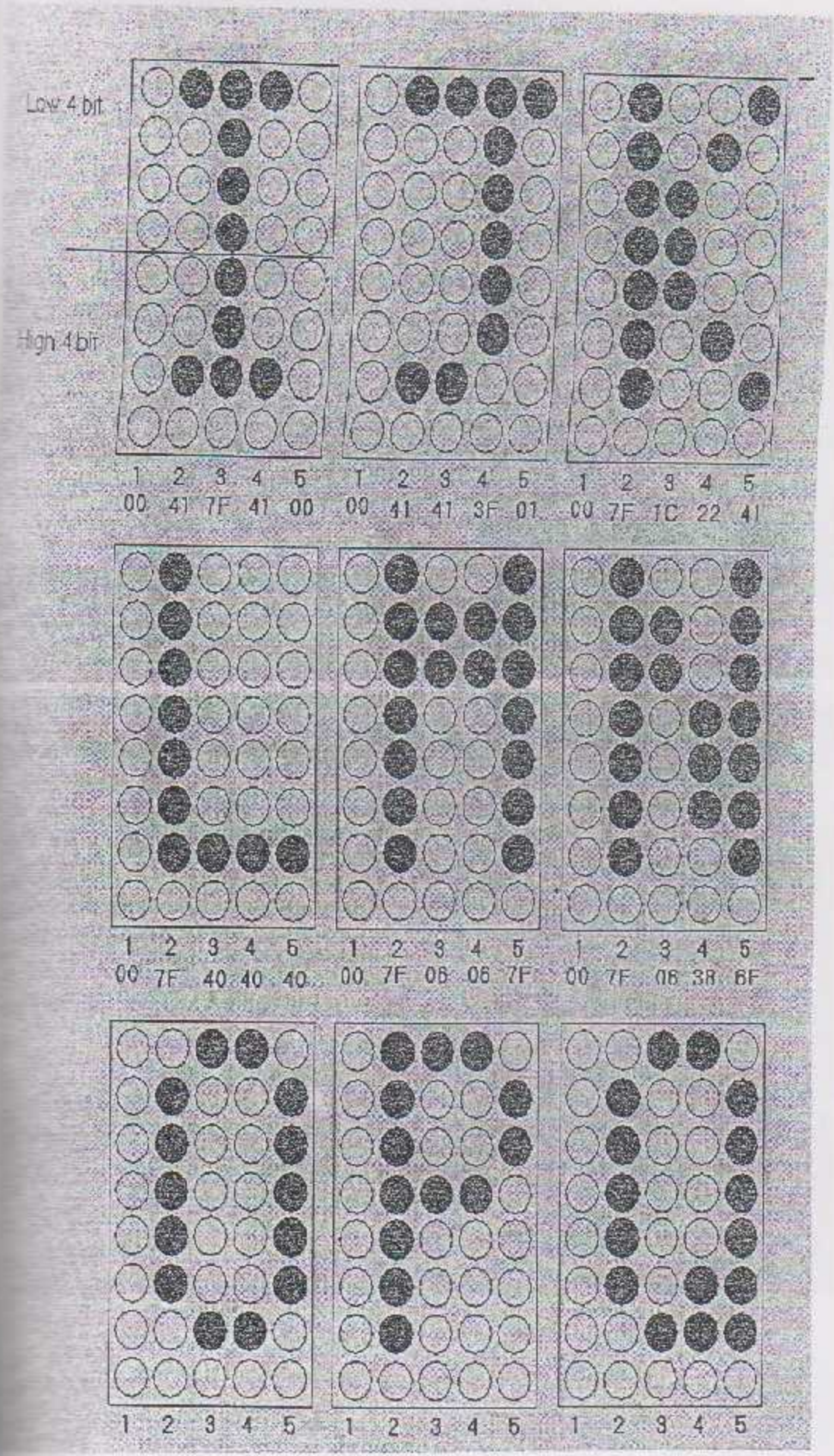
Websites

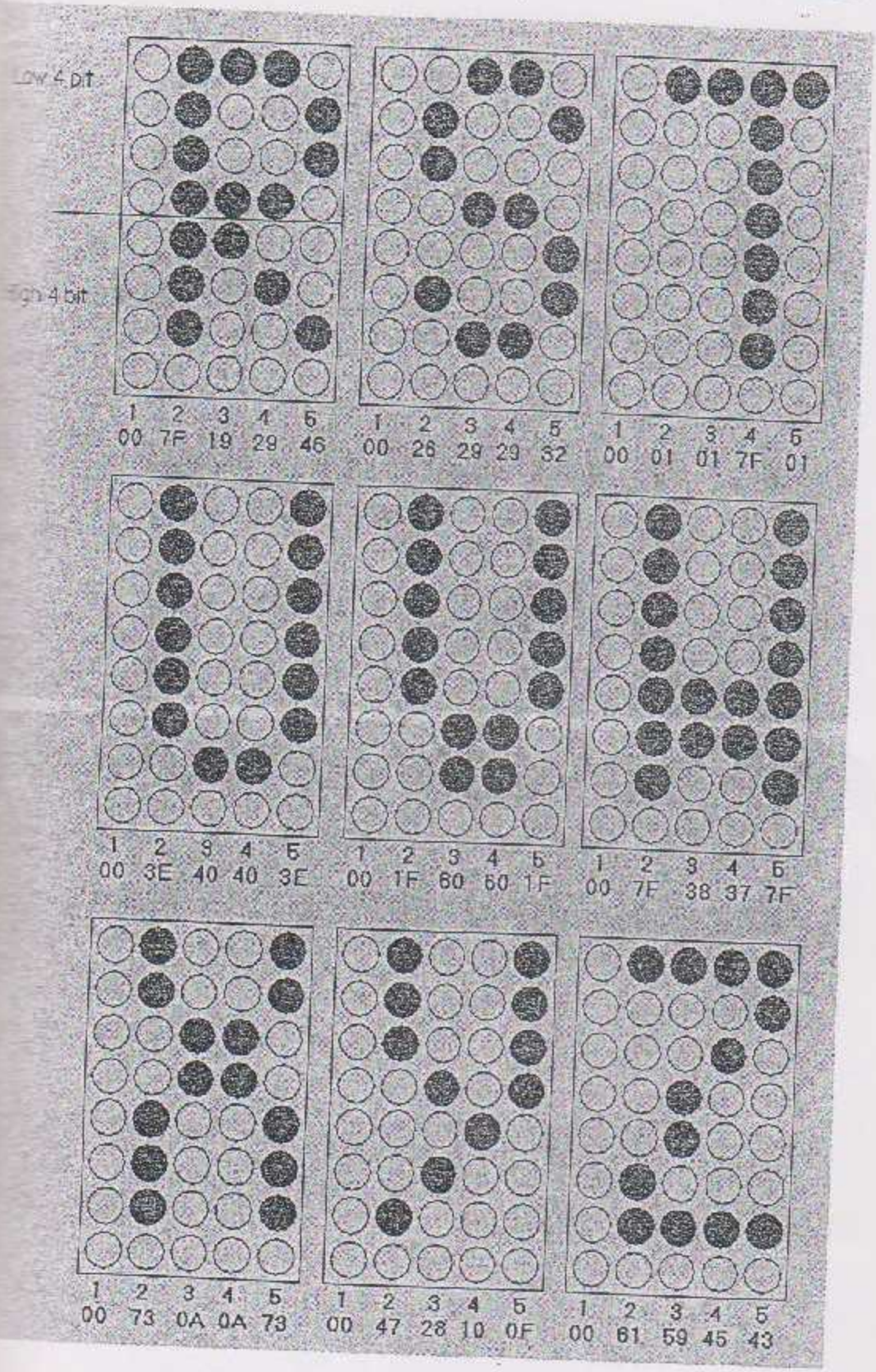
http://www.epanorama.net
http://www.en.wikipedia.org/wiki/Main_Page
http://www.instructables.com
http://www.hyperphysics.phy-astr.
http://www.mywisecowl.com/
http://www.electronic-kits-and-projects.com/kit-files/3000/3029.pdf
http://www.kpsec.freeuk.com/components/led.htm
http://www.quasarelectronics.com/componentid.htm
http://www.avr.4mg.com/custom3.html
http://www.jap.hu/electronic/
http://www.electronickits.com/kit/complete/elec/ek1620.htm
http://www.hobbyprojects.com/
http://www.electronickits.com/
http://www.newdatasheet.com/
http://www.datasheet4u.com/
http://www.alldatasheet.com/

Appendix A


Low 4 bit
High 4 bit
1 2 3 4 5
00 7F 41 41 7F
1 2 3 4 5
00 42 7F 40 00
1 2 3 4 5
00 00 79 49 4F
1 2 3 4 5
00 49 49 49 4F
1 2 3 4 5
00 1E 10 7C 10
1 2 3 4 5
00 4F 49 49 79
1 2 3 4 5
00 7F 49 49 79
1 2 3 4 5
00 01 71 09 07
1 2 3 4 5
00 7F 49 49 7F


Low 4 bit
High 4 bit
1 2 3 4 5
00 4F 49 49 7F
1 2 3 4 5
00 7E 11 11 7E
1 2 3 4 5
00 7F 49 49 36
1 2 3 4 5
00 3E 41 41 22
1 2 3 4 5
00 7F 41 41 3E
1 2 3 4 5
00 7F 49 49 41
1 2 3 4 5
00 7F 09 09 01
1 2 3 4 5
00 3E 41 51 32
1 2 3 4 5
00 7F 08 08 7F


Low 4 bit
High 4 bit
1 2 3 4 5
00 41 7F 41 00
1 2 3 4 5
00 41 41 3F 01
1 2 3 4 5
00 7F 1C 22 41
1 2 3 4 5
00 7F 40 40 40
1 2 3 4 5
00 7F 06 06 7F
1 2 3 4 5
00 7F 06 3B 6F
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5


Low 4 bit
High 4 bit
1 2 3 4 5
00 7F 19 29 46
1 2 3 4 5
00 26 29 29 32
1 2 3 4 5
00 01 01 7F 01
1 2 3 4 5
00 3E 40 40 3E
1 2 3 4 5
00 1F 60 60 1F
1 2 3 4 5
00 7F 38 37 7F
1 2 3 4 5
00 73 0A 0A 73
1 2 3 4 5
00 47 28 10 0F
1 2 3 4 5
00 61 59 45 43

