Semi-Auto Jig For LCD Testing

By

Norhafiza Binti Rahim

FINAL PROJECT REPORT

Submitted to

Electrical & Electronics Engineering Department

In partial fulfillment of the requirement for the

Bachelor of Engineering (Hons)

(Electrical & Electronics Engineering)

DECEMBER 2004

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

[SEMI-AUTO JIG FOR LCD TESTING]

by

[Norhafiza Binti Rahim]

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:

[illegible]

[Abu Bakar Sayuti Hj Mohd Saman]
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

December 2004

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]

[Norhafiza Binti Rahim]

ABSTRACT

LCD is normally used in laptop computers, digital clocks and watches, microwave ovens, CD players and many other electronic devices and they offer some real advantages over display technologies. In a radio application, the LCD is used to display the function of the radio as possible as the menus to run the radio functions. Normally, manufacturer plugs in the LCD panel on the radio to display the required segment by using the LCD driver. The segment displays can be viewed as they press the radio button or keypads. Repeating job like plug in the LCD onto- the radio panel brings lots of problems. This testing method is not productive because of the time taken to plug in the LCD and take it out from the radio panel is too long. There will be side damage as we will re-assemble the radio panel after testing the LCD panel. The purpose to build the jig is to check the segment display on the specified LCD. The testing procedure should be handled in easier way in order to test the segment displays. The literature review consists of the fundamental theory of each related topic involved in this project. The topics involved are related to LCD applications, PIC programming, Visual Basic Programming and also circuit simulation using tools available. Serial communication between the serial port and user interface allow user to send the signal to each of the LCD pitch. Software In implementing this project, there are several main processes taken. The project is divided intro three parts which is hardware part or LCD tester jig, user interfacing and serial communication. The circuit is designed to send out the signal to turn on the LCD segment display controlling. In the "Testing History" the result test is recorded according to number of testing done, date, time, total LCD passed, total LCD rejected and also total LCD tested. The jig is able to check the segment display as required. User can assemble the LCD on the jig without continuous side damage. The testing window can help user to implement the LCD checking with a smooth testing flow. It also reduces the time taken to implement this LCD checking.

ACKNOWLEDGEMENTS

Alhamdulillah, with the greatest gratitude to the Almighty **Allah** for his gracious blessings throughout the whole period this project was undertaken. My deepest gratitude to my parents, who have provided me with their love and undivided attention through the sweet and sour moments. To my supervisor, **Mr Abu Bakar Sayuti Hj Mohd Saman**, thank you so much for your guidance and support throughout this project. His words of wisdom have encouraged me to rise again during the times I fall.

The compliment should also go to all Electrical and Electronics Engineering Laboratory technicians for bundles of information and assistance in completing this project especially to Ms. Siti Hawa and En. Isnani

I would like to take this opportunity to thank all my friends who has contributed to this project. Your assistance is duly acknowledged and noted.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Liquid crystal display (LCD) is a display having conductive segments or dots deposited on the inside surfaces of two transparent glass plates separated by a crystal in liquid form. When energized with AC voltage in the presence of light, the selected segments will provide black-tone or gray readout. We use LCD in laptop computers, digital clocks and watches, microwave ovens, CD players and many other electronic devices. LCDs are common because they offer some real advantages over other display technologies. They are thinner and lighter and draw much less power than cathode ray tubes (CRTs). For example, in Radio application, the LCD is used to display the function of the radio as possible as the menus to run the radio functions. Normally, manufacturer plugs in the LCD panel on the radio to display the required segment by using the LCD driver. The segment displays can be viewed as they press the radio button or keypads.

1.2 Problem Statement

1.2.1 Problem Identification

Repeating job like plug in the LCD onto- the radio panel brings lots of problems. This testing method is not productive because of the time taken to plug in the LCD and take it out from the radio panel is too long. There will be side damage as we will re-assemble the radio panel after testing the LCD panel. The slotted pin for voltage supply of the radio is also can effected. Furthermore, the problem identification each electronic component on the radio panel is more complicated as the LCD is tested using the driver which link with other IC and microcontrollers. The requirements of this project are to design a jig to test the LCD and find the best way for user to interface the testing methods.

1.2.2 Significant of the Project

The purpose to build the jig is to check the segment display on the specified LCD. The findings of this project would help the user to identify the display problem on the specified LCD. The testing procedure should be handled in easier way in order to test the segment displays. This project is designed to come out with a new testing method and a testing jig so that the test procedures become smoother and efficient. The primary function of the jig is only to test the segment display of the LCD in all ON condition. The design of the jig can reduce time taken to implement the LCD testing. The jig is able to detect the current level which flow at each pitch of LCD. The buzzer will sound as the current flow at each of LCD pitch is out of range. This buzzer will alert user if there is a segment display problem on the LCD. If this happen, the LCD panel is considered as having missing segment or functional defect.

1.3 Objective and Scope of Study

1.3.1 The Relevancy of the Project

This project is relevant to give exposure to the students regarding the self studying, knowledge findings, software usability and hands-on tasks on order to complete the whole project. For the first semester, the target is to complete the literature study of the theory and engineering principles, system simulations and analysis. For the second semester, this project will focus on circuit simulation and troubleshooting. Then, it should end with the jig prototype.

1.3.2 Feasibility of the Project within the Scope and Time frame

The scope of the project can be divided into 2 parts of these two semesters. For the first semester, literature reviews regarding LCD applications, the serial port, PIC Programming and user interfacing should be gained and understand the concept. In the second semester involves the circuit simulations and analysis of the output by using the software and starts to prepare for the hardware. This project can be completed within the time frame given.

The literature review or theory related to this project is explained in Chapter 2. The topics related involved the understanding about the LCD used in this project, the visual Basic Programming, introduction to the serial communication and also the peripheral interface programming.

The methodology to achieve the objectives of this project is described in Chapter 3 of this report. The focus of this chapter is to explain the procedures taken to implement this project. The finding on this project is discussed in Chapter 4. the conclusion and recommendation is stated in Chapter 5.

CHAPTER 2

LITERATURE REVIEW AND THEORY

The literature review consists of the fundamental theory of each related topic involved in this project. The topics involved are related to LCD applications, PIC programming, Visual Basic Programming and also circuit simulation using tools available.

2.1 Liquid Crystal Display (LCD)

Liquid crystal display (LCD) displays utilize two sheets of polarizing material with a liquid crystal solution between them. An electric current passed through the liquid causes the crystals to align so that light cannot pass through them. Each crystal, therefore, is like a shutter, either allowing light to pass through or blocking the light. LCD can be light on with 5 V ac voltages **[1]**.

Some research on how the LCD works have been done. There is far more to building an LCD than simply creating a sheet of liquid crystals. The combination of four facts makes LCD possible:

- Light can be polarized.
- Liquid crystals can transmit and change polarized light.
- The structure of liquid crystals can be changed by electric current.
- There are transparent substances that can conduct electricity.

An LCD is a device that uses these four facts in a surprising way. LCD is made from take two pieces of polarized glass. A special polymer that creates microscopic grooves in the surface is rubbed on the side of the glass that does not have the polarizing film on it. The grooves must be in the same direction as the polarizing film and added with a coating of nematic liquid crystals to one of the filters. The grooves will cause the first layer of molecules to align with the filter's orientation. Then add the second piece of glass with the polarizing film at a right angle to the first piece. Each successive layer of TN molecules will gradually twist until the uppermost layer is at a 90-degree angle to the bottom, matching the polarized glass filters.

As light strikes the first filter, it is polarized. The molecules in each layer then guide the light they receive to the next layer. As the light passes through the liquid crystal layers, the molecules also change the light's plane of vibration to match their own angle. When the light reaches the far side of the liquid crystal substance, it vibrates at the same angle as the final layer of molecules. If the final layer is matched up with the second polarized glass filter, then the light will pass through.

When we apply an electric charge to liquid crystal molecules, they untwist. When they straighten out, they change the angle of the light passing through them so that it no longer matches the angle of the top polarizing filter. Consequently, no light can pass through that area of the LCD, which makes that area darker than the surrounding areas.

The layers would look like this:

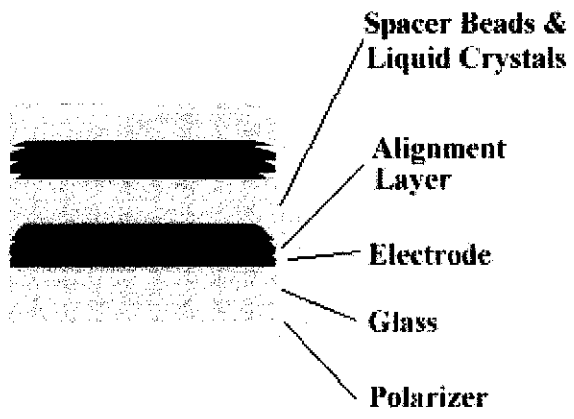


Figure 2.1: LCD layers

The electrode is hooked up to a power source like a battery. When there is no current, light entering through the front of the LCD will simply hit the mirror and bounce right back out. But when the battery supplies current to the electrodes, the liquid crystals between the common-plane electrode and the electrode shaped like a rectangle untwist and block the light in that region from passing through. That makes the LCD show the rectangle as a black area.

There are two main types of LCDs used in computers, passive matrix and active matrix. Passive-matrix LCDs use a simple grid to supply the charge to a particular pixel on the display. Creating the grid is quite a process. It starts with two glass layers called substrates. One substrate is given columns and the other is given rows made from a transparent conductive material. This is usually indium-tin oxide. The rows or columns are connected to integrated circuits that control when a charge is sent down a particular column or row. The liquid crystal material is sandwiched between the two glass substrates, and a polarizing film is added to the outer side of each substrate. To turn on a pixel, the integrated circuit sends a charge down the correct column of one substrate and a ground activated on the correct row of the other. The row and column intersect at the designated pixel, and that delivers the voltage to untwist the liquid crystals at that pixel.

LCD required an external light source. Liquid crystal materials emit no light of their own. Small and inexpensive LCDs are often reflective, which means to display anything they must reflect light from external light sources. Look at an LCD watch: The numbers appear where small electrodes charge the liquid crystals and make the layers untwist so that light is not transmitting through the polarized film.

In general terms, in order to protect the liquid crystal material from deteriorating, cells are addressed by alternating current (AC), not direct current (DC). LCDs require very little power to operate, typically less than 5mA. For this LCD the it is operate in up to 1.2 mA.

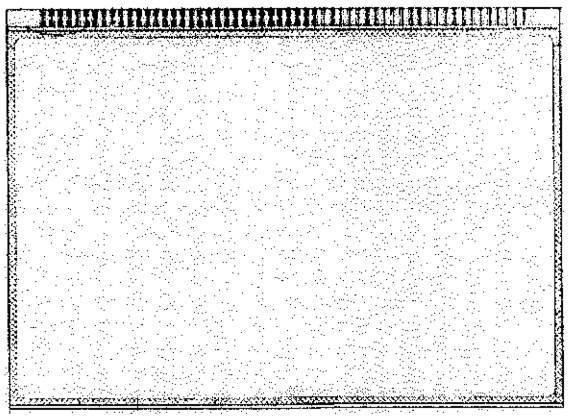


Figure 2.2: LCD used in this project

The LCD used in this project is a passive matrix LCD produced by Varitronix (M) Sdn Bhd [3]. This LCD has 56 pitches which mean that each pitch has its on character.

