

A PROJECT REPORT ON
**TROUBLESHOOTING, MAINTENANCE AND REPAIR
OF MICROCOMPUTER SYSTEMS AND
PERIPHERALS**

BY

OLADUNGBEHIN O. SHADRACH
(93/3650)

SUBMITTED TO:

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

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE
AWARD OF BACHELOR DEGREE (B.ENG.) IN
ELECTRICAL/COMPUTER ENGINEERING.

MARCH 2000

DECLARATION


I hereby declare that this work "TROUBLESHOOTING, MAINTENANCE AND REPAIR OF MICROCOMPUTER SYSTEMS AND PERIPHERALS" is an original conducted by me under the supervision and guidance of Mr. Danjuma of the Department of Electrical / Computer Engineering, Federal University of Technology, Minna during the 1998/1999 academic session

OLADUNGBEHIN O. SHADRACH

NAME

93/3650

REG. NUMBER


8.17/02/2000

SIGN/DATE:

DEDICATION

This project write-up is dedicated to my Lord Jesus Christ and to all “they that love and follow him”.

I will also like to dedicate it to my dear brother Frank Akinlani of blessed memory.

CERTIFICATION

This is to certify that this project work "Troubleshooting, maintenance and repair of microcomputers systems and peripherals" was conducted and presented by OLADUNBEHIN O. SHADRACH of Electrical / Computer Engineering Department, Federal University Of Technology, Minna, in fulfilment of the requirements for the award of Bachelor degree in Electrical / Computer Engineering.

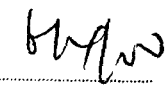
MR DANJUMA I. A.
SUPERVISOR


SIGNATURE

04/04/2000
DATE

DR. ADEDIRAN Y. A.
HEAD OF DEPARTMENT


SIGNATURE


DATE

EXTERNAL EXAMINER


SIGNATURE

6/4/2000
DATE

2.2.3	GENERAL METHODS OF TROUBLESHOOTING A PC PROBLEM	11
2.3	PROBLEM DEFINITION	13
2.3.1	POWER – UP (POWER UNIT).....	13
2.3.2	SYSTEM UNIT	14
2.3.3	INPUT DEVICES	17
2.3.4	DISKS AND DISK DRIVES	18
2.3.5	MONITORS – (DISPLAY UNIT)	21
2.3.6	PRINTERS	24

CHAPTER THREE

COMPUTER PROGRAMMING

3.1	PROGRAMMING AND PROGRAMMING LANGUAGES	26
3.2	AN OVERVIEW OF PROGRAMMING LANGUAGES.....	27
3.3	PROGRAMMING IN C OR C++	28
3.4	PROGRAM DESIGN	29

CHAPTER FOUR

CONCLUSION

4.1	CONCLUSION.....	32
4.2	RECOMMENDATIONS	32
4.3	REFERENCES	33

APPENDIX

LIST OF FIGURES

Fig. 1.1	Flowchart for general steps in problem solving	10
Fig. 2.1	Flowchart for troubleshooting power problems	14
Fig. 2.2	Flowchart for troubleshooting system unit problems	16
Fig. 2.3	Flowchart for troubleshooting disks and disk drive problems	20
Fig. 2.4(a)	Rectifier diodes.....	22
Fig. 2.4(b)	Full-wave diode bridge rectifier	22
Fig. 2.5	Flowchart for troubleshooting monitor problems.....	23
Fig. 2.6	Flowchart for troubleshooting printer problems	25
Fig. 3.1	Flowchart for the main menu program module	30
Fig. 3.2	Flowchart for a device troubleshooting program module	31

1.1

AIMS AND OBJECTIVES

This project is aimed at providing the users with the basic knowledge and techniques of troubleshooting and maintaining their microcomputer systems and common Peripheral devices. Users will also be acquainted with some of the terms associated with computer troubleshooting, maintenance and repairs. It should be noted that this project is not aimed at making its target beneficiaries professional computer engineers. Rather it will provide them with the basics to enable them maintain their system effectively and efficiently. Consequently, it will help the users to save cost of inviting an expert in most cases.

With this project, the user will have the following benefits.

- * Get familiar with the component parts of the computer system unit.
- * Be able to connect and / or install peripheral devices such as the printers.
- * Know what the computer is made up of.
- * Users should be able to handle test equipment without fear or favour.
- * They should be able to perform routine maintenance of the entire system.
- * Be able to distinguish between hardware and software.
- * Diagnose or troubleshoot most computer problems and fix them.

1.2

TOOLS AND EQUIPMENT

Below is a list of common tools and equipment needed for effective troubleshooting and repairs of computers, accessories and peripherals.

- 1). A biro and a jotter (note pad)
- 2). Anti-static strip or anti-static wrist band (optional)
- 3). Digital multimeter
- 4). Soldering iron
- 5). Solder sucker
- 6). Parts tube
- 7). Screw drivers set
- 8). Hex and socket drivers set
- 9). Chip puller
- 10). Blower
- 11). Drive head cleaners (for floppy and CD-ROM drives)
- 12). Diagnostic software (e.g. Anti-virus start-up disks)
- 13). Pliers.

1.3

SCOPE OF THIS PROJECT

This project is designed to meet the immediate needs of its target users in areas of troubleshooting, maintenance and fixing of most computer problems. So much resources (money and time) are usually involved when a computer system is taken to professionals for repairs. In most cases, the problem could be a simple one that an experienced user can fix himself. You don't necessarily need to be a computer "Whiz" or "Guru" to fix majority of PC problems. All you need is just a knowledge of a few principles and methods of troubleshooting.

The primary aim of this project is to identify all common problems of microcomputers irrespective of their make or model and acquaint the user with step by step or an easy solution to them. It is not limited to the computer system alone. This project also involves troubleshooting and fixing of problems of computer peripheral devices such as the printers. Proper application of this project will thus make the use of personal microcomputers interesting and cost effective.

This project is limited to the problems of stand-alone microcomputer systems. It is of very little or no importance in a computer network problem. Steps of problem-solving techniques contained in this project may not be applicable to other types of computers e.g. Super computers, mainframes etc. This project does not intend to equip the user with such knowledge as to tear apart and fix the circuits of a microprocessor. Neither does it intend to make the user more qualified than or equivalent to the professional computer engineers. In a situation where any problem persists after the user have applied the steps for solving it as specified in this project, it is recommended that such a system be taken to professionals for repairs.

1.4

LITERATURE REVIEW

Over the years, many people and corporate bodies have carried out much work on similar or related topics. Some of these are as reviewed below.

Que Corporation has published several editions of the books "Upgrading and Repairing PCs" and "Upgrading and Maintaining PCs". written by Scott Mueller. For the purpose of this project work, the sixth edition of the first - Upgrading and Repairing PCs published in 1996 by Que Corporation was consulted. The book provides the user with necessary precautions and steps to be taken in troubleshooting and fixing PC problems. It also provides an enumerated list of the tools and equipment necessary for the job and their proper usage (1, Pgs 25 - 50).

In 1998, Sanhills Publishing Co. published the book "Troubleshooting - the book of simple solutions for most computer problems - from PCNovice and Smart computing. The book highlight most problems of personal computers, which are user fixable and steps for solving them. (2, Pgs 4-9; 112 - 139; 149 - 151; 155 - 170).

As his final year project, Ekuniyi Olumuyiwa of the department of Electrical and Computer Engineering, Federal University of Technology, Minna during the 1996/97 academic session, carried out computer-assisted analysis and repair of i80386 based microcomputers.

IBM Corporations among other establishments has released a number of books on maintenance and repair of microcomputers. One of such is the IBM hardware handbook. The book highlights good maintenance tips and steps to fix common problems microcomputers. (7, Pgs 35 - 58). It also deals with the installation/assembly and configuration of microcomputers taking the IBM Aptiva computer as a case study.

In 1997, Gradient Software released interactive video tutorial, utilities and diagnostic software titled - Build a Pentium computer (Master Tech '97). It is an audio-visual tutorial, which teaches the user practically, how to assemble a Pentium computer, troubleshoot and fix common Pentium computer problems. It also provides a wide variety of computer terminologies.

1.5

PROJECT OUTLINE

This project consists of a title page which states clearly the title of the project - "Troubleshooting, maintenance and repair of microcomputers and peripherals ". The first few pages contain the declaration, certification, dedication, acknowledgement and an abstract of the project.

Chapter one, which is basically an introductory chapter, is dedicated to introducing every aspect of the project topic. The aims and objectives of this project and its scope are clearly enumerated. The tools necessary to achieve the aims of this report write-up and a review of background literature in respect to the project topic are also contained in this chapter.

Chapter two deals explicitly with the project topic. It enumerates the different components that make up the term "hardware" as applied to microcomputers, methods of troubleshooting and fixing general and specific PC problems. Flow charts are also included to illustrate step-wise solution to problems.

Chapter three dwells on the programming aspect of this project. It makes an overview of programming and programming languages and a brief introduction of C - Language, the language used for the tutorial program designed for this project. As part of this chapter, a tutorial program is designed using C++ programming language for screen learning of the troubleshooting and repair steps enumerated in preceding chapters.

Chapter four concludes this work with some recommendations and a list of references to acknowledge sources of information and material for this project. An appendix of computer terms and their meaning is also included.

CHAPTER TWO

SOLVING HARDWARE PROBLEMS

2.1 INTRODUCTION TO COMPUTER HARDWARE MAINTENANCE AND REPAIRS

The term computer hardware refers to the physical components that make up the computer system. These machine parts are subject to ageing effect. Therefore, a good maintenance / repair culture is inevitable if the system is to function as expected all its life span.

Basically, the microcomputer system consists of input device(s), system unit and an output device(s). The primary input device is the keyboard. It is used to input text and characters including control characters into the system unit. Other input devices in common use includes the mouse, joystick, scanner (used to input graphics and 'printed text' into the computer.

The monitor is the primary output device of the microcomputer. Majority of the monitors available for desktop PCs uses a CRT (Cathode Ray Tube) for display. The user can input (from the keyboard) a displayable character (e.g. alphanumeric and special characters) or a control character (e.g. Return key or Function keys). A key pressed at the keyboard is processed by the input interface adapter and sent to a transducer as a code. The transducer interprets this code and sends the required signal to the display unit to display the indicated character or perform the requested control action. Other output devices are printers, modems etc.

The system unit consists of the microprocessor (commonly called the Central Processing Unit – CPU). A system based on a single microprocessor chip is termed a microcomputer. It also contains the memories (i.e. Memory units). These are either Random Access Memory (RAM) or Read Only Memory (ROM). Only the system main memory, which is a RAM, is of direct importance to this work. The memories and CPU and any interface card are all mounted on the motherboard. The system unit also includes the disk drives.

To maintain these hardware components, make sure the entire computer or peripheral device is regularly cleaned with an appropriate solution such as the contact "cleaning solution" and always keep

them covered when not in use. Presence of dust or other particles can easily lead to fatal problems. Be careful that no liquid substance gets into or on any of these devices.

Computer systems should be used where there is adequate ventilation or on the better side, in an air-conditioned room. Excessive heat can readily damage the microprocessor or memory chips. The computer system should be powered through a dependable power regulatory device such as the UPS (Un-interruptible Power Supply). This will ensure maximum protection of RAM chips, hard disk, power unit and the entire system against the effects of power fluctuations.

2.2 FINDING AND FIXING HARDWARE PROBLEMS

2.2.1 PRECAUTIONS

In carrying out any troubleshooting or assembly of a system, the following precautions must always be observed.

- 1) Turn off the computer (i.e. power down the system using the standard operating system's procedures). Remove the power cables. Don't forget to turn off and unplug any peripheral device(s) attached to the system e.g. printers
- 2) Never move the computer while it is turned ON. This may damage the hard disk while it is working.
- 3) Avoid plugging or unplugging the mouse or keyboard (especially the PS/2 types) while the computer is ON. This can easily damage the computer's mainboard.
- 4) Once you open up a system (or unit) case, touch the unpolished case parts to discharge any electrostatic build up on your body. This is very efficient where an anti-static strip or wristband is not used. Even a tiny static charge can damage the microprocessor, memory chips etc. Repeat this action frequently when installing hardware like adapter cards, hard disk, disk drives, memory modules etc.