Appendix B

: Writing the final program of PC

: Control of Dot Matrix Display (CDMD)

```
#include<conio.h>
#include<stdio.h>
#include<dos.h>
#define cp 0x37A                    // define address of data
#define dp 0x378                   // define address of control
main()
{
int i,j;
char s[16];                          //number of chrar
scanf ("%s",s);                     //read of chrar

outport(cp,0);                       // reset of counter

    while  (1)
    {
    for (j=0;j<16;++j)               //  send all char
    {
    if (s[j]=='A') {                // display char A

    for (int i=0;i<400;++i)          // delay
    outport(dp,0x7e);               // send data of coulmn 1
    outport(cp,7);                  // send clk
    outport(cp,3);                   // send rest

    for (i=0;i<400;++i)              // delay
    outport(dp,0x11);               // send data of coulmn 2

    outport(cp,7);                  // send clk
    outport(cp,3);                  // send rest


    for (i=0;i<400;++i)             // delay
    outport(dp,0x11);               // send data of coulmn 3
    outport(cp,7);                  // send clk
    outport(cp,3);                 // send rest
    for (i=0;i<400;++i)             // delay
    outport(dp,0x7e);               // send data of coulmn 4
    outport(cp,7);                  // send clk
    outport(cp,3);                 // send rest
    }
```

```
if (s[j]=='B')// display char B same method display char A

    {
    for (int i=0;i<400;++i)
    outport(dp,0x7f);
    outport(cp,7);
    outport(cp,3);

    for (i=0;i<400;++i)
    outport(dp,0x49);
    outport(cp,7);
    outport(cp,3);

    for (i=0;i<400;++i)
    outport(dp,0x49);
    outport(cp,7);
    outport(cp,3);

    for (i=0;i<400;++i)
    outport(dp,0x36);
    outport(cp,7);
    outport(cp,3);
     }

if (s[j]=='C') //display char B same method display char A
     {
     for (i=0;i<200;++i)
     outport(dp,0x3e);
     outport(cp,7);
     outport(cp,3);

     for (i=0;i<200;++i)
     outport(dp,0x41);
     outport(cp,7);
     outport(cp,3);

     for (i=0;i<200;++i)
     outport(dp,0x41);
     outport(cp,7);
     outport(cp,3);

     for (i=0;i<200;++i)
     outport(dp,0x22);
     outport(cp,7);
     outport(cp,3);

     }
```

```
if (s[j]=='D') //display char B same method display char A
      {
      for (i=0;i<200;++i)
      outport(dp,0x7f);
      outport(cp,7);
      outport(cp,3);

      for (i=0;i<200;++i)
      outport(dp,0x41);
      outport(cp,7);
      outport(cp,3);

      for (i=0;i<200;++i)
      outport(dp,0x41);
      outport(cp,7);
      outport(cp,3);

      for (i=0;i<200;++i)
      outport(dp,0x3e);
      outport(cp,7);
      outport(cp,3);


      }

if (s[j]=='E') //display char B same method display char A
      {
      for (i=0;i<200;++i)
      outport(dp,0x7f);
      outport(cp,7);
      outport(cp,3);

      for (i=0;i<200;++i)
      outport(dp,0x49);
      outport(cp,7);
      outport(cp,3);

      for (i=0;i<200;++i)
      outport(dp,0x49);
      outport(cp,7);
      outport(cp,3);

      for (i=0;i<200;++i)
      outport(dp,0x41);
      outport(cp,7);
      outport(cp,3);
      }
```

```
if (s[j]=='F') //display char B same method display char A
      {
      for (i=0;i<10;-+i)
      outport(dp,0x00);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)
      outport(dp,0x7F);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)
      outport(dp,0x09);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;-+i)
      outport(dp,0x09);
      outport(cp,7);
      outport(cp,3);

          }


if (s[j]=='G') //display char B same method display char A
      {
      for (i=0;i<200;-+i)
      outport(dp,0x3c);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;-+i)
      outport(dp,0x41);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;+-i)
      outport(dp,0x51);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)
      outport(dp,0x32);
      outport(cp,7);
      outport(cp,3);

          }

if (s[j]=='H')//display char B same method display char A
      {
      for (i=0;i<200;++i)
      outport(dp,0x7f);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;+i)
      outport(dp,0x08);
```

```
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)
        outport(dp,0x08);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)
        outport(dp,0x7f);
        outport(cp,7);
        outport(cp,3);

            }


if (s[j]=='I') //display char B same method display char A
        {
        for (i=0;i<200;++i)
        outport(dp,0x41);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)
        outport(dp,0x7f);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)
        outport(dp,0x41);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)
        outport(dp,0x00);
        outport(cp,7);
        outport(cp,3);

            }


if (s[j]=='J') //display char B same method display char A
        {
        for (i=0;i<200;++i)
        outport(dp,0x41);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)
        outport(dp,0x41);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)
        outport(dp,0x3f);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)
```

```
        outport(dp,0x01);
        outport(cp,7);
        outport(cp,3);

            }


if (s[j]=='K') //display char B same method display char A
        {
        for (i=0;i<200;++i)
        outport(dp,0x7f);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;-+i)
        outport(dp,0x1c);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)
        outport(dp,0x22);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)
        outport(dp,0x41);
        outport(cp,7);
        outport(cp,3);

            }


if (s[j]=='L') //display char B same method display char A
        {
        for (i=0;i<200;++i)
        outport(dp,0x7f);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;+-i)
        outport(dp,0x40);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;+-i)
        outport(dp,0x40);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)
        outport(dp,0x40);
        outport(cp,7);
        outport(cp,3);

            }
```

```
if (s[j]=='M') //display char B same method display char A
      {
      for (i=0;i<200;++i)
      outport(dp,0x7f);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)
      outport(dp,0x02);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)
      outport(dp,0x02);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)
      outport(dp,0x7f);
      outport(cp,7);
      outport(cp,3);

          }


if (s[j]=='N') //display char B same method display char A
      {
      for (i=0;i<200;++i)
      outport(dp,0x7e);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)
      outport(dp,0x10);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)
      outport(dp,0x20);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)
      outport(dp,0x7e);
      outport(cp,7);
      outport(cp,3);

          }


if (s[j]=='O') //display char B same method display char A
      {
      for (int i=0;i<200;++i)
      outport(dp,0x3e);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)
```

```
        outport(dp,0x41);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)

        outport(dp,0x41);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)

        outport(dp,0x3e);
        outport(cp,7);
        outport(cp,3);
         }

if (s[j]=='P') //display char B same method display char A
        {
        for (int i=0;i<200;++i)
        outport(dp,0xef);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)

        outport(dp,0x09);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)

        outport(dp,0x09);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)

        outport(dp,0x06);
        outport(cp,7);
        outport(cp,3);
         }

if (s[j]=='Q') //display char B same method display char A
        {
        for (int i=0;i<200;++i)
        outport(dp,0x3e);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)

        outport(dp,0x41);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)
```

```
        outport(dp,0x61);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)

        outport(dp,0x7c);
        outport(cp,7);
        outport(cp,3);
         }

if (s[j]=='R') //display char R same method display char A
      {
      for (int i=0;i<200;++i)
      outport(dp,0x7f);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)

      outport(dp,0x19);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)

      outport(dp,0x29);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)

      outport(dp,0x46);
      outport(cp,7);
      outport(cp,3);
       }

if (s[j]=='S') //display char S same method display char A
      {
      for (int i=0;i<200;++i)
      outport(dp,0x26);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)

      outport(dp,0x29);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)

      outport(dp,0x29);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)
```

```
        outport(dp,0x32);
        outport(cp,7);
        outport(cp,3);
         }

if (s[j]=='T') //display char B same method display char A
        {
        for (int i=0;i<200;++i)
        outport(dp,0x01);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)

        outport(dp,0x01);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)

        outport(dp,0x7f);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)

        outport(dp,0x01);
        outport(cp,7);
        outport(cp,3);
         }

if (s[j]=='U') //display char B same method display char A
        {
        for (int i=0;i<200;++i)
        outport(dp,0x3f);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)

        outport(dp,0x40);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)

        outport(dp,0x40);
        outport(cp,7);
        outport(cp,3);
        for (i=0;i<200;++i)

        outport(dp,0x3f);
        outport(cp,7);
        outport(cp,3);
```

```
if (s[j]=='V') //display char B same method display char A
     {
     for (int i=0;i<200;++i)
     outport(cp,0x1f);
     outport(cp,7);
     outport(cp,3);
     for (i=0;i<200;++i)

     outport(dp,0x40);
     outport(cp,7);
     outport(cp,3);
     for (i=0;i<200;++i)

     outport(dp,0x40);
     outport(cp,7);
     outport(cp,3);
     for (i=0;i<200;++i)

     outport(dp,0x1f);
     outport(cp,7);
     outport(cp,3);
      }

if (s[j]=='W') //display char B same method display char A
     {
     for (int i=0;i<200;++i)
     outport(dp,0x7e);
     outport(cp,7);
     outport(cp,3);
     for (i=0;i<200;++i)

     outport(dp,0x20);
     outport(cp,7);
     outport(cp,3);
     for (i=0;i<200;++i)

     outport(dp,0x20);
     outport(cp,7);
     outport(cp,3);
     for (i=0;i<200;++i)

     outport(dp,0x7e);
     outport(cp,7);
     outport(cp,3);
      }
      if (s[j]=='X')
     {
     for (int i=0;i<200;++i)
     outport(dp,0x73);
     outport(cp,7);
```

```
      outport(cp,3);
      for (i=0;i<200;++i)

      outport(dp,0x0c);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)

      outport(dp,0x0c);
      outport(cp,7);
      outport(cp,3);
      for (i=0;i<200;++i)

      outport(dp,0x73);
      outport(cp,7);
      outport(cp,3);
       }

if (s[j]=='Y') //display char B same method display char A
     {
     for (int i=0;i<200;++i)
     outport(dp,0x47);
     outport(cp,7);
     outport(cp,3);
     for (i=0;i<200;++i)

     outport(dp,0x28);
     outport(cp,7);
     outport(cp,3);
     for (i=0;i<200;++i)

     outport(dp,0x10);
     outport(cp,7);
     outport(cp,3);
     for (i=0;i<200;++i)

     outport(dp,0x0f);
     outport(cp,7);
     outport(cp,3);
      }

if (s[j]=='Z') //display char B same method display char A
     {
     for (int i=0;i<200;++i)
     outport(dp,0x71);
     outport(cp,7);
     outport(cp,3);
     for (i=0;i<200;++i)

     outport(dp,0x49);
     outport(cp,7);
```

```
 outport(cp,3);
 for (i=0;i<200;++i)

 outport(dp,0x45);
 outport(cp,7);
 outport(cp,3);
 for (i=0;i<200;++i)

outport(dp,0x43);
outport(cp,7);
outport(cp,3);
 }

 }
outport(cp,0);                    // reset of counter

}
```

