

**DESIGN AND CONSTRUCTION OF A MICROCOMPUTER
BASED, 14 PINS DIGITAL IC TESTER.**

BY

INUWA. M. BALA
96/5111EE

SUBMITTED TO

**THE DEPARTMENT OF ELECTRICAL AND COMPUTER
ENGINEERING,
SCHOOL OF ENGINEERING AND ENGINEERING
TECHNOLOGY,
FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA.**

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE
AWARD OF BACHELOR'S DEGREE IN ELECTRICAL AND
COMPUTER ENGINEERING

FEBRUARY 2002.

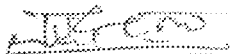
**DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
SCHOOL OF ENGINEERING AND ENGINEERING
TECHNOLOGY
FEDERAL UNIVERSITY OF TECHNOLOGY MINNA,
NIGER STATE.**

A PROJECT SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIREMENT FOR THE AWARD OF THE BACHELOR OF
ENGINEERING (B. ENG) IN ELECTRICAL AND COMPUTER
ENGINEERING.

.....
Candidate

INUWA. M. BALA

.....
Date

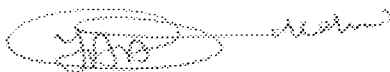


.....
SUPERVISOR

Engr. J. Kolo

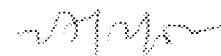
.....
27/2/202

.....
Date



.....
Head of department

Dr. (Engr.) Y. A. ADEDIRAN



.....
Date

.....
External Examiner

.....
Date

DEDICATION

This project is dedicated to the Almighty Allah (SWT), the Most High, the giver and sustainer of life; whose Grace, Glory and Blessings saw me through the rigors of university life. Moreover, to my late Grand parents *Alh. Garba Angulu* and *Mallamu Zainab Garba*. May their souls rest in perfect peace (Amen).

ACKNOWLEDGEMENT

All praises be to Allah (SWT) for all the abundant blessings and protection given to me for a smooth completion of my Academic work.

My profound gratitude goes to all that contributed immensely for my upliftment. I am particularly grateful to my beloved parents *Alhaji Wada Garba Lapai* and *Hajiyu Fatima Uwa Wada* for their infinite support and contributions throughout my stay in the University. To my dear Uncle Alhaji Isa Iya Pai for his moral and financial support, you are indeed great.

My sincere appreciation goes to the entire members of the family especially, Mrs. Gambo Aliyu, Mrs. Aminat shuaibu, Mrs. Hussainat Isah, Mal. Haruna Garba, Mal. Danladi Mohammed Inurwa, Miss Hafsat Wada, Master Abdullahi Wada, Miss Zainab Wada, Mrs. Pricilla Tina Zuwanden, etc you are all wonderful!

Special thanks to the management and staff of Dee – great computers and Telecomms Engineering, Minna. Especially Engr. Ernest. D. Essienton (M/D), Engr. Waheed. A. Salami (System Engineer), Miss Mercy Shalom (Secretary) and Mr. F. P. Etuk (NITEL Agaie), for their priceless support.

I wish to use this opportunity to say a big thank you to the entire members of staff of the Electrical and Computer Engineering department, especially the HOD and my project supervisor Engr. Jonathan Kolo, for believing in me throughout the period of this project.

My thanks goes to friends and colleagues such as Jimada checheko, Raini Kabira Alade, Nada Adam Suleiman, the entire members of Estate palace Bosso, and many others too numerous to mention, for their various forms of support and encouragement.

Finally, to every other person I could not acknowledge due to space, thanks for everything, the memory lingers on. May Allah (SWT) bless you all abundantly.

DECLARATION

I hereby declare that this project is an Original work of mine and has never been presented in any form for the award of diploma or degree certificate. All information derived from published and unpublished work has been duly acknowledged



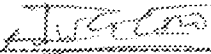
INUWA. M. BALA
96/5111EE

27-02-2008.

Date

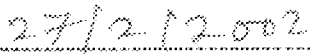
CERTIFICATION

This is to certify that this project was carried out by Inuwa. M. Bala of the department of Electrical and Computer Engineering, Federal University of Technology Minna under the Supervision of Engr. J. Kolo.



Supervisor
Engr. J. Kolo

Head of department
Dr. (Engr.) Y. A. Adediran.



Date

Date

ABSTRACT

The single most significant development in digital system design in recent years has been the microprocessor, a central processing unit integrated on single chip of silicon. The processing power and economics of the microprocessor have had a tremendous impact on the way digital systems are designed and their scope of application.

This project is about the microprocessor. Its related integrated circuit. The hardware design of microprocessor based systems. Its purpose is first to provide a thorough understanding of the basic hardware and software concepts necessary for the design of the microprocessor based systems, and to further provide an in depth knowledge of specific devices. The attendant practical considerations and design effectively systems using them.

The goal of this project has been to introduce the necessary hardware and software concepts in an elementary, systematic, and integrated fashion and to build upon these concepts logically.

A microcomputer system is made up of different types of components, which includes a microprocessor, memory units, input/output registers and peripheral devices. The interconnection of these components takes into account the nature and timing of the signal that appears at the interfaces between components for the purpose of achieving compatibility of signals. This is referred to as *interfacing*. Microcomputer system is divided into two areas of concern:

One area involves the interconnection of the components such as memory unit and input/output registers to the uses of a microprocessor. Such interfacing is concerned with the timing and control of the buses and the selection of component so as to effect data transfer at a given time between the selected component and the microprocessor.

The other area of concern involves interfacing external components to the microcomputer, such as peripheral devices, data channel, and controllers. Such interfacing does not directly involve the buses of the microprocessor.

The project at hand is a type and form of the first mentioned, and has its statements as follows. the design and implementation of the hardware and software of a microcomputer based 14-pin digital IC tester. The device has six data bus and three control lines interfaced with *IBM PC-AT ISA PORT* via an expansion slot. The port timing is to be software programmed using C++ builder.