- 5) Before connecting power to mainboard and other components of the system unit, it is advisable to test the output of the power unit to ensure desired output supply. Abnormal or excessive supply can easily damage the components.
- 6) Do not power the system until you have ascertained that all components and cables are rightly and firmly connected.

2.2.2 GENERAL STEPS IN HARDWARE PROBLEM SOLUTION

The first and primary solution to any problem is identifying the problem itself. Computer system problem could be very frustrating but simple and logical steps could fix a good number of them.

Although problems of computers (hardware or software) can be numerous, common stepwise actions are usually involved in troubleshooting and fixing vital mishaps in computer systems.

Generally the process of troubleshooting takes the following steps.

- (i) Identify the problem, i.e. problem definition. This is an easy decision to make. For example, system cannot read from a floppy disk, no sound output could be heard from multimedia speakers when playing an audio or video CD.
- (ii) Finding what component part is causing the problem. That is the component that is not functioning properly (hardware or software). A system not being able to read from a floppy disk could be due to faulty disk drive or bad floppy disk etc.
- (iii) Determining why the component isn't functioning properly. A system not being able to read a floppy disk even though the floppy drive is okay could be due to lost disk format, files in bad clusters, incorrect disk type/capacity i.e. drive-disk capacity incompatibility etc.
- (iv) Making simple changes to get the component to work properly. A drive that can no longer read a disk in it even though the disk is very good could work after cleaning it using a drive head cleaner.
- (v) Finally, if a component error cannot be fixed, it may need to be changed or replaced where necessary. For example, a damaged memory module will need to be replaced for the computer to work.

Below is a flowchart representing these steps.

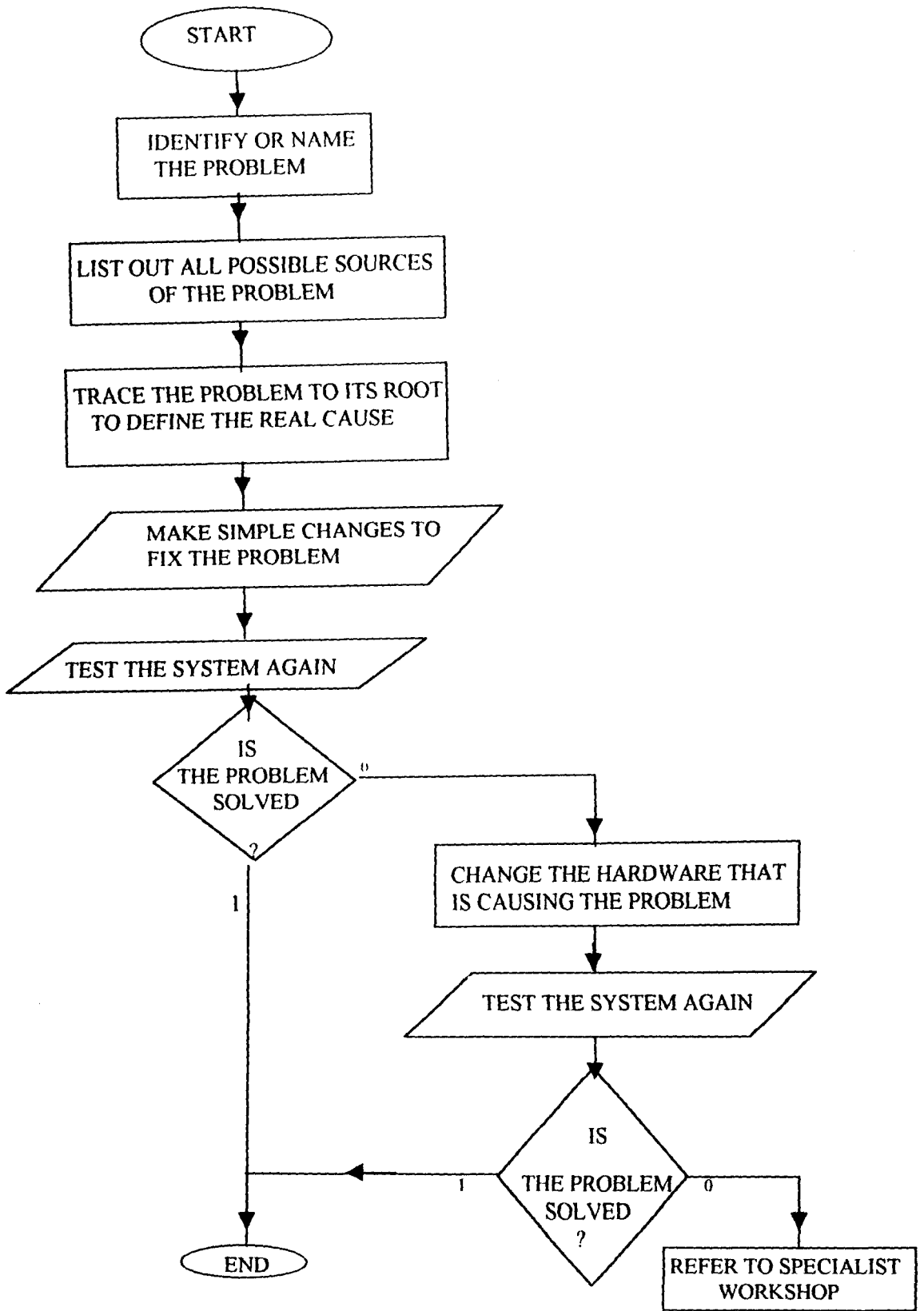


FIG 1.1 FLOWCHART FOR GENERAL STEPS IN PROBLEM SOLVING

2.2.3 GENERAL METHODS OF TROUBLESHOOTING A PC PROBLEM

Enumerated below are the methods involved in troubleshooting a computer problem.

i). A PROCESS OF ELIMINATION

The troubleshooting steps listed earlier (section 2.2.2) are most often performed through a process of elimination. This process helps you remove every element of doubt concerning the origin of the problem. This makes the process of elimination a highly cost effective method of troubleshooting PC problems.

As an example, consider a problem with a diskette drive. Start with the obvious question – Is the drive bad? Or is there something wrong with the diskette? Check the diskette first by trying to use it in another PC. If the diskette is alright, there may be loose cables inside. Open the case and make sure that the power and signal cables to the drive are firmly connected. If this did not fix the problem, try using the signal cable in another system. It may be bad. If the cable is okay then the drive may be bad, try using it in another system too. If the drive is working well, then the drive controller must be the problem. If otherwise, the problem has then be established as faulty drive.

ii). CHANGES

Another method of troubleshooting a PC problem is to determine what is changed or added (hardware or software) since the last time the computer worked properly.

Obviously, adding a new hardware may affect the PC's performance. For example, a modem newly installed may be using the same interrupt request (IRQ) as another component. This will cause at least the modem not to work. Simply change the IRQ of such new hardware as may be provided in its "installation manual".

A newly installed software may change the settings your PC uses to locate various components and peripherals. As an example, the default printer port may be changed from LPT1 to LPT2. If this happen,

the PC may complain of not finding or recognising the printer which you have been using on your system prior to this time. In such a case, locate and correct the settings problem.

iii). SELF TEST

The computer can report the presence of some problems. The Power On Self Test (POST) runs each time you turn ON or restart the computer. POST takes an inventory of the computer's components and determines if any isn't working well. It displays a problem using an error code and a written message. For instance, if the keyboard is not connected, POST display

"105 – 8042 Command Not Accepted
Keyboard Communication Failure"

Where an error message cannot be displayed, POST uses beeps to announce the presence of a problem. Different problems are indicated by different number of beeps. For example, five (5) beeps indicates a problem with the microprocessor. A single short beep means the computer has passed POST while a short beep followed by a continuous one indicates absence or unrecognised memory (RAM) module.

iv). HIDDEN PROBLEM

A problem in one component may manifest itself with symptoms in some other components. For example small fluctuations in the energy supplied from the power supply unit may damage the SIMMs. If such SIMMs modules are simply changed, the problem will continue to occur until the power supply problem is fixed. If a problem re-occur, it is necessary to trace the root of such a problem from the components supplying power and data to the malfunctioning component.

2.3

PROBLEM DEFINITION

Under this section, I shall highlight most of the common problems of the microcomputer by component hardware / unit and provide methods of troubleshooting and fixing them. It should be noted that only problems associated with hardware are discussed in this project. Many problems of the computer could be due to software problems and are outside the scope of this project. Consequently, they are not discussed.

2.3.1 POWER – UP (POWER UNIT)

When the screen is blank and nothing seems to work including the system's power LED, make sure power cables are firmly connected to power source (mains) and the back of the computer including the monitor. Ensure that the mains socket is switched 'ON'. If the system is powered from a power strip, surge suppresser or an UPS, make sure it is switched 'ON' and where supply indicators are present on such devices, check to see that they are ON. If the problem persist, check the power switches on your set to make sure they are in the ON position. If nothing still works, it may be a problem with the power outlet (blackout or brownout). Check supply voltage with the multimeter. If the problem still persists, it may be a blown fuse or MOV in the power module. A damaged rectifier component in a monitor's rectifier module could cause blank-out of only the display. If this happen, refer to the monitor troubleshooting section for solution. Below is a flowchart summarising the above procedures for troubleshooting the power supply unit.

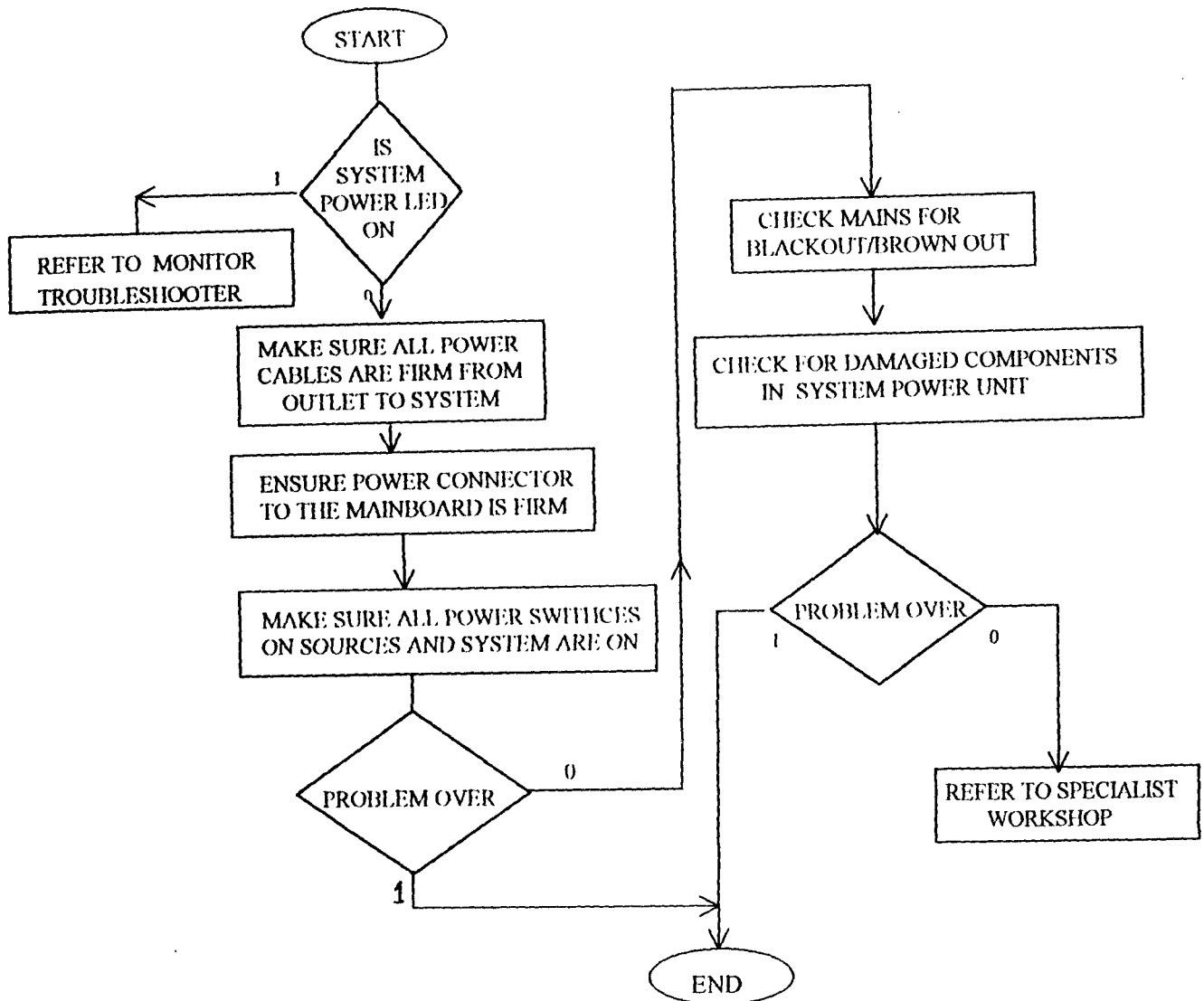


Fig. 2.1 **FLOWCHART FOR TROUBLESHOOTING POWER PROBLEMS**

2.3.2 SYSTEM UNIT

The most common problem of the system unit fixable by the user is failure to initialize or start-up. That failure to pass POST. This problem could be identified as the system makes a grinding noise and stops without displaying any information or a single beep shortly after switch-on is followed by a continuous one. On the other hand, it may display the BIOS “Welcome” screen but stops the initialization process at the point of ‘counting’ the RAM size. These symptoms all indicate a faulty memory (RAM) chip. With the appropriate screw driver, open the system unit’s case (observe all necessary precautions), test the supply of the power unit for accuracy since an abnormal power supply can easily damage the memory chip. Remove the RAM

chipboard and replace with the same type (SIMMs or DIMMs). If power supply is okay, power up and try again. If otherwise, fix power problem before powering up to avoid damage to the new RAM chip.