Appendix C

Program of PIC

```
      LIST p=16f84a
      #include p16f84a.inc
      __CONFIG _XT_OSC & _CP_OFF & _WDT_OFF & _PWRTE_OFF



V1          EQU         0X0C              ;counter
V2          EQU         0X0D              ;counter
V3          EQU         0X0E              ;counter

            ORG         0X00 ; start origin 00 in memmory

            BSF         STATUS,5
            CLRF TRISB                    ; define PB output
            CLRF TRISA
            BSF         TRISA,2           ;  define PA2 input
            BCF         STATUS,5
            BSF         PORTA,0           ; define PA0 input
            BCF         PORTA,1           ;define PA1 input
LOPTI
            ;;;;;;;;;;;;;;;;;;;;;;;;      ; Display chara P
            ;;;;;;;;;;;;;;;;;;;;;;;;
            MOVLW       0X7F        ; load 7f in rigestar w
            MOVWF       PORTB       ; send data on PB
            CALL DELY0              ; delay
            CALL CLK                ; send clk
            MOVLW       0X09        ;load 09 in rigestar w
            MOVWF       PORTB       ; send data on PB
            CALL DELY0              ; delay
            CALL CLK                ; send clk
            MOVLW       0X09        ;load 09 in rigestar w
            MOVWF       PORTB       ; send data on PB
            CALL DELY0              ; delay
            CALL CLK                ; send clk
            MOVLW       0X06        ;load 06 in rigestar w
            MOVWF       PORTB       ; send data on PB

            CALL DELY0              ; delay

            CALL CLK                ; send clk
```

```
; display char I same method display char P

        MOVLW     0X41
        MOVWF     PORTB
        CALL DELY0
        CALL CLK
        MOVLW     0X7F
        MOVWF     PORTB
        CALL DELY0
        CALL CLK
        MOVLW     0X41
        MOVWF     PORTB
        CALL DELY0
        CALL CLK
        MOVLW     0X00
        MOVWF     PORTB
        CALL DELY0
        CALL CLK
        ;;;;;;;;;;
        BTFSC     PORTA,2
        GOTO TL

        ; display char C same method display char P

        MOVLW     0X3E
        MOVWF     PORTB
        CALL DELY0
        CALL CLK
        MOVLW     0X41
        MOVWF     PORTB
        CALL DELY0
        CALL CLK
        MOVLW     0X41
        MOVWF     PORTB
        CALL DELY0
        CALL CLK
        MOVLW     0X22
        MOVWF     PORTB
        CALL DELY0
        CALL CLK

        ; display space  same method display char P

        MOVLW     0X0
        MOVWF     PORTB
        CALL DELY0
        CALL CLK
        MOVLW     0X0
        MOVWF     PORTB
        CALL DELY0
        CALL CLK
```

```
MOVLW     0X0
MOVWF     PORTB
CALL DELY0
CALL CLK
MOVLW     0X0
MOVWF     PORTB
CALL DELY0
CALL CLK
; display char T same method display char P

MOVLW     0X01
MOVWF     PORTB
CALL DELY0
CALL CLK
MOVLW     0X01
MOVWF     PORTB
CALL DELY0
CALL CLK
MOVLW     0X7F
MOVWF     PORTB
CALL DELY0
CALL CLK
MOVLW     0X01
MOVWF     PORTB
CALL DELY0
CALL CLK
; display char E same method display char P

MOVLW     0X7F
MOVWF     PORTB
CALL DELY0
CALL CLK
MOVLW     0X49
MOVWF     PORTB
CALL DELY0
CALL CLK
MOVLW     0X49
MOVWF     PORTB
CALL DELY0
CALL CLK
MOVLW     0X41
MOVWF     PORTB
CALL DELY0
CALL CLK
; display char S same method display char P

MOVLW     0X26
MOVWF     PORTB
CALL DELY0
CALL CLK
MOVLW     0X49
```

```
        MOVWF   PORTB
        CALL DELY0
        CALL CLK
        MOVLW   0X49
        MOVWF   PORTB
        CALL DELY0
        CALL CLK
        MOVLW   0X32
        MOVWF   PORTB
        CALL DELY0
        CALL CLK
        ; display char T same method display char P

        MOVLW   0X01
        MOVWF   PORTB
        CALL DELY0
        CALL CLK
        MOVLW   0X01
        MOVWF   PORTB
        CALL DELY0
        CALL CLK
        MOVLW   0X7F
        MOVWF   PORTB
        CALL DELY0
        CALL CLK
        MOVLW   0X01
        MOVWF   PORTB
        CALL DELY0
        CALL CLK
TL      ;;;;;;;;;;;;;;      ; rest the counter
        BSF     PORTA,1         ; send clk of PA1
        BCF     PORTA,1         ; send rest of PA1

        GOTO LOPTI          ; delay

        ;;;;;;;;;;;;;;      ; clk the counter


CLK     BCF     PORTA,0         ; send rest of PA0
        NOP
        BSF     PORTA,0         ; send clk of PA0
        RETURN

        ;;;  ; delay between two succissive data

DELY0   MOVLW   D'15'
        MOVWF   V2
LOOP1   MOVWF   0XFF
        MOVWF   V3
LOOP1   DECFSZ  V3,1
        GOTO LOOP1
```

```
DECFSZ    V2,1
GOTO LOOPI
RETURN

END
```

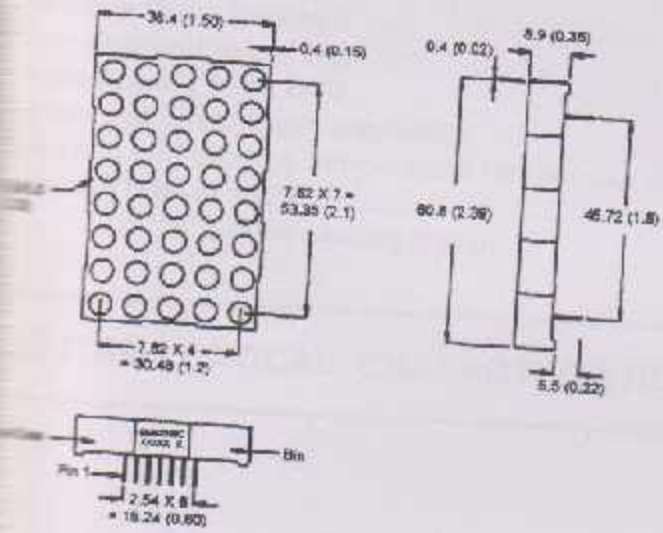
Appendix D

FAIRCHILD
SEMICONDUCTOR™

2.3 INCH (58.4 mm) 5 X 8 DOT MATRIX STICK DISPLAY

AlGaAs Red GMA2285C
AlGaAs Red GMC2285C

PACKAGE DIMENSIONS



Dimensions are in mm (inch).
Tolerances are ± 0.25 (0.1) unless otherwise noted.
All pins are 0.5 (.02).

DESCRIPTION

The GMX2285C 5 X 8, Single Hetero Junction AlGaAs Red dot matrix display. It has a grey face with neutral segment color.

FEATURES

2.5" (58.4mm) character height.
Low power requirement.
Wide 130° viewing angle.
High brightness and contrast
5 X 8 array with X-Y select.
X-Y stackable.
Easy mounting on P.C. board.

MODEL NUMBER

Part Number	Colour	Description
GMA2285C	AlGaAs Red	Common anode row.
GMC2285C	AlGaAs Red	Common Cathode row.

(For other color options, contact your local area Sales Office)

FAIRCHILD
SEMICONDUCTOR™

2.3 INCH (58.4 mm) 5 X 8 DOT MATRIX STICK DISPLAY

ABSOLUTE MAXIMUM RATING (T_A = 25°C unless otherwise specified)

	AlGaAs Red	Units
ak forward current per segment (Duty cycle 1/10, 10KHz)	200	mA
inous IF per segment	30	mA
er dissipation per segment	100*	mW
rate linearly from 25°C	0.5	mW/°C
erse voltage VR per segments	5	Volts
erating and storage temperature range		-25°C to +85°C
dering time at 260°C (1/16" below seating plane)		3 sec

ECTRO - OPTICAL CHARACTERISTICS (T_A = 25°C unless otherwise specified)

	AlGaAs Red	Test Condition
inous Intensity/Dot	5000ucd	I_F = 20mA
average (Typical)		
ard voltage (V_F)		
typical	1.8V	I_F = 20 mA
maximum	2.5V	I_F = 20 mA
wavelength (nm)	660nm	I_F = 20 mA
ctral line half width (nm)	20nm	I_F = 20mA
rse breakdown voltage V_R	5V	I_R = 100uA

FAIRCHILD

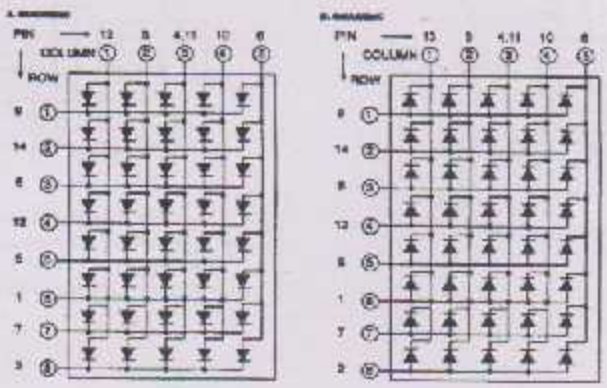
SEMICONDUCTOR™

2.3 INCH (58.4 mm) 5 X 8 DOT MATRIX STICK DISPLAY

PIN CONNECTION:

GMA2285C		GMC2285C	
Pin Number	Function	Pin Number	Function
1	Anode Row 6	1	Cathode Row 6
2	Anode Row 8	2	Cathode Row 8
3	Cathode Column 2	3	Anode Column 2
4	Cathode Column 3	4	Anode Column 3
5	Anode Row 5	5	Cathode Row 5
6	Cathode Column 5	6	Anode Column 5
7	Anode Row 7	7	Cathode Row 7
8	Anode Row 3	8	Cathode Row 3
9	Anode Row 1	9	Cathode Row 1
10	Cathode Column 4	10	Anode Column 4
11	Cathode Column 3	11	Anode Column 3
12	Anode Row 4	12	Cathode Row 4
13	Cathode Column 1	13	Anode Column 1
14	Anode Row 2	14	Cathode Row 2

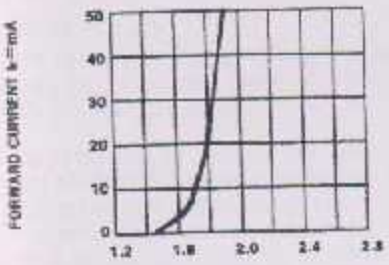
SCHEMATIC:



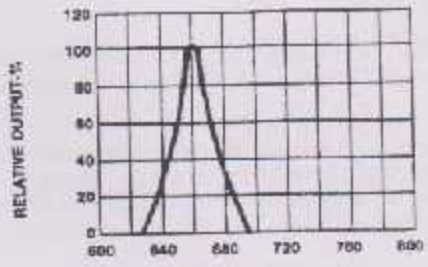
FAIRCHILD
SEMICONDUCTOR™

2.3 INCH (58.4 mm) 5 X 8 DOT MATRIX STICK DISPLAY

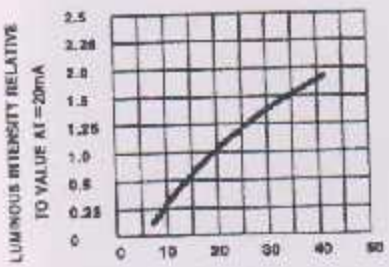
RAPHICAL DETAIL: AlGaAs Red (T_A = 25°C unless otherwise specified)



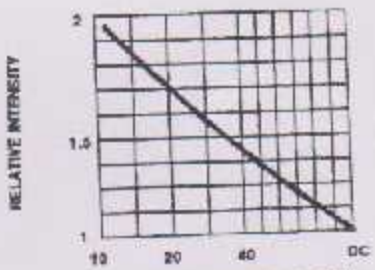
FORWARD VOLTAGE (VF)-VOLTS
Fig.1 FORWARD CURRENT VS. FORWARD VOLTAGE.