TABLE OF CONTENTS

Cover Page	i	
Title Page	ii	
Dedication	iii	
Acknowledgement	iv	
Declaration	v	
Certification	vi	
Abstract	vii	
Table of Content	ix	
CHAPTER ONE		
INTRODUCTION AND LITERATURE REVIEW		
1.0 Introduction	1	
1.1 Aims And Objectives	2	
1.2 Literature Review	2	
CHAPTER TWO		
2.0 CIRCUIT METHODOLOGY AND HARDWARE		4
2.1 Octal D Flip Flop	5	
2.1.2 Transparent D Latch	6	
2.1.3 Three State Buffer	7	
2.1.4 Inverter or NOT gate	9	
2.2 Decoder	10	
2.2.1 Open Collector Buffer	10	
2.3 Parallel Port Interfaces	11	
2.3.1 Handshaking	13	
2.4 The Microcomputer Bus Structure	14	
2.5 The Power Supply	15	
2.6 Circuit Diagram	16	

CHAPTER THREE

CONSTRUCTION PROCEDURE

3.1 Tools Used	17
3.2.1 Mode of Operation	18
3.2.2 Software	19
3.3 Testing	20
3.4 Result	20
3.5 Discussion of Result	21
3.5.1 Limitations	22
3.5.2 Problems encountered in the construction of the Project	22

CHAPTER FOUR

CONCLUSION AND RECOMMENDATION

4.1 Conclusion	23
4.2 Recommendation	23
References	24
Appendix	25

CHAPTER ONE

INTRODUCTION AND LITERATURE REVIEW

1.0 INTRODUCTION

This century witnessed great inventions of man resulting into an age where the computer is infiltrating all aspects of our working lives. We are accustomed to its prominence in the world of business and finance where it has assumed many of the books keeping and communication functions of most organizations and companies. It has become an essential tool for engineering design and now it is rapidly becoming indispensable for manufacturing especially, in this era, where the task of manually evaluating complicated system of interrelated activities is almost impossible.

Before now, computers are used to store and process large amount of information, and to solve complicated mathematical problems. As newer technology evolved, additional activities were added to this list. These include controlling systems and devices, simulation, information management system, military, space, etc.

The age long inventory control function can now be monitored constantly by a real time computerized system. Presently, Industrial Engineers use computers in production and test equipment and for analyses and special studies. This project will provide an insight on a new way of using computer to interface through the parallel (lpt.) port.

The micro computer based, 14-pins digital IC tester, is inevitable in the world of industrial instrumentation and reliability management, digital integrated circuit monitoring becomes important especially in the cause of components trouble-shooting in which it is always required to diagnose the IC state before any replacement is carried out to faulty device or equipment.

Using a computer to test a digital IC gives a good degree of sensitivity, speed and accuracy and an insight to industrial instrumentation. It is paramount therefore, to note that, if the computer can be used to test or monitor the Output State of an IC, then it could be made to control or even monitor an entire system.

1.1 AIMS AND OBJECTIVES

The aim of this project is to produce a microcomputer-based digital IC testing device, which can be used as:

- (i) A monitoring device to test the output voltage levels of digital ICs.
- (ii) An insight into interfacing through the parallel port of an IBM compatible PC.
- (iii) A step in achieving microcomputer digital IC instrumentation.

1.2 LITERATURE REVIEW.

Keeping in line with the trend that has become synonymous with the field of engineering, that is, getting much work done with minimal efforts, control system engineering has thrived tremendously upon this philosophy and several opportunities have been opened up for the advancement of humanity.

Since the 1950s, system theory has evolved from a semiheuristic discipline directed toward the design and analysis of electronic and / or mechanical systems consisting of a handful of components into a very sophisticated technology capable of treating complex and large systems with myriad applications. The theory must deal not only with electronic and mechanical systems, the complexities of which have increased by several orders of magnitudes, but also with vast number of real life systems in society, the economy, industry, and government.

The rapid development of powerful computers, the advent of computer graphics, and the ability to interface computers and machine tools have had a profound effect on modern industry. Computer control has made possible numerically controlled machines, robots, and flexible manufacturing systems and will lead to more comprehensive automation in the near future.

Initially system engineers attempted to cope with this increasing system complexity through the development of sophisticated numerical techniques in order to apply classical systems theory to large systems. This approach reached a point of diminishing returns, and it became apparent that new system theoretical techniques would be necessary to handle large and complex systems. However, through the use of simulation on a computer, answers can be given to "what if" types of questions which provide significant input to the decision maker.

Over the years, tremendous efforts have been put in place by many instrumentation engineers to monitor or measure variations in electrical and mechanical quantities, giving birth to enhanced monitoring devices that can test discrete voltage levels.

The Electrical Research Association (*ERA*) in the United Kingdom first developed a logic probe to test the *HIGH-LOW* transition state of a digital IC, although it was manually operated it actually served the purpose it was meant for.

This project therefore, is an advancement in this regard, where a micro-computer is used to conduct the testing automatically via a channel linking the parallel port with a hard-wired logical probe circuit. The port timing is to be software-programmed using *C++ builder*.

CHAPTER TWO

2.0 CIRCUIT METHODOLOGY AND HARDWARES

The microcomputer based 14-pins digital IC tester (*MBDIT-14*) is a peripheral hardware connected to the parallel port of an IBM compatible PC.

The device receives and transmits data through the process of *READ* or *WRITE* controlled by software, device driver. Thus it can be said to have two principal aspects hardware and software

The device driver makes it possible for the computer to recognize incoming data and then analyses it or even control the external hardware, in this case the logic probe unit (*LPU*).

Its principle of operation is based on the *READ/WRITE* operation of the computer effected by sending control signal on to the computer's control line, which is generated by the software. At this time, the computer bi-directional data bus receives or sends data either to the *LPU* or the computer based on what is set on the control line. The data is usually logic voltage levels (i.e. +5v as *HIGH* and 0v as *LOW*), since, the variables is meant to test are digital ICs

The intelligent peripheral hardware is made up of two units namely:

- (i) The Read/Write unit
- (ii) The probe unit

Which are wired bi-directionally to the microcomputer, as could be seen in figure 2.0.

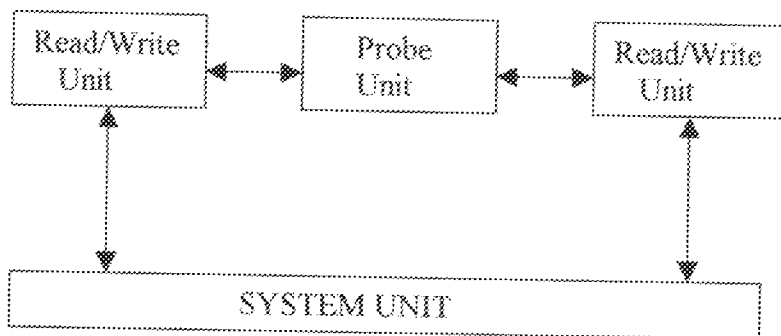


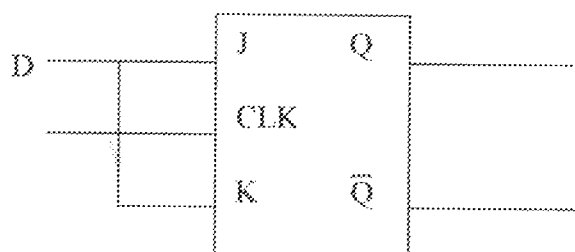
Fig 2.0 Block diagram of the microcomputer based 14-pins digital IC tester.

