PC Care Manual

Diagnosing and Maintaining Your MS-DOS, CP/M or Macintosh System

Chris Morrison and Teresa S. Stover

PC Care Manual

Diagnosing and Maintaining Your MS-DOS, CP/M or Macintosh System

No. 2991
$24.95

PC Care Manual

Diagnosing and Maintaining Your MS-DOS, CP/M or Macintosh System

Chris Morrison and Teresa S. Stover

TAB BOOKS Inc.
Blue Ridge Summit, PA 17294

FIRST EDITION
FIRST PRINTING

Library of Congress Cataloging in Publication Data

Morrison, Chris.
PC care manual : diagnosing and maintaining your MS-DOS, DP/M, or Macintosh system / Chris Morrison and Teresa S. Stover.
p. cm.
Includes index.
ISBN 0-8306-0991-1 ISBN 0-8306-2991-2 (pbk.)
1. Microcomputers—Maintenance and repair. I. Stover, Teresa S.
II. Title.
TK7887.M66 1987 87-26235
621.391'6—dc19 CIP

Questions regarding the content of this book
should be addressed to:

Reader Inquiry Branch
Editorial Department
TAB BOOKS Inc.
Blue Ridge Summit, PA 17294

Contents

Acknowledgments

We would like to thank the following individuals for taking the time to review the manuscript and for providing invaluable input toward making this book more readable, consistent, and clear:

Diana Price
Pat Mahony
Kevin Wheeler
Craig Stover
Jessie Spencer-Cooke
Rick Williams

We would also like to express our appreciation to Ilva Klar for creating several of the line drawings, and to Craig Stover for contributing his photographic skills to this work.

Thanks to you all for your enthusiasm and support of this project, which certainly helped carry us through.

Introduction

Computers are remarkable machines; perhaps one of the most revolutionary inventions in our history. The presence of the computer dominates our era and our lives in many forms. Particular prominence belongs to the microcomputer: what is known as the "personal computer." You might use a personal computer at work, for your business, for school, or as a hobby.

Although computers are indeed amazing, they are still mere machines, and therefore are not infallible. Personal computers can malfunction and break down, just like any other machine or appliance you use. Computers permeate all levels and aspects of our society, so computer breakdowns usually create some degree of inconvenience, if not chaos.

You have probably been using a personal computer for a year or so, and although you do not know a great deal about how the computer functions, or what all the hardware does, you do know how to use your application software to get your work done.

But your computer is beginning to show signs of age and wear. Perhaps you've already experienced trips to a computer service center, with their attendant repair bills. Although you are not a computer technician, and you do not know dc and ac electronics, you want more control over the health of your computer, saving yourself time and money.

This book will help you attain that goal. You need no previous experience with computer programming or repair for this book to be useful to you. All you need is a conscientious desire to take better care of your personal computer and to diagnose problems as they emerge.

This book provides you with three basic tools for taking care of your personal computer: preventive maintenance, diagnostics when something goes wrong, and general repair once you know what the problem is.

The pleasant results of using these tools are that you will:

- Have a better understanding of the internal workings of your system.
- Have a healthier system less prone to extended periods of "downtime."
- Gain a new understanding of BASIC programming.
- Save time from not having to deal with a broken computer.
- Save money from fewer trips to computer service.
- Generally gain more control from knowledge about your computer.

PREVENTIVE MAINTENANCE

To avoid breakdowns and the troubles associated with them, you should take certain preventive measures to keep your system healthy and running at its best.

The first chapter outlines general, everyday practices that will help keep your computer healthy. It also explains periodic preventive maintenance procedures to follow that will stave off system problems. Troubleshooting techniques are explained to help you start ferreting out problems.

Each succeeding chapter specializes in a particular subsystem of the typical personal computer system: the keyboard, monitor, printer, disk drive, and serial communication interface. Within these chapters, there are diagnostic program modules, troubleshooting procedures, and general repair guidelines.

DIAGNOSTICS

To aid you in the often difficult task of troubleshooting, the *System Diagnostic Program* is available. This BASIC program consists of five modules, each designed to perform a series of tests on a different component. You have the choice of buying the program on a diskette in either MS-DOS, Macintosh, or CP/M formats. Alternatively, you can type the program in yourself, because complete program listings are provided in the appendices.