The character of each pitch of this LCD is listed in table below:

Pin No.				
56	(6)	I	H	G
55	J	D	E	F
54	(4)	---	---	(5)
53	L	C	B	A
52	1D	1E	1F	1G
51	FM	1M	1N	1A
50	1	1L	1H	1J
49	2	1C	1K	1B
48	AM	2E	2F	2G
47	2D	2M	2N	2A
46	ASM	2L	2H	2J
45	(3)	2C	2K	2B
44	M	3E	3F	3G
43	3D	3M	3N	3A
42	(2)	3L	3H	3J
41	STEREO	3C	3K	3B
40	N	4E	4F	4G
39	4D	4M	4N	4A
38	(1)	4L	4H	4J
37	DISC	4C	4K	4B
36	O	5E	5F	5G
35	5D	5M	5N	5A
34	CD-IN	5L	5H	5J
33	EQ-ON	5C	5K	5B
32	ROCK	6E	6F	6G
31	6D	6M	6N	6A
30	POP	6L	6H	6J
29	K	6C	6K	6B
28	P1	7E	7F	7G
27	7D	7M	7N	7A
26	---	7L	7H	7J
25	---	7C	7K	7B
24	P2	8E	8F	8G
23	8D	8M	8N	8A
22	TRACK	8L	8H	8J
21	---	8C	8K	8B
20	---	11F	11E	11N
19	11A	11H	11G	11D
18	---	11J	11M	11L
17	P3	11B	11K	11C
16	---	12F	12E	12N
15	12A	12H	12G	12D
14	---	12J	12M	12L
13	LOCAL	12B	12K	12C
12	P4	10B	10K	10C
11	---	10J	10M	10L
10	10A	10H	10G	10D
9	---	10F	10E	10N
8	VOCAL	9B	9K	9C
7	CLAS'L	9J	9M	9L
6	9A	9H	9G	9D
5	JAZZ	9F	9E	9N
4	---	---	---	COM 4
3	---	---	COM 3	---
2	---	COM 2	---	---
1	COM 1	---	---	---

Table 2.1: The character in each LCD pitch

2.2 Peripheral Interface Controller (PIC)

Peripheral Interface Controller (PIC) is the IC which was developed to control peripheral devices, alleviating the load from the main CPU [2]. Compared to a human being, the brain is the main CPU and the PIC is equivalent to the autonomic nervous system. PIC is a type of microcontrollers which are essentially 8 bit microprocessors with small RAM, ROM and simple peripherals packaged on a single chip. 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. The PIC is convenient for making calculations. The memory, the input/output ports and so on are incorporated into the IC. The efficiency and the functions are limited, but the PIC can do the job of many IC's with software. So, the circuit can be compact.

Term "port" refers to a group of pins on a microcontroller which can be accessed simultaneously, or on which we can set the desired combination of zeros and ones, or read from them an existing status. Physically, port is a register inside a microcontroller which is connected by wires to the pins of a microcontroller. Ports represent physical connection of Central Processing Unit with an outside world. Microcontroller uses them in order to monitor or control other components or devices. The figure 2.3 and 2.4 show the pin diagram of the PIC used in this project.

Pin Diagram

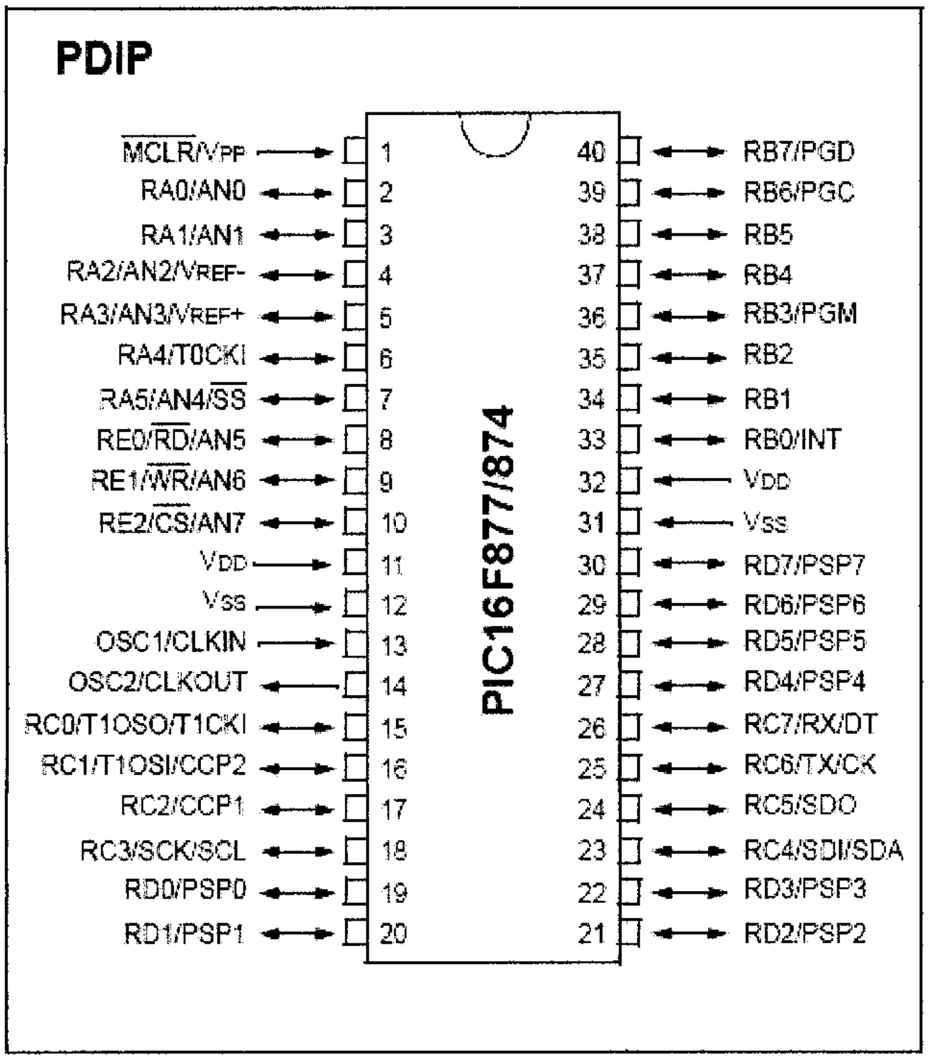


Figure 2.3: Pin Diagram of PIC16F877

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.

The PIC16F84A 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.

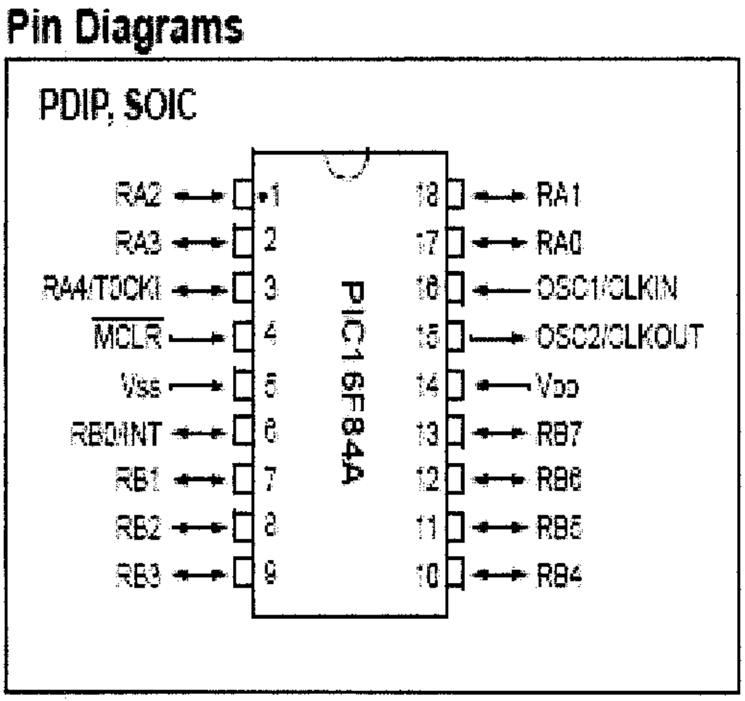


Figure 2.4: Pin Diagram of PIC16F84A

2.3 Serial communication

The serial port is an I/O (Input/Output) device. An I/O device is just a way to get data into and out of a computer [4]. There are many types of I/O devices such as serial ports, parallel ports, disk drive controllers, ethernet boards, universal serial buses, etc. Most PC's have one or two serial ports. Each has a 9-pin connector (sometimes 25-pin) on the back of the computer.

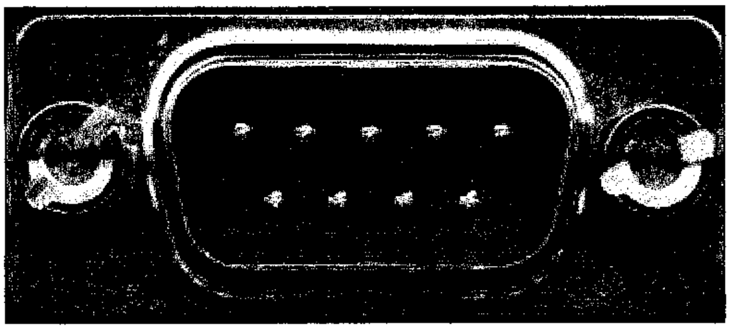


Figure 2.5: Male serial connector.

Computer programs can send data (bytes) to the transmit pin (output) and receive bytes from the receive pin (input). The other pins are for control purposes and ground. The serial port is much more than just a connector. It converts the data from parallel to serial and changes the electrical representation of the data. Inside the computer, data bits flow in parallel (using many wires at the same time). Serial path is a stream of bits on a single wire on the transmit or receive pin of the serial connector. For the serial port to create such a flow, it must convert data from parallel (inside the computer) to serial on the transmit pin (and conversely).

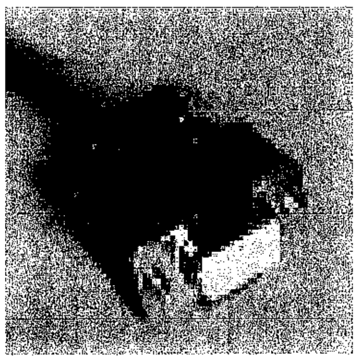


Figure 2.6: Female serial connector and cable

RS232 is a voltage loop interface for two-way (full-duplex) communication represented by voltage levels with respect to system ground (common). A common ground between the PC and the associated device is necessary. Maximum serial cable length is defined: 75 feet at 9,600 bps, but today cables up to 1,000 feet are used successfully.

RS 232 serial port (9-pin) DTE-device (PC) male connector, female cable connector	
PIN	DESCRIPTION
1	Data Carrier Detect
2	Received Data
3	Transmitted Data
4	DTE (Data Terminal) Ready
5	Signal Ground
6	DCE (Data Set) Ready
7	Request to Send
8	Clear to Send
9	Ring Indicator

Table 2.2: RS232 pin number and description

The serial port on the PC is a full-duplex device meaning that it can send and receive data at the same time. In order to be able to do this, it uses separate lines for transmitting and receiving data. Suppose we are working with three lines only, and that one line is used for sending data, other for receiving, and the third one is used as a reference line for both the input and the output side. In order for this to work, we need to set the rules of exchange of data. These rules are called protocol. Protocol is therefore defined in advance so there would not be any misunderstanding between the sides that are communicating with each other.

The logical unit "1" is set up on the transmitting line until transfer begins. Once the transfer starts, we lower the transmission line to logical "0" for a period of time (which we will designate as T), so the receiving side will know that it is receiving data, and so it will activate its mechanism for reception. Go back to the transmission side and start putting logic zeros and ones onto the transmitter line in the order from a bit of the lowest value to a bit of the highest value. Let each bit stay on line for a time period which is equal to T, and in the end, or after the 8th bit, let us bring the logical unit "1" back on the line which will mark the end of the transmission of one data. The protocol we've just described is called in professional literature NRZ (Non-Return to Zero).