Another start-up problem common with microcomputer is the influence of computer viruses, modern RAM BIOS versions has a "virus guard software" incorporated. At start-up, this scans the memory for viruses. If a virus is detected, it will report and start-up process halt to avoid spread. To fix the problem, start-up from an anti-virus start-up diskette (e.g. Norton anti-virus start-up disks). This will scan the memory boot records and program files of the hard disk and remove any virus detected.

Also, the absence of primary input device (the keyboard) will cause initialization failure, POST should display an error code/message if this happen. A similar action will occur if a key is held locked down during the start-up. For example, if the keyboard is not connected, POST will display the message

"105 – 8042 Command Not Accepted
Keyboard Communication Failure"

To fix, simply plug in the keyboard connector firmly into its port and restart the system.

Some start-up errors may not be displayable (e.g. microprocessor error), POST uses beep code to signal such errors. For example:

- 1-short beep means DRAM refresh failure
- 2-short beeps means parity circuit failure
- 4-short beeps means system timer failure
- 5-short beeps means processor failure

NOTE:

The beep codes may vary depending on the ROM BIOS type and version of a particular microcomputer system.

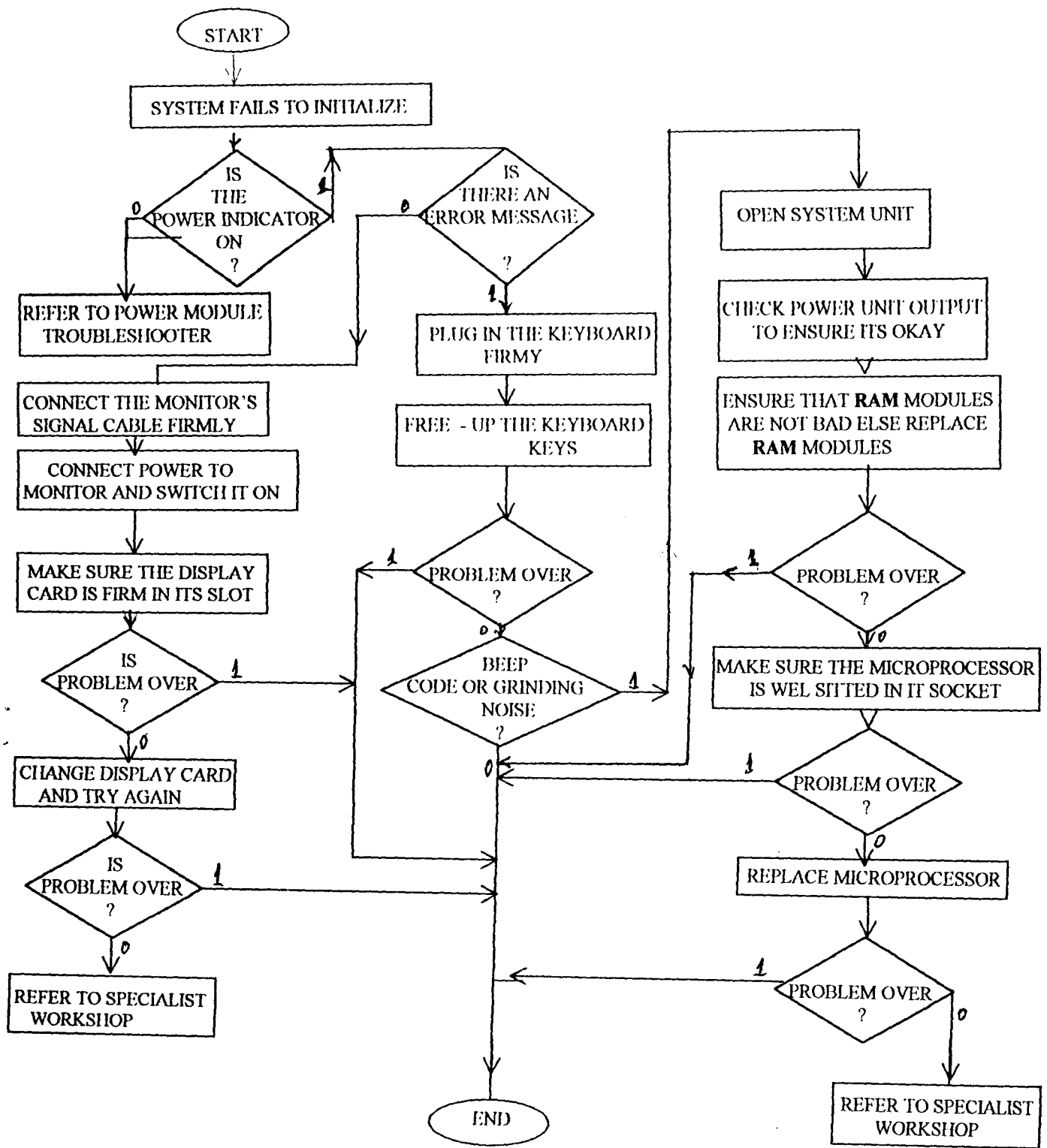


FIG 2.2 **FLOWCHART FOR TROUBLESHOOTING SYSTEM UNIT**

2.3.4 DISKS AND DISK DRIVES

Each time you turn on or restart the computer, POST check for installed hardware (including disk drives). The LED indicator on the floppy disk drive should come on for a short time, otherwise, power or signal cable is/are not connected firmly to it or the drive may have gone bad. Next, the system looks for boot-up command. As a default, the system is usually configured to look for these commands first from a floppy disk so that if no floppy disk is available, it switches to boot from the hard disk. If a bootable disk (e.g. DOS) is left in the floppy drive, when the system is turned ON or restarted, it will boot into DOS (Disk Operating System) rather than the familiar windows. However, if the disk contains no boot-up command, the system report an error message

“None System Disk or Disk Error
Replace and Strike ANY Key”

Simply remove the diskette and press any key to boot from the hard disk or replace it with a bootable diskette and press a key to boot from floppy disk (to DOS prompt).

Occasionally, at start-up, POST may report the error message

“Disk Drive Error” or “Incorrect Drive A Type”.

These messages indicate loose connecting cable to floppy drive A. open the system and ensure cables are plugged in snugly. Reboot. If problem persists, enter system setup and make sure the correct drive type and capacity are specified.

Other error messages associated with disks and disk drives are

General Failure Reading Drive A: the diskette in drive A: is bad

Not ready Reading Drive A: There is no diskette in drive or the drive head can no longer read a disk

Missing Operating System: The operating system's system files are missing or corrupted or (for a hard disk) the disk parameters are not correct. Re-install the operating system. If problem persists, correct drive parameters as provided on drive label else refer to specialist.

Disk I/O Error: The BIOS ROM is not compatible with the drive parameters
or there is no active partition in a multi-partitioned hard disk.

When you issue a file access command unto an unformatted hard disk partition, POST will report the following message

“Invalid Media Type”

If the command is on an unformatted floppy disk, POST report the same error with the message

**“Diskette in Drive A: is Not Formatted
Format Now ? (Y / N)”**

If you are sure that the diskette in drive A: is unformatted, press key ‘Y’ to format it or format the hard disk partition. Else press ‘N’ to quit and try another diskette. If error persists, clean the drive with the drive head cleaner and try again. If same error still repeats, then the drive must be bad. Change it.

Another disk drive problem is a situation where access to the drive is denied with POST displaying the message “Invalid Drive Specification”. If the drive you are trying to access is a CD-ROM drive, install its driver and restart the computer from the hard disk (especially where the operating system is DOS).

If on the other hand, it is a hard disk drive, with the start-up disk in drive A:, Enter the command “FDISK” and follow the screen instructions to create disk partition(s) in the drive, format it and install the operating system and other relevant software. Note that these steps should ONLY be taken when the hard disk is a new one. If the error occur on a formally working hard disk, consult a specialist.

The flowchart below shows a quick reference to the disks and disk drives troubleshooting steps.

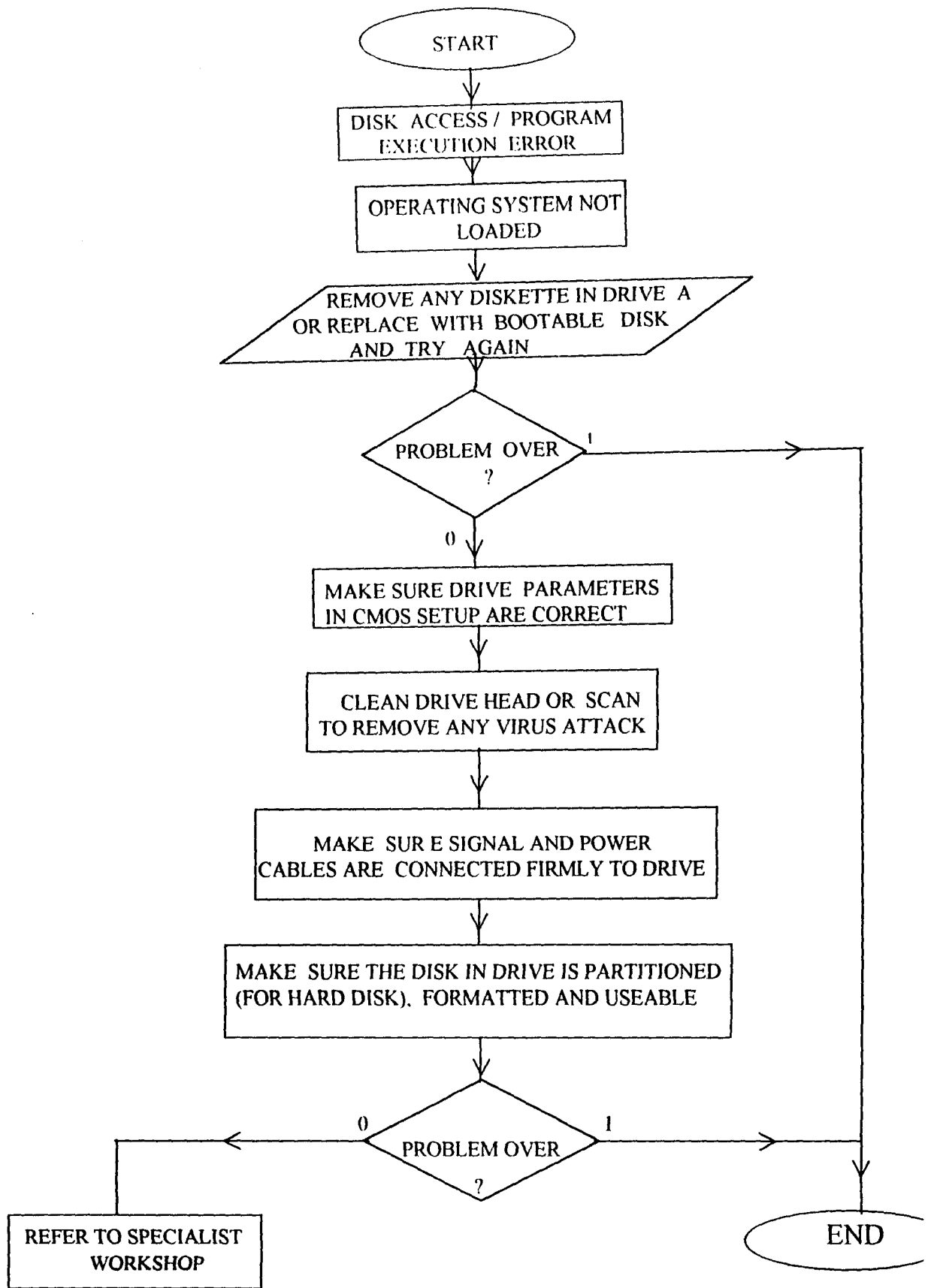


Fig.2.3 FLOWCHART FOR TROUBLESHOOTING DISK AND DISK DRIVE PROBLEMS

2.3.5 MONITORS – (DISPLAY UNIT)

If the monitor is blank and you are certain that the power and signal cables are firmly connected and the power LED indicator confirms good power supply, try adjusting the monitor's control knobs (brightness, contrast etc.). If nothing happens, restart the system and adjust or disable monitor's Advance Power Management (APM) settings or screen saver time-outs. If this could not solve the problem, shut down and try again using a known good working monitor. If the problem persists, the display card is most likely the problem. Shut down, open the system unit case, ensure that the display adapter card is firmly connected by removing it and fixing it back into its slot/connector (or another slot where applicable). If this fix the problem then it's a simple one. If not, change the display card and try again.