The System Diagnostic Program enables you to gather information to determine if the subsystem being tested has a problem, and to pinpoint what that problem is. Each chapter has a section that describes the conditions that warrant running a particular test. Instructions for running the module are provided, along with illustrations of expected results. After running these tests, you'll be able to make a diagnosis and take action.

Program listings and detailed, line-by-line explanation of program *code* are also included in the appropriate chapters. This is particularly useful if you know how to program in BASIC, and would like to understand how the program works and what the logic is. This can also help you modify the program for your own specific purposes, or in creating new modules for additional subsystems and peripherals. However, if you are not familiar with BASIC, you can easily skip these program listings and explanations without missing out on anything about using the programs for diagnosis.

The System Diagnostic Program is presented in MS-DOS Microsoft BASIC. Most examples refer to the IBM PC/AT/XT series, as well as the IBM "clones" such as the Compaq, Leading Edge, Epson, and Tandy. However, the Macintosh and CP/M systems are discussed as well. Where the systems differ, either in the code or descriptions, this is pointed out.

If your personal computer is not explicitly covered in this book, you can still use these diagnostics if you have Microsoft BASIC and are conversant enough in BASIC to perform the minor code changes necessary to adapt these programs to your computer's format. Refer to Appendix D for more information about adapting the program to your computer.

It is not necessary to know how to read and write BASIC programs in order to use the diagnostic routines in this book. All the programs are ready for you to use. All you need to do is create a backup of the program and then load it into your system.

REPAIR GUIDELINES

Once you've run the computer through some diagnostic tests and you have an idea of what the problem is, you can proceed with a "treatment plan" by fixing it yourself, using the Troubleshooting and Repair Guidelines provided in each chapter for each subsystem. If it's a complex problem, you can take the system to a computer service center with a more educated, in-depth explanation, and probably a diagnosis, of what must be done. This saves the technician time, thereby saving you money.

EXERCISER

Once you've taken your system in and had it fixed, you want to make sure it really is working correctly. Your personal computer is a system, and, as such, when something goes wrong in one part of the computer, it could actually be caused by something in another part of the computer. The Exerciser Module of the System Diagnostic Program lets you make sure that the system as a whole is working right.

The diagnostic routines for each subsystem are presented separately in each chapter, but each routine is actually one module of the larger System Diagnostic Program. When you run this entire diagnostic program after having a problem repaired, all system functions are exercised and run for a period of time. This is much like test-driving your car after repair. The program "burns-in" and tests the system to confirm that it works correctly and that there are no recurring problems.

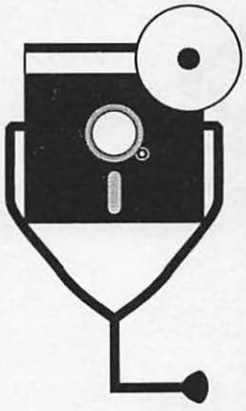
CREATING YOUR OWN DIAGNOSTICS

If you know BASIC programming, this book provides you with the techniques for developing new diagnostic routines for peripherals not covered in this book. Suppose you have another peripheral, such as a mouse or a video camera, for which you want to develop a diagnostic program. The information in this book

introduces a systematic diagnostic approach that will help you predict the type of problems that occur with various kinds of equipment.

This systematic approach also leads you to develop the questions to gather the needed information. Once you find out the kinds of problems that can occur and the information you need to diagnose these problems, you can write a program or add a new routine appropriate to such a troubleshooting activity. Chapter 8 discusses this in detail.

In every case, we have done our best to ensure the accuracy and reliability of the information presented in this book. Keep in mind that we are using the IBM PC as our standard baseline system example, and other computers might behave a little differently. The similarities should be sufficient, however, to give you the instructions necessary to help you pinpoint problems when they arise.



Chapter 1
Getting Started

This chapter is designed to help you better understand your computer, to use the System Diagnostic Program to troubleshoot problems, and to help get these problems fixed. First, general aspects of the System Diagnostic Program are discussed, and each of the individual component modules are summarized.