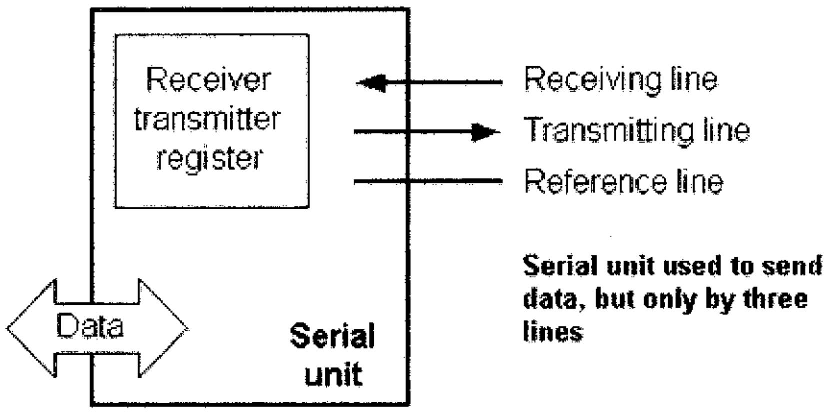

Receiver transmitter register
Receiving line
Transmitting line
Reference line
Serial unit used to send data, but only by three lines
Data
Serial unit

Figure 2.7: Serial unit used to send data, but only by three lines

There are two basic types of serial communications, synchronous and asynchronous. With synchronous communications, the two devices initially synchronize themselves to each other, and then continually send characters to stay in sync. Even when data is not really being sent, a constant flow of bits allows each device to know where the other is at any given time. That is, each character that is sent is either actual data or an idle character. Synchronous communications allows faster data transfer rates than asynchronous methods, because additional bits to mark the beginning and end of each data byte are not required. The serial ports on IBM-style PCs are asynchronous devices and therefore only support asynchronous serial communications.

Asynchronous means "no synchronization", and thus does not require sending and receiving idle characters. However, the beginning and end of each byte of data must be identified by start and stop bits. The start bit indicates when the data byte is about to begin and the stop bit signals when it ends. The requirement to send these additional two bits causes asynchronous communication to be slightly slower than synchronous however it has the advantage that the processor does not have to deal with the additional idle characters.

An asynchronous line that is idle is identified with a value of 1 (also called a mark state). By using this value to indicate that no data is currently being sent, the devices are able to distinguish between an idle state and a disconnected line. When a character is about to be transmitted, a start bit is sent. A start bit has a value of 0 (also called a space state). Thus, when the line switches from a value of 1 to a value of 0, the receiver is alerted that a data character is about to be sent.

2.4 Visual Basic Programming

Visual Basic is a computer programming language [5]. A programming language uses words with a specific meaning, connected together in a specific order, to form a statement. A program is created when several statements are assemble together to accomplish a specific task. A program is a set of statements written in a computer language in order to accomplish a specific task. In this project, Visual Basic is used to create a program that will display different graphical images on the screen.

The user interface defines what a user of the program sees on the screen and how the user interacts with the program. In writing a computer program in Visual Basic, various tools to design the end-user interface, for the program that was created. Visual Basic is an event-driven programming language which means that different windows on the screen can respond to events. An event is an action that occurs as a result of some user activity.

Visual Basic is also an object-oriented programming language. Object-oriented means that the programmer creates objects that end-user will use to perform task. An object can be a button that the user clicks or a box that will contain text. In Visual Basic, tools to create objects are called controls. Each type of control has different characteristics (properties), responds to events that are discrete for each type of control, and perform a unique set of actions (methods).

In Visual Basic, a control is considered a class. A class is a template for an object, defining the object's supported properties, methods and events. In other words, a class defines what an object does and how it behaves. When creating a control instance on a form, thus create an instance of a class.

Object-oriented design describes a solution to a problem in terms of active data elements called objects. Objects are a composition of data and actions. The data represents the state of the object. The actions use the objects themselves as the basis for describing how the solution will work.

There are three basic stages in creating a program;

- Designing the user interface that the end-user will see, this involves placing controls onto the form and modifying their properties.
- Writing statements to perform each task required by the program.
- Looping in which the programmer test the program, corrects any errors, then test the program again.

The Project menu enables to add forms or modules to the project and to alter the project properties. The format menu enables to size and align the controls on the form to make its appearance more uniform and tidy. The Option menu item in the Tools menu allows changing various options that suits. The properties window displays the properties for the selected item or control, enabling to change it for the best operation.

In order to create the Visual Basic code that will display a graphical image and its corresponding file name, the actions taken when an event occurs must be defined. An event procedure is a set of Visual Basic statement that execute when a user performs an action on an object, such as clicking a command button.

There are some components in Visual Basic Programming required to perform a serial communication between Visual Basic programming and serial port which is call MScomm. This component can allow a communication between serial port and the Visual Basic Programming. The component that allows user to create a database is known as Data component.

The figure below shows the flow chart on how user can build a user interfacing using Visual Basic.

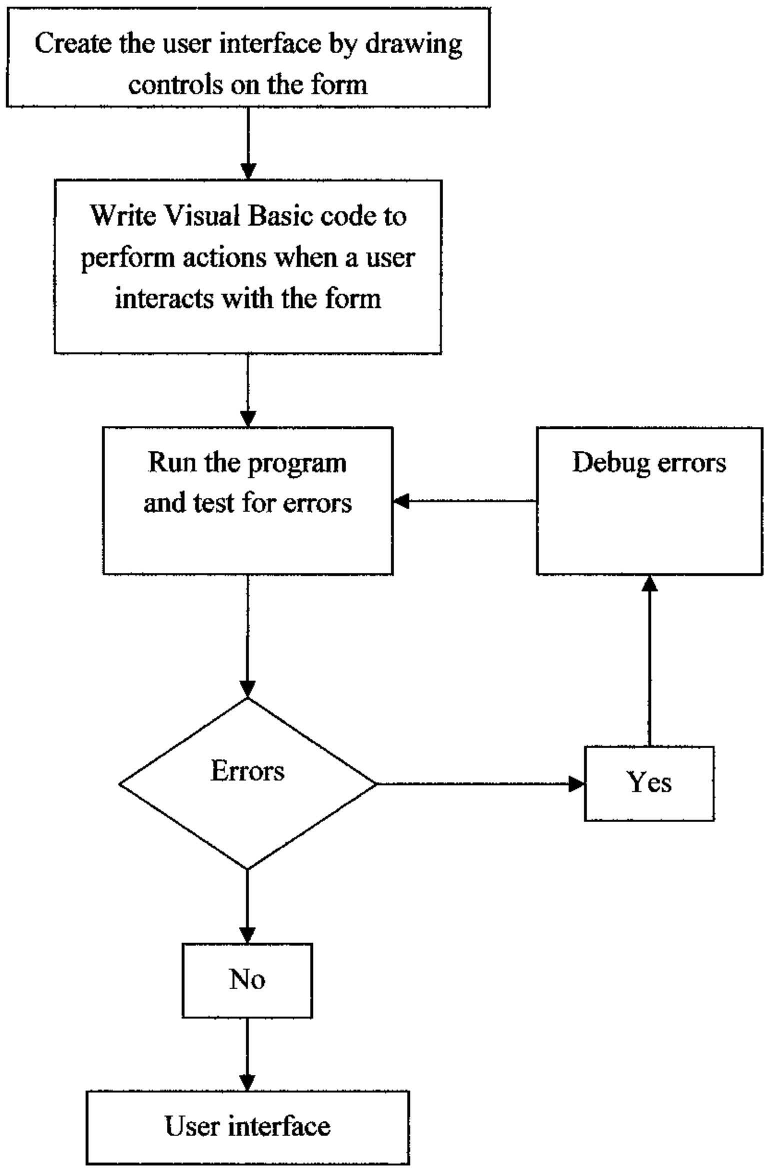

Create the user interface by drawing controls on the form
Write Visual Basic code to perform actions when a user interacts with the form
Run the program and test for errors
Debug errors
Errors
Yes
No
User interface

Figure 2.8: Flow chart for user interface using Visual Basic Programming.

CHAPTER 3

METHODOLOGY OR PROJECT WORK

3.1 Procedure Identification

In implementing this project, there are several main processes taken. The Gantt chart for this project is as appendix X. The procedures in designing out the jig are as follows:

3.1.1 Identify the system breakdown of this project

The project is divided intro three parts:

- Hardware part or LCD tester jig
- Testing window or User interfacing
- Serial communication

The system breakdown is shown in Figure 3.1.

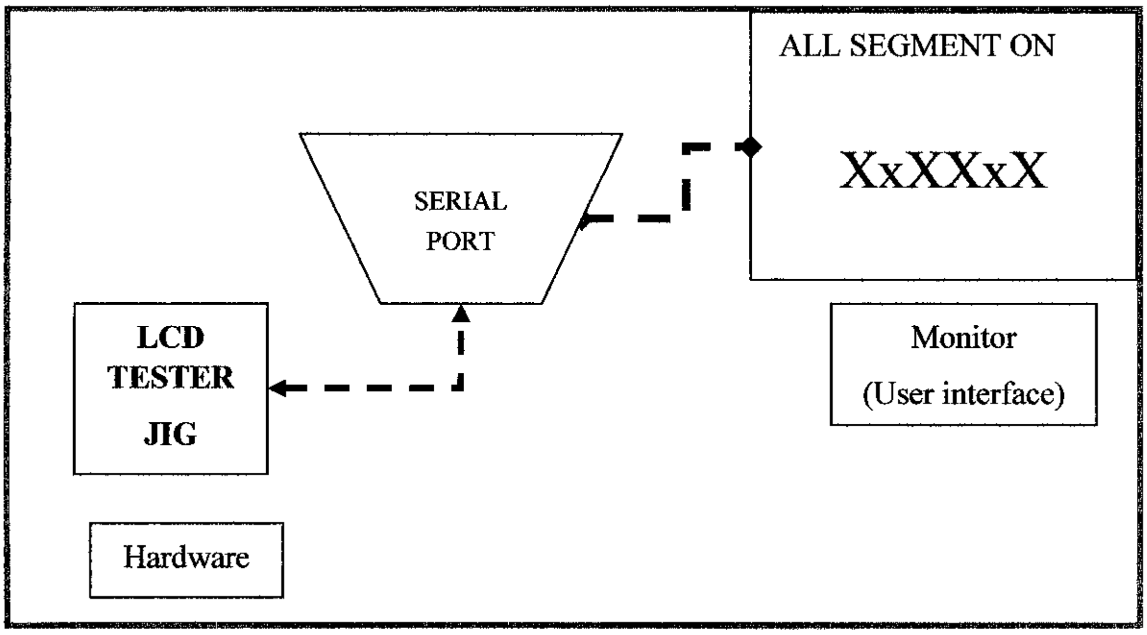


Figure 3.1: System Breakdown of the Project

3.1.2 Understand the needs of each part of system from this system breakdown.

Hardware part or LCD tester jig

- The tester should able to test the segment display on the LCD

Testing window or user interfacing

- User can send test sequences or test number in the testing window.
- User can view the specified segment displays on the LCD as same as on the testing window.
- The testing result is stored as database.
- This user interfacing must be user friendly system.

Serial communication

- Serial port is used to communicate data from the computer to the tester and vise versa.

3.1.3 Specify the test sequences of a specified segments display

The test sequence or number of testing to be performed in this system is important to be identified. This is because the sequence of testing is related to the test result that will be stored in the database.

3.1.4 Identify the number of output or pitches on the specified LCD and segments characters.

The segment will be on directly by supplying 5V to the LCD pitches. Each pitch has their on segment characters.

3.1.5 Design the circuit to send out the signal to the serial port and simulate them using Multisim 6.

The circuit is divided into two parts. The main circuit is simulated to send out the signal to each of the LCD pitch.

3.1.6 Write the coding for the PICs used to send out the signal to the serial port with C programming and burn it to the PICs

3.1.7 Build up the testing window for user interface using Visual Basic Programming 6.0.

3.1.8 Communicate the hardware with the user interface and perform troubleshooting.

3.2 Tools required

3.2.1 Hardware

3.2.1.1 Header for LCD Tester Jig

The tools required for the tester header are as follows:

- LCD (specified/customized) produced by Varitronix (M) Sdn Bhd. Refer to Figure 2.2
- Copper conductor
- Zebra connector or known as elastomer.