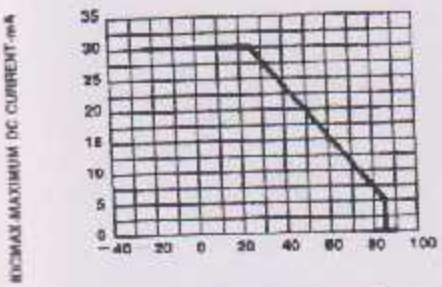
WAVELENGTH (λ)-nm
Fig.2 SPECTRAL RESPONSE



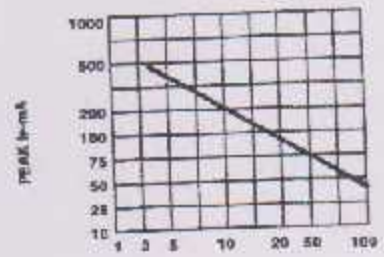
IF-FORWARD CURRENT-mA
Fig.3 RELATIVE LUMINOUS INTENSITY VS. FORWARD CURRENT



DUTY CYCLE % PER SEGMENT
(AVERAGE IF=10mA)
Fig.5 LUMINOUS INTENSITY VS. DUTY CYCLE



TA AMBIENT TEMPERATURE °C
Fig.4 MAXIMUM ALLOWABLE DC CURRENT PER SEGMENT VS. A FUNCTION OF AMBIENT TEMPERATURE.



DUTY CYCLE %
Fig. 6 MAX PEAK CURRENT VS. DUTY CYCLE %
(REFRESH RATE f=1 KHz)

FAIRCHILD

SEMICONDUCTOR™

2.3 INCH (58.4 mm) 5 X 8 DOT MATRIX STICK DISPLAY

CLAIMER

CHILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO PRODUCTS HEREIN TO IMPROVE RELIABILITY, FUNCTION OR DESIGN. FAIRCHILD DOES NOT ASSUME BILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN; HER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS.

SUPPORT POLICY

CHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES STEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF FAIRCHILD SEMICONDUCTOR PORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury of the user.
2. A critical component in any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

© 2000 Fairchild Semiconductor Corporation

MICROCHIP

PIC16F84A

18-pin *Enhanced* Flash/EEPROM 8-Bit Microcontroller

Devices Included in this Data Sheet:

- PIC16F84A
- Extended voltage range device available (PIC16LF84A)

High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- 1024 words of program memory
- 68 bytes of data RAM
- 64 bytes of data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 special function hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt on change
  - Data EEPROM write complete

Peripheral Features:

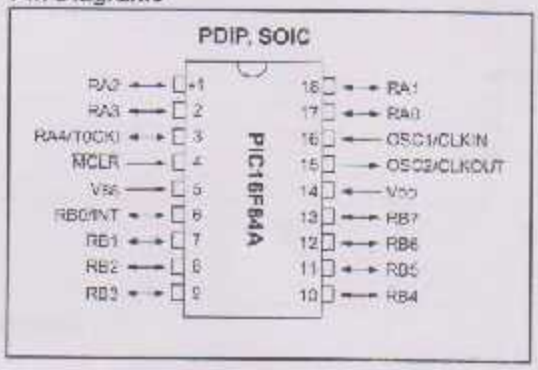
- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Special Microcontroller Features:

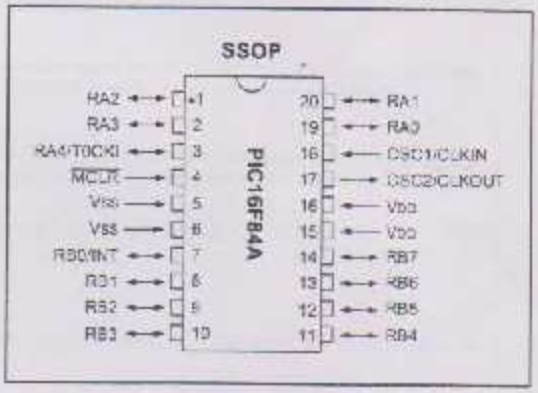
- 1000 erase/write cycles *Enhanced* Flash program memory
- 1,000,000 typical erase/write cycles EEPROM data memory
- EEPROM Data Retention > 40 years
- In-Circuit Serial Programming (ICSP™) - via two pins
- Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Code-protection
- Power saving SLEEP mode
- Selectable oscillator options

Pin Diagrams

PDIP, SOIC



SSOP



CMOS *Enhanced* Flash/EERPOM Technology:

- Low-power, high-speed technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.0V to 5.5V
  - Industrial: 2.0V to 5.5V
- Low power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 15 µA typical @ 2V, 32 kHz
  - < 0.5 µA typical standby current @ 2V

© 1998 Microchip Technology Inc.

Table of Contents

To Our Valued Customers

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please check our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, e.g., DS30000A is version A of document DS30000.

Errata

An errata sheet may exist for current devices, describing minor operational differences (from the data sheet) and recommended workarounds. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (602) 786-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Corrections to this Data Sheet

We constantly strive to improve the quality of all our products and documentation. We have spent a great deal of time to ensure that this document is correct. However, we realize that we may have missed a few things. If you find any information that is missing or appears in error, please:

- Fill out and mail in the reader response form in the back of this data sheet.
- E-mail us at webmaster@microchip.com.

We appreciate your assistance in making this a better document.

 © 1998 Microchip Technology Inc.

1.0 DEVICE OVERVIEW

This document contains device-specific information for the operation of the PIC16F84A device. Additional information may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023), which may be downloaded from the Microchip website. The Reference Manual should be considered a complementary document to this data sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

The PIC16F84A belongs to the mid-range family of the PICmicro™ microcontroller devices. A block diagram of the device is shown in Figure 1-1.

The program memory contains 1K words, which translates to 1024 instructions, since each 14-bit program memory word is the same width as each device instruction. The data memory (RAM) contains 68 bytes. Data EEPROM is 64 bytes.

There are also 13 I/O pins that are user-configured on a pin-to-pin basis. Some pins are multiplexed with other device functions. These functions include:

- External interrupt
- Change on PORTB interrupt
- Timer0 clock input

Table 1-1 details the pinout of the device with descriptions and details for each pin.

FIGURE 1-1: PIC16F84A BLOCK DIAGRAM

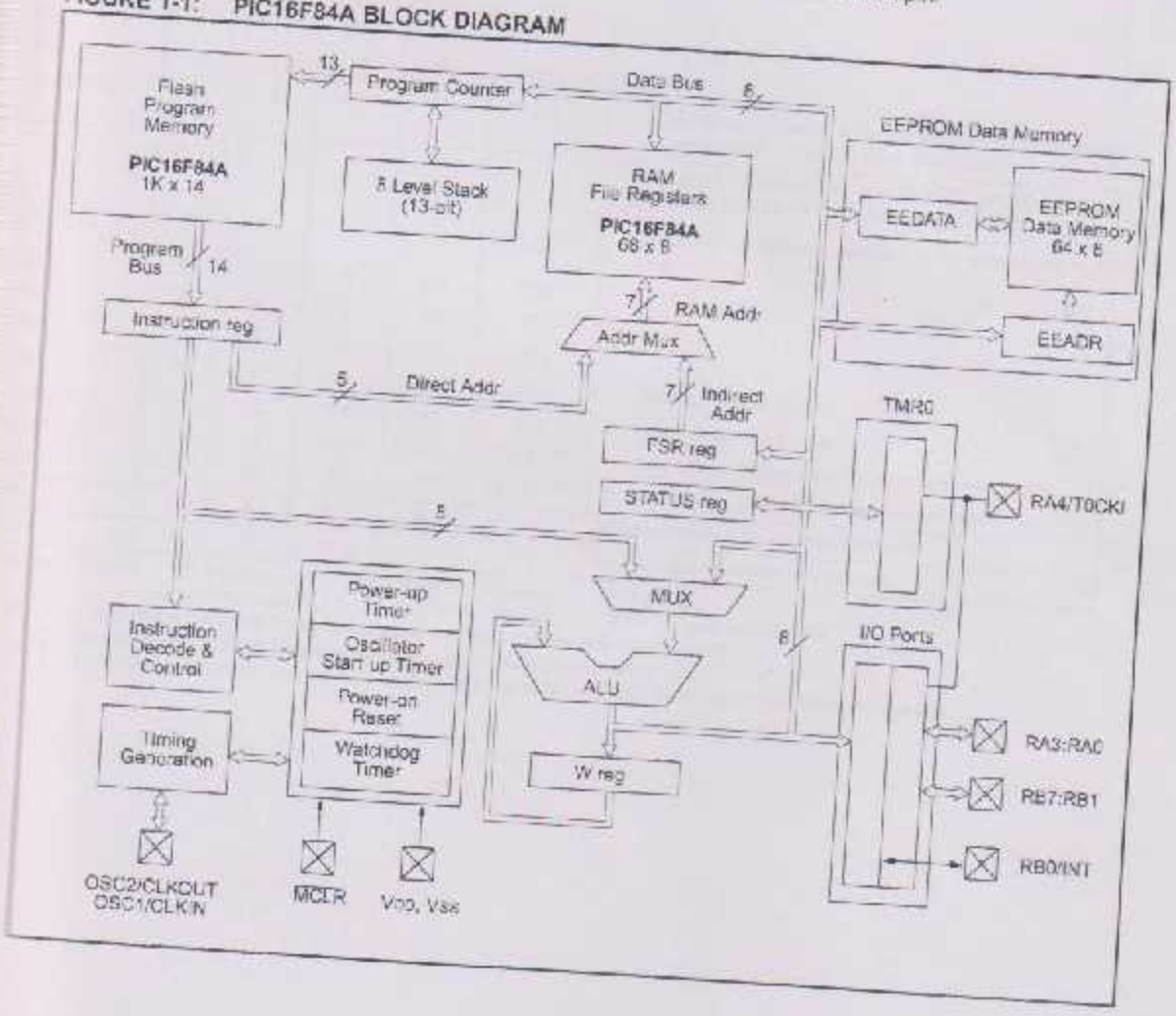


TABLE 1-1 PIC16F84A PINOUT DESCRIPTION

Pin Name	DIP No.	SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	15	I	ST/CMOS (3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	19	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
MCLR	4	4	4	I/P	ST	Master clear (reset) input/programming voltage input. This pin is an active low reset to the device.
						PORTA is a bi-directional I/O port.
RA0	17	17	19	I/O	TTL	
RA1	18	18	20	I/O	TTL	
RA2	1	1	1	I/O	TTL	
RA3	2	2	2	I/O	TTL	
RA4/T0CKI	3	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	7	I/O	TTL/ST (1)	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	8	I/O	TTL	
RB2	8	8	9	I/O	TTL	
RB3	9	9	10	I/O	TTL	
RB4	10	10	11	I/O	TTL	Interrupt on change pin.
RB5	11	11	12	I/O	TTL	Interrupt on change pin.
RB6	12	12	13	I/O	TTL/ST (2)	Interrupt on change pin. Serial programming clock.
RB7	13	13	14	I/O	TTL/ST (2)	Interrupt on change pin. Serial programming data.
VSS	5	5	5,6	P	—	Ground reference for logic and I/O pins.
VDD	14	14	16,15	P	—	Positive supply for logic and I/O pins.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

© 1998 Microchip Technology Inc.

2.0 MEMORY ORGANIZATION

There are two memory blocks in the PIC16F84A. These are the program memory and the data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle.

The data memory can further be broken down into the general purpose RAM and the Special Function Registers (SFRs). The operation of the SFRs that control the "core" are described here. The SFRs used to control the peripheral modules are described in the section discussing each individual peripheral module.

The data memory area also contains the data EEPROM memory. This memory is not directly mapped into the data memory, but is indirectly mapped. That is, an indirect address pointer specifies the address of the data EEPROM memory to read/write. The 64 bytes of data EEPROM memory have the address range 0h-3Fh. More details on the EEPROM memory can be found in Section 5.0.