This book not only provides the tools with which you can troubleshoot your computer's problems, but also outlines general repair guidelines once you do pinpoint the problem. To give you a good foundation for repairing your computer, this chapter describes the necessary tools and techniques. General principles of computer system preventive maintenance, troubleshooting, and repair are covered.

SYSTEM OVERVIEW

A typical personal computer system consists of the following components:

- keyboard
- monitor
- system unit
- printer
- disk drive(s)
- serial communication interface

You might have more peripherals on your particular system, but this is a basic configuration, resembling the system in Fig. 1-1.

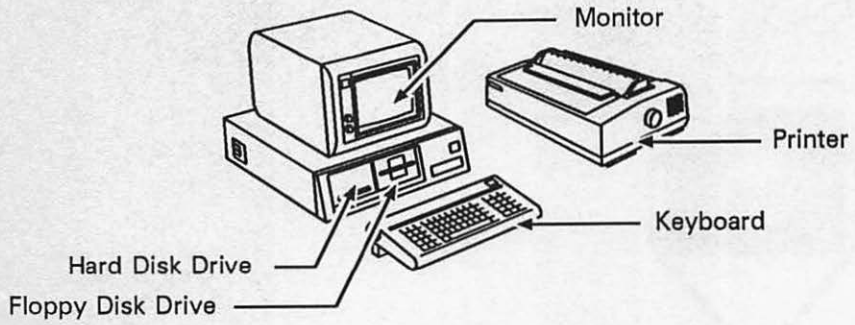


Fig. 1-1. Personal computer system.

Although the components are covered in more detail in their individual chapters, a brief description of each is provided here. This general overview describes how the various components interact to perform the work expected of a computer system.

The Keyboard

The keyboard is the computer's primary input device. You type in characters which can form commands, move the cursor, and enter data. The system unit receives the code of the characters you've pressed, sends them through the system unit for processing and interpretation, and displays the appropriate results on the monitor screen. The keyboard, as input from the outside world, is your principal means for getting your computer to do what is needed.

The Monitor

The monitor is the visual display for your computer work. It is also known as a *cathode ray tube* (CRT) or video display. The monitor is the output device that displays your input data from the keyboard, mouse, modem, or other input device. It also shows the result of system processing performed within the system unit. A standard monitor is the IBM monochrome monitor, which displays the familiar green characters on a black background. Color monitors and high-resolution monitors are also available.

The System Unit

The system unit is the computer "brains." This is the single most important component of the personal computer. The system unit is the main housing of the computer that holds the *central processing unit* (CPU), as well as the various controller boards and interfaces for the computer's peripherals. The CPU is the microprocessor chip that runs the computer. It issues commands that cause calculations to be made and processes to be run, coordinates the various input and output devices, and transfers and processes data to and from the outside world.

The system unit for the IBM PC includes an Intel 8088 CPU chip, 256 kilobytes of memory, a disk controller, and both parallel and serial interfaces.

The Apple Macintosh system unit consists of a Motorola 68000 CPU chip, 128 kilobytes to one megabyte of memory, an internal disk drive, a nine-inch high-resolution monitor, a parallel printer port, and a serial communication port.

A typical CP/M system unit includes a Zilog Z80 CPU, 64 kilobytes of memory, a parallel printer port, and a serial communication port.

The following chapters cover all standard computer components except for the system unit. This is because the system unit has a built-in set of diagnostic tests for checking memory and making sure that all the necessary components are properly connected. This built-in system unit diagnostic program is run automatically whenever you boot up the system. If any of the tests fail, an error message is displayed on the screen. This error message indicates what your next step should be, whether to tighten up a peripheral connection, or take the system unit in for servicing.

The Printer

Like the monitor, the printer is an output device. However, while the monitor provides output on the screen, the printer provides output on paper, or *hardcopy*. A parallel or serial connection between the system unit and the printer causes data and processing results to be transferred from the system to be typed on the printer. You might have a dot-matrix, letter-quality, or laser printer.

The Disk Drive

The disk drive, along with a disk of some kind, is used to store and recall software programs and data. Your system might use one disk drive or two disk drives. It might use a floppy disk drive using floppy diskettes as the storage media, or a hard disk drive using a hard metal platter.