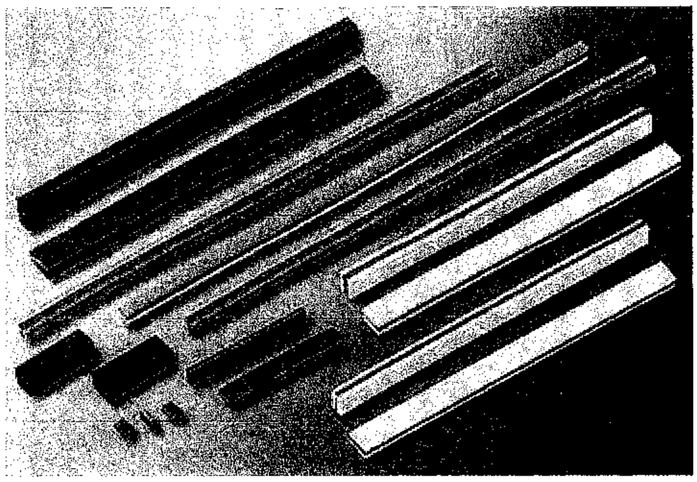


Figure 3.2: Elastomer.

- LCD reflector

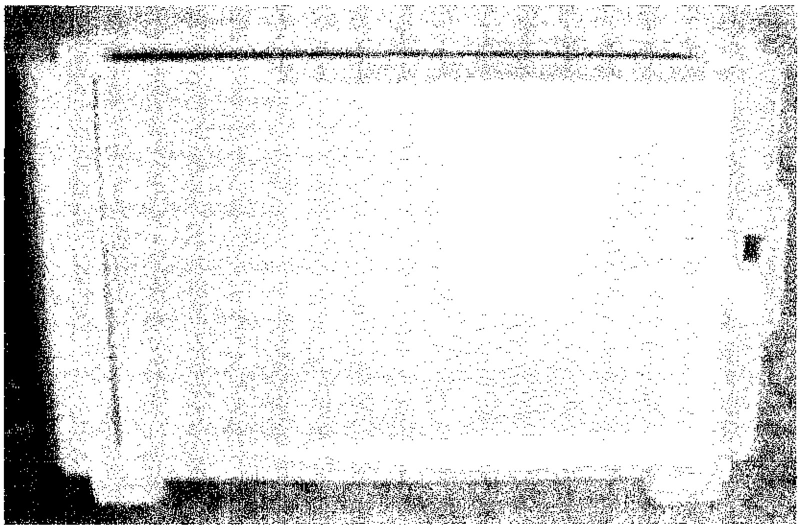


Figure 3.3: LCD reflector

3.2.1.2 Circuit for LCD Tester Jig

The electronic components required for the tester circuit are as follows:

- PIC16F877 2 pcs
- PIC16F84A 1 pc
- RS232
- Serial Female connector
- Serial Male connector
- Serial cable
- Capacitors
- Resistor
- Voltage regulator
- LEDs
- Push button

3.2.2 Software

The software required to implement this project involved:

- Visual Basic 6.0 Programming to build the user interface coding.
- C programming for PIC and serial communication.
- PIC C compiler
- PIC burner
- WARP 13
- Multisim

CHAPTER 4

RESULT AND DISCUSSION

The literature review helps to go further on data analysis and project implementations.

4.1 Circuit design

This project consists of two circuit part. The first circuit is designed to send out the signal to turn on the LCD segment controlling. The second circuit is designed for the tester jig header.

Circuit design for segment activation and controlling is shown in **Appendix F**. From the circuit, two PICs are used to activate the LCD segments. Each of PIC16F877, only 28 output pin are used assigned to the first of 28 LCD pitch and other 28 is assigned to second PIC16F877. The PIC16F84A is used as a main controller to control the 56 output pin of the two PIC16f877. The bit assigned is as in table below.

Control bits	Test assigned
00	Test 1
01	Test 2
10	Test 3

Table 4.1: Control bit assigned for each test number

The second circuit design consist only copper line functions as a conductor for the tester header. This circuit board is attached with the LCD reflector. The power is supplied to the LCD pitch through the zebra connector.

The PCB layout of the copper conductor as in figure below has been done.

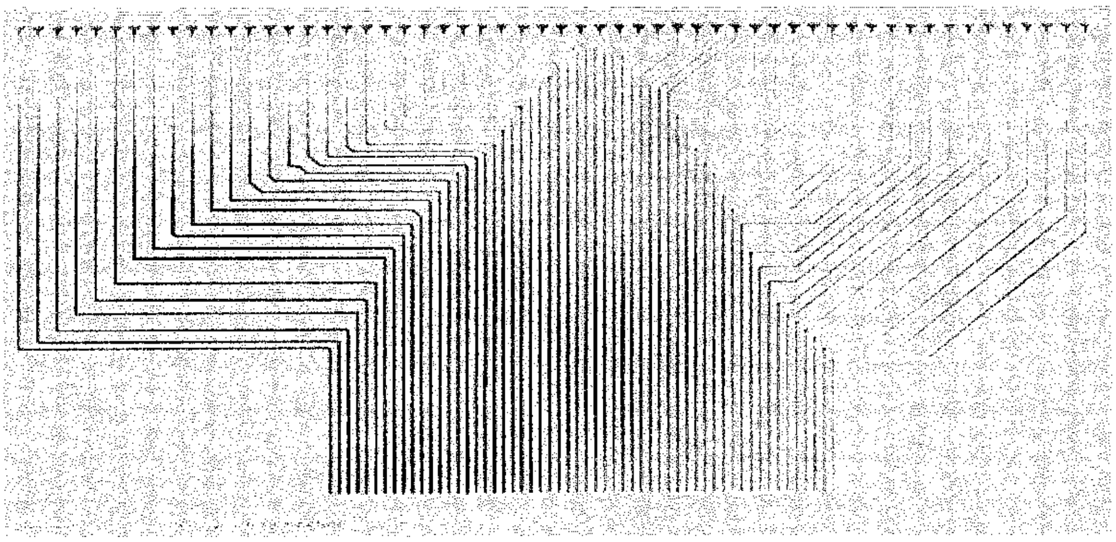


Figure 4.1: A copper conductor for tester header

The coding for these three PICs in C programming language is shown in Appendix A, B and C.

4.2 Testing Sequences

The number of LCD pitch to activate is identified before we identify the testing sequences. The table below show the LCD pitch need to be activated.

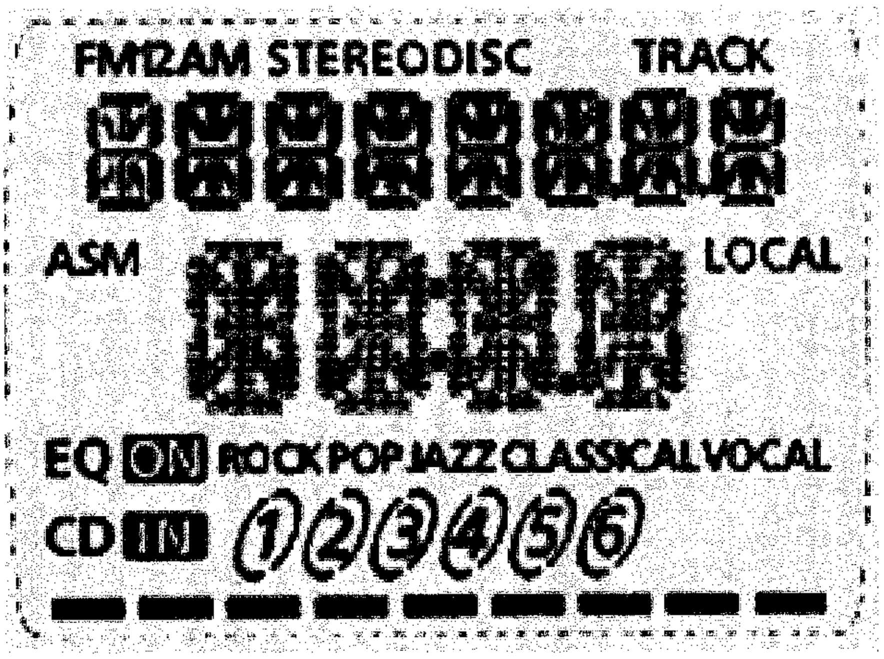
	Test 1	Test 2	Test 2	Test 2	Test 2	Test 3
Pin	1	2	3	4	5	6
56	1	1	1	1	1	0
55	1	1	1	1	1	0
54	1	1	1	1	1	0
53	1	1	1	1	1	0
52	1	0	0	0	0	0
51	1	1	0	0	0	0
50	1	1	0	0	0	0
49	1	1	0	0	0	0
48	1	1	0	0	0	0
47	1	0	0	0	0	0
46	1	1	0	0	0	0
45	1	1	0	0	0	0
44	1	1	0	0	0	0
43	1	0	0	0	0	0
42	1	1	0	0	0	0
41	1	1	0	0	0	0
40	1	1	0	0	0	0
39	1	0	0	0	0	0
38	1	1	0	0	0	0
37	1	1	0	0	0	0
36	1	1	0	0	0	0
35	1	0	0	0	0	0
34	1	1	0	0	0	0
33	1	1	0	0	0	0
32	1	1	0	0	0	0
31	1	0	0	0	0	0
30	1	1	0	0	0	0

29	1	1	0	0	0	0
28	1	1	0	0	0	0
27	1	0	0	0	0	0
26	1	0	0	0	0	0
25	1	0	0	0	0	0
24	1	1	0	0	0	0
23	1	0	0	0	0	0
22	1	1	0	0	0	0
21	1	0	0	0	0	0
20	1	0	0	0	0	0
19	1	0	0	0	0	0
18	1	0	0	0	0	0
17	1	1	0	0	0	0
16	1	0	0	0	0	0
15	1	0	0	0	0	0
14	1	0	0	0	0	0
13	1	1	0	0	0	0
12	1	1	0	0	0	0
11	1	0	0	0	0	0
10	1	0	0	0	0	0
9	1	0	0	0	0	0
8	1	1	0	0	0	0
7	1	1	0	0	0	0
6	1	0	0	0	0	0
5	1	1	0	0	0	0
4	COM4	1	1	1	COM 4	COM4
3	COM3	1	1	COM3	1	COM3
2	COM2	1	COM2	1	1	COM2
1	COM1	COM1	1	1	1	COM1

Table 4.2: The LCD pitch need to be activated

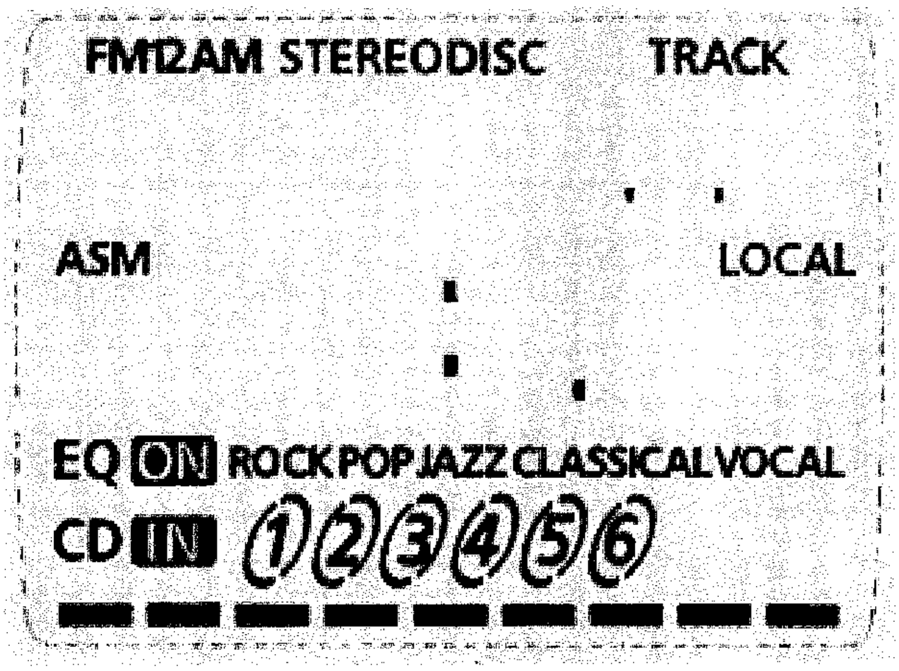
The expected display for the three tests:

Test 1: to on all segments on the LCD



. Figure 4.2: Expected Output Display of Test 1

Test 2: to on the fixed segments



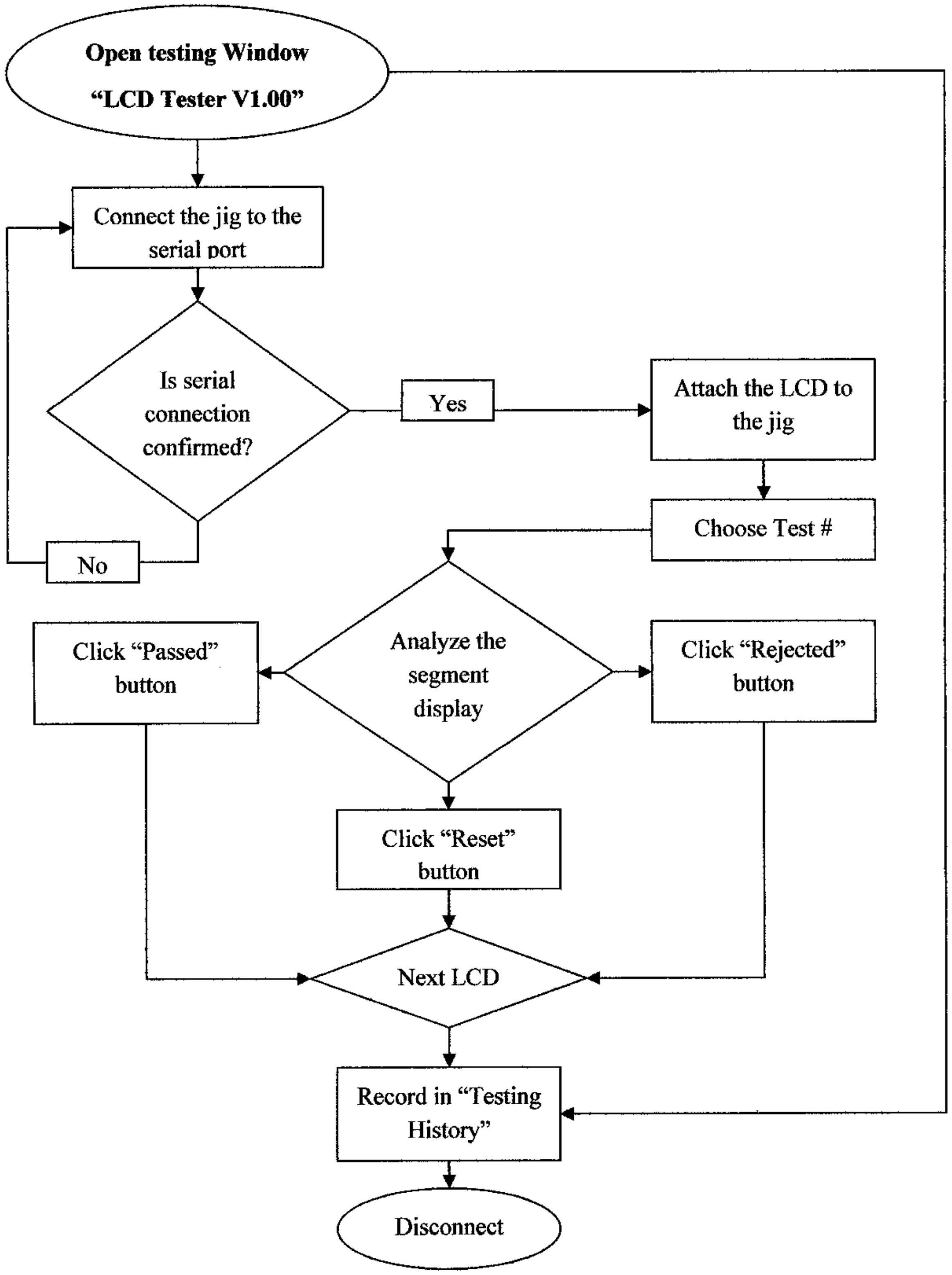
. Figure 4.3: Expected Output Display of Test 2

Test 3: to off all segments.

Figure 4.4: Expected Output Display of Test 3

4.3 Visual Basic Programming

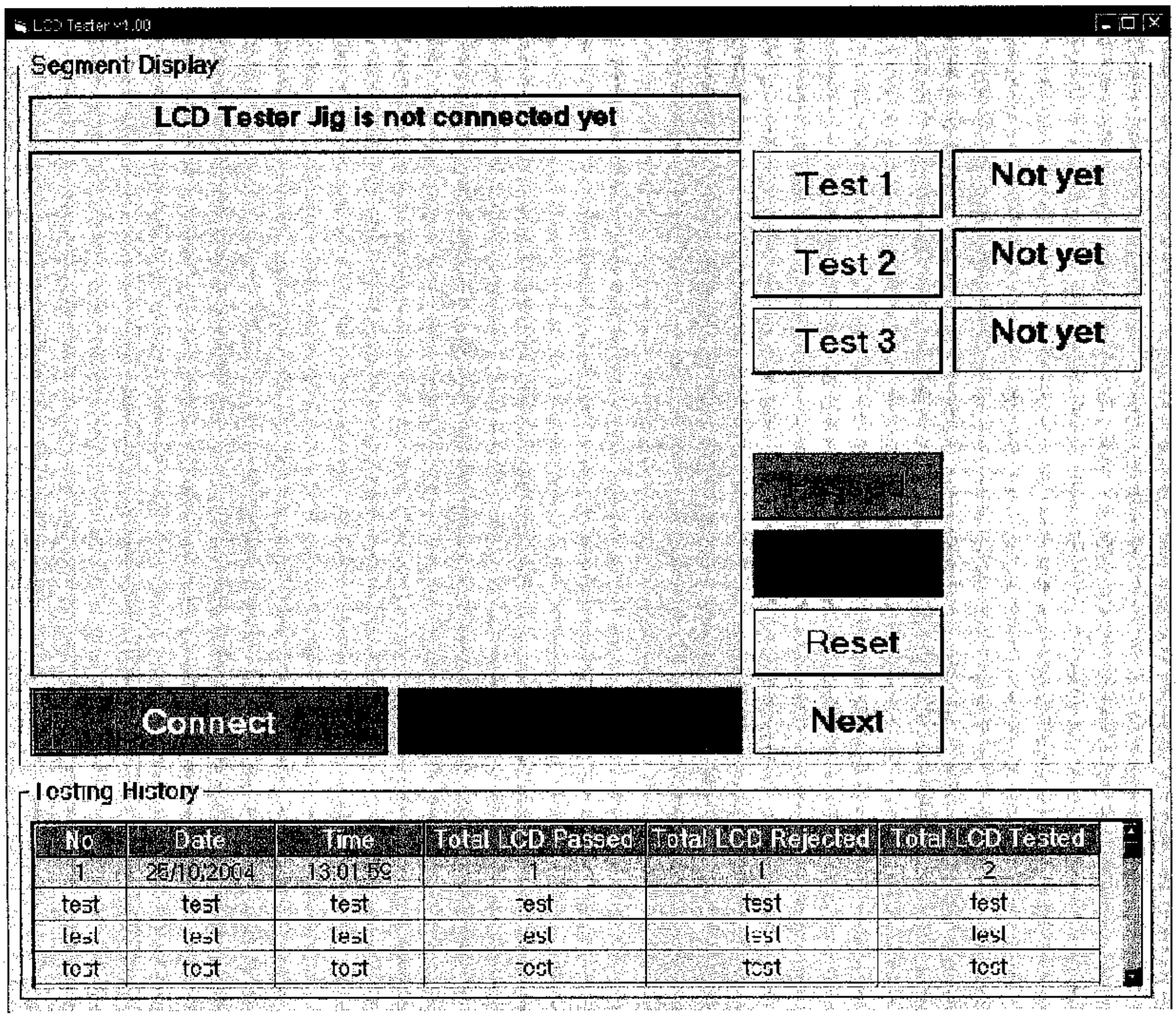
The testing window for this project is created using visual Basing Programming 6.0. The testing window only has one main window. The testing flow of this system is shown in the chart below:



Open testing Window "LCD Tester V1.00"
Connect the jig to the serial port
Is serial connection confirmed?
Yes
No
Attach the LCD to the jig
Choose Test #
Analyze the segment display
Click "Passed" button
Click "Rejected" button
Click "Reset" button
Next LCD
Record in "Testing History"
Disconnect

Figure 4.5: Testing Flow Chart

Each specified segment display will appear as an acknowledgement to the test number when the command button is clicked. Once the “Passed” or “Rejected” button is entered, the test result for that test number will be recorded. If the user not satisfied with the test, “Reset” button can clear the test number and perform that test again.



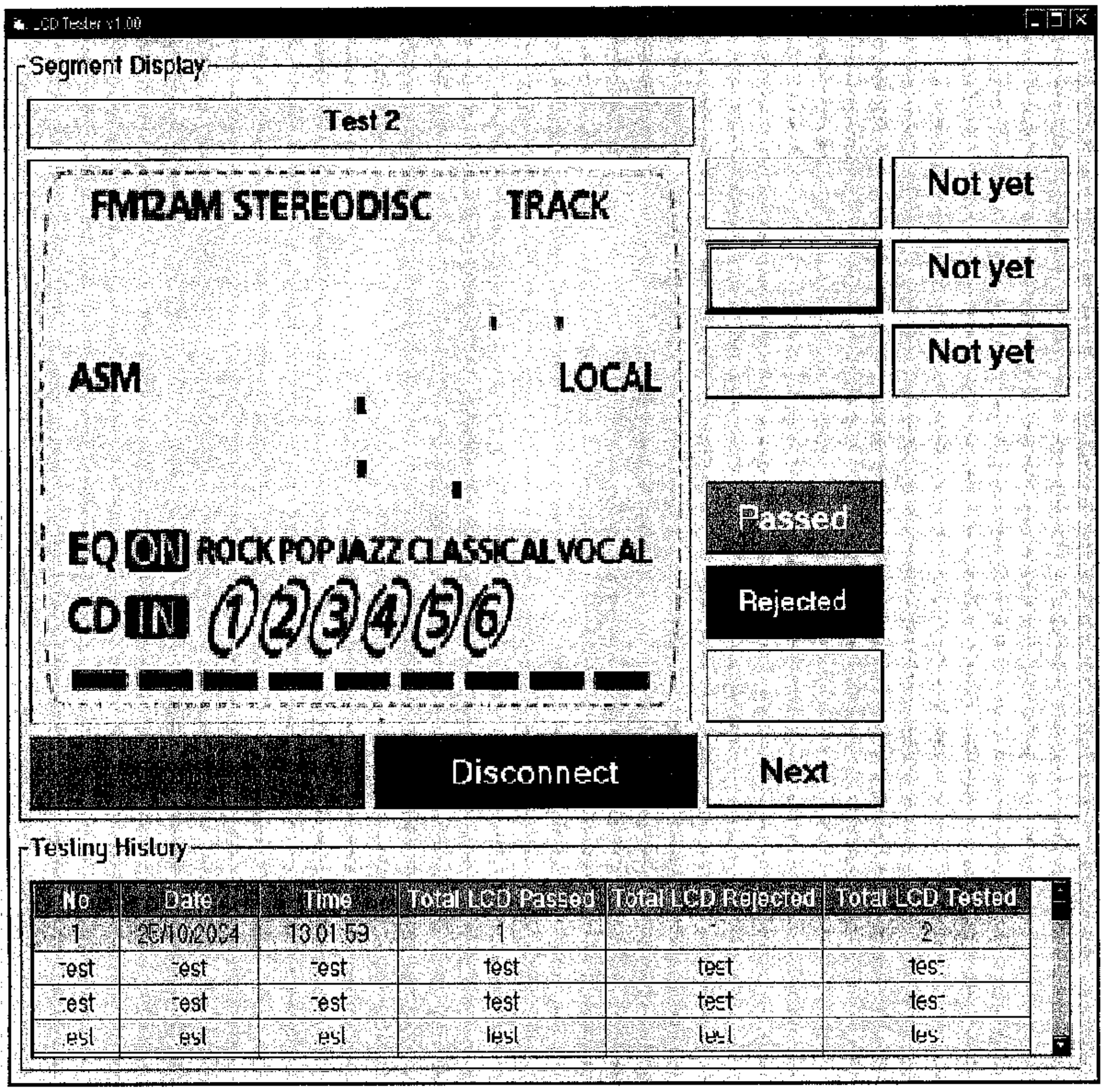
LCD Tester Jig is not connected yet

No	Date	Time	Total LCD Passed	Total LCD Rejected	Total LCD Tested
1	25/10/2004	13:01:59	1	1	2
test	test	test	test	test	test
test	test	test	test	test	test
test	test	test	test	test	test

Figure 4.6: Testing window

In the “Testing History” the result test is recorded according to number of testing done, date, time, total LCD passed, total LCD rejected and also total LCD tested. The latest result will be updated in the testing history database. The result is confirmed after the ‘next’ button is clicked.