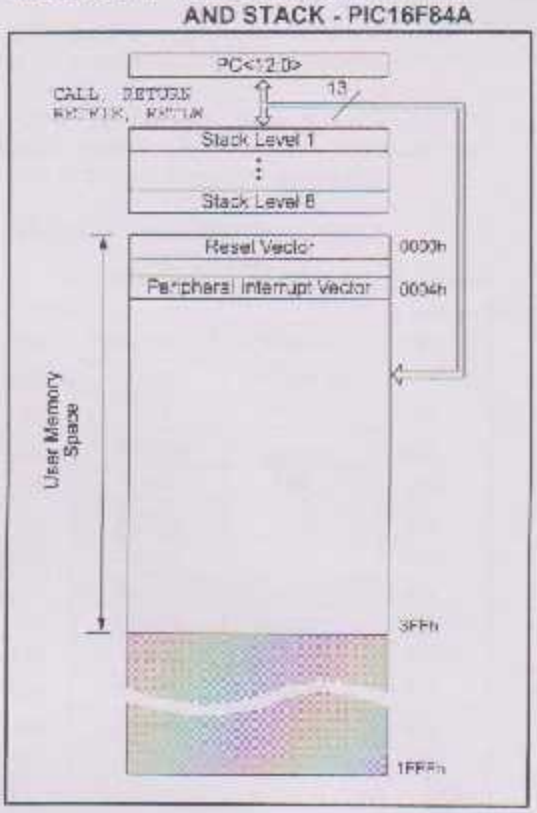
Additional information on device memory may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

2.1 Program Memory Organization

The PIC16FXX has a 13-bit program counter capable of addressing an 8K x 14 program memory space. For the PIC16F84A, the first 1K x 14 (0000h-03FFh) are physically implemented (Figure 2-1). Accessing a location above the physically implemented address will cause a wraparound. For example, for locations 20h, 420h, 820h, C20h, 1020h, 1420h, 1820h, and 1C20h will be the same instruction.

The reset vector is at 0000h and the interrupt vector is at 0004h.

FIGURE 2-1: PROGRAM MEMORY MAP AND STACK - PIC16F84A



2.2 Data Memory Organization

The data memory is partitioned into two areas. The first is the Special Function Registers (SFR) area, while the second is the General Purpose Registers (GPR) area. The SFRs control the operation of the device.

Portions of data memory are banked. This is for both the SFR area and the GPR area. The GPR area is banked to allow greater than 116 bytes of general purpose RAM. The banked areas of the SFR are for the registers that control the peripheral functions. Banking requires the use of control bits for bank selection. These control bits are located in the STATUS Register. Figure 2-1 shows the data memory map organization.

Instructions MOVWF and MOVF can move values from the W register to any location in the register file ("F"), and vice-versa.

The entire data memory can be accessed either directly using the absolute address of each register file or indirectly through the File Select Register (FSR) (Section 2.4). Indirect addressing uses the present value of the RP0 bit for access into the banked areas of data memory.

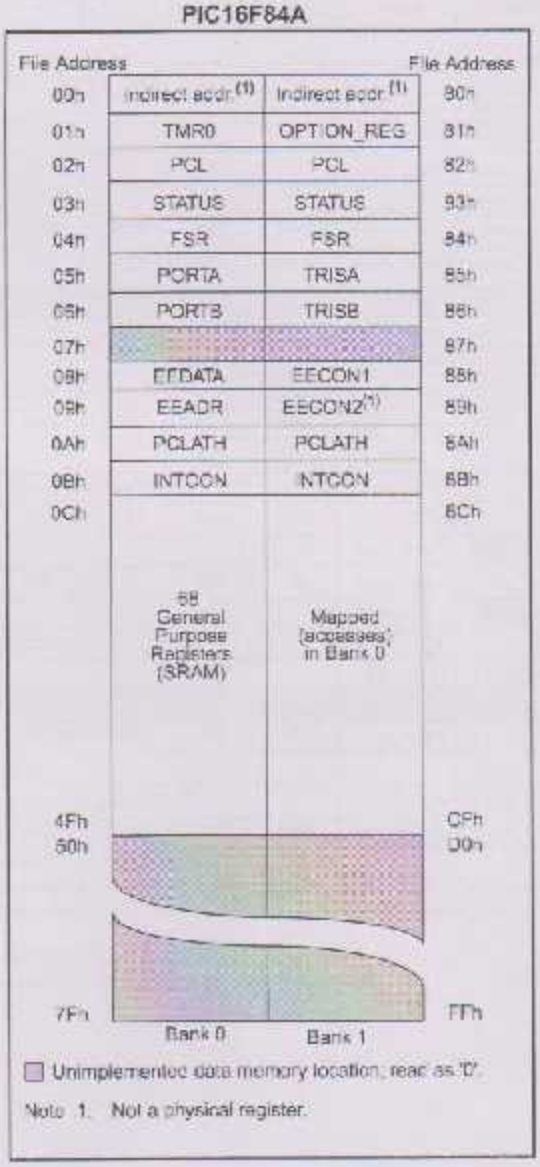
Data memory is partitioned into two banks which contain the general purpose registers and the special function registers. Bank 0 is selected by clearing the RP0 bit (STATUS<5>). Setting the RP0 bit selects Bank 1. Each Bank extends up to 7Fh (128 bytes). The first twelve locations of each Bank are reserved for the Special Function Registers. The remainder are General Purpose Registers implemented as static RAM.

2.2.1 GENERAL PURPOSE REGISTER FILE

Each General Purpose Register (GPR) is 8 bits wide and is accessed either directly or indirectly through the FSR (Section 2.4).

The GPR addresses in bank 1 are mapped to addresses in bank 0. As an example, addressing location 0Ch or 8Ch will access the same GPR.

FIGURE 2-1: REGISTER FILE MAP - PIC16F84A



© 1998 Microchip Technology Inc.

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers (Figure 2-1 and Table 2-1) are used by the CPU and Peripheral functions to control the device operation. These registers are static RAM.

The special function registers can be classified into two sets, core and peripheral. Those associated with the core functions are described in this section. Those related to the operation of the peripheral features are described in the section for that specific feature.

TABLE 2-1 REGISTER FILE SUMMARY

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note3)
Bank 0											
00h	INDF	Uses contents of FSR to address data memory (not a physical register)								---------	---------
01h	TMR0	8-bit real-time clock/counter								xxxx xxxx	uuuu uuuu
02h	PCL	Low order 8 bits of the Program Counter (PC)								0000 0000	0000 0000
03h	STATUS (2)	IRP	RP1	RP0	TO	PD	Z	DC	C	0001 1xxx	000q quuu
04h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
05h	PORTA (4)	---	---	---	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
06h	PORTB (5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
07h		Unimplemented location, read as '0'								---------	---------
08h	EEDATA	EEPROM data register								xxxx xxxx	uuuu uuuu
09h	EEADR	EEPROM address register								xxxx xxxx	uuuu uuuu
0Ah	PCLATH	---	---	---	Write buffer for upper 5 bits of the PC (1)					---x xxxx	---u uuuu
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
Bank 1											
80h	INDF	Uses contents of FSR to address data memory (not a physical register)								---------	---------
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	0000 0000
83h	STATUS (2)	IRP	RP1	RP0	TO	PD	Z	DC	C	0001 1xxx	000q quuu
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	uuuu uuuu
85h	TRISA	---	---	---	PORTA data direction register					---1 1111	---1 1111
86h	TRISB	PORTB data direction register								1111 1111	1111 1111
87h		Unimplemented location, read as '0'								---------	---------
88h	EECON1	---	---	---	EEIF	WRERR	WREN	WR	RD	---0 x000	---0 q000
89h	EECON2	EEPROM control register 2 (not a physical register)								---------	---------
0Ah	PCLATH	---	---	---	Write buffer for upper 5 bits of the PC (1)					---0 0000	---0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q = value depends on condition

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> is never transferred to PCLATH.

2: The TO and PD status bits in the STATUS register are not affected by a MCLR reset.

3: Other (non power-up) resets include: external reset through MCLR and the Watchdog Timer Reset.

4: On any device reset, these pins are configured as inputs.

5: This is the value that will be in the port output latch.

2.2.2.1 STATUS REGISTER

The STATUS register contains the arithmetic status of the ALU, the RESET status and the bank select bit for data memory.

As with any register, the STATUS register can be the destination for any instruction. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, CLRF STATUS will clear the upper-three bits and set the Z bit. This leaves the STATUS register as 000u uluu (where u = unchanged).

Only the BCF, BSF, SWAPF and MOVWF instructions should be used to alter the STATUS register (Table 7-2) because these instructions do not affect any status bit.

Note 1: The IRP and RP1 bits (STATUS<7:6>) are not used by the PIC16F84A and should be programmed as cleared. Use of these bits as general purpose R/W bits is NOT recommended, since this may affect upward compatibility with future products.

Note 2: The C and DC bits operate as a $\overline{borrow}$ and digit $\overline{borrow}$ out bit, respectively, in subtraction. See the SUBLW and SUBWF instructions for examples.

Note 3: When the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. The specified bit(s) will be updated according to device logic.

FIGURE 2-1: STATUS REGISTER (ADDRESS 03h, 83h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **IRP**: Register Bank Select bit (used for indirect addressing)
The IRP bit is not used by the PIC16F84A. IRP should be maintained clear.

bit 6-5: **RP1:RP0**: Register Bank Select bits (used for direct addressing)
00 = Bank 0 (00h - 7Fh)
01 = Bank 1 (80h - FFh)
Each bank is 128 bytes. Only bit RP0 is used by the PIC16F84A. RP1 should be maintained clear.

bit 4: **$\overline{TO}$**: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3: **$\overline{PD}$**: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2: **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1: **DC**: Digit carry/$\overline{borrow}$ bit (for ADDWF and ADDLW instructions) (For $\overline{borrow}$ the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0: **C**: Carry/$\overline{borrow}$ bit (for ADDWF and ADDLW instructions)
1 = A carry-out from the most significant bit of the result occurred
0 = No carry-out from the most significant bit of the result occurred
Note: For $\overline{borrow}$ the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

© 1998 Microchip Technology Inc.

2.2.2.2 OPTION_REG REGISTER

The OPTION_REG register is a readable and writable register which contains various control bits to configure the TMR0/WDT prescaler, the external INT interrupt, TMR0, and the weak pull-ups on PORTB.

Note: When the prescaler is assigned to the WDT (PSA = '1'), TMR0 has a 1:1 prescaler assignment.

FIGURE 2-1: OPTION_REG REGISTER (ADDRESS 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **RBPU**: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled (by individual port latch values)

bit 6: **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5: **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4: **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3: **PSA**: Prescaler Assignment bit
1 = Prescaler assigned to the WDT
0 = Prescaler assigned to TMR0

bit 2-0: **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

2.2.2.3 INTCON REGISTER

The INTCON register is a readable and writable register which contains the various enable bits for all interrupt sources.

Note: Interrupt flag bits get set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>).