The Serial Communication Interface

The serial communication interface is the means for your computer to "talk" to the outside world. It can connect to, or *interface* with, a modem, which sends and receives data to and from another computer over telephone lines. It can interface with a printer, transmitting data to be printed. It can also interface directly with another computer.

THE SYSTEM DIAGNOSTIC PROGRAM

The System Diagnostic Program is divided into several modules. Each module includes the necessary program instructions that can exercise a particular device on your computer and test for possible problems. The devices that can be tested with this program are those of a typical configuration: keyboard, monitor, disk drive, printer, and serial communication interface.

System Diagnostic Program Flow and Module Summary

The block diagram in Fig. 1-2 outlines the structure of the System Diagnostic Program. As you can see, lines 10-30 are set aside for the explanation and storage of any constants and variables that may be used within the program. The numbers within each of the blocks are the line numbers where the code for that subsystem module is located.

The first module code is the *menu driver,* or the System Diagnostic Main Menu. The program code for the main menu, residing in lines 40-210, is described later in this chapter. The code for the other modules is listed and explained in the appropriate subsystem chapter. For example, the code for the Monitor Diagnostic Module is explained in Chapter 3.

The second module in the program is the Keyboard Diagnostic Module. This module runs a test that echoes and gives the numeric code, or *ASCII value,* of any key you press. If you press a function key, the function key name and/or combination is displayed on the screen. The Keyboard Diagnostic Module resides in lines 220-770.

Fig. 1-2. Diagnostic program block diagram.

The third module is the Monitor Diagnostic Module. This module consists of the following six tests, residing in lines 790-1350:

- The Display Test fills the entire screen with a specified character. This lets you check that each location on the monitor is capable of displaying data.
- The Alignment Test displays a series of vertical and horizontal lines to let you check that the monitor is properly aligned.
- The Character Set Test displays your monitor's full set of available display and graphic characters.
- The Intensity Test lets you view the various intensity modes that your monitor has available, so that you can check and adjust the brightness and contrast of the display.
- The Scroll Test provides a sliding character set to let you check that the monitor is scrolling correctly.
- The Status Line Test determines whether or not you have a 25th status line display available.

The Printer Diagnostic Module is similar in many ways to the Monitor Diagnostic Module, and is stored at lines 1360-2040. It contains the following:

- The Printer Setup routine lets you customize the diagnostic to work specifically for your particular printer.
- The Sliding Alpha Test lets you check each letter of the printer's available character set.
- The Display Character Print Test displays the primary character set that the printer recognizes.
- The Echo Character Print Test lets you check whether or not a particular character can be printed.
- The Horizontal Tab Test checks all horizontal tab positions.
- The Line Feed Test checks various line space gradations, including feeding a new page of a continuous form.

The Disk Drive Diagnostic Module tests and exercises the two main functions of a disk drive: reading and writing to the diskette. This test supports floppy disk drives as well as hard disk drives. Lines 2050-2390 contain this disk drive test.

Lines 2400-2580 contain the Serial Communication Diagnostic Module, which lets you test the serial communication interface, including a modem interface. For this test, you press a key on the keyboard, and the character is echoed on the screen. This "loopback" test determines whether or not the modem interface is functioning.

The final program module is the System Exerciser Module at lines 2590 to 2920. If you purchase a new computer, or if you've just brought your computer back from servicing, it's a good idea to "burn in" the system. Burn-in is a continuous testing mode that exercises all system functions. It is particularly useful after the computer is moved, or when it is initially set up, because that is when

most computer problems occur. These types of problems are generally referred to as *infant mortality*. This module allows you to burn in and check virtually all system functions (the keyboard is not tested) automatically and unattended.

RUNNING THE SYSTEM DIAGNOSTIC PROGRAM

If you have purchased the diskette containing the System Diagnostic Program, (ordering information available in the back of this book), the first thing you need to do, as with any new program diskette, is make a backup copy and keep the original in a safe place.

If you have not purchased the diskette, access the Microsoft BASIC on your system (BASICA for the IBM PC, GWBASIC for a PC clone, Microsoft BASIC for the Macintosh, and MBASIC for a CP/M systems), and key in the appropriate version of the code as listed in the appendices. Appendix A provides the MS-DOS program listing, Appendix B provides the Macintosh program listing, and Appendix C provides the CP/M system program listing.