The specified segment display is viewed as an acknowledgment to the signal send from the user interfacing window or the testing window to the tester. For example, if the user click the 'Test 2' button, control bit 01 will be sent to serial port, as acknowledgement, the display in Figure 4.11 will appear.



No	Date	Time	Total LCD Passed	Total LCD Rejected	Total LCD Tested
1	25/10/2004	13:01:59	1	1	2
test	test	test	test	test	test
test	test	test	test	test	test
test	test	test	test	test	test

Figure 4.7: Main testing window – The Test 2 is clicked.

Coding for testing window is shown in **Appendix D**.

The jig is look like the figure below.

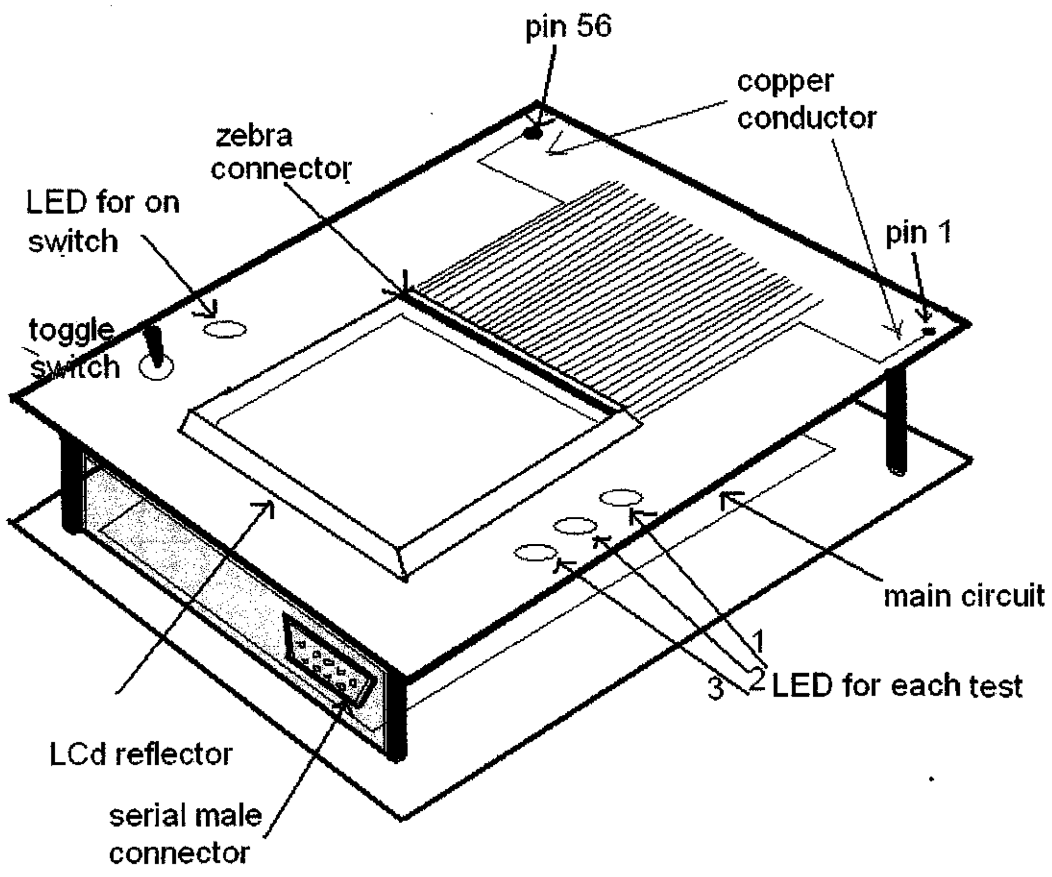


Figure 4.8: LCD Jig Tester

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusions

The jig is able to check the segment display as required. User can assemble the LCD on the jig without continuous side damage. The testing is only done to two sample of LCD. The testing window can help user to implement the test with a smooth testing flow. Besides, the testing flow with this jig can reduce time taken to implement LCD checking. Using PIC can reduce the usage of ICs in the circuit. The user interface also consist of database that can record the test results according to the date, time , total number of LCD passed, total number of LCD rejected and also total number of LCD checked. This database is important to for user to follow up the performance of LCD quality.

5.2 Recommendations

The sample of LCD used in this project is limited due to cost of the LCD. It is better if more samples can be used to perform the LCD checking.

However, I would like to recommend with some extended objectives on this project, so that the next student will take opportunity to achieve the objectives below:

- This project can be applied to another LCD with variety number of LCD pitch.
- The current flow on each LCD pitch can be sense and monitored in testing window.

REFERENCES

[1] http://howstuffworks.com

[2] http://www.elektronika.co.yu/micropic.htm

[3] LCD Datasheet, Varitronix (M) Sdn. Bhd.

[4] http://www.taltech.com/TalTechWeb/resource/Intro-sc.htm

[5] Programming with Visual Basic 6.0, Diane Zak, pg16- 209, Course.

APPENDICES

Appendix A - Coding for PIC16F84A (main controller for PIC16F877)

```
#include <16F84A.h>
#fuses  HS,NOPROTECT,NOWDT
#use    delay(clock=10000000)
#use rs232(baud=9600,xmit=PIN_A0,rcv=PIN_A1)

void main()
{
          int rcv_data;
  output_bit(PIN_A3,1);
  output_bit(PIN_A2,0);

          output_bit(PIN_B0,1);
          output_bit(PIN_B1,1);
          output_bit(PIN_B2,1);

          while(true)
  {
                    rcv_data=getch();
    if (rcv_data==0x41)
    {
              //control bits
                                output_bit(PIN_A3,0);
                                output_bit(PIN_A2,0);
              //LED
                                output_bit(PIN_B0,0);
                                output_bit(PIN_B1,1);
                                output_bit(PIN_B2,1);
      printf("Test1. ");
    }
    else
    if (rcv_data==0x42)
    {
              //control bits
                                output_bit(PIN_A3,0);
                                output_bit(PIN_A2,1);
              //LED
                                output_bit(PIN_B0,1);
                                output_bit(PIN_B1,0);
                                output_bit(PIN_B2,1);
      printf("Test2. ");
    }
    else
    if (rcv_data==0x43)
    {
            //control bits
                            output_bit(PIN_A3,1);
                            output_bit(PIN_A2,0);
            //LED
                            output_bit(PIN_B0,1);
                            output_bit(PIN_B1,1);
                            output_bit(PIN_B2,0);
     printf("Test3. ");
    }
  }
}
```

Appendix B - Coding for PIC16F877 (output for first 28 LCD pitches)

```
#include <16F877.h>
#use    delay(clock=10000000)
#fuses  HS,NOPROTECT,NOWDT,NOLVP
byte       const       test_output[6][28]={
{0,1,1,1,1,0,1,1,0,0,0,1,1,0,0,0,1,0,0,0,0,1,0,1,0,0,0,1},
{1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
 {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
}
void main()
{
          while(true)
  {
                    //Test 1
          if (input(PIN_A1)==0 && input(PIN_A0)==0)
    {
                              test_function(4);
    }
    else
                    //Test 2 (need to refresh 4 cycles)
          if (input(PIN_A1)==0 && input(PIN_A0)==1)
    {
                              test_function(0);
                              test_function(1);
                              test_function(2);
                              test_function(3);
    }
    else
//Test 3
          if (input(PIN_A1)==1 && input(PIN_A0)==0)
    {
                              test_function(5);
    }
  }
}
void      test_function(int base_test)
{
          output_bit(PIN_B0,test_output[base_test][0]);
          output_bit(PIN_B1,test_output[base_test][1]);
          output_bit(PIN_B2,test_output[base_test][2]);
          output_bit(PIN_B3,test_output[base_test][3]);
          output_bit(PIN_B4,test_output[base_test][4]);
          output_bit(PIN_B5,test_output[base_test][5]);
          output_bit(PIN_B6,test_output[base_test][6]);
          output_bit(PIN_B7,test_output[base_test][7]);
          output_bit(PIN_C0,test_output[base_test][8]);
          output_bit(PIN_C1,test_output[base_test][9]);
          output_bit(PIN_C2,test_output[base_test][10]);
          output_bit(PIN_C3,test_output[base_test][11]);
          output_bit(PIN_C4,test_output[base_test][12]);
          output_bit(PIN_C5,test_output[base_test][13]);
          output_bit(PIN_C6,test_output[base_test][14]);
          output_bit(PIN_C7,test_output[base_test][15]);
          output_bit(PIN_D0,test_output[base_test][16]);
          output_bit(PIN_D1,test_output[base_test][17]);
          output_bit(PIN_D2,test_output[base_test][18]);
          output_bit(PIN_D3,test_output[base_test][19]);
          output_bit(PIN_D4,test_output[base_test][20]);
          output_bit(PIN_D5,test_output[base_test][21]);
          output_bit(PIN_D6,test_output[base_test][22]);
          output_bit(PIN_D7,test_output[base_test][23]);
          output_bit(PIN_E0,test_output[base_test][24]);
          output_bit(PIN_E1,test_output[base_test][25]);
          output_bit(PIN_A2,test_output[base_test][26]);
          output_bit(PIN_A3,test_output[base_test][27]);
}
```