The Read/Write unit consists of basically octal transparent D-latch, tristate buffer, inverter, and decoder while the probe unit is made up of the IC socket (probe) and open collector buffer.

2.1 OCTAL D FLIP FLOP

2.1.1 In a synchronous digital system it is often necessary to have signal delayed by exactly one cycle of the clock and a specified flip flop is available for this operation called the D (delay) flip-flop.

The output after a clock pulse equals the input before the pulse. In figure 2.0 are shown the symbol and characteristic table. Optional are the clear-preset inputs and the complement \bar{Q} of the output. A D flip-flop can be made from a JK flip-flop by connecting the J and K inputs, with the connection serving as the data input when periodic pulses are applied to the clock input, the output is that of the input delayed by one clock pulse. As in below:



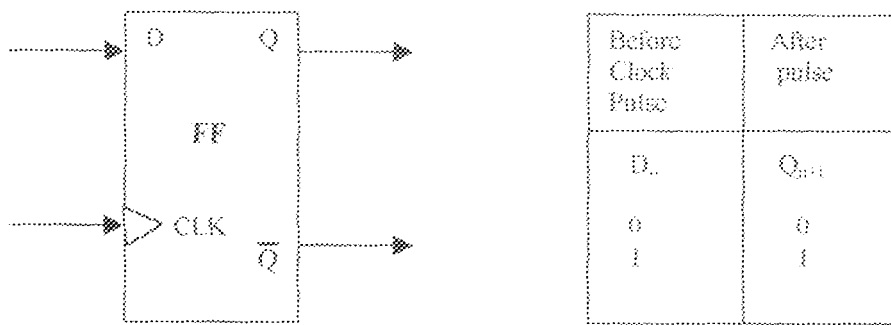


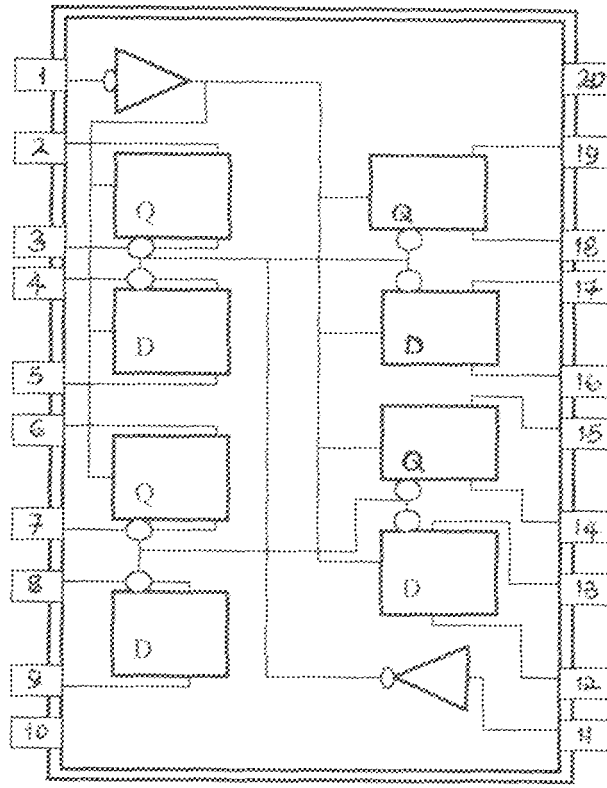
Fig 2.1 D flip-flop and characteristic table.

2.1.2 TRANSPARENT D LATCH

A transparent D latch otherwise called clocked D latch differs from a D flip-flop in that one-bit delay is eliminated. The network is such that when the clock pulse triggers the gate, the output is coupled directly to the input D, and Q equal D. The output is then held or latched, in this state until the next pulse triggers the gate. The clock simply acts as an enable input to the latch. It has important application in registers especially for temporary data storage.

The figure below provides the logic diagram and function table for the 74LS373 octal D-type transparent latch.

Output enable	Enable latch	D	Output
0	1	1	1
0	1	0	0
0	0	x	Q_n
1	x	x	Z



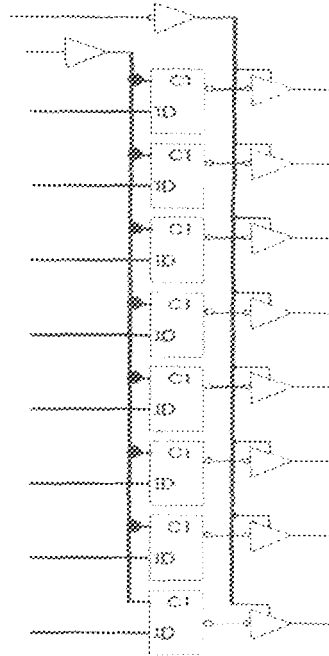
*Figure 2.1.1 function table and logic diagram of 74LS373 Octal
Transparent D latch.*

2.1.3 THREE STATE BUFFERS

These are buffers with three output states. Two of the states are the common logic states 0 and 1. When the buffer is enabled by the control input its output is either logic 0 or 1. The output impedance is low when the buffer is enabled, allowing the output to source or sink current to the circuits it drives. The third state, which exists when the buffer is disabled, does not provide a conventional logic level output but, rather, causes the output to be a very high impedance. This high impedance state prevents the output from driving or loading any circuit connected to it. Thus, when

disabled, three-state outputs are electrically disconnected from any logic circuits to which they are physically connected and are said to be floating.

The figure below provides the logic diagram and function table for the 74LS244 octal non-inverting 3-state buffer.



	Inputs				Outputs			
G	A1	A2	A3	A4	Y1	Y2	Y3	Y4
1	X	X	X	X	Z	Z	Z	Z
0	X	X	X	X	A1	A2	A3	A4

Figure 2.1.2 logic diagram and function table of 74LS244-tristate buffer.

2.1.4 INVERTER OR NOT GATE

This is the simplest gate, with one input and one output. Its action is to produce a *HIGH* level at output (1), if the input is *LOW* (0), and vice-versa. That is, the output goes *LOW* if the input is *HIGH*. The circuit implementation, logic diagram and truth table is provided in the figure below.