FIGURE 2-1: INTCON REGISTER (ADDRESS 0Bh, 8Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, read as '0'
- n = Value at POR reset

bit 7: **GIE**: Global Interrupt Enable bit
1 = Enables all un-masked interrupts
0 = Disables all interrupts

Note: For the operation of the interrupt structure, please refer to Section *.

bit 6: **EEIE**: EE Write Complete Interrupt Enable bit
1 = Enables the EE write complete interrupt
0 = Disables the EE write complete interrupt

bit 5: **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4: **INTE**: RB0/INT Interrupt Enable bit
1 = Enables the RB0/INT interrupt
0 = Disables the RB0/INT interrupt

bit 3: **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2: **T0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 has overflowed (must be cleared in software)
0 = TMR0 did not overflow

bit 1: **INTF**: RB0/INT Interrupt Flag bit
1 = The RB0/INT interrupt occurred
0 = The RB0/INT interrupt did not occur

bit 0: **RBIF**: RB Port Change Interrupt Flag bit
1 = When at least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

© 1998 Microchip Technology Inc.

2.3 PCL and PCLATH

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 13 bits wide. The low byte is called the PCL register. This register is readable and writable. The high byte is called the PCH register. This register contains the PC<12:8> bits and is not directly readable or writable. All updates to the PCH register go through the PCLATH register.

2.3.1 STACK

The stack allows a combination of up to 8 program calls and interrupts to occur. The stack contains the return address from this branch in program execution.

Midrange devices have an 8 level deep x 13-bit wide hardware stack. The stack space is not part of either program or data space and the stack pointer is not readable or writable. The PC is PUSHed onto the stack when a `CALL` instruction is executed or an interrupt causes a branch. The stack is POPed in the event of a `RETURN`, `RETLW` or a `RETFIE` instruction execution. PCLATH is not modified when the stack is PUSHed or POPed.

After the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

2.4 Indirect Addressing; INDF and FSR Registers

The INDF register is not a physical register. Addressing INDF actually addresses the register whose address is contained in the FSR register (FSR is a *pointer*). This is indirect addressing.

EXAMPLE 2-1: INDIRECT ADDRESSING

- Register file 05 contains the value 10h
- Register file 06 contains the value 0Ah
- Load the value 05 into the FSR register
- A read of the INDF register will return the value of 10h
- Increment the value of the FSR register by one (FSR = 06)
- A read of the INDF register now will return the value of 0Ah.

Reading INDF itself indirectly (FSR = 0) will produce 00h. Writing to the INDF register indirectly results in a no-operation (although STATUS bits may be affected).

A simple program to clear RAM locations 20h-2Fh using indirect addressing is shown in Example 2-2.

EXAMPLE 2-2: HOW TO CLEAR RAM USING INDIRECT ADDRESSING

```
        movlw   0x20    ;initialize pointer
        movwf   FSR     ;  to RAM
NEXT    clrf    INDF    ;clear INDF register
        incf    FSR     ;inc pointer
        btfss   FSR,4   ;all done?
        goto    NEXT    ;NO, clear next
CONTINUE
        :               ;YES, continue
```

An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 2-1. However, IRP is not used in the PIC16F84A.

DATA SHEET

For a complete data sheet, please also download:

- The IC06 74HC/HCT/HCU/HCMOS Logic Family Specifications
- The IC06 74HC/HCT/HCU/HCMOS Logic Package Information
- The IC06 74HC/HCT/HCU/HCMOS Logic Package Outlines

74HC/HCT154
4-to-16 line decoder/demultiplexer

roduct specification
ile under Integrated Circuits, IC06

September 1993

hilips
emiconductors

PHILIPS

-16 line decoder/demultiplexer

74HC/HCT154

ES

demultiplexing capability

es 4 binary-coded inputs into one of 16 mutually ve outputs

enable gate for strobing or expansion

capability: standard

egory: MSI

AL DESCRIPTION

C/HCT154 are high-speed Si-gate CMOS devices pin compatible with low power Schottky TTL They are specified in compliance with JEDEC l no. 7A.

The 74HC/HCT154 decoders accept four active HIGH binary address inputs and provide 16 mutually exclusive active LOW outputs.

The 2-input enable gate can be used to strobe the decoder to eliminate the normal decoding "glitches" on the outputs, or it can be used for the expansion of the decoder.

The enable gate has two AND'ed inputs which must be LOW to enable the outputs.

The "154" can be used as a 1-to-16 demultiplexer by using one of the enable inputs as the multiplexed data input.

When the other enable is LOW, the addressed output will follow the state of the applied data.

REFERENCE DATA

V; T_{amb} = 25 °C; $t_r = t_f$ = 6 ns

OL	PARAMETER	CONDITIONS	TYPICAL		UNIT
			HC	HCT	
PH	propagation delay A_n, $\overline{E}_n$ to $\overline{Y}_n$	C_L = 15 pF; V_{CC} = 5 V	11	13	ns
	input capacitance		3.5	3.5	pF
	power dissipation capacitance per package	notes 1 and 2	60	60	pF

is used to determine the dynamic power dissipation (P_D in µW)

$P_D = C_{PD} \times V_{CC}^2 \times f_i + \Sigma (C_L \times V_{CC}^2 \times f_o)$ where:

input frequency in MHz

output frequency in MHz

$C_L \times V_{CC}^2 \times f_o$) = sum of outputs

= output load capacitance in pF

= supply voltage in V

HC: the condition is V_i = GND to V_{CC}

HCT: the condition is V_i = GND to V_{CC} − 1.5 V

RING INFORMATION

4HC/HCT/HCU/HCMOS Logic Package Information".

ESCRIPTION	SYMBOL	NAME AND FUNCTION
NO.		
3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17	$\overline{Y}_0$ to $\overline{Y}_{15}$	outputs (active LOW)
	$\overline{E}_0$, $\overline{E}_1$	enable inputs (active LOW)
9	GND	ground (0 V)
	A_0 to A_3	address inputs
22, 21, 20	V_{CC}	positive supply voltage

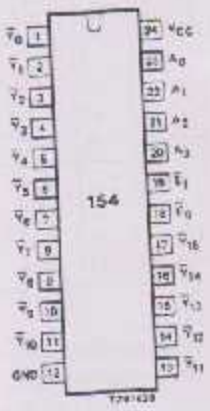


Fig.1 Pin configuration.

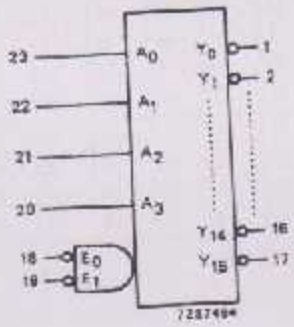


Fig.2 Logic symbol.

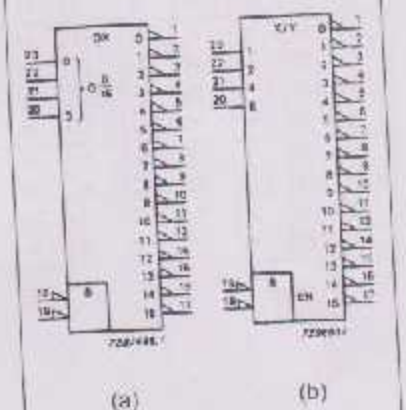


Fig.3 IEC logic symbol.

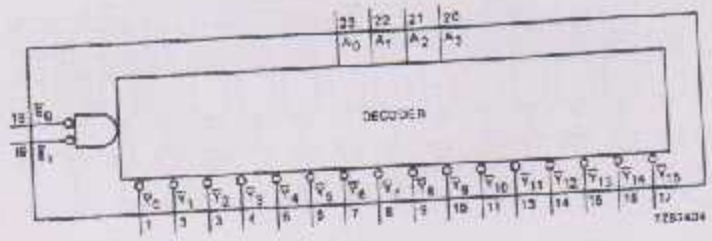


Fig.4 Functional diagram.

FUNCTION TABLE

INPUTS						OUTPUTS															
$\overline{E}_0$	$\overline{E}_1$	A_0	A_1	A_2	A_3	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$	$\overline{Y}_8$	$\overline{Y}_9$	$\overline{Y}_{10}$	$\overline{Y}_{11}$	$\overline{Y}_{12}$	$\overline{Y}_{13}$	$\overline{Y}_{14}$	$\overline{Y}_{15}$
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	H	H	L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L

Note

1. H = HIGH voltage level
   L = LOW voltage level
   X = don't care

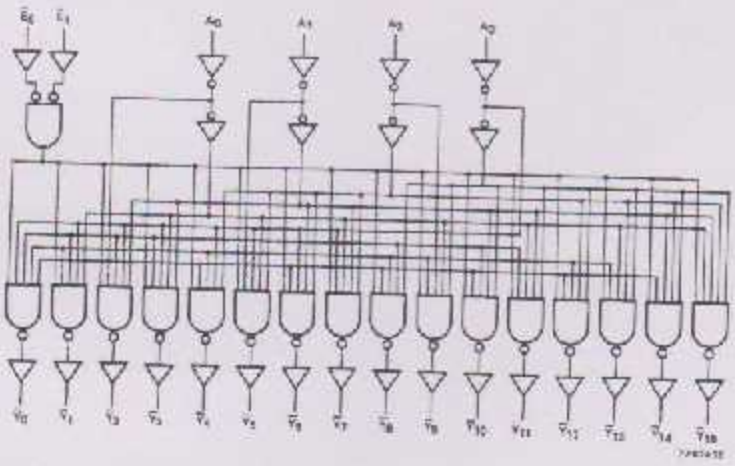


Fig.6 Logic diagram.

DC CHARACTERISTICS FOR 74HC

For the DC characteristics see *"74HC/HCT/HCU/HCMOS Logic Family Specifications"*.

Output capability: standard
I_{CC} category: MSI

AC CHARACTERISTICS FOR 74HC

GND = 0 V; $t_r = t_f$ = 6 ns; C_L = 50 pF

SYMBOL	PARAMETER	T_{amb} (°C)							UNIT	TEST CONDITIONS	
		74HC								V_{CC} (V)	WAVEFORMS
		+25			−40 to −85		−40 to +125				
		min.	typ.	max.	min.	max.	min.	max.			
t_{PHL} / t_{PLH}	propagation delay A_n to $\overline{Y}_n$		36 13 10	150 30 26		190 38 33		225 45 38	ns	2.0 4.5 6.0	Fig.6
t_{PHL} / t_{PLH}	propagation delay $\overline{E}_n$ to $\overline{Y}_n$		39 14 11	150 30 26		190 38 33		225 45 38	ns	2.0 4.5 6.0	Fig.7
t_{THL} / t_{TLH}	output transition time		19 7 6	75 15 13		95 19 16		110 22 19	ns	2.0 4.5 6.0	Figs 6 and 7

VDE | UL | CSA | SETI | SEMKO | DEMKO | NEMKO | BART

6-Pin DIP Optoisolators Transistor Output

4N25*
4N25A*
4N26*
[CTR = 20% Min]
4N27
4N28
[CTR = 10% Min]
*Motorola Preferred Devices

The 4N25/A, 4N26, 4N27 and 4N28 devices consist of a gallium arsenide infrared emitting diode optically coupled to a monolithic silicon phototransistor detector.