Appendix C - Coding for PIC16F877 (output for last 28 LCD pitches)

```
#include <16F877.h>
#use    delay(clock=10000000)
#fuses  HS,NOPROTECT,NOWDT,NOLVP

void        test_function(int base_test);
byte        const       test_output[6][28]={
{1,1,0,1,1,1,0,1,1,1,0,1,1,1,0,1,1,1,0,1,1,1,1,0,1,1,1,1},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1},
 {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1},
 {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1},
  {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
}

void main()
{
          while(true)
  {
                    //Test 1
          if (input(PIN_A1)==0 && input(PIN_A0)==0)
    {
                              test_function(4);
    }
    else
                    //Test 2 (need to refresh 4 cycles)
          if (input(PIN_A1)==0 && input(PIN_A0)==1)
    {
                              test_function(0);

                              test_function(1);
                              test_function(2);
                              test_function(3);
    }
    else
                    //Test 3
          if (input(PIN_A1)==1 && input(PIN_A0)==0)
    {
                              test_function(5);
    }
  }
}
```

```
void        test_function(int base_test)
{
            output_bit(PIN_B0,test_output[base_test][0]);
            output_bit(PIN_B1,test_output[base_test][1]);
            output_bit(PIN_B2,test_output[base_test][2]);
            output_bit(PIN_B3,test_output[base_test][3]);
            output_bit(PIN_B4,test_output[base_test][4]);
            output_bit(PIN_B5,test_output[base_test][5]);
            output_bit(PIN_B6,test_output[base_test][6]);
            output_bit(PIN_B7,test_output[base_test][7]);
            output_bit(PIN_C0,test_output[base_test][8]);
            output_bit(PIN_C1,test_output[base_test][9]);
            output_bit(PIN_C2,test_output[base_test][10]);
            output_bit(PIN_C3,test_output[base_test][11]);
            output_bit(PIN_C4,test_output[base_test][12]);
            output_bit(PIN_C5,test_output[base_test][13]);
            output_bit(PIN_C6,test_output[base_test][14]);
            output_bit(PIN_C7,test_output[base_test][15]);
            output_bit(PIN_D0,test_output[base_test][16]);
            output_bit(PIN_D1,test_output[base_test][17]);
            output_bit(PIN_D2,test_output[base_test][18]);
            output_bit(PIN_D3,test_output[base_test][19]);
            output_bit(PIN_D4,test_output[base_test][20]);
            output_bit(PIN_D5,test_output[base_test][21]);
            output_bit(PIN_D6,test_output[base_test][22]);
            output_bit(PIN_D7,test_output[base_test][23]);
            output_bit(PIN_E0,test_output[base_test][24]);
            output_bit(PIN_E1,test_output[base_test][25]);
            output_bit(PIN_A2,test_output[base_test][26]);
            output_bit(PIN_A3,test_output[base_test][27]);
}
```

Appendix D - Coding for testing window in Visual Basic Programming 6.0

```
'For connection
Public currentport
Public currentbaudrate
Public is_connected

'For tester
Public state_test
Private Sub cmdnext_Click()
   Call WRITE_DB
   Call READ_DB

   textdisplay.Visible = True
   display(0).Visible = False
   display(1).Visible = False
   display(2).Visible = False

   resulttest(0).Text = "Not yet"
   resulttest(1).Text = "Not yet"
   resulttest(2).Text = "Not yet"
   resulttest(0).BackColor = &HC0FFC0
   resulttest(1).BackColor = &HC0FFC0
   resulttest(2).BackColor = &HC0FFC0

   cmdnext.Enabled = False
   cmdpass.Enabled = False
   cmdfail.Enabled = False
   cmdreset.Enabled = False

   testnumber.Text = "History saved and LCD Tester Jig has been reset"

End Sub

Private Sub Form_Load()
   'Title initialization
   Me.Caption = App.Title & " v" & App.Major & "." & App.Minor & App.Revision

   'autopath
   Data1.DatabaseName = App.Path & "\Database\testing_history.mdb"

   'Read from database
   Call READ_DB
```

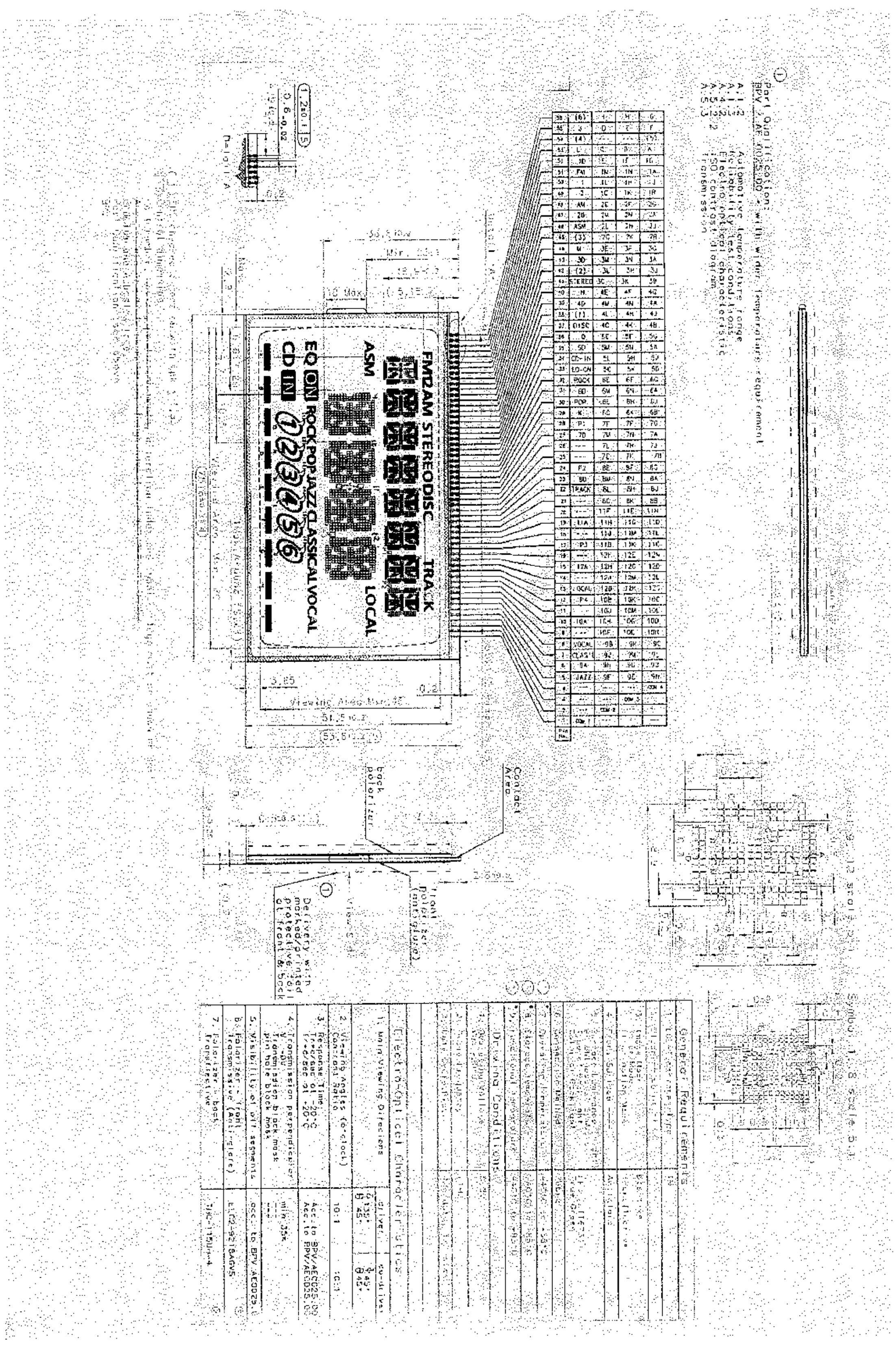
```
    'Object initialization
    Call BUTTON_ENABLE(False)

    'Grid initialization
    flex_history.ColWidth(0) = 1000
    flex_history.ColWidth(1) = 1600
    flex_history.ColWidth(2) = 1600
    flex_history.ColWidth(3) = 2400
    flex_history.ColWidth(4) = 2500
    flex_history.ColWidth(5) = 2400

    flex_history.TextMatrix(0, 0) = "No"
    flex_history.TextMatrix(0, 1) = "Date"
    flex_history.TextMatrix(0, 2) = "Time"
    flex_history.TextMatrix(0, 3) = "Total LCD Passed"
    flex_history.TextMatrix(0, 4) = "Total LCD Rejected"
    flex_history.TextMatrix(0, 5) = "Total LCD Tested"

    flex_history.ColAlignmentFixed(0) = 4
    flex_history.ColAlignmentFixed(1) = 4
    flex_history.ColAlignmentFixed(2) = 4
    flex_history.ColAlignmentFixed(3) = 4
    flex_history.ColAlignmentFixed(4) = 4
    flex_history.ColAlignmentFixed(5) = 4

    flex_history.ColAlignment(0) = 4
    flex_history.ColAlignment(1) = 4
    flex_history.ColAlignment(2) = 4
    flex_history.ColAlignment(3) = 4
    flex_history.ColAlignment(4) = 4
    flex_history.ColAlignment(5) = 4

    'Connection
    currentport = 1
    currentbaudrate = 9600
    is_connected = False

End Sub
```

```
Private Sub cmdconnect_Click()
    Call SERIAL_CONNECT


    'If successful
    If is_connected = True Then


      'Enable buttons
      Call BUTTON_ENABLE(True)


    End If
End Sub

Private Sub cmddisconnect_Click()
    Call SERIAL_DISCONNECT
    Call BUTTON_ENABLE(False)
End Sub

Private Sub SERIAL_CONNECT()
  'set the active serial port
  MSComm1.CommPort = currentport

  'set the baudrate,parity,databits,stopbits for the connection
  MSComm1.Settings = currentbaudrate & ",N,8,1"

  'enable the oncomm event for every received character
  'RThreshold=1,comEvReceive=enabled
  'RThreshold=0,comEvReceive=disabled
  MSComm1.RThreshold = 1

  'disable the oncomm event for send characters
  'SThreshold=1,comEvSend=enabled
  'SThreshold=0,comEvSend=disabled
  MSComm1.SThreshold = 0

  On Error GoTo errorhandler
  'open the serial port
  MSComm1.PortOpen = True
  is_connected = True

  'This exit sub is to prevent the normal flow (without error) goes into error handler
  Exit Sub

errorhandler:
 a1 = MsgBox(Err.Description & vbCrLf & "[Error no. = " & Err.Number & "]", vbExclamation, "Error")
 is_connected = False

End Sub

Private Sub SERIAL_DISCONNECT()

 'Close port if and only if it is currently connected
 If is_connected = True Then

   MSComm1.PortOpen = False
   is_connected = False
 End If
End Sub

Private Sub BUTTON_ENABLE(state)
 If state = True Then

   textdisplay.Visible = True
   testnumber.Text = "LCD Tester Jig is already connected"
   cmdtest(0).Enabled = True
   cmdtest(1).Enabled = True
   cmdtest(2).Enabled = True

   resulttest(0).Text = "Not yet"
   resulttest(1).Text = "Not yet"
   resulttest(2).Text = "Not yet"
   resulttest(0).BackColor = &HC0FFC0
   resulttest(1).BackColor = &HC0FFC0
   resulttest(2).BackColor = &HC0FFC0

   cmdpass.Enabled = False
   cmdfail.Enabled = False
   cmdreset.Enabled = False
   cmdnext.Enabled = False

   cmddisconnect.Enabled = True
   cmdconnect.Enabled = False
```