If the display card is not the problem then open the monitor and test (using the digital multimeter) for continuity of all the cables in the monitor's signal cable. Extreme care must be taken, as dangerously high voltage could remain stored up in the monitor for a length of time. Check for broken or displaced pin in the 15-pin DB connector at the end of the signal cord. If the problem persists, refer to a specialist workshop.

Another common problem of the monitor is a situation where images on the screen appear distorted with a wavering, scrambled display and moving dots. Here, the images appear as moving lines and text become unreadable. To fix, adjust the monitor window controls. Speakers that are not magnetically shielded, fans, motors etc. near the device could cause magnetic interference which consequently resulted in this problem. Make sure you remove them. A display card not seated firmly or held down too tight in its slot could cause wavering display. Follow the simple step of fixing display problem given above.

If the monitor doesn't come ON whereas power supply mains and cords are ascertained alright, it may be a problem in the power stabilization circuit of the monitor. Open the monitor (observe all safety/voltage precautions). Test the fuse for continuity and the rectifier diodes. Each of the diodes should show a low resistance in one direction and a very high (or infinite) resistance in the reverse. The

diagram below shows the power diode symbolic and practical appearance (Fig. 2.4 (a)) and the full bridge rectifier

(Fig. 2.4 (b)). Replace and damaged fuse or diode and try again. If problem continue contact a specialist.



Fig 2.4 (a) RECTIFIER DIODES

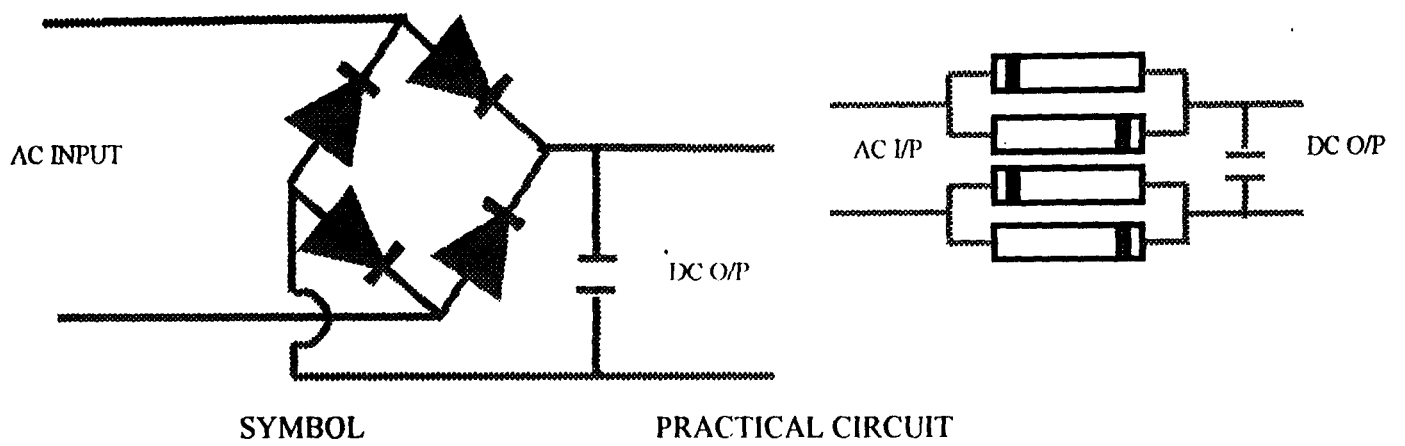


Fig 2.4 (b) FULL-WAVE BRIDGE DIODE RECTIFIER

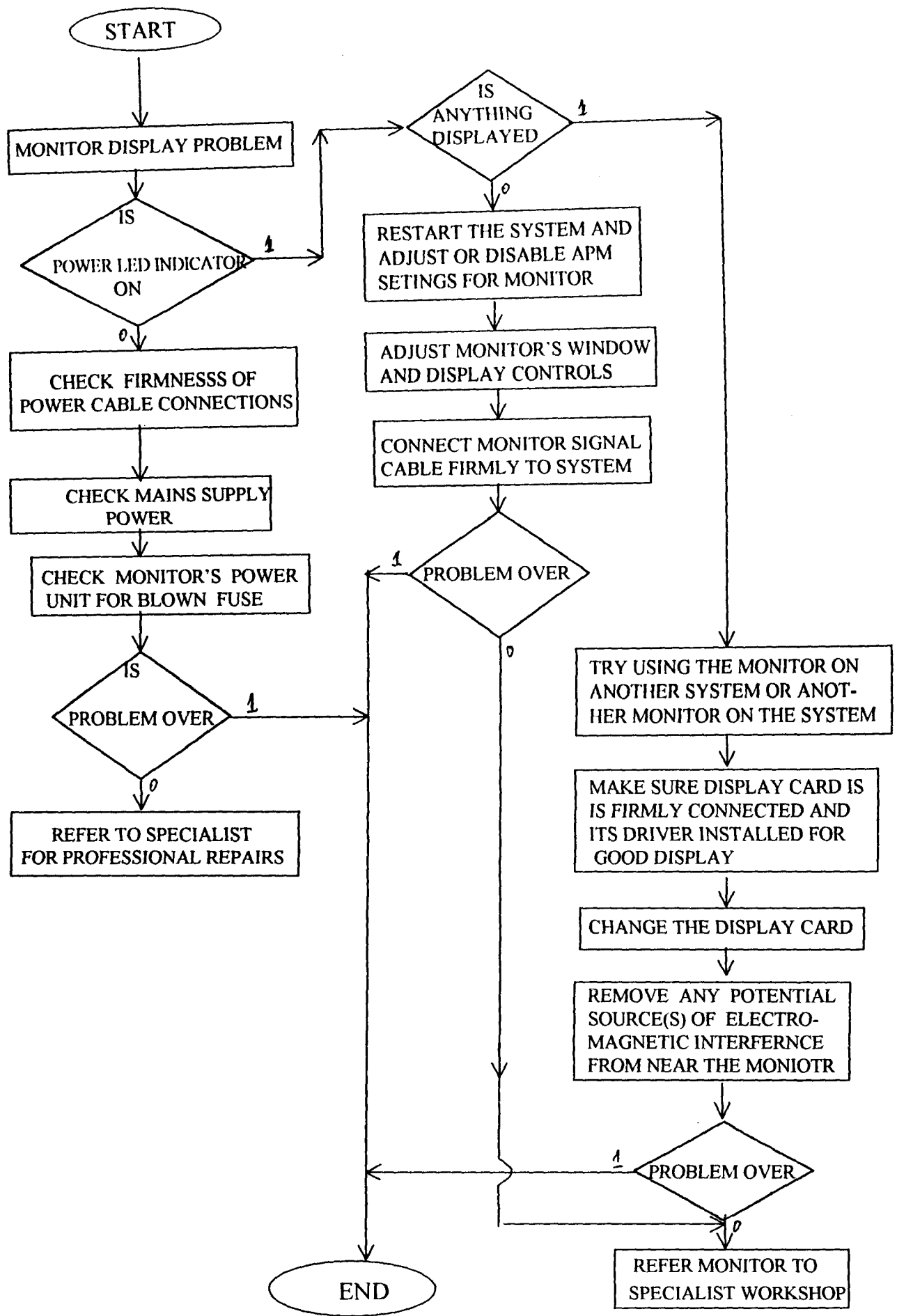


Fig. 2.5 FLOWCHART FOR TROUBLESHOOTING MONITOR PROBLEMS

2.3.6

PRINTERS

The printer is the output device with which the computer output information on a paper page.

Computer printers are either dot matrix, LaserJet or an Inkjet printer. The dot matrix print information using dots by striking an inked ribbon (with pins on the print-head) against the paper resting on a carriage board. The LaserJet print by means of guided tiny laser lights and powdered toner. The Inkjet print by firing liquid ink from a cartridge which slide along a bar just above the paper course.

No matter the type, computer printers are subject to failure to print. To fix this problem, make sure power outlet cable to printer is okay and the printer is on (i.e. the Ready or Online indicator light is ON). Try printing a Self Test Page. If this prints fine, then there is a communication problem between the printer and the CPU.

Replace the printer parallel cable connecting the device to the computer with a certified one. If problem persists, check the computer to ensure that the printer's driver is installed and the correct driver is selected. Check your printer's documentation for compatible LPT mode. Set the correct LPT mode in the peripheral devices option of the CMOS setup. If the problem persists, try using the same printer on another system (be sure to select the correct driver). If it print fine, replace the printer's port with an I/O card. Be sure to disable any unwanted port or connector to avoid hardware conflict. If otherwise, refer to service centre for professional repairs.

If the problem is a paper jam, open any cover as necessary to gain access to the paper and remove it.

Clean up the printer using a soft hand brush. Pick up any object inside the printer with a dry handkerchief. Objects gaining access into the printer (e.g. the fuser compartment of a LaserJet printer) or presence of dust particles can easily cause a paper jam. If the printer does not pick paper or complain of "printer out of paper" when papers are loaded, remove the papers and load back properly. If this did not fix the problem, open the printer, locate and clean the paper and paper feeder sensors. They are normally close to the top front end of the paper tray. Presence of particles in a sensor gap could cause malfunction. The paper feeder could be dirty, clean it. Don't force any turning part as this could damage motors controlling movement of paper. If your inkjet printer makes a grinding noise while printing,

open the cover , clean the carriage bar properly with tissue paper. Apply oil (e.g. machine oil), repeat cleaning and lubrication then try printing again. If problem persist, call a professional.