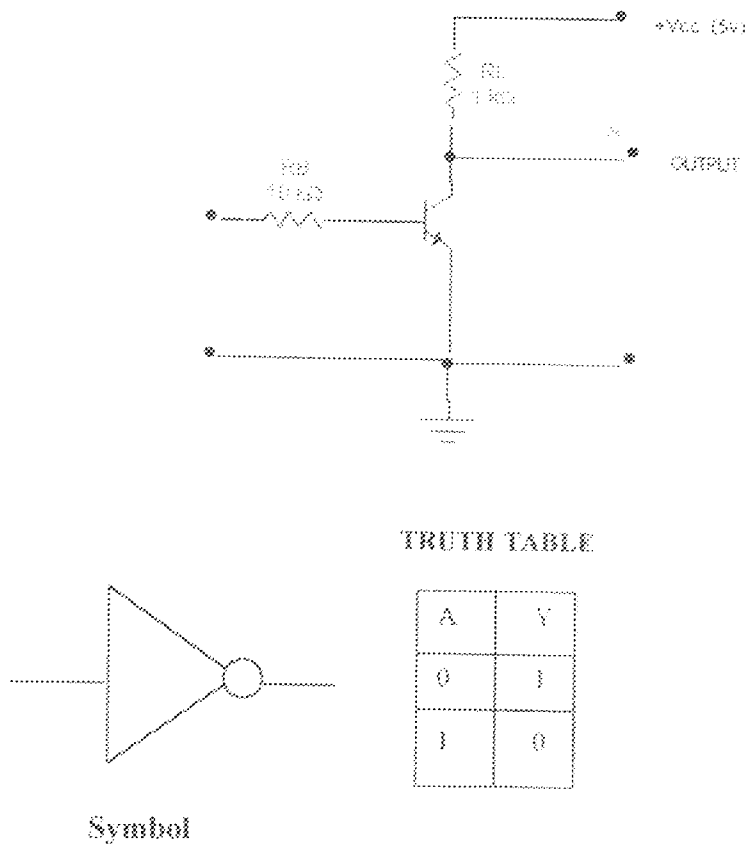
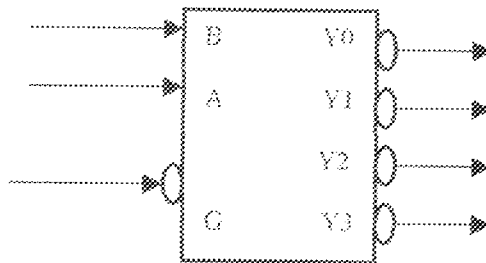


Figure 2.1.3 Diagram showing circuit implementation, logic representation and truth table of an inverter.

2.2 DECODER

The decoder is an essential component of every random access memory. The decoder has n input lines and 2^n output lines. One and only one output line will have the value logical 1 for each combination of input values. A simple implementation of a 2 to 2^2 or 2-bit decoder is illustrated below.



INPUTS			OUTPUTS			
ENABLE	SELECT					
G	B	A	Y ₀	Y ₁	Y ₂	Y ₃
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Fig 2.2 Logic symbol and function table 74LS139 decoder

2.2.1 OPEN COLLECTOR BUFFER

The collector of an output transistor is wired in series with a resistor, a value of $1k\Omega$ being typical. The resistor is known as pull-up resistor. If outputs of several open collector buffers are tied together with single pull-up resistor each, then, wired logic is performed. This is so, in order to allow several devices share a common bus. At any time all but one of the gates on the bus are in their *HIGH* State. The remaining

gate may stay in the *HIGH* State or drive the bus to the *LOW* State depending on whether it want to transmit a logical 1 or 0 on the bus.

The pull-up resistor plays no part other than drawing an inconsequential 5mA from the supply. When the input of the transistor is *LOW*, the path to 0v through the transistor is lifted and the point A is pulled-up toward +5v, i.e. logic 1 and at saturation the point A of the transistor goes *LOW*. This type of transistor output may be connected directly or through a tristate buffer to a computer input port, the 1k Ω pull-up resistor limits the current into the port to a safe value of no more than 5mA.

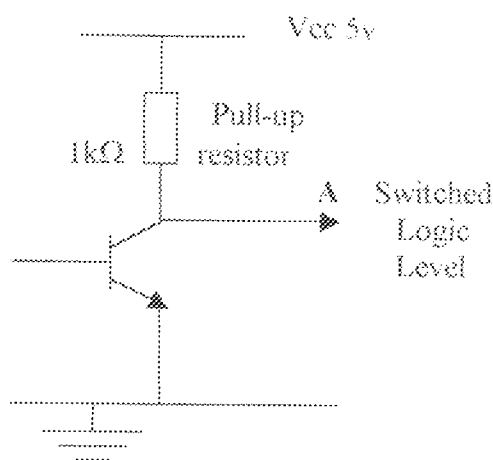


Fig 2.2.1 Arrangement for a switched level input to a computer.

2.3 PARALLEL PORT INTERFACES

Parallel port interfaces use a group of binary digit data lines (commonly eight) for connecting a programmable logic controller (PLC) with an I/O device. The most common standard is the printer interface, the IBM compatible based on the centronics pin assignment (see fig 2.3). This is a one-way, high data rate, and short-distance connection from a PLC (or computer) to a printer, which incorporates handshaking

signals for regulating the flow of data. A strobe pulse from the PLC or computer transfers parallel data into the printer's internal memory buffer. The acknowledge and busy signals are used for handshaking the data.

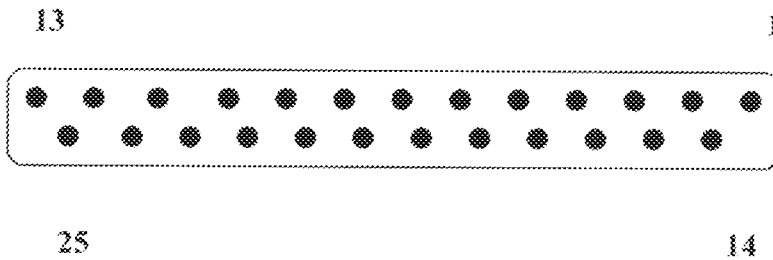


fig 2.3 port view back of a computer

The parallel port on most IBM compatible starts with ID of *0378 HEX*. There are up to three ports on it, the data port, the status port and the control port. The starting address is used for the data port and the next address is used for the status port and then followed by the control port.