- Most Economical Optoisolator Choice for Medium Speed, Switching Applications
- Meets or Exceeds All JEDEC Registered Specifications
- ***To order devices that are tested and marked per VDE 0884 requirements, the suffix "V" must be included at end of part number. VDE 0884 is a test option.***

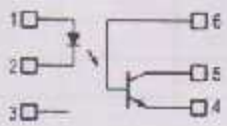
Applications

- General Purpose Switching Circuits
- Interfacing and coupling systems of different potentials and impedances
- I/O Interfacing
- Solid State Relays

STYLE 1 PLASTIC

STANDARD THRU HOLE
CASE 730A–04

SCHEMATIC



PIN 1. LED ANODE
2. LED CATHODE
3. N.C.
4. EMITTER
5. COLLECTOR
6. BASE

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

Rating	Symbol	Value	Unit
INPUT LED			
Reverse Voltage	V_R	3	Volts
Forward Current — Continuous	I_F	60	mA
LED Power Dissipation @ T_A = 25°C with Negligible Power in Output Detector Derate above 25°C	P_D	120 1.41	mW mW/°C
OUTPUT TRANSISTOR			
Collector–Emitter Voltage	V_{CEO}	30	Volts
Emitter–Collector Voltage	V_{ECO}	7	Volts
Collector–Base Voltage	V_{CBO}	70	Volts
Collector Current — Continuous	I_C	150	mA
Detector Power Dissipation @ T_A = 25°C with Negligible Power in Input LED Derate above 25°C	P_D	150 1.76	mW mW/°C
TOTAL DEVICE			
Isolation Surge Voltage[1] (Peak ac Voltage, 60 Hz, 1 sec Duration)	V_{ISO}	7500	Vac(pk)
Total Device Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	250 2.94	mW mW/°C
Ambient Operating Temperature Range[2]	T_A	−55 to +100	°C
Storage Temperature Range[2]	T_{stg}	−55 to +150	°C
Soldering Temperature (10 sec, 1/16" from case)	T_L	260	°C

1. Isolation surge voltage is an internal device dielectric breakdown rating.
For this test, Pins 1 and 2 are common, and Pins 4, 5 and 6 are common.
2. Refer to Quality and Reliability Section in Opto Data Book for information on test conditions.

Preferred devices are Motorola recommended choices for future use and best overall value.
Global Optoisolator is a trademark of Motorola, Inc.

© Motorola, Inc. 1995

MOTOROLA

ELECTRICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted)[1]

Characteristic		Symbol	Min	Typ[1]	Max	Unit
INPUT LED						
Forward Voltage (I_F = 10 mA)	T_A = 25°C T_A = −55°C T_A = 100°C	V_F	— — —	1.15 1.3 1.05	1.5 — —	Volts
Reverse Leakage Current (V_R = 3 V)		I_R	—	—	100	μA
Capacitance (V = 0 V, f = 1 MHz)		C_J	—	18	—	pF
OUTPUT TRANSISTOR						
Collector–Emitter Dark Current (V_{CE} = 10 V, T_A = 25°C	4N25,25A,26,27 4N28	I_{CEO}	— —	1 1	50 100	nA
(V_{CE} = 10 V, T_A = 100°C)	All Devices	I_{CEO}	—	1	—	μA
Collector–Base Dark Current (V_{CB} = 10 V)		I_{CBO}	—	0.2	—	nA
Collector–Emitter Breakdown Voltage (I_C = 1 mA)		$V_{(BR)CEO}$	30	45	—	Volts
Collector–Base Breakdown Voltage (I_C = 100 μA)		$V_{(BR)CBO}$	70	100	—	Volts
Emitter–Collector Breakdown Voltage (I_E = 100 μA)		$V_{(BR)ECO}$	7	7.8	—	Volts
DC Current Gain (I_C = 2 mA, V_{CE} = 5 V)		h_{FE}	—	500	—	—
Collector–Emitter Capacitance (f = 1 MHz, V_{CE} = 0)		C_{CE}	—	7	—	pF
Collector–Base Capacitance (f = 1 MHz, V_{CB} = 0)		C_{CB}	—	19	—	pF
Emitter–Base Capacitance (f = 1 MHz, V_{EB} = 0)		C_{EB}	—	9	—	pF
COUPLED						
Output Collector Current (I_F = 10 mA, V_{CE} = 10 V)	4N25,25A,26 4N27,28	I_C (CTR)[2]	2 (20) 1 (10)	7 (70) 5 (50)	— —	mA (%)
Collector–Emitter Saturation Voltage (I_C = 2 mA, I_F = 50 mA)		$V_{CE(sat)}$	—	0.15	0.5	Volts
Turn–On Time (I_F = 10 mA, V_{CC} = 10 V, R_L = 100 Ω)[3]		t_{on}	—	2.8	—	μs
Turn–Off Time (I_F = 10 mA, V_{CC} = 10 V, R_L = 100 Ω)[3]		t_{off}	—	4.6	—	μs
Rise Time (I_F = 10 mA, V_{CC} = 10 V, R_L = 100 Ω)[3]		t_r	—	1.2	—	μs
Fall Time (I_F = 10 mA, V_{CC} = 10 V, R_L = 100 Ω)[3]		t_f	—	1.9	—	μs
Isolation Voltage (f = 60 Hz, t = 1 sec)[4]		V_{ISO}	7500	—	—	Vac(pk)
Isolation Resistance (V = 500 V)[4]		R_{ISO}	10^{11}	—	—	Ω
Isolation Capacitance (V = 0 V, f = 1 MHz)[4]		C_{ISO}	—	0.2	—	pF

1. Always design to the specified minimum/maximum electrical limits (where applicable).
2. Current Transfer Ratio (CTR) = I_C/I_F x 100%.
3. For test circuit setup and waveforms, refer to Figure 11.
4. For this test, Pins 1 and 2 are common, and Pins 4, 5 and 6 are common.

TYPICAL CHARACTERISTICS

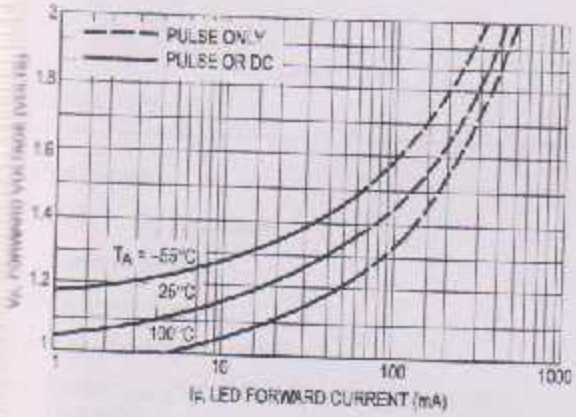


Figure 1. LED Forward Voltage versus Forward Current

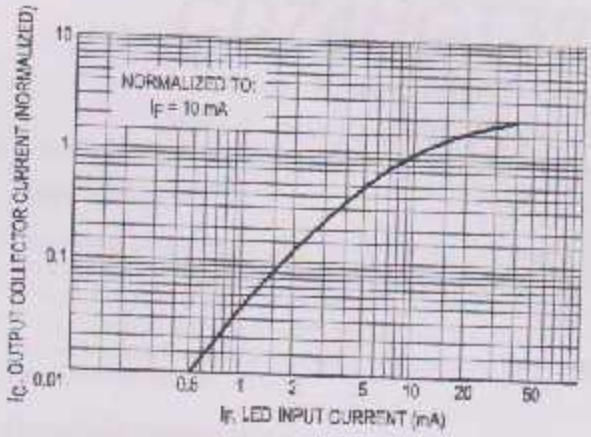


Figure 2. Output Current versus Input Current

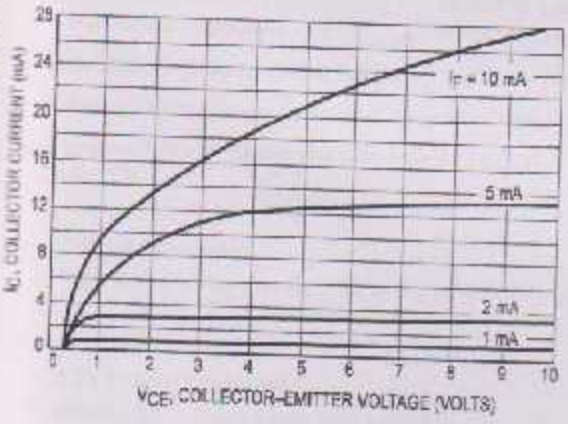


Figure 3. Collector Current versus Collector–Emitter Voltage

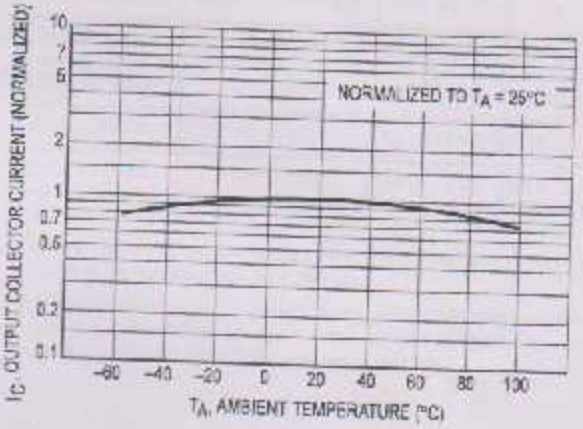


Figure 4. Output Current versus Ambient Temperature

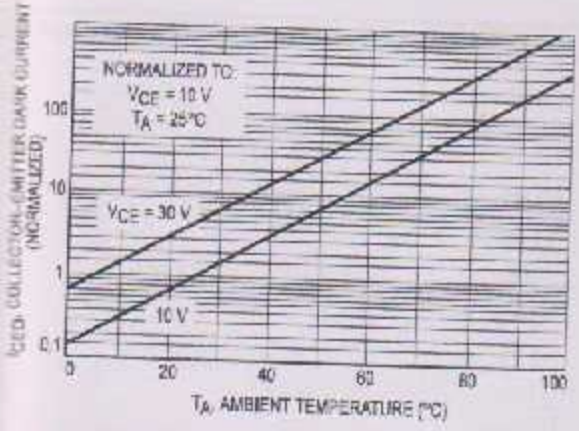


Figure 5. Dark Current versus Ambient Temperature

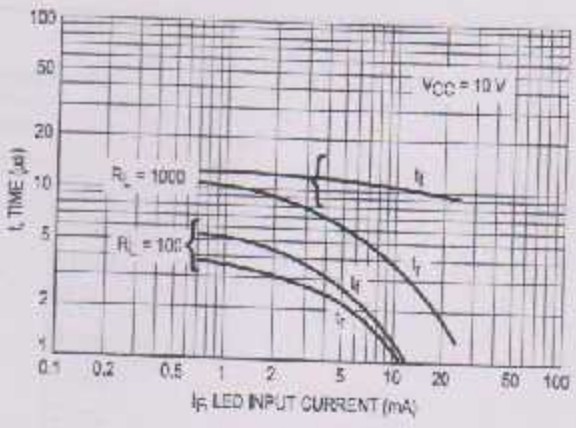


Figure 6. Rise and Fall Times (Typical Values)

HARRIS Semiconductor

CD74HC393, CD74HCT393

High Speed CMOS Logic Dual 4 -Stage Binary Counter

September 1997

Features

- Fully Static Operation
- Buffered Inputs
- Common Reset
- Negative-Edge Clocking
- Typical f_{MAX} = 60 MHz at V_{CC} = 5V, C_L = 15pF, T_A = 25°C
- Fanout (Over Temperature Range)
  - Standard Outputs . . . . . 10 LSTTL Loads
  - Bus Driver Outputs . . . . . 15 LSTTL Loads
- Wide Operating Temperature Range . . . -55°C to 125°C
- Balanced Propagation Delay and Transition Times
- Significant Power Reduction Compared to LSTTL Logic ICs
- HC Types
  - 2V to 6V Operation
  - High Noise Immunity: N_{IL} = 30%, N_{IH} = 30%of V_{CC} at V_{CC} = 5V
- HCT Types
  - 4.5V to 5.5V Operation
  - Direct LSTTL Input Logic Compatibility, V_{IL} = 0.8V (Max), V_{IH} = 2V (Min)
  - CMOS Input Compatibility, $I_I \leq 1\mu A$ at V_{OL}, V_{OH}

Description

The Harris CD74HC393 and CD74HCT393 are 4-stage ripple-carry binary counters. All counter stages are master-slave flip-flops. The state of the stage advances one count on the negative transition of each clock pulse; a high voltage level on the MR line resets all counters to their zero state. All inputs and outputs are buffered.