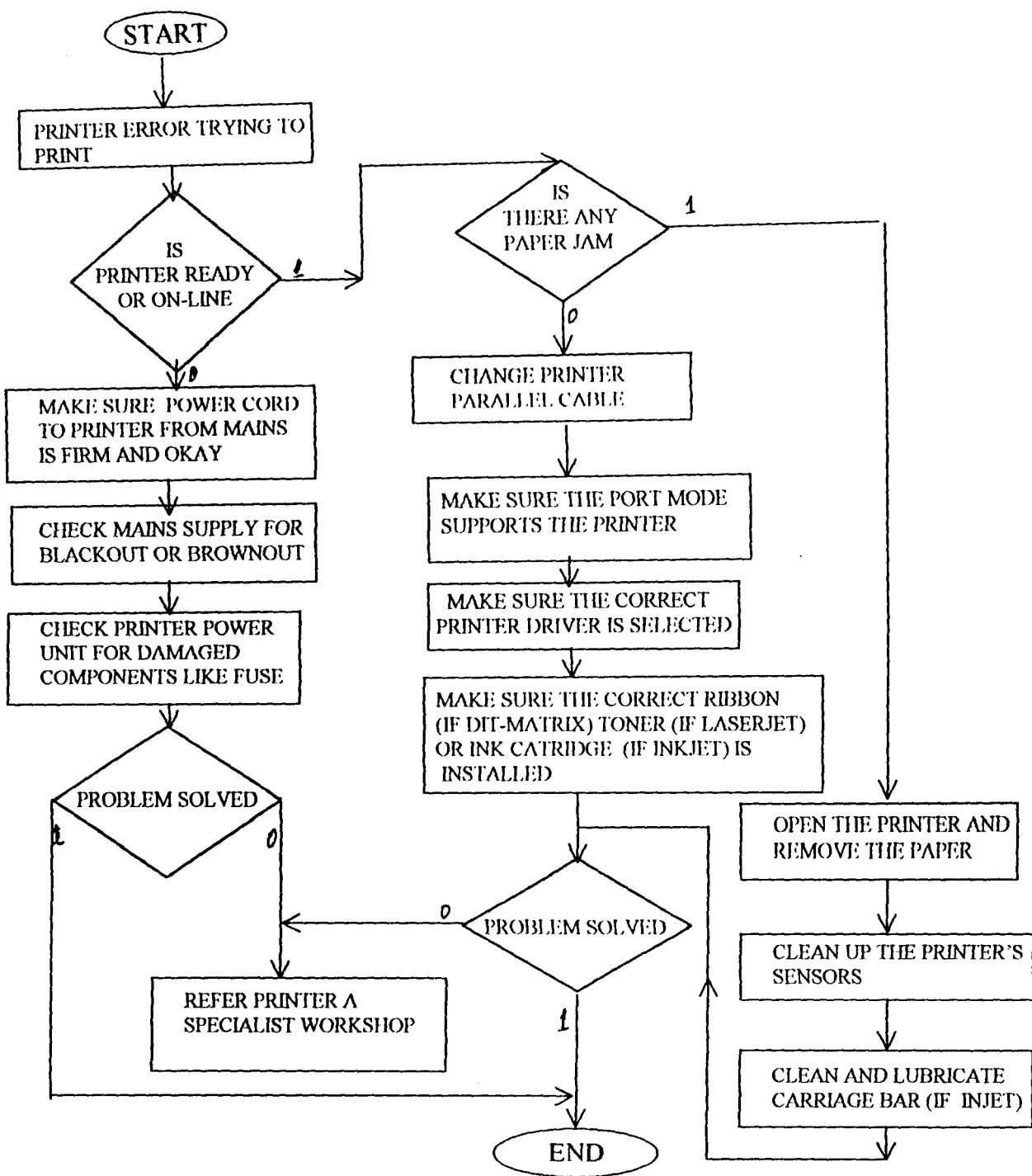


Fig. 2.6 **FLOWCHART FOR TROUBLESHOOTING PRINTER PROBLEMS**

CHAPTER THREE

COMPUTER PROGRAMMING

3.1 PROGRAMMING AND PROGRAMMING LANGUAGES

A computer program refers to a set of ordered and well-planned instructions used to control the operation and performance of the computer to achieve or perform a specific task. These “instruction sets” must follow certain rules as allowed in the programming language in use. The act of using these programs to manipulate the computer is termed ‘programming’.

Just as communication between any two persons can only be possible through the use of a language understood by both persons involved, programming a computer is achieved via a programming language, which the programmer and the computer must understand.

Basically, a computer programming language consist of all the symbols, characters, words and their usage rule that permit a person to communicate with the computer. Generally, we have three major types of programming languages. These are -- Machine language, Assembly language and High level languages.

Every programming language must accept certain types of written instructions that enable a computer system to perform a number of familiar operations. These instructions must fall into the following categories:

1. Input / output instructions.
2. Calculations instructions.
3. Logical / comparison instructions.
4. Storage, removal and movement instructions.

Although, these instruction categories are common to all programming languages, there is marked difference in symbols, characters, codes etc.

3.2 AN OVERVIEW OF PROGRAMMING LANGUAGES

i). MACHINE LANGUAGES

Within the scope of this project, the only language understood by the computer is the machine language. It consists of strings of binary numbers. An instruction in machine language consists of two parts. The first is the command operation or operation code (OPCODE) which tells the computer what function to perform. The second path called the OPERAND tells the computer where to find or store data or other instructions to be manipulated. Writing or debugging a machine language program is very tedious, time consuming and extensive.

A program written in other language (e.g. a high level language) must be translated or interpreted to the computer in machine language for the computer system to understand and execute them.

ii). ASSEMBLY LANGUAGE

In this programming language, symbols are used to represent operation codes of the machine language instruction. These symbols are termed mnemonics. Although the assembly language is quite similar to machine language, use of mnemonics to represent "OPCODES" makes it relatively easier to use. An assembly language program designed for one machine will most likely not work in another computer. The symbols used vary among different makes and models of computer.

iii). HIGH LEVEL LANGUAGE

The development of mnemonic techniques and instructions as used in assembly language gradually led to the development of high level languages. A number of high level languages now exist with each oriented towards solving a particular type of problem. Advantages of high level languages include:

- 1) Unlike assembly language, a high level language is applicable in different makes of computer
- 2) They are easier to use and learn
- 3) They provide better documentation
- 4) Writing a high-level language program requires less time.

Major of high level programming languages available today are:

- i). **BASIC** which stands for Beginners All-purpose Instruction Symbolic Code
- ii). **FORTRAN** : This stands for Formula translation designed for engineering and scientific applications
- iii). **COBOL**: This is an abbreviation for Common Business Orientation Language for use in business applications
- iv). **PL/1**: Stands for Programming Language one. It was designed to solve both scientific and business type problems.
- v). **PASCAL**: This was designed for use in both scientific and file processing applications
- vi). **C – LANGUAGE**: This is a compiler language. It finds good use in scientific, engineering and file processing applications.

3.3 PROGRAMMING IN C OR C++

C is a compiler type of programming language. It involves the use of collection of commands and functions in the form of familiar-looking words that convert into binary code instructions to be executed by the computer. Over the past few years, C-language has become the language of choice among computer programmers for its outstanding features. These include:

i). SPEED

Among the various high-level programming languages, C is closer to the assembly language in terms of speed of program execution. This is possible because some C command directly address the physical hardware of the computer. This makes compiled C-programs execute very fast. The speed of C is so high that it can be used to design operating systems and even other compiler software.

ii). PORTABILITY

Although, C-programming language is closer to assembly language, it is not machine dependent as does assembly language. A source program written in C can run on any computer. A C program is

compiled into an executable file that can run on any computer without having to go through the compiler.

iii). CONTROL

Specifying the order in which statements are to be executed in a computer program is termed control. C as a programming language is very efficient in this area. That is the programmer can easily specify execution order for different program modules. This is possible by using C commands such as the if ...else statement, do ... while statement, case ... switch statement etc. C is also capable of switching between files with the use of the “fopen” (file open) and “fclose” (file closure) commands.

Any meaningful C-program will start with the ‘function main call’ “main()” and the entire program listing is enclosed in brace “{}”. Each line of instruction is terminated with a semicolon (;).

C-Language comprise header (*.h) files which can be called as necessary using the #include command. For example, if the program is to use a disk file or print out information, you will need to call the stdio.h file at the beginning with the command “#include stdio.h”

While writing a C-program, all variables to be used must be declared before they are used. A variable declared as an integer cannot be used as a string or floating-point variable.

3.4 PROGRAM DESIGN

The tutorial program designed for this project work as specified earlier in this write-up was designed from “Turbo C++” version 1.0 for DOS. The entire program is designed in graphics mode. Two programming techniques were used in this program design. These are – “Modula Programming Technique (MPT) and Object Oriented Programming (OOP) in Turbo C++.

To explain the Modula Programming Technique as used in this project, each module consists of two files (a program file and a data file). For example, the main menu (i.e. central control) program file calls the main menu data file for information to output. When an option, say system unit, is selected, the system unit program control file is loaded which in itself calls the system unit data file for information to output. As a module is loaded (depending on the option selected), the floppy disk access indicator light will be seen to come on each time. This shows that the program is opening a file for reading. The program is designed to be an interactive one. It thus involve the full participation of the user.

Each module (program file) consist of structures and functions. Each function performs a specific task for instance, creation of objects, action when any particular key is pressed, display of information on screen etc. This is one of the excess of Object Oriented Programming.

A major limitation of the program is the inability to use the mouse even despite the creation of button objects.

The program design is such that when the software is started, it opens with a “welcome” screen which contain information such as project topic, name of student/programmer, name of project supervisor etc. when a keyboard key is pressed, it proceed to the central control module (called the MAIN MENU). Here, provided a valid option is selected, the program loads the appropriate module else an error message is displayed. In each of the specific hardware type program module, the problems associated with such hardware that are user fixable are displayed. The user simply need to selected a valid problem definition and the program will display a step solution for the problem else an error message will again be generated. Sounds of different frequencies are included signal different actions.

Below are the flowcharts for the operation of the tutorial program developed for this project work. The disk is attached for reference and testing.