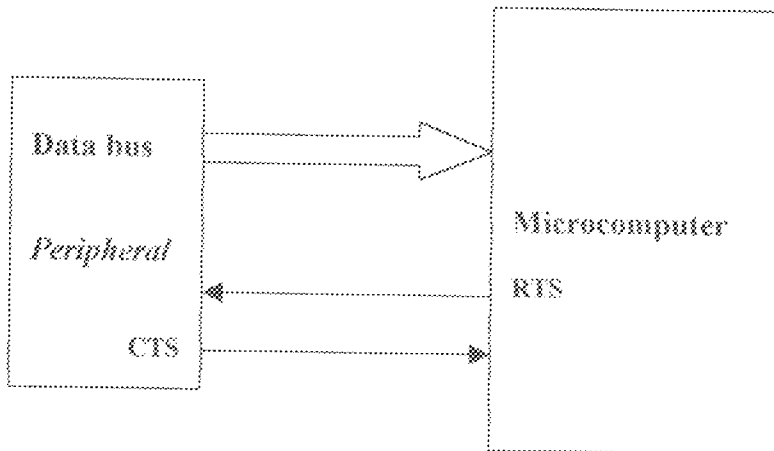
A breakdown of the parallel port pin out is given in the table below.

TABLE 2.0 PINS AND FUNCTIONS OF THE PARALLEL PORT INTERFACE CONNECTOR.

Pin number	Name	Input/output
1	STROBE	Output
2-9	DATA 0 - DATA 7	Output
10	ACKNOWLEDGE	Input
11	BUSY	Input
12	PAPER END	Input
13	SELECT	Input
14	AUTO FEED XT	Output
15	ERROR	Input
16	INIT	Output
17	SELECT IN	Output
18-25	SIGNAL GROUND (0v)	-----

2.3.1 HANDSHAKING

To achieve this, the computer and peripheral are connected together by appropriate control lines, either two or three in number, they are also linked by the data bus, on which the actual transfer of data will be made (see figure 2.3.1).



RTS=Request

CTS=clear send

Fig 2.3.1 computer and peripheral communicating with handshaking lines (RTS and CTS)

The control lines are known as the handshaking lines. Where only two such lines are used, one is employed by the sending end to 'request' the receiving end to respond to this request. In the event that the receiving device is not ready for new data, it is said to 'busy'. The logic level on the handshaking lines just referred to will indicate this state. Handshaking can be used at either input or output of the computer.

2.4 THE MICROCOMPUTER BUS STRUCTURE

A bus can be considered as a set of lines over which digital information (e.g. 16-bit address or 8-bit data) can be transferred in parallel. This method of interconnection has a significant impact on the structure and operation of the system.

Digital devices sharing a bus must be tristate. This means that when the output lines are not in use they are put into a high impedance state so that they will not load the bus. Therefore, an output line can have one of three possible conditions, which are logic 0 (low), logic 1 (high) and output disconnected. Tristate devices incorporate a chip select (also called chip enable) input which is used to isolate its data output lines that are placed into the high impedance or tristate.

In most systems, there are three distinct buses:

- Data bus
- Address bus
- Controls bus

The Data bus: This is a bi-directional path on which data can flow between the microprocessor, memory and I/O. An 8-bit microprocessor has a data bus, which is 8 line wide. A 16-bit microprocessor has a data bus, which is 16 lines wide.

The Address bus: Is a unidirectional set of lines, which carry binary number addresses. Addresses are generated by the *CPU* during the execution of a program to specify the source and destination points of the various data item to be moved along the data bus an address identifies a particular memory location or *I/O* port

The control bus: It consists of a set of signals generated by the CPU to control the devices in the system. An example is the Read/Write control line which selects one of two operations, either a write operation where the CPU is outputting data on the data bus or a read operation where the CPU is inputting data from the data bus.

2.5 THE POWER SUPPLY

The project piece requires a +5V regulated DC power supply, which was generated from the NEPA mains of between 220/240V AC through the following processes.

- (i) Stepping down the 220/240V AC to 9V AC with a transformer.
- (ii) Conversion of the lower AC to DC using full wave rectification.
- (iii) Filtering/smoothing of the DC output ripple.
- (iv) Stabilizing the DC output using 7805 IC regulator.
- (v) Power display LED.

The implemented power supply circuit is shown below.