Try out each program function to make sure that there are no syntax errors, which would indicate typographical errors made while entering the program.

After doing all the work of typing the program in, be sure to save the program to disk and make a backup copy of it. It's imperative that you key the program in **before** you have a system problem. If you wait until you suspect a malfunction, you might not be able to enter the program because of this malfunction, or your test results might be inaccurate and misleading.

Running the Diagnostic on an MS-DOS System

If you have an MS-DOS system with two floppy disk drives, and you wish to run the System Diagnostic Program, follow these steps:

1. Boot up the MS-DOS operating system.
2. Insert the DOS diskette containing BASICA or GWBASIC into Drive A.
3. At the A> prompt, enter BASICA or GWBASIC. BASIC is loaded into the system, and the OK prompt is displayed.
4. Place the diagnostic diskette into Drive B.
5. Enter RUN "B:DIAG".

If you have an MS-DOS system with one floppy disk drive, do the following:

1. Place the DOS/BASIC diskette into Drive A and enter BASICA OR GWBASIC as applicable. You will receive the OK prompt from BASIC.
2. Remove the DOS/BASIC diskette from the disk drive.
3. Insert the diskette containing the System Diagnostic Program.
4. Enter RUN "DIAG" to activate the diagnostic program.

Running the Diagnostic on a Macintosh System

If you have a Macintosh system with two disk drives, follow these steps to run the System Diagnostic Program:

1. Place the Microsoft BASIC program diskette into the disk drive. The BASIC icon is displayed.

2. Double-click on the BASIC program icon to open BASIC.
3. Insert the diskette containing the System Diagnostic Program into the second disk drive. The System Diagnostic Program icon is displayed.
4. Double-click on the disk icon to open the directory. Select and double-click on the Diag program icon to open it up.
5. Open the Command window and type RUN. The System Diagnostic Main Menu is displayed.

If your Macintosh is a single disk drive system, follow these steps:

1. Place the Microsoft BASIC program diskette into the disk drive. The BASIC icon is displayed.
2. Double-click on the BASIC program icon to open BASIC.
3. Eject the BASIC diskette and insert the System Diagnostic Program diskette. Its icon is displayed.
4. Double-click on the System Diagnostic Program icon to open its directory.
5. Open the Command window and type RUN. The System Diagnostic Main Menu is displayed.

Running the Diagnostic on a CP/M System.

If you have a CP/M system, follow the same steps as for an MS-DOS system. Once you see the A> prompt displayed, enter MBASIC. BASIC is loaded into the system, and the OK prompt is displayed. Then enter RUN "B:DIAG:"

THE MAIN MENU

Once the program is loaded, the screen displays the System Diagnostic Main Menu. This menu gives you several options as to which diagnostic module to run, as shown in Fig. 1-3.

A selection is made by pressing the one-character command which is the first letter of the name of the desired test. For example, if you wish to run the keyboard test, enter K or k. The program is not character case-sensitive.

When you press a valid selection, the screen clears and either presents a new menu or begins the desired test immediately. If you enter an invalid command, such as Z or R, the system beeps and returns to the selection mode to let you try again.

By pressing the Q, the program *quits,* or ceases operation, and returns you to the operating system, so that you can run a different program.

How the Main Menu Module Works

Now that you see the general structure of the main menu module, you can start to see how the program works, line by line (Fig. 1-4). Note that lines 10 through 30 shown here are not used in the Macintosh or CP/M versions.