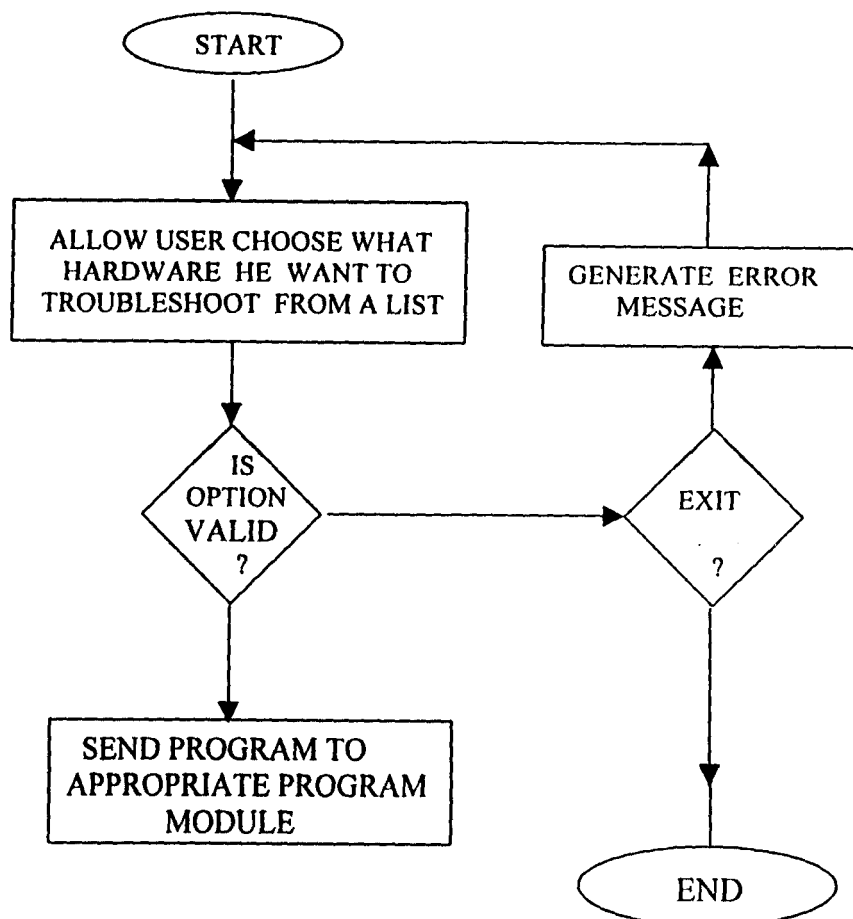


Fig. 3.1 FLOWCHART FOR THE MAIN MENU PROGRAM MODULE

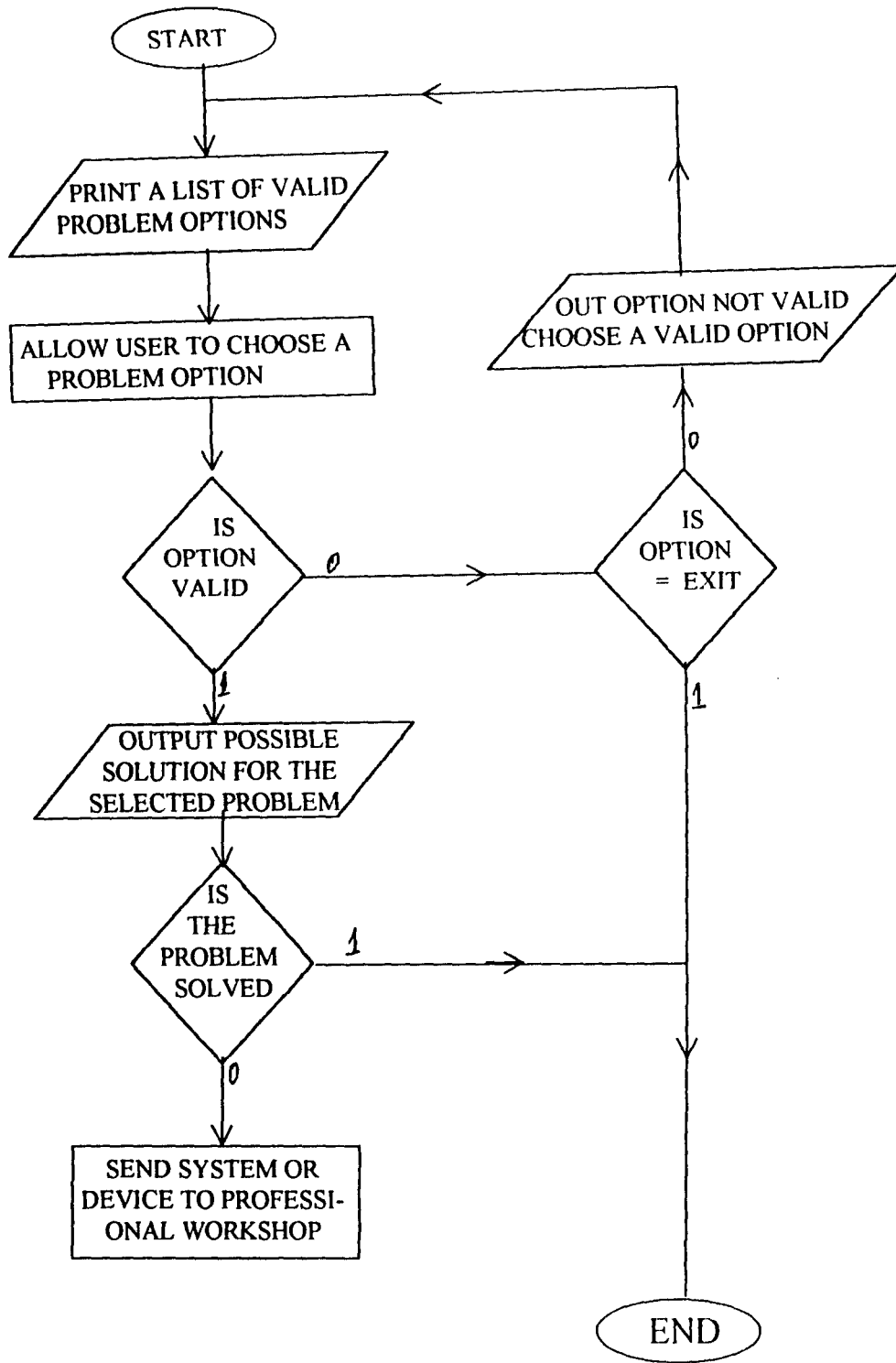


Fig. 3.2 GENERAL FLOWCHART FOR A DEVICE TROUBLESHOOTING PROGRAM MODULE

CHAPTER FOUR

4.1 CONCLUSION

The contents of this project report cover most of the common hardware problems of personal microcomputers. It should not be taken as an end in itself but as a means to an end. I strongly believe that the aims and objectives of starting this project have been met. That is acquainting the users and private owners of microcomputers with basic principles and skills of troubleshooting most common problems of their PCs. Thereby making the use of these machines cost effective.

As a benefit of this project work, I have become exposed to the practical aspects of computer engineering. This has come to being by way of researches through books on the subject matter and consulting with practising computer engineering personnel and firms. Therefore, I can assure the validity and dependability of the contents of this report as regards the topic matter.

4.2 RECOMMENDATIONS

The department should try to acquire some computer systems for the purpose of practical training of students of the department. It is disheartening to note that some of the students of Electrical / Computer Engineering Department (up to 300L) have never seen a computer motherboard practically not to mention troubleshooting or fixing its problems.

REFERENCES

1. **UPGRADING AND REPAIRING PCs** - - - - - 6th Edition
By Scott Mueller
Published 1996 by Que[®] Corporation
2. **TROUBLESHOOTING** - - The book of simple solutions for most
computing problems
By PCNOVICE & Smart Computing
Published 1998 by Sandhills Publishing
120 West Harvest Drive
Lincoln
3. **BUILD A PENTIUM COMPUTER** - - An interactive video tutorial,
(MASTER TECH '97) utilities and diagnostic software
By Gradient Software Inc.
4. **COMPUTER ASSISTED ANALYSIS AND REPAIRS OF i80386
BASED MICROCOMPUTERS** (A Final Year Project)
By Ekuniyi Olumuyiwa
Elect / Comp. Engrg. Dept., F.U.T. Minna
February 1997
5. **A SHORT STRAIGHT LOOK AT THE MICROCOMPUTER**
By Daniel B. Ayo (PhD)
Published 1996 by RMRDC, Abuja
6. **C HOW TO PROGRAM**
By H. M. Deitel & P. J. Deitel
Deitel and Associates
Published 1992 by Prentice - Hall, Inc.
Englewood Cliffs, New Jersey
7. **IBM HARDWARE HANDBOOK**
By IBM Corporation
8. **IBM PERSONAL COMPUTER TROUBLESHOOTING AND REPAIR**
By IBM Corporation
9. **OBJECT ORIENTED PROGRAMMING WITH TURBO C++**
By Keith Weiskamp and Loren Heiny
Published 1998, by John Willey & Sons, Inc.
10158 - 0012, New York.

IC	Integrated Circuit
IDE	Integrated Drive Electronics
IRQ	Interrupt Request
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MCA	Micro Channel Architecture
MCGA	Muti-Colour Graphics Adapter
Microprocessor	A solid state Central Processing Unit that function much like a computer on a chip. It is an IC that accept coded instructions for execution
Modem	MOdualtor DEModulator. A device that convert electrical signals from a computer into an audio form transmittable over telephone lines, or vice versa
Module	An assembly that contain a complete circuit or sub-circuit
Motherboard	The main circuit board in the computer. also called a system board or main board
Peripheral	Any piece of equipment used in computer systems that is an attachment to the computer e.g. hard disks, printers etc.
POST	Power On Self Test
RAM	Random Access Memory
ROM	Read Only Memory
SIMM	Single Inline Memory Module
Troubleshooting	The task of determining the cause of a problem
UPS	Un-interruptible Power Supply
VESA	Video Electronics Standard Architecture
VGA	Video Graphics Adapter
Virus (Computer virus)	A type of resident program designed to attach itself to other programs. At a later time, when the virus is runnig, it causes and undesirable action to take place
ZIF	Zero Insertion Force

APPENDIX B

PROGRAM LISTING

```
// CENRAL CONTROL PROGRAM FILE

#include<stdio.h>
#include<stdlib.h>
#include<graphics.h>
#include<conio.h>
#include<dos.h>
#include <process.h>
#include <stdio.h>
#include <errno.h>

#define KEY1 0x0231
#define KEY2 0x0332
#define KEY3 0x0433
#define KEY4 0x0534
#define KEY5 0x0635
#define ESC 0x011b
#define ZF 0x40
#define TRUE 1
#define FALSE 0
typedef struct{
    int x1,y1,x2,y2;
}vertexpoints;
typedef struct{
    char *sdata;
}stringdata;
typedef struct{ vertexpoints sent,ret;
    int fcolor,fpattern,blcolor,trcolor,thk,txtcolor,txtdir;
    int txtfont,txtsize,stringnum;
}animatedbar;
stringdata sd[100];
animatedbar ab[20];
struct viewporttype vp;
struct REGPACK reg;

int gdriver,gmode,errorcode,waste;
int firstmenu,lastmenu,cursorpostn,maxx,maxy;

void initiallizgr(void);
void winshape(void);
void mainmenu(void);
void optionwin(void);
void submenul(void);
void designbar(int p);
void writestringat (int p,int strgnum);
void ritestringat (int x,int y,int a,int j,int p,int strgnum);
void boton(int a,int b,int c,int d,int e,int f,int j,int s,int t,char fc,\
    char tr,char bl,char tc,char tf);
void filldata(void);
void clearfill(vertexpoints vps,int fillcolor,int fpattern);
int makeachoice(void);
void quit(void);
void syst(int argc, char *st[]);
void mon(int argc, char *st[]);
void inpt(int argc, char *st[]);
void dsk(int argc, char *st[]);
void prn(int argc, char *st[]);

main(){
    initiallizgr();
    filldata();
    submenul();
    return 0;
}
```

```

void initializegr(void){
    int i;
    gdriver = DETECT;
    initgraph(&gdriver,&gmode,"c:\\tc\\bgi");
    errorcode = graphresult();
    if(errorcode != grOk){
        printf("graph error : %\n",grapherrormsg(errorcode));
        printf("press any key to halt:");
        getch();
        exit(1);
    }
    maxx = getmaxx();
    maxy = getmaxy();
}

void winshape(void){
    cleardevice();
    filldata();
    boton(0,0,0,0,maxx,25,0,1,1,LIGHTRED,WHITE,LIGHTGRAY,DARKGRAY,TRIPLEX_FONT);
    boton(1,1,25,0,maxx,maxy-40,0,4,1,LIGHTBLUE,WHITE,LIGHTGRAY,WHITE,TRIPLEX_FONT);
    boton(2,2,maxy-20,0,maxx,20,0,5,2,LIGHTGRAY,WHITE,LIGHTGRAY,RED,SMALL_FONT);
    boton(3,3,3,maxx-75,70,19,0,4,2,LIGHTGRAY,DARKGRAY,DARKGRAY,RED,SMALL_FONT);
    writestringat(3,204);
    writestringat(0,0);
}

void mainmenu(void){
    winshape();
    writestringat(2,200);
    boton(3,3,66,90,maxx-180,maxy-
110,0,2,0,LIGHTGRAY,DARKGRAY,DARKGRAY,BLACK,DEFAULT_FONT);
    boton(4,4,45,90,maxx-180,21,0,1,1,RED,WHITE,DARKGRAY,WHITE,TRIPLEX_FONT);
    boton(5,5,maxy-45,90,maxx-180,15,0,4,0,LIGHTGRAY,BLACK,DARKGRAY,BLUE,SMALL_FONT);
    boton(6,6,70,95,maxx-190,20,0,1,0,LIGHTGRAY,LIGHTGRAY,LIGHTGRAY,BLUE,SANS_SERIF_FONT);
}

void optionwin(void){
    makeachoice();
    submenul();
}

void submenul(void){
    int i,k=1;
    mainmenu();
    boton(7,11,128,125,40,25,30,2,3,LIGHTGRAY,WHITE,DARKGRAY,BLUE,DEFAULT_FONT);
    ritestringat(90,20,4,55,3,11);
    writestringat(4,10);
    writestringat(5,67);
    for(i=7; i<=12; i++){
        writestringat(i,k);
        k++; }
}

optionwin();
}

void designbar(int p){
    int dx,dy,i;
    setlinestyle(SOLID_LINE,0,NORM_WIDTH);
    setfillstyle(ab[p].fpattern,ab[p].fcolor);
    bar(ab[p].sent.x1,ab[p].sent.y1,ab[p].sent.x2,ab[p].sent.y2);
    dx = ab[p].sent.x2-ab[p].sent.x1;
    dy = ab[p].sent.y2-ab[p].sent.y1;
    for( i = 0 ; i <= ab[p].thk ; i++){
        moveto(ab[p].sent.x1+i,ab[p].sent.y1+i);
        setcolor(ab[p].blcolor);
        linerel(0,dy-i*2);
        linerel(dx-i*2,0);
        setcolor(ab[p].trcolor);
        linerel(0,-(dy-i*2));
        linerel(-(dx-i*2),0);
    }
}