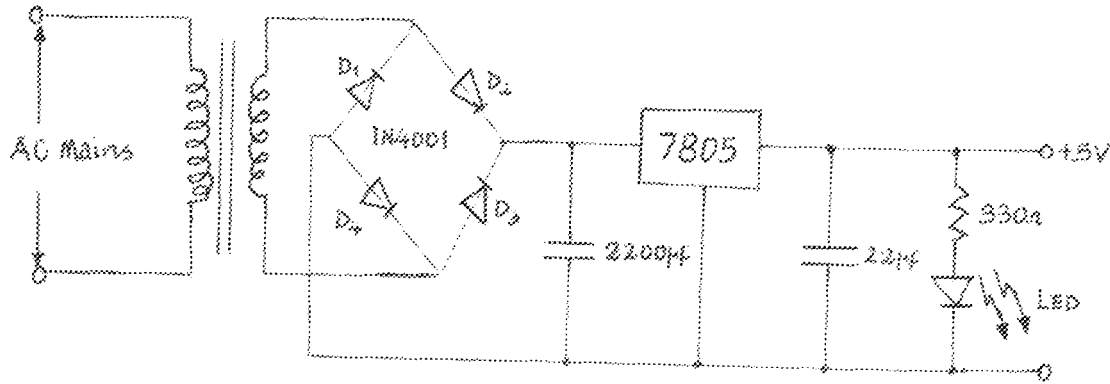


Fig 2.4 POWER SUPPLY UNIT

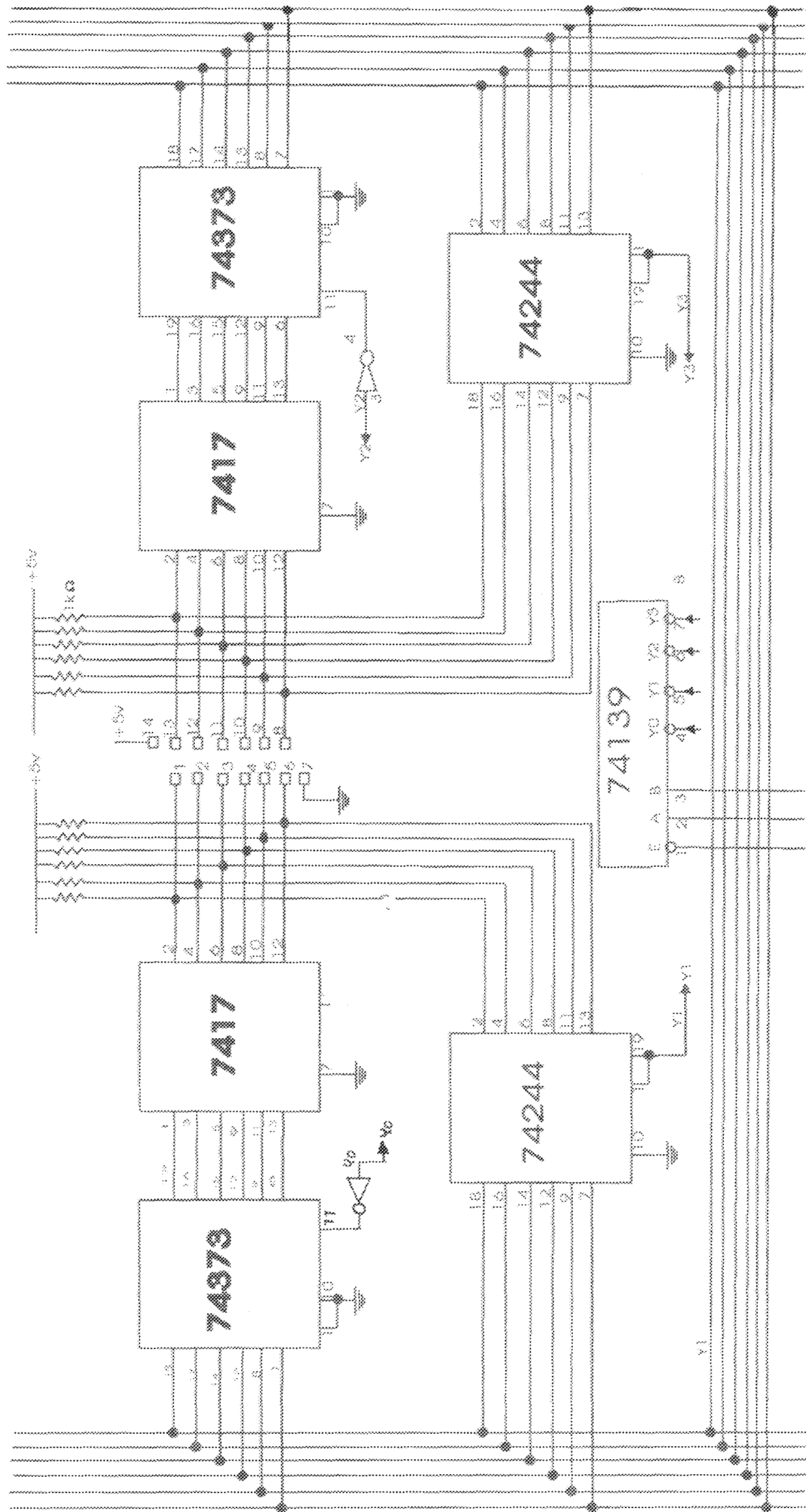


Fig 2.5 Schematic diagram of the microcomputer based 14 pins Digital IC tester

CHAPTER THREE

3.0 CONSTRUCTION PROCEDURE

On completion of the project design, components needed for its construction were bought. The construction started with the implementation of the design on the breadboard.

This was done in modules proceeding from one module to another after due testing. The modules are:

- ◆ The power supply unit (*PSU*)
- ◆ The probe circuit

The construction was then carried out on a veroboard, using the following proceeding

- (i) The veroboard was planned into layout to avoid clumsy arrangement and components short-circuiting.
- (ii) Components such as resistors, capacitor and regulators were soldered directly to the veroboard.
- (iii) All ICs used were placed on IC sockets.
- (iv) A digital multimeter was then used to ensure that proper contact and continuity is achieved between the components used. The software was written, and debugged using *C++ builder*, when both the software and hardware were ready, testing was done separately without interfacing the probe circuit to the PC and finally the piece was interfaced and tested.

3.1 TOOLS USED

The use of instruments such as soldering iron, soldering lead, sucker, breadboard, veroboard, connecting leads, screw driver, long nose plier, cutter and digital multimeter, the pre listed steps mentioned above were taken to construct the *MBDIT-14*.

3.2.1 MODE OF OPERATION

The Digital IC tester consist of two symmetrical blocks (as in figure 2.5) with 6-bit bi-directional data bus and 3-bit unidirectional control bus, the buses so mentioned were gotten from an 8-bit addressable latch, open collector buffer with 1 K Ω pull up resistor and a tri-state buffer wired together.

The mode of operation is one of the two conditions either a *READ* or a *WRITE* effected on to the control bus through timing signal generated by the software into the input lines of the 2-to-4 lines decoder to select the individual control line which in turn enables the appropriate block to *READ* from or *WRITE* on to. The truth table for the MBDIT-14 is shown below.

Control Input			Operation
E	A	B	
1	X	X	Disable all blocks
0	0	0	WRITE A
0	0	1	WRITE B
0	1	0	READ A
0	1	1	READ B

From the table above, if the condition (000) is achieved any data on the computer's data bus is written to the addressable latch (74373) labeled A which is coupled directly to an open collector buffer (7417), thereby writing to the first half of the IC under investigation. The condition (001) also, enables a write to the second half of the IC. Conditions (010) and (011) enables the computer to read from both block A and B of the IC, the generated output of the IC is then taking to a matching unit in the software where it is compared with the manufacturer's function table of the IC and an appropriate message is displayed to tell whether the IC is good or bad.

The step by step connection for the *MBDIT-14* probe is shown below.

Step 1 connect the *MBDIT-14* to the computer via the parallel port

Step 2 connect the *MBDIT-14* to its power source and switch on

Step 3 Load the *MBDIT-14* software on the screen, which will aid the user to test the state of the digital IC.

3.2.2 SOFTWARE

These are the instruction programs that enable the hardware to function, in other words, they are information which the CPU can interpret to generate the impulses that send signals to hardware components to behave in certain manners.

In this project the function of the software is to read or write to either the logic probe unit interface or the system unit, it also check the status of the digital IC voltage and then report the status to an output device i.e. the monitor.

The software is developed using C++ builder is high level language with some low - level capability, it is a structured programming language with object oriented programming features. It was referred to as A then B, then C, then C++.

With recent development it has been upgraded to what we know as C++ builder.