Line 10 of Fig. 1-4 turns off the function key display at the bottom of the screen. The FOR-TO statement then loads nulls for the values of the function keys. This

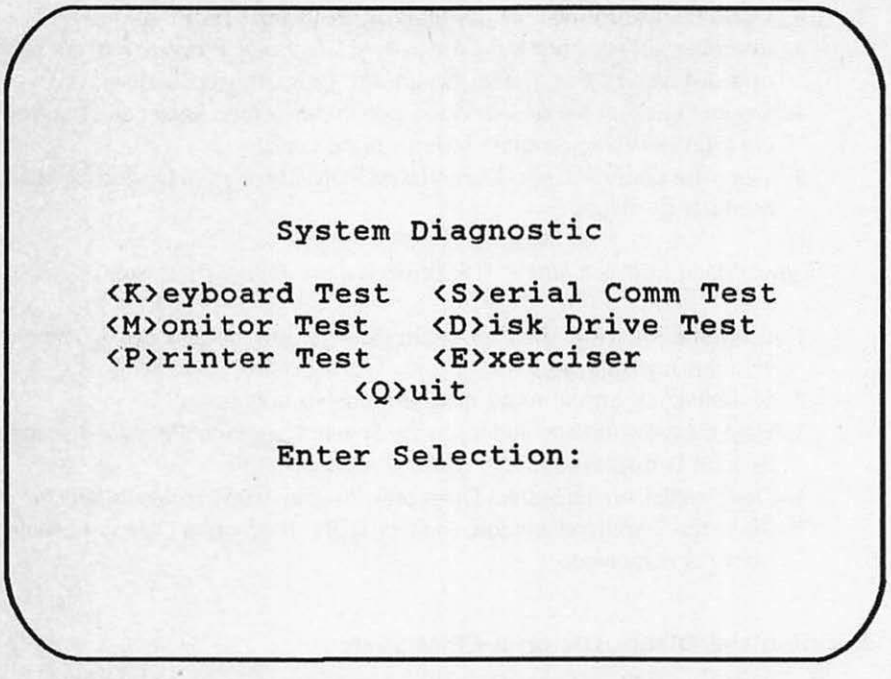


Fig. 1-3. Diagnostic program main menu.

only applies to MS-DOS systems. It essentially frees up the 25th status line and disables the function keys so that any programmed values are disabled:

```
10  KEY OFF:FOR I=1 TO 10:KEY I,"":NEXT I:REM System Diagnostic
Program
```

Line 20 is a DATA statement, and as you can see, it has the layout of the "QWERTY" keyboard. This is used in conjunction with the DIM (dimension) statements of A1$, A2$, A3$, and A4$. These will be used for determining that a CTRL (control key) sequence, as opposed to a normal shift or unshifted sequence, has been pressed:

```
20  DATA "q","w","e","r","t","y","u","i","o","p"
    :DATA "a","s","d","f","g","h","j"," k","l"
    :DATA "z","x","c","v","b","n","m"
    :DATA "1","2","3","4","5","6","7","8","9","0","-","=",
    :DIM A1$(10):DIM A2$(9):DIM A3$(7):DIM A4$(12)
```

Line 30 then loads set values from the DATA statements into their appropriate variable arrays, A1$ through A4$:

```
10 KEY OFF:FOR I=1 TO 10:KEY I,"":NEXT I:REM System
   Diagnostic Program
20 DATA "q","w","e","r","t","y","u","i","o","p"
   :DATA "a","s","d","f","g","h","j","k","l"
   :DATA "z","x","c","v","b","n","m"
   :DATA "1","2","3","4","5","6","7","8","9","0","-","=",
   :DIM A1$(10):DIM A2$(9):DIM A3$(7):DIM A4$(12)
30 FOR I=1 TO 10 :READ A1$(I):NEXT I:FOR I=1 TO 9:READ
   A2$(I):NEXT I:FOR I=1 TO 7:READ A3$(I):NEXT I:FOR I=1 TO
   12:READ A4$(I):NEXT I
40 REM: System Diagnostic Main Menu Module
50 CLS:PRINT TAB(12): "System Diagnostic":PRINT:PRINT TAB(3)
   "<K>eyboard Test  <S>erial Comm Test":PRINT TAB(3)
   "<M>onitor Test   <D>isk Drive Test":PRINT TAB(3)
   "<P>rinter Test   <E>xerciser":PRINT TAB(16) "<Q>uit":
   PRINT:PRINT TAB(12) "Enter Selection:"
60 AN$="":AN$=INKEY$:IF AN$="" THEN GOTO 60
70 IF AN$="K" THEN 220
80 IF AN$=k" THEN 220
90 IF AN$="M" THEN 790
100 IF AN$="m" THEN 790
110 IF AN$="P" THEN 1360
120 IF AN$="p" THEN 1360
130 IF AN$="D" THEN 2050
140 IF AN$="d" THEN 2050
150 IF AN$=S" THEN 2400
160 IF AN$=s" THEN 2400
170 IF AN$="E" THEN 2600
180 IF AN$="e" THEN 2600
190 IF AN$="Q" THEN END
200 IF AN$="q" THEN END
210 BEEP:GOTO 60
```