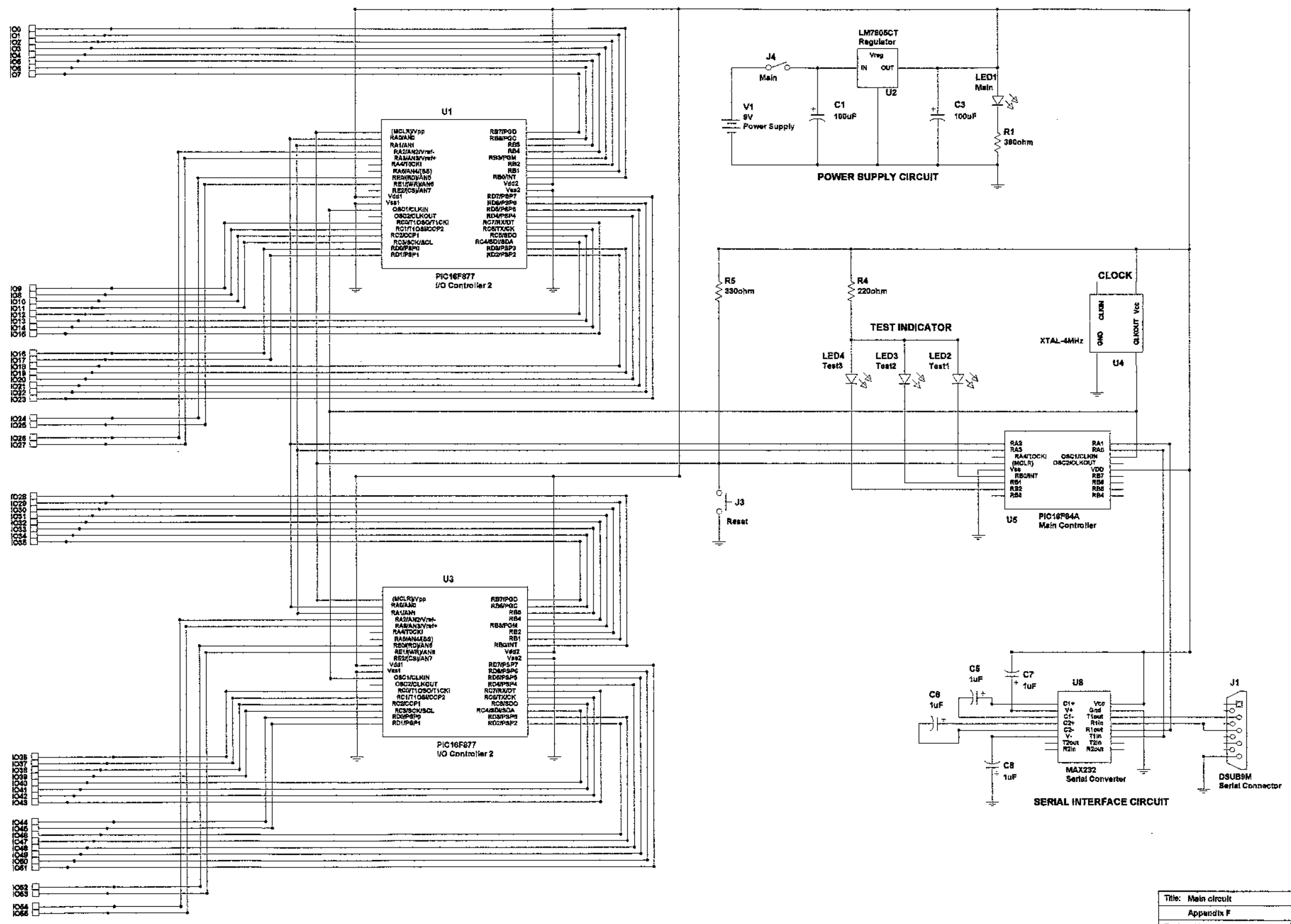
```
        state_test = 0
    ElseIf state = False Then

        textdisplay.Visible = True
        display(0).Visible = False
        display(1).Visible = False
        display(2).Visible = False

        testnumber.Text = "LCD Tester Jig is not connected yet"
        cmdtest(0).Enabled = False
        cmdtest(1).Enabled = False
        cmdtest(2).Enabled = False

        resulttest(0).Text = "Not yet"
        resulttest(1).Text = "Not yet"
        resulttest(2).Text = "Not yet"
        resulttest(0).BackColor = &HC0FFC0
        resulttest(1).BackColor = &HC0FFC0
        resulttest(2).BackColor = &HC0FFC0

        cmdpass.Enabled = False
        cmdfail.Enabled = False
        cmdreset.Enabled = False
        cmdnext.Enabled = False

        cmddisconnect.Enabled = False
        cmdconnect.Enabled = True

        state_test = 0
    End If
End Sub

Private Sub cmdtest_Click(Index As Integer)

    If Index = 0 Then
        textdisplay.Visible = False
        display(0).Visible = True
        display(1).Visible = False
        display(2).Visible = False

        cmdpass.Enabled = True
        cmdfail.Enabled = True
        cmdreset.Enabled = True
        testnumber.Text = "Test 1"

        state_test = 1
    ElseIf Index = 1 Then
        textdisplay.Visible = False
        display(0).Visible = False
        display(1).Visible = True
        display(2).Visible = False

        cmdpass.Enabled = True
        cmdfail.Enabled = True
        cmdreset.Enabled = True

        testnumber.Text = "Test 2"

        state_test = 2

    ElseIf Index = 2 Then
        textdisplay.Visible = False
        display(0).Visible = False
        display(1).Visible = False
        display(2).Visible = True

        cmdpass.Enabled = True
        cmdfail.Enabled = True
        cmdreset.Enabled = True

        testnumber.Text = "Test 3"

        state_test = 3
    End If

End Sub

Private Sub cmdreset_Click()
    textdisplay.Visible = True
    display(0).Visible = False
    display(1).Visible = False
    display(2).Visible = False

    resulttest(0).Text = "Not yet"
    resulttest(1).Text = "Not yet"
    resulttest(2).Text = "Not yet"
    resulttest(0).BackColor = &HC0FFC0
    resulttest(1).BackColor = &HC0FFC0
    resulttest(2).BackColor = &HC0FFC0

    cmdnext.Enabled = False
    cmdpass.Enabled = False
    cmdfail.Enabled = False
    cmdreset.Enabled = False

    testnumber.Text = "LCD Tester Jig has been reset"
End Sub

Private Sub cmdpass_Click()

    If state_test = 1 Then
        resulttest(0).Text = "Passed"
        resulttest(0).BackColor = &HC000&

    ElseIf state_test = 2 Then
        resulttest(1).Text = "Passed"
        resulttest(1).BackColor = &HC000&

    ElseIf state_test = 3 Then
        resulttest(2).Text = "Passed"
        resulttest(2).BackColor = &HC000&
    End If

    cmdnext.Enabled = True
End Sub

Private Sub cmdfail_Click()

    If state_test = 1 Then
        resulttest(0).Text = "Rejected"
        resulttest(0).BackColor = &HFF&

    ElseIf state_test = 2 Then
        resulttest(1).Text = "Rejected"
        resulttest(1).BackColor = &HFF&

    ElseIf state_test = 3 Then
        resulttest(2).Text = "Rejected"
        resulttest(2).BackColor = &HFF&

    End If

    cmdnext.Enabled = True
End Sub

Private Sub READ_DB()

    'Count no of data
    db_counter = 0
    Data1.Refresh
    Data1.Recordset.MoveFirst
    Do While Not Data1.Recordset.EOF
        db_counter = db_counter + 1

        Data1.Recordset.MoveNext
    Loop

    total_db = db_counter
    flex_history.Rows = total_db + 1

    'Read data
    db_counter = 0
    Data1.Refresh
    Data1.Recordset.MoveFirst
    Do While Not Data1.Recordset.EOF
        db_counter = db_counter + 1

        flex_history.TextMatrix(db_counter, 0) = Data1.Recordset.Fields(0)
        flex_history.TextMatrix(db_counter, 1) = Data1.Recordset.Fields(1)
        flex_history.TextMatrix(db_counter, 2) = Data1.Recordset.Fields(2)
        flex_history.TextMatrix(db_counter, 3) = Data1.Recordset.Fields(3)
        flex_history.TextMatrix(db_counter, 4) = Data1.Recordset.Fields(4)
        flex_history.TextMatrix(db_counter, 5) = Data1.Recordset.Fields(5)

        Data1.Recordset.MoveNext
    Loop
End Sub
```

```
Private Sub WRITE_DB()
  'Write to database
  Data1.Recordset.AddNew
  Data1.Recordset.Fields(0) = "test"
  Data1.Recordset.Fields(1) = "test"
  Data1.Recordset.Fields(2) = "test"
  Data1.Recordset.Fields(3) = "test"
  Data1.Recordset.Fields(4) = "test"
  Data1.Recordset.Fields(5) = "test"
  Data1.Recordset.Update
End Sub
```

Appendix E- LCD Datasheet



Appendix F- Main circuit



Title: Main circuit		
Appendix F		
Designed by: Norhafiza	Document N 0001	Revision 1,0
Checked by: Azizan Hashim	Date June 18, 2004	Size A2
Approved by: Abu Bakar	Sheet 1 of 1	

Appendix G - Gantt chart for the 2 Semesters Final Year Project

Appendix G- Gantt chart for the 2 Semesters Final Year Project

No.	Detail/ Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	Selection of Project Topic																									
	-Propose Topic	■																								
2	Preliminary Research Work		■																							
	-Introduction																									
	-Objective																									
	-List of references/literature			■																						
	-Project planning			■	■																					
3	Project Work																									
	-Reference/Literature					■	■																			
	-understand system breakdown						■	■																		
	-understand the testing sequences and expected result																									
4	Design the testing window and coding (VB)																									
	- self study on Visual Basic							■	■	■	■	■	■	■	■	■	■	■								
	-testing window coding and debugging											■	■	■	■	■	■	■	■	■	■					
5	Practical/Laboratory Work																									
	- circuit design and troubleshooting										■	■	■	■	■	■	■	■	■	■	■	■	■			
6	Hardware and software interfacing																									
	-coding for PIC16F84A (main controller)																		■	■	■	■				
	-coding for PIC16F877 (output)																				■	■	■	■		
7	Troubleshooting and finishing																						■	■	■	■

■ Process ▨ Time line for the first semester ▨ Time line for the second semester

Appendix H - PIC16F84A Datasheet

PIC16F84A

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

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

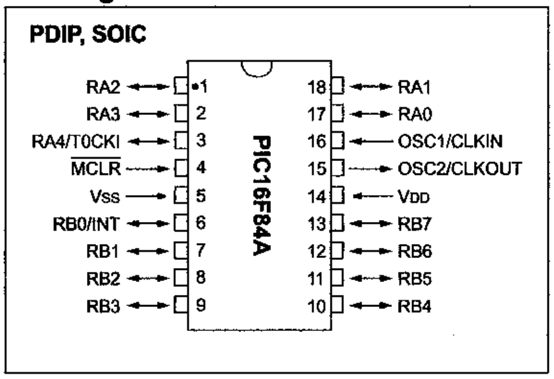
Peripheral Features:

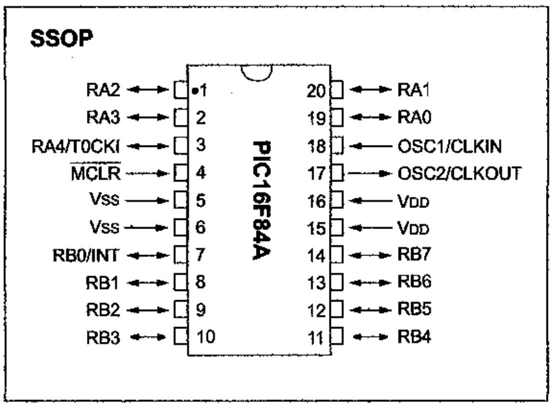
- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Special Microcontroller Features:

- 10,000 erase/write cycles *Enhanced* FLASH Program memory typical
- 10,000,000 typical erase/write cycles EEPROM Data memory typical
- 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





CMOS Enhanced FLASH/EEPROM 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

Appendix I - PIC16F877 Datasheet

MICROCHIP

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

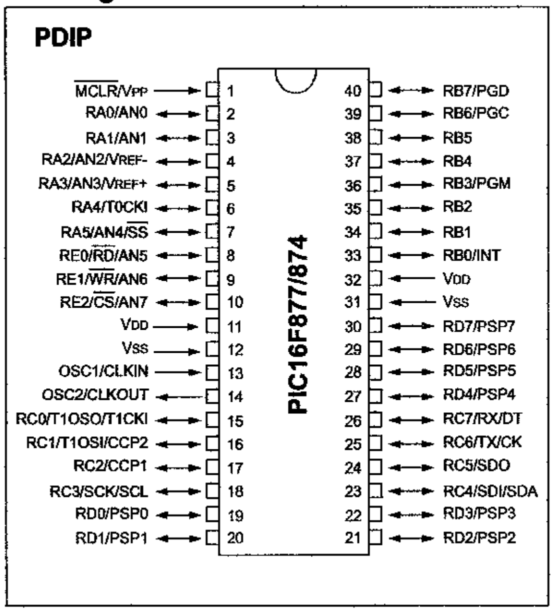
Devices Included in this Data Sheet:

- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 μA typical @ 3V, 32 kHz
  - < 1 μA typical standby current

Pin Diagram



Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^{2}C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)