C was first invented in 1980 by Bjarne Straustrop a Bell laboratory in Murray hall, New Jersey; it was changed to C++ in 1983. Since then it has undergone two major revisions one in 1995 and the other in early and late 90's. C++ builder is most commonly used in the development of operating systems and hardware device drivers. It is very good due to its machine language compatibility. The software in this project has been made to be user friendly and mouse enabled.

3.3 TESTING

The testing of the hardware started from the construction stage as each of the modules on completion was tested before moving on to the next one. The circuit was in each case checked for continuity using digital multimeter, also the required characteristics of each component was tested before and after soldering. Finally, after interfacing the resultant testing devices on to the PC and plugging an IC to be tested into the logic probe, the *LOW-HIGH* state of the IC under investigation was tested.

RESULT

The testing part of the program generated the following results.

IC TYPE	STATUS	REPORT
7400	GOOD	ALL I/O PINS OK
7402	BAD	I/O PINS 2&4 BAD
7404	BAD	I/O PINS 1&2 BAD
7408	GOOD	ALL I/O PINS OK
7416	GOOD	ALL I/O PINS OK
7417	GOOD	ALL I/O PINS OK
7432	GOOD	ALL I/O PINS OK
7474	GOOD	ALL I/O PINS OK
7486	GOOD	ALL I/O PINS OK

3.5 DISCUSSION OF RESULT.

The reading of the IC's output voltage after testing were compared with their manufacturers function table and were found to have a very accurate and agreeable values. The truth table of the ten (10) tested ICs are provided below.

7400

Inputs		Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

7402

Inputs		Output
A	B	Y
0	0	1
0	0	0
1	0	0
1	1	0

7404

Input	Output
A	Y
0	1
1	0

7408

Inputs		Output
A	B	Y
0	0	0
0	1	0
1	0	1
1	1	1

7416

Input	Output
A	Y
0	0
1	1

7417

Input	Output
A	Y
0	1
1	0

7432

Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

7486

Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

7474

Inputs				Output	
Preset	Clear	Clock	D	Q ₁	Q ₂
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	1*	1*
1	1	↑	1	1	0
1	1	↑	0	0	1
1	1	0	X	Q ₁	Q ₂

3.5.1 LIMITATION

The microcomputer based 14 pins digital IC tested (*MDDIT-14*) can only be used to test IC's with only 14 pin configuration and is restricted to testing only digital ICs.

3.5.2 PROBLEMS ENCOUNTERED IN THE CONSTRUCTION OF THE PROJECT.

During the process of design and construction of this project, the following problems were encountered

- The project being a relatively new area of intensive study (at project work level), reference materials were not readily available.
- The non-availability of an 8-bits addressable latch with open collector buffer leaves me with no other option than to improvise a separate open collector buffer to an ordinary 8-bit addressable latch.
- Some components were damaged and were replaced several times.
- There was no available device to test my digital ICs.

CHAPTER FOUR

4.0 CONCLUSION AND RECOMMENDATION

4.1 CONCLUSION

This project work has proved to be interesting; it covers the areas of software and hardware implementation. However, its application to test digital IC is a millionth of fraction of what it could be used for.

This project could be put to use in high prospects areas as process control, industrial control, military, scientific analysis, and the list is inexhaustible.

4.2 RECOMMENDATION

Having successfully concluded this project work, within the scope required at this level, and seeing its beautiful prospects, I hereby recommend to the department, school and the relevant university authorities to explore ways of developing projects like this and further sell/expose it to the outside world in order to fulfil engineering dreams for the future.

REFERENCES

1. Ejud.N. G and Tyler.G. H (1984) " A Handbook of essential engineering information and data", Mc Graw-Hill book company.
2. Robert .A. M (1992) " The Encyclopaedia of physical science and technology", second edition, vol. 4, Mc Graw-Hill, Inc.
3. Charles .A.H (1978) "Electronic circuits digital and analogue", John Wiley and Sons, Inc.
4. John .F. Wakerly (1976) "logic design projects using standard integrated circuits", John Wiley and Sons, Inc.
5. Fredrick .J. H and Gerald .R. P (1987) "Digital systems hardware organization and design", Third edition, John Wiley & Sons, Inc
6. Alan J.Crispin (1991) " Programmable Logic Controllers and their engineering applications", Mc Graw-Hill Companies.
7. Kenneth .L.Short (1987) "Microprocessors And Programmed logic", Second Edition, Prentice Hall Incorporation

APPENDIX
SOURCE CODE LISTING

```

#ifndef __TESTDEVICE_H
#define __TESTDEVICE_H

#include <bool.h>

enum    ChipIeff{

        NAME ,
        INMASK ,
        OUTMASK

};

enum    OpCode{

        WRITE ,
        READ ,
        PULSE ,
        MATCH ,
        ON_FAIL ,
        ON_PASS ,
        MSG ,
        STOP

};

struct TestCode{

        unsigned char*    Codes ;
        int                Size ;
        TestCode( int ) ;
        ~TestCode() ;

};

struct ChipInfo{

        char*    Name ;
        unsigned int    InMask ;
        unsigned int    OutMask ;
        TestCode* TestProg ;
        ChipInfo( char* , unsigned int ,
        unsigned int) ;
        ~ChipInfo() ;

};

struct Strings{

        char**    Str ;
        int    Size ;
        Strings( int ) ;
        ~Strings() ;

};

class DeviceDriver{

public :
        //Constructor & destructor
        DeviceDriver() ;
        ~DeviceDriver() ;

private :

```

```

//Variables
ChipInfo*      Info ;
Strings*       Messages ;
unsigned int    ProgIndex ;

/*Test program index*/
                unsigned int
InRegister ;   /*Input register*/
                bool
                GoodMatch;

//Functions : Run
                bool
/*Executes next instruction*/
                Execute() ;

                void
                Skip() ;

//Functions : I/O
                void
/*Writes to IC inputs */
                Write( unsigned int ) ;

                void
/*Reads IC output logic states*/
                Read() ;

//Functions : Miscellaneous
                void
                Match(
unsigned int ) ; /*Matches parameter with InReg*/

                public:
//Variables
                bool
                ShowMsg ;
                int
                Delay ;

//Functions
                void
                SetChipInfo(
ChipInfo* ) ;

                void
                SetMessages(
Strings* ) ;

                bool
                Detect() /*
Detects hardware and runs
                * self-diagnostic tests
                * Remove any chip from probes
                */

                bool
                Run() ;
};

#endif

#ifdef __TESTDEVICE__CPP
#include <string.h>
#include <stdio.h>
#include <dos.h>
#include "Device.h"

#define WRITE_A 0x00
#define WRITE_B 0x01