Fig. 1-4. Main menu module listing.

```
30 FOR I=1 TO 10 :READ A1$(I):NEXT I:FOR I=1 TO 9:READ
   A2$(I):NEXT I:FOR I=1 TO 7:READ A3$(I):NEXT I:FOR I=1
   TO 12:READ A4$(I):NEXT I
```

CP/M AND MAC USERS. Lines 10 through 30 as shown here are not used in the Macintosh and CP/M versions of this program.

Line 40 is a REMARK statement stating that this is the Main Menu Module:

```
40 REM:  System Diagnostic Main Menu Module
```

Line 50 clears the screen with the CLS command and then displays all the headings for the System Diagnostic Main Menu:

```
50 CLS:PRINT TAB(12): "System Diagnostic":PRINT:PRINT TAB(3)
   "<K>eyboard Test <S>erial Comm Test":PRINT TAB(3)
   "<M>onitor Test <D>isk Drive Test":PRINT TAB(3)
   "<P>rinter Test <E>xerciser":PRINT TAB(16) "<Q>uit":
   PRINT:PRINT TAB(12) "Enter Selection:"
```

Line 60 then searches for a key by calling up the keyboard input subroutine that prompts and waits for a key to be pressed. If no key is pressed, the variable A$ checks to see if it's null, and it loops in on itself. If a key is pressed, the program returns to line 70:

```
60 GOSUB 2930
```

Lines 70 through 200 check to see if you've pressed a valid key sequence. If K or k is pressed, the program branches to line 220 for the Keyboard Diagnostic Module. If M or m is pressed, the program branches to line 790 for the Monitor Diagnostic Module. If P or p is pressed, the program branches to line 1360 for the Printer Diagnostic Module. If D or d is pressed, the program branches to line 2050 for the Disk Drive Diagnostic Module. If S or s is pressed, the program branches to line 2400 for the Serial Communication Diagnostic Module. If E or e is pressed, the program branches to line 2600 for the Exerciser Module. If Q or q is pressed, the program ends and exits:

```
 70 IF A$="K" THEN 220
 80 IF A$="k" THEN 220
 90 IF A$="M" THEN 790
100 IF A$="m" THEN 790
110 IF A$="P" THEN 1360
120 IF A$="p" THEN 1360
130 IF A$="D" THEN 2050
140 IF A$="d" THEN 2050
150 IF A$=S" THEN 2400
160 IF A$="s" THEN 2400
170 IF A$="E" THEN 2600
180 IF A$="e" THEN 2600
190 IF A$="Q" THEN END
200 IF A$="q" THEN END
```

If you wish to add a new peripheral, you can very easily add the command check for it within this section.

If an invalid key is pressed, the program proceeds to line 99, which causes a beep to sound, and then loops back to the section accepting keystrokes and you can try again:

```
210 BEEP:GOTO 25
```

Program Modification and Adaptation

As you have seen, the System Diagnostic Program is written in Microsoft BASIC, and is structured to accommodate modification and additions.

BASIC is a universal programming language available on nearly every microcomputer available. This means you could use this diagnostic program and book for any microcomputer if you know the variations between Microsoft BASIC for the IBM PC and the BASIC used on your computer. Most differences involve keyboard and screen manipulation, referring to the manner in which a pressed key is read internally in the system, and how the character is displayed on the screen.

The examples used throughout this book are based on the IBM PC system, with the diagnostic module listings shown in the PC BASIC version. However, as mentioned earlier, the appendices provide the complete program listings for the Macintosh and CP/M systems in addition to the MS-DOS system code listing.

Appendix D provides general rules that are used to convert the diagnostic program from the MS-DOS format to either the CP/M or Macintosh format. It