```



```

        ab[p].ret.d.x1 =      ab[p].sent.x1 +      ab[p].thk+1;
        ab[p].ret.d.y1 =      ab[p].sent.y1 +      ab[p].thk+1;
        ab[p].ret.d.x2 =      ab[p].sent.x2 -      (ab[p].thk+1);
        ab[p].ret.d.y2 =      ab[p].sent.y2 -      (ab[p].thk+1);
    }
    void writestringat (int p,int strgnum){
        int x,y;
        getviewsettings(&vp);
        setviewport (ab[p].ret.d.x1,ab[p].ret.d.y1,ab[p].ret.d.x2,ab[p].ret.d.y2,1);
        x = (ab[p].ret.d.x2-ab[p].ret.d.x1)/2;
        y = (ab[p].ret.d.y2-ab[p].ret.d.y1)/2;
        settxtstyle(ab[p].txtfont,ab[p].txtdir,ab[p].txtsize);
        setcolor(ab[p].txtcolor);
        settxtjustify(CENTER_TEXT,CENTER_TEXT);
        outtextxy(x,y,sd[strgnum].sdata);
        ab[p].stringnum = strgnum;
        setviewport (vp.left,vp.top,vp.right,vp.bottom,1);
    }
    void ritestringat (int x,int y,int a,int j,int p,int strgnum){
        int m;
        getviewsettings(&vp);
        setviewport (ab[p].ret.d.x1,ab[p].ret.d.y1,ab[p].ret.d.x2,ab[p].ret.d.y2,1);
        settxtstyle(ab[p].txtfont,ab[p].txtdir,ab[p].txtsize);
        setcolor(ab[p].txtcolor);
        settxtjustify(LEFT_TEXT,CENTER_TEXT);
        for (m=0; m<=a; m++){
            outtextxy(x,y=y+j,sd[strgnum+m].sdata);
        }
        ab[p].stringnum = strgnum;
        setviewport (vp.left,vp.top,vp.right,vp.bottom,1);
    }
    void filldata(void){
#include "mainmenu.h"
    }
    void boton(int a,int b,int c,int d,int e,int f,int j,int s,int t,char fc,char
        tr,char bl,char tc,char tf){
        int i, k=16, p;
        getviewsettings(&vp);
        ab[a].sent.y1 = c;
        for(i=a; i<=b ;i++){
            ab[i].sent.x1 = d;
            ab[i].sent.x2 = ab[i].sent.x1 + e;
            ab[i].sent.y2 = ab[i].sent.y1 + f;
            if(i >= a)
                ab[i].fcolor =      fc;
            ab[i].fpattern = SOLID_FILL;
            ab[i].thk = t;
            ab[i].trcolor = tr;
            ab[i].blcolor = bl;
            ab[i].txtcolor = tc;
            ab[i].txtfont = tf;
            ab[i].txtsize = s;
            designbar(i);
            ab[i+1].sent.y1 = ab[i].sent.y2 + j;
            k++;
        }
    }
    void clearfill(vertexpoints vps,int fillcolor,int fpattern){
        getviewsettings(&vp);
        setviewport (vps.x1,vps.y1,vps.x2,vps.y2,1);
        clearviewport();
        setviewport (vp.left,vp.top,vp.right,vp.bottom,1);
            setfillstyle(fpattern,fillcolor);
            bar(vps.x1,vps.y1,vps.x2,vps.y2);
        }
    }

```

```

int makeachoice(void){
    char *st;
    unsigned int a;
    do{
        reg.r_ax = 0x1 << 8;
        intr(0x16,&reg);
    }while(reg.r_flags & ZF);
    a=reg.r_ax;
    waste = getch();

    if(a==KEY1){sound(1500);delay(250);nosound();syst(0,st);    }
    else if(a==KEY2){sound(1500);delay(250);nosound();inpt(0,st);    }
    else if(a==KEY3){sound(1500);delay(250);nosound();dsk(0,st);    }
    else if(a==KEY4){sound(1500);delay(250);nosound();mon(0,st);    }
    else if(a==KEY5){sound(1500);delay(250);nosound();prn(0,st);    }
    else if(a==ESC){ quit();    }
    else {
        ab[5].txtcolor=RED;
        clearfill(ab[5].ret,d,ab[5].fcolor,ab[5].fpattern);
        writestringat(5,61);
        sound(35);delay(200);nosound();
        delay(1800);
    }return(a);
}

void quit(void){
    winshape();
    ritestringat(100,140,1,50,1,230);
    delay(3000);
    closegraph();
    exit(1);
    return;}

void syst(int argc, char *st[]){ execvp("system.ex_", st);}
void mon(int argc, char *st[]){ execvp("monitor.ex_", st);}
void inpt(int argc, char *st[]){ execvp("input_d.ex_", st);}
void dsk(int argc, char *st[]){ execvp("dskdrv.ex_", st);}
void prn(int argc, char *st[]){ execvp("printer.ex_", st);}

```

```
// MAINMENU (CENTRAL CONTROL PROGRAM MODULE) DATA FILE
```

```
sd{0}.sdata = "TROUBLESHOOTING MICROCOMPUTER HARDWARE PROBLEMS ";
sd{1}.sdata = "1";
sd{2}.sdata = "2";
sd{3}.sdata = "3";
sd{4}.sdata = "4";
sd{5}.sdata = "5";

sd{10}.sdata = " M A I N           M E N U";
sd{11}.sdata = "SYSTEM UNIT           ";
sd{12}.sdata = "INPUT DEVICES         ";
sd{13}.sdata = "DISKS AND DISK-DRIVES  ";
sd{14}.sdata = "MONITOR - DISPLAY UNIT ";
sd{15}.sdata = "PRINTERS              ";

sd{61}.sdata = "Selected OPTION is invalid -- Press Key 1 to 5 or ESC to quit";
sd{67}.sdata = "Type a number on a button to select its corresponding option";
sd{200}.sdata = "Select the hardware type you want to troubleshoot";
sd{204}.sdata = "ESC->Quit";
sd{230}.sdata = "S H U T T I N G   D O W N";
sd{231}.sdata = " P L E A S E   W A I T ";
```

```
// SYSTEM UNIT TROUBLESHOOTING PROGRAM MODULE FILE
```

```
#include<stdio.h>
#include<stdlib.h>
#include<graphics.h>
#include<conio.h>
#include<dos.h>
```

```
#define KEY1 0x0231
#define KEY2 0x0332
#define KEY3 0x0433
#define KEY4 0x0534
#define ESC 0x011b
#define ZF 0x40
```

```
typedef struct(
    int x1,y1,x2,y2;
    )vertexpoints;
typedef struct(
    char *sdata;
    )stringdata;
typedef struct(    vertexpoints sent,ret;
    int fcolor,fpattern,blcolor,trcolor,thk,txtcolor,txtdir;
    int txtfont,txtsize,stringnum;
    )animatedbar;
stringdata sd[250];
animatedbar ab[20];
```

```
struct viewporttype vp;
struct REGPACK reg;
int gdriver,gmode,errorcode,waste;
int firstmenu,lastmenu,maxx,maxy;
```

```
void winshape(void);
void status(void);
void unitwin(void);
void optionwin(void);
void submenu(void);
void trsysunit(void);
void sys_1(void);
void notcome(void);
void no_ini(void);
void again(void);
void turnoff(void);
void initiallizegr(void);
void designbar(int p);
void writestringat (int p,int strgnum);
void ritestringat (int x,int y,int a,int j,int p,int strgnum);
void menuwin(void);
void boton(int a,int b,int c,int d,int e,int f,int j,int s,int t,char fc,char tr,\
    char bl,char tc,char tf);
void filldata(void);
void clearfill(vertexpoints vps,int fillcolor,int fpattern);
int makeachoice(void);
void quit(void);
void homel(int argc, char *st[]);
void home(int argc, char *st[]);
```

```
main(){
    initiallizegr();
    filldata();
    submenu();
    getch();
    return 0;
}
```

```
void initiallizegr(void){
    int i;
    gdriver = DETECT;
    initgraph(&gdriver,&gmode,"c:\\tc\\bgi");
    errorcode = graphresult();
    if(errorcode != grOk){
        printf("graph error : %\n",grapherrormsg(errorcode));
        printf("press any key to halt:");
        getch();
        exit(1);
    }
}
```

```

    maxx = getmaxx();
    maxy = getmaxy();
}

void winshape(void) {
    cleardevice();
    filldata();
    boton(0,0,0,0,maxx,25,0,1,1,LIGHTRED,LIGHTGRAY,LIGHTGRAY,DARKGRAY,TRIPLEX_FONT);
    boton(2,2,25,0,maxx,maxy-43,0,4,1,LIGHTBLUE,LIGHTGRAY,LIGHTGRAY,WHITE,TRIPLEX_FONT);
    boton(1,1,maxy-18,0,maxx,18,0,4,2,LIGHTGRAY,WHITE,LIGHTGRAY,RED,SMALL_FONT);
    boton(3,3,3,maxx-75,70,19,0,4,2,LIGHTGRAY,DARKGRAY,DARKGRAY,RED,SMALL_FONT);
    writestringat(3,204);
    writestringat(0,0);
}

void unitwin(void) {
    winshape();
    boton(4,4,66,35,maxx-70,maxy-110,0,5,0,LIGHTGRAY,DARKGRAY,DARKGRAY,BLACK,SMALL_FONT);
    boton(3,3,45,35,maxx-70,21,0,1,1,RED,WHITE,DARKGRAY,WHITE,TRIPLEX_FONT);
    boton(5,5,maxy-45,35,maxx-70,15,0,4,0,LIGHTGRAY,BLACK,DARKGRAY,BLUE,SMALL_FONT);
    boton(6,6,70,120,maxx-
240,20,0,1,0,LIGHTGRAY,LIGHTGRAY,LIGHTGRAY,BLUE,SANS_SERIF_FONT);
}

void sys_1(void) {
    makeachoice();
    submenu();
}

void notcome(void) {
    unitwin();
    writestringat(3,15);
    writestringat(5,203);
    writestringat(6,20);
    ritestringat(10,30,12,18,4,21);
    ab[4].txtsize=1;ab[4].txtfont=DEFAULT_FONT;
    ritestringat(60,335,1,13,4,7);
    clearfill(ab[1].retd,ab[1].fcolor,ab[1].fpattern);
    writestringat(1,210);
    getch();
    return;
}

void no_ini(void) {
    unitwin();
    writestringat(3,15);
    clearfill(ab[1].retd,ab[1].fcolor,ab[1].fpattern);
    writestringat(1,211);
    writestringat(5,203);
    writestringat(6,34);
    ritestringat(6,30,17,18,4,35);
    getch();

    unitwin();
    sound(700);delay(20);nosound();
    writestringat(3,15);
    clearfill(ab[1].retd,ab[1].fcolor,ab[1].fpattern);
    writestringat(1,211);
    writestringat(5,203);
    writestringat(6,34);
    ritestringat(9,30,14,18,4,53);
    getch();

    sound(700);delay(20);nosound();
    unitwin();
    writestringat(3,15);
    clearfill(ab[1].retd,ab[1].fcolor,ab[1].fpattern);
    writestringat(1,210);
    writestringat(5,203);
    writestringat(6,34);
    ritestringat(9,30,14,18,4,68);
    ab[4].txtsize=1;ab[4].txtfont=DEFAULT_FONT;
    ritestringat(60,335,1,13,4,7);
    getch();
    return;
}

```

```

void turnoff(void) {
unitwin();
writestringat(3,15);
clearfill(ab[1].retd,ab[1].fcolor,ab[1].fpattern);
writestringat(1,210);
writestringat(5,203);
writestringat(6,90);
ritestringat(9,25,20,15,4,91);
ab[4].txtsize=1;ab[4].txtfont=DEFAULT_FONT;
ritestringat(60,335,1,13,4,7);
getch();
return;
}

```

```

void optionwin(void) {
winshape();
writestringat(1,205);
boton(1,1,101,125,maxx-250,231,0,1,0,LIGHTGRAY,DARKGRAY,DARKGRAY,BLACK,DEFAULT_FONT);
boton(2,2,80,125,maxx-250,21,0,1,1,RED,WHITE,DARKGRAY,WHITE,TRIPLEX_FONT);
boton(3,3,maxy-150,125,maxx-250,15,0,4,0,LIGHTGRAY,BLACK,DARKGRAY,BLUE,SMALL_FONT);
}