```

```

#define PEAI_A      0x12
#define READ_E     0x13

#define VALID_BUS  0x04
#define DATAPORT   0x0370
#define CTRLPORT   DATAPORT + 2
#define DATA_MASK 0x3F3F

//
static unsigned int ToWord( unsigned char a, unsigned char b )
{
    unsigned int Result ;
    unsigned char* Ptr((unsigned char*)&Result) ;

    Ptr[0] = a ;
    Ptr[1] = b ;

    return Result ;
}

//Class 'TestCode'
TestCode::TestCode( int S)
{
    this->Codes = new unsigned char[S] ;
    if( this->Codes ) this->Size = S ;
    else this->Size = 0 ;
}

TestCode::~TestCode()
{
    delete[] this->Codes ;
}

//Class 'ChipInfo'
ChipInfo::ChipInfo( char* Str, unsigned int In , unsigned int Out )
:InMask(In),OutMask(Out),TestProg(0)
{
    this->Name = new char[strlen(Str) + 1] ;
    if( this->Name ) strcpy( this->Name, Str ) ;
    InMask &= DATA_MASK ;
    OutMask &= DATA_MASK ;
}

ChipInfo::~ChipInfo()
{
    delete[] this->Name ;
}

//class 'Strings'
Strings::Strings( int S)
{
    int Count ;
    this->Str = new char*[S] ;
    if( this->Str )
    {
        for( Count = 0 ; Count < S ; Count++ ) this->Str[Count] = 0 ;
        this->Size = S ;
    }
    else this->Size = 0 ;
}

```

```

Strings::~Strings()
{
    int Count ;
    if( this->Str )
        for( Count = 0 ; Count < this->Size ; Count++ ) delete[] this->Str[Count] ;

    delete[] this->Str ;
}

//Class 'DeviceDriver'
DeviceDriver::DeviceDriver()
:Info(0),Messages(0),ProgIndex(0),InRegister(0)
,GoodMatch(false),ShowMsg(true),Delay(1)
{}

DeviceDriver::~DeviceDriver()
{}

bool DeviceDriver::Execute()
{
    unsigned char* Mem(this->Info->TestProg->Codes) ;
    int a , b ;
    unsigned int PulseMask ,c,d;
    bool Result(true) ;

    switch( Mem[this->ProgIndex] )
    {
        case WRITE:
            this->ProgIndex++ ;
            a = this->ProgIndex ;
            this->ProgIndex++ ;
            b = this->ProgIndex ;
            this->Write( ToWord(Mem[a],Mem[b]) ) ;
            break ;

        case READ:
            this->Read() ;
            break ;

        case PULSE:
            this->ProgIndex++ ;
            a = this->ProgIndex ;
            this->ProgIndex++ ;
            b = this->ProgIndex ;

            PulseMask = ToWord(Mem[a],Mem[b]) ;
            this->Read() ;
            c = (~this->InRegister) & PulseMask ;
            d = this->InRegister & (~PulseMask) ;

            this->Write( c | d ) ;
            this->Write( this->InRegister ) ;

            break ;

        case MATCH:
            this->ProgIndex++ ;
            a = this->ProgIndex ;
            this->ProgIndex++ ;
            b = this->ProgIndex ;

```

```

        this->Match( ToWord(Mem[a],Mem[b]) );
break ;

case    ON_FAIL:
    this->ProgIndex++ ;
    if( !this->GoodMatch )return this->Execute() ;
    else this->Skip() ;
break ;

case    ON_PASS:
    this->ProgIndex++ ;
    if( this->GoodMatch )    return this->Execute() ;
    else this->Skip() ;
break ;

case    MSG:
    this->ProgIndex++ ;
    if( this->Messages && this->ShowMsg )
    {
        a = Mem[this->ProgIndex] ;
        if( a < Messages->Size )
            puts( this->Messages->Str[a] ) ;
    }
break ;

case    STOP:
default:
    Result = false ;
break ;
}

this->ProgIndex++ ;
return Result ;
}

void    DeviceDriver::Skip()
{
switch( this->Info->TestProg->Codes[this->ProgIndex] )
{
    case    WRITE:
        this->ProgIndex += 2 ;
        break ;

    case    READ:
        break ;

    case    FULSE:
        this->ProgIndex += 2 ;

        break ;

    case    MATCH:
        this->ProgIndex += 2 ;
        break ;

    case    MSG:
        this->ProgIndex++ ;
        break ;
}

this->ProgIndex++ ;
}

```

```

void      DeviceDriver::SetChipInfo( ChipInfo* I)
{
    if( !I || !I->TestProg ) this->Info = 0 ;
    else this->Info = I ;
}

void      DeviceDriver::SetMessages( Strings* M)
{
    this->Messages = M ;
}

void      DeviceDriver::Match( unsigned int D)
{
    if( this->InRegister == D ) this->GoodMatch = true ;
    else this->GoodMatch = false ;
}

void      DeviceDriver::Write( unsigned int D )
{
    unsigned char* Ptr((unsigned char*)&D ) ;

    D ^= this->Info->InMask ;

    //write to side A
    outporth(CTRLPORT,WRITE_A | VALID_BUS) ;
    outporth(DATAPORT,Ptr[0]) ;

    //write to side B
    outporth(CTRLPORT,WRITE_B | VALID_BUS) ;
    outporth(DATAPORT,Ptr[1]) ;

    //Disable bus
    outport(CTRLPORT,0) ;
}

void      DeviceDriver::Read()
{
    unsigned char a,b ;

    //read side A
    outporth(CTRLPORT,READ_A | VALID_BUS) ;
    a = inporth(DATAPORT) ;

    //read side B
    outporth(CTRLPORT,READ_B | VALID_BUS) ;
    b = inporth(DATAPORT) ;

    //Disable bus
    outport(CTRLPORT,0) ;

    this->InRegister = ToWord(a,b) ;

    this->InRegister ^= this->Info->OutMask ;
}

bool      DeviceDriver::Detect()
{
    int Data(1) ;
    int Count ;
}

```



```

for( Count = 0 ; Count < 16 ; Count++ )
{
    this->Write(Data & DATA_MASK) ;
    delay(1000) ;
    this->Read() ;
    if( this->InRegister != (Data & DATA_MASK) ) return false ;
    Data = Data << 1 ;
}

return true ;
}

bool    DeviceDriver::Run()
{
    if( !this->Info || !this->Messages ) return false ;

    this->GoodMatch = false ;
    this->ProgIndex = 0 ;

    while( this->Execute() ) delay(this->Delay);

    return true ;
}

#endif

```