Ordering Information

PART NUMBER	TEMP. RANGE (°C)	PACKAGE	PKG. NO.
CD74HC393E	-55 to 125	14 Ld PDIP	E14.3
CD74HCT393E	-55 to 125	14 Ld PDIP	E14.3
CD74HC393M	-55 to 125	14 Ld SOIC	M14.15
CD74HCT393M	-55 to 125	14 Ld SOIC	M14.15

NOTES:

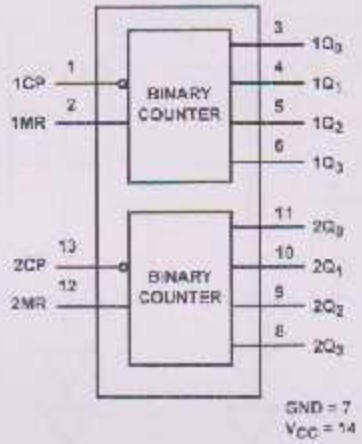
1. When ordering, use the entire part number. Add the suffix 96 to obtain the variant in the tape and reel.
2. Wafer or die for this part number is available which meets all electrical specifications. Please contact your local sales office or Harris customer service for ordering information.

Pinout

CD74HC393, CD74HCT393
(PDIP, SOIC)
TOP VIEW

CAUTION: These devices are sensitive to electrostatic discharge. Users should follow proper IC Handling Procedures.
Copyright © Harris Corporation 1997

Functional Diagram



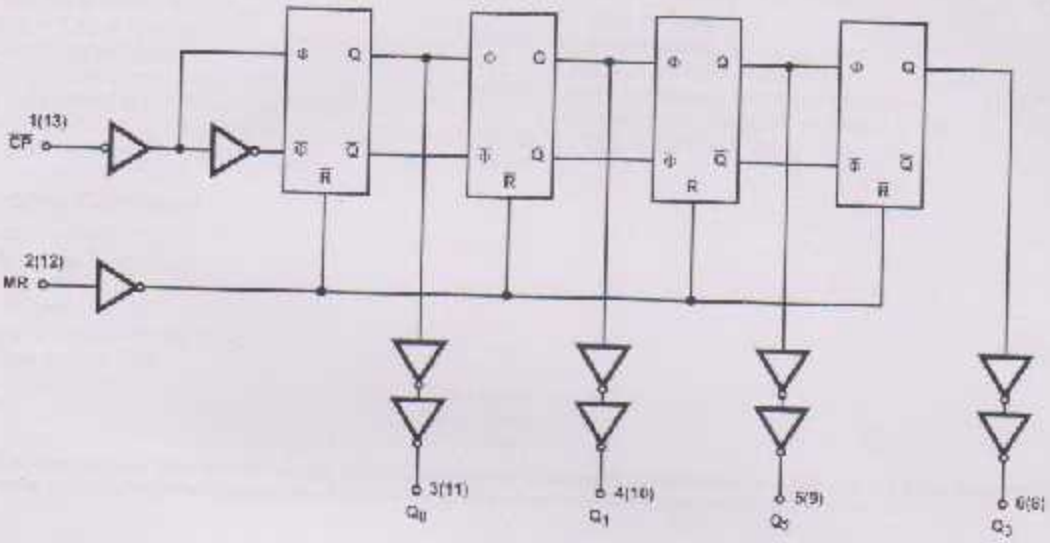
TRUTH TABLE

CP COUNT	OUTPUTS			
	Q_0	Q_1	Q_2	Q_3
0	L	L	L	L
1	H	L	L	L
2	L	H	L	L
3	H	H	L	L
4	L	L	H	L
5	H	L	H	L
6	L	H	H	L
7	H	H	H	L
8	L	L	L	H
9	H	L	L	H
10	L	H	L	H
11	H	H	L	H
12	L	L	H	H
13	H	L	H	H
14	L	H	H	H
15	H	H	H	H

CP COUNT	MR	OUTPUT
↑	L	No Change
↓	L	Count
X	H	LLLL

NOTE: H = High Voltage Level, L = Low Voltage Level, X = Don't Care.
↑ = Transition from Low to High Level, ↓ = Transition from High to Low.

Logic Diagram



Absolute Maximum Ratings

DC Supply Voltage, V_{CC} ..... -0.5V to 7V
DC Input Diode Current, I_{IK}
For V_I < -0.5V or V_I > V_{CC} + 0.5V ..... ±20mA
DC Output Diode Current, I_{OK}
For V_O < -0.5V or V_O > V_{CC} + 0.5V ..... ±20mA
DC Output Source or Sink Current per Output Pin, I_O
For V_O > -0.5V or V_O < V_{CC} + 0.5V ..... ±25mA
DC V_{CC} or Ground Current, I_{CC} or I_{GND} ..... ±50mA

Operating Conditions

Temperature Range (T_A) ..... -55°C to 125°C
Supply Voltage Range, V_{CC}
HC Types ..... 2V to 6V
HCT Types ..... 4.5V to 5.5V
DC Input or Output Voltage V_I, V_O ..... 0V to V_{CC}
Input Rise and Fall Time
2V ..... 1000ns (Max)
4.5V ..... 500ns (Max)
6V ..... 400ns (Max)

Thermal Information

Thermal Resistance (Typical, Note 3) θ_{JA} (°C/W)
PDIP Package 90
SOIC Package 175
Maximum Junction Temperature ..... 150°C
Maximum Storage Temperature Range ..... -65°C to 150°C
Maximum Lead Temperature (Soldering 10s) ..... 300°C
(SOIC - Lead Tips Only)

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

NOTE:

3. θ_{JA} is measured with the component mounted on an evaluation PC board in free air.

DC Electrical Specifications

PARAMETER	SYMBOL	TEST CONDITIONS V_I (V)	TEST CONDITIONS I_O (mA)	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
HC TYPES												
High Level Input Voltage	V_{IH}	-	-	2	1.5	-	-	1.5	-	1.5	-	V
				4.5	3.15	-	-	3.15	-	3.15	-	V
				6	4.2	-	-	4.2	-	4.2	-	V
Low Level Input Voltage	V_{IL}	-	-	2	-	-	0.5	-	0.5	-	0.5	V
				4.5	-	-	1.35	-	1.35	-	1.35	V
				6	-	-	1.8	-	1.8	-	1.8	V
High Level Output Voltage CMOS Loads	V_{OH}	V_{IH} or V_{IL}	-0.02	2	1.9	-	-	1.9	-	1.9	-	V
			-0.02	4.5	4.4	-	-	4.4	-	4.4	-	V
			-0.02	6	5.9	-	-	5.9	-	5.9	-	V
High Level Output Voltage TTL Loads			-	-	-	-	-	-	-	-	-	V
			-4	4.5	3.98	-	-	3.84	-	3.7	-	V
			-5.2	6	5.48	-	-	5.34	-	5.2	-	V
Low Level Output Voltage CMOS Loads	V_{OL}	V_{IH} or V_{IL}	0.02	2	-	-	0.1	-	0.1	-	0.1	V
			0.02	4.5	-	-	0.1	-	0.1	-	0.1	V
			0.02	6	-	-	0.1	-	0.1	-	0.1	V
Low Level Output Voltage TTL Loads			-	-	-	-	-	-	-	-	-	V
			4	4.5	-	-	0.26	-	0.33	-	0.4	V
			5.2	6	-	-	0.26	-	0.33	-	0.4	V
Input Leakage Current	I_I	V_{CC} or GND	-	6	-	-	±0.1	-	±1	-	±1	µA
Quiescent Device Current	I_{CC}	V_{CC} or GND	0	6	-	-	8		80	-	160	µA

DC Electrical Specifications (Continued)

PARAMETER	SYMBOL	TEST CONDITIONS V_I (V)	TEST CONDITIONS I_O (mA)	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
HCT TYPES												
High Level Input Voltage	V_{IH}	-	-	4.5 to 5.5	2	-	-	2	-	2	-	V
Low Level Input Voltage	V_{IL}	-	-	4.5 to 5.5	-	-	0.8	-	0.8	-	0.8	V
High Level Output Voltage CMOS Loads	V_{OH}	V_{IH} or V_{IL}	-0.02	4.5	4.4	-	-	4.4	-	4.4	-	V
High Level Output Voltage TTL Loads			-4	4.5	3.98	-	-	3.84	-	3.7	-	V
Low Level Output Voltage CMOS Loads	V_{OL}	V_{IH} or V_{IL}	0.02	4.5	-	-	0.1	-	0.1	-	0.1	V
Low Level Output Voltage TTL Loads			4	4.5	-	-	0.26	-	0.33	-	0.4	V
Input Leakage Current	I_I	V_{CC} and GND	0	5.5	-	-	±0.1	-	±1	-	±1	µA
Quiescent Device Current	I_{CC}	V_{CC} or GND	0	5.5	-	-	8	-	80	-	160	µA
Additional Quiescent Device Current Per Input Pin: 1 Unit Load	ΔI_{CC}	V_{CC} -2.1	-	4.5 to 5.5	-	100	360	-	450	-	490	µA

NOTE: For dual-supply systems theoretical worst case (V_I = 2.4V, V_{CC} = 5.5V) specification is 1.8mA.

HCT Input Loading Table

INPUT	UNIT LOADS
$n\overline{CP}$	0.4
nMR	1

NOTE: Unit Load is ΔI_{CC} limit specified in DC Electrical Table, e.g., 360µA max at 25°C.

Prerequisite for Switching Specifications

PARAMETER	SYMBOL	V_{CC} (V)	25°C MIN	25°C TYP	25°C MAX	-40°C TO 85°C MIN	-40°C TO 85°C MAX	-55°C TO 125°C MIN	-55°C TO 125°C MAX	UNITS
HC TYPES										
Maximum Clock Frequency	f_{MAX}	2	6	-	-	5	-	4	-	ns
		4.5	30	-	-	24	-	20	-	ns
		6	35	-	-	28	-	24	-	ns
Clock Pulse Width	t_W	2	80	-	-	100	-	120	-	ns
		4.5	16	-	-	20	-	24	-	ns
		6	14	-	-	17	-	20	-	ns
Reset Recovery Time	t_{REC}	2	5	-	-	5	-	5	-	ns
		4.5	5	-	-	5	-	5	-	ns
		6	5	-	-	5	-	5	-	ns