```

```

void submenu(void) {
int i, k=1;
optionwin();
boton(5,8,135,140,40,25,20,2,2,LIGHTGRAY,WHITE,DARKGRAY,BLUE,DEFAULT_FONT);
ritestringat(70,4,3,45,1,11);
writestringat(2,10);
writestringat(3,201);

for(i=5; i<=8; i++){
writestringat(i,k);
k++;
}
sys_1();
}

```

```

void designbar(int p) {
int dx,dy,i;
setlinestyle(SOLID_LINE,0,NORM_WIDTH);
setfillstyle(ab[p].fpattern,ab[p].fcolor);
bar(ab[p].sent.x1,ab[p].sent.y1,ab[p].sent.x2,ab[p].sent.y2);
dx = ab[p].sent.x2-ab[p].sent.x1;
dy = ab[p].sent.y2-ab[p].sent.y1;
for( i = 0 ; i <= ab[p].thk ; i++){
moveto(ab[p].sent.x1+i,ab[p].sent.y1+i);
setcolor(ab[p].blcolor);
linere1(0,dy-i*2);
linere1(dx-i*2,0);
setcolor(ab[p].trcolor);
linere1(0,-(dy-i*2));
linere1(-(dx-i*2),0);
}
ab[p].retd.x1 = ab[p].sent.x1 + ab[p].thk+1;
ab[p].retd.y1 = ab[p].sent.y1 + ab[p].thk+1;
ab[p].retd.x2 = ab[p].sent.x2 - (ab[p].thk+1);
ab[p].retd.y2 = ab[p].sent.y2 - (ab[p].thk+1);
}

```

```

void writestringat (int p,int strgnum){
int x,y;
getviewsettings(&vp);
setviewport(ab[p].retd.x1,ab[p].retd.y1,ab[p].retd.x2,ab[p].retd.y2,1);
x = (ab[p].retd.x2-ab[p].retd.x1)/2;
y = (ab[p].retd.y2-ab[p].retd.y1)/2;
settextstyle(ab[p].txtfont,ab[p].txtmdir,ab[p].txtsize);
setcolor(ab[p].txtcolor);
if(p != 9) {
settextjustify(CENTER_TEXT,CENTER_TEXT);
outtextxy(x,y,sd[strgnum].sdata);
}
else{

```

```

    settextjustify(CENTER_TEXT,BOTTOM_TEXT);
    outtextxy(x,y,sd[strgnum].sdata);
    settextjustify(CENTER_TEXT, TOP_TEXT);
    outtextxy(x,y,sd[strgnum+1].sdata);
}
ab[p].stringnum = strgnum;
setviewport(vp.left, vp.top, vp.right, vp.bottom, 1);
}

```

```

void ritestringat (int x,int y,int a,int j,int p,int strgnum){
    int m;
    getviewsettings(&vp);
    setviewport(ab[p].ret.d.x1,ab[p].ret.d.y1,ab[p].ret.d.x2,ab[p].ret.d.y2,1);

```

```

    settextstyle(ab[p].txtfont,ab[p].txtedir,ab[p].txtsize);
    setcolor(ab[p].txtcolor);
    if(p != 9){
        settextjustify(LEFT_TEXT,LEFT_TEXT);
        for (m=0; m<=a; m++){
            outtextxy(x,y+y+j,sd[strgnum+m].sdata);
        }

```

```

    ab[p].stringnum = strgnum;
    setviewport(vp.left, vp.top, vp.right, vp.bottom, 1);
}

```

```

void filldata(void){
#include "stringing.h"
}

```

```

void boton(int a,int b,int c,int d,int e,int f,int j,int s,int t,char fc,char tr,char bl,
char tc,char tf){

```

```

    int i, k=16, p;
    getviewsettings(&vp);
    ab[a].sent.y1 = c; //60;
    for(i=a; i<=b; i++){
        ab[i].sent.x1 = d; //42; //maxx/8;
        ab[i].sent.x2 = ab[i].sent.x1 + e; //55; //maxx-10;
        ab[i].sent.y2 = ab[i].sent.y1 + f; //40;
        if(i >= a)
            ab[i].fcolor = fc; //LIGHTMAGENTA;
        ab[i].fpattern = SOLID_FILL;
        ab[i].thk = t;
        ab[i].trcolor = tr; //LIGHTMAGENTA;
        ab[i].blcolor = bl; //LIGHTMAGENTA;
        ab[i].txtcolor = tc;
        ab[i].txtfont = tf;
        ab[i].txtsize = s; //3;
        designbar(i);
        ab[i+1].sent.y1 = ab[i].sent.y2 + j; //0;
        k++;
    }
}

```

```

void clearfill(vertexpoints vps,int fillcolor,int fpattern){
    getviewsettings(&vp);
    setviewport(vps.x1,vps.y1,vps.x2,vps.y2,1);
    clearviewport();
    setviewport(vp.left, vp.top, vp.right, vp.bottom, 1);
    setfillstyle(fpattern, fillcolor);
    bar(vps.x1,vps.y1,vps.x2,vps.y2);
}

```

```

int makeachoice(void)
{
    unsigned int a; char *st;
    do{
        reg.r_ax = 0x1 << 8;
        intr(0x16, &reg);
    }while(reg.r_flags & ZF);
    a=reg.r_ax;
    waste = getch();
}

```

```
if(a==KEY1){sound(500);delay(250);nosound();notcome();}
else if(a==KEY2){sound(500);delay(250);nosound();no_ini();}
else if(a==KEY3){sound(500);delay(250);nosound();turnoff();}
else if(a==KEY4){sound(300);delay(500);nosound();home(0,st);}
else if(a==ESC){quit();}
else {
    ab[3].txtcolor=RED;
    clearfill(ab[3].ret,ab[3].fcolor,ab[3].fpattern);
    writestringat(3,202);
    sound(35);delay(200);nosound();
    delay(1800);
}return(a);
}

void quit(void){
winshape();
ritestringat(100,140,1,50,2,230);
delay(3000);
closegraph();
exit(1);
}

void home(int argc, char *st[]){execvp("mainmenu.exe", st);}
```


// DATA FILE FOR SYSTEM UNIT TROUBLESHOOTING PROGRAM MODULE

```
sd[0].sdata = "TROUBLESHOOTING MICROCOMPUTER HARDWARE PROBLEMS";
sd[1].sdata = "1";
sd[2].sdata = "2";
sd[3].sdata = "3";
sd[4].sdata = "4";
sd[7].sdata = " If the above steps could not solve the problem, refer ";
sd[8].sdata = "unit to Service/Repair center for professional attention !";
```

```
sd[10].sdata = "SYSTEM UNIT HARDWARE PROBLEMS";
sd[11].sdata = "System did not come ON";
sd[12].sdata = "System came ON but could not initialize";
sd[13].sdata = "System turns off without warning";
sd[14].sdata = "Exit to main menu";
sd[15].sdata = "T R O U B L E S H O O T I N G   S Y S T E M   U N I T";
```

// SYSTEM DID NOT COME ON

```
sd[20].sdata = "SYSTEM DID NOT COME ON";
sd[21].sdata = "- Make sure that the power cables are connected firmly to the system";
sd[22].sdata = " unit, monitor and mains outlets.";
sd[24].sdata = "- Check to ensure that the power buttons on the system unit, monitor";
sd[25].sdata = " and any intermediate power supply unit such as a UPS, SurgeArrester";
sd[26].sdata = " or voltage stabilizer in use are switched ON. Confirm from device LED";
sd[27].sdata = " indicator.";
sd[29].sdata = "- Read the output of intermediate unit for adequate supply using ";
sd[30].sdata = " the multimeter.";
sd[32].sdata = "- If nothing works, open the system case and test the power module for";
sd[33].sdata = " damaged components like fuses or power (rectifier) diodes.";
```

// SYSTEM COLD NOT INITIALIZE

```
sd[34].sdata = "SYSTEM COULD NOT INITIALIZE";
sd[35].sdata = " If the power indicator of the system unit is ON but the unit could not";
sd[36].sdata = " initialize (or start-up), follow the simple steps below to rectify";
sd[37].sdata = " problem.";
sd[38].sdata = "- confirm the efficiency of power supply to the system using the meter.";
sd[40].sdata = "- Make sure that the keyboard is firmly connected to the unit.";
sd[42].sdata = "- Free up the keyboard keys. A key held locked down will disrupt smooth";
sd[43].sdata = " start-up of the system. Regular cleaning of the keyboard will reduce";
sd[44].sdata = " this error to bear minimum.";
sd[46].sdata = "- If these could not solve the problem, open the unit and ensure that the ";
sd[47].sdata = " microprocessor is firmly seated in its socket.";
sd[49].sdata = "- Make sure the extended memory (RAM) chip is firm in its slot. If not,";
sd[50].sdata = " remove and re-install it properly.";
sd[53].sdata = "- If the system makes one short beep followed by a long one, it indicates";
sd[54].sdata = " a faulty RAM memory module. If only one of this is in use, replace it.";
sd[55].sdata = " If more than one are used, remove them one after the other restarting ";
sd[56].sdata = " the computer in each case to determine the affected module(s). Replace";
sd[57].sdata = " faulty modules with a new one of the same type; EDORAM or SDRAM ";
sd[58].sdata = " (SIMMs or DIMMs).";
sd[60].sdata = "- Check to make sure that if there is key-lock on the system, it is";
sd[61].sdata = " opened. A locked keylock means the keyboard is disabled and will";
sd[62].sdata = " therefore not allow the system to initialize.";
sd[64].sdata = "- If the system beep five(5) times after power on and could not start-up";
sd[65].sdata = " it means there is a problem with the micro-processor, remove it and";
sd[66].sdata = " re-install. If problem continue, replace the micro-processor.";
sd[68].sdata = " If the system did not beep at all and the extended RAM memory is ";
sd[69].sdata = " ascertained good, then follow these steps to fix the problem.";
sd[71].sdata = "- Check that the processor and memory modules are correctly installed.";
sd[72].sdata = " take note of module alignment or else you may damage the memory module.";
sd[73].sdata = " The notch on the processor chip should align with the pin-1 of the CPU.";
sd[74].sdata = " socket (usually a Zero Insertion Force - ZIF- type). The microprocessor";
sd[75].sdata = " should sit flush in the socket -- Don't force it. Don't forget to mount";
sd[76].sdata = " the CPU cooling fan or else, the microprocessor can easily get damaged ";
sd[77].sdata = " due to excessive undissipated heat.";
sd[79].sdata = "- If you added hardware, remove it and restart the computer. If this";
sd[80].sdata = " solve the problem, consult the installation manual of the hardware you";
sd[81].sdata = " are trying to install for proper setting of jumpers and configuration";
sd[82].sdata = " of the hardware. If not, contact a professional.";
```

// SYSTEM TURNS OFF WITHOUT WARNING

```
sd[90].sdata = "SYSTEM TURNS OFF WITHOUT WARNING";
sd[91].sdata = "- Make sure that the power cords are securely plugged into the power";
sd[92].sdata = " outlet.";
sd[94].sdata = "- Make sure that the cables are connected correctly and securely to";
sd[95].sdata = " the system unit.";
sd[97].sdata = "- Check for a blown fuse, tripped circuit breaker especially in the";
sd[98].sdata = " circuit or device supplying power to the system unit.";
sd[100].sdata = "- Unplug the system from the electrical outlet, wait for at least ";
sd[101].sdata = " 15 seconds and then plug the unit back. Switch ON the computer.";
```

```
//          STATUS
sd[201].sdata = "Type the number on a button to select its corresponding option";
sd[202].sdata = "Option selected is not valid -- Try again after 2 seconds";
sd[203].sdata = "Follow steps carefully to resolve hardware problem. Else, see a
professional";
sd[204].sdata = "ESC->Quit";
sd[205].sdata = "Select a problem type to get troubleshooting steps";
sd[210].sdata = "Press ANY key to Return to MENU";
sd[211].sdata = "Press ANY key to continue";
sd[230].sdata = "S H U T T I N G   D O W N";
sd[231].sdata = "   P L E A S E   W A I T   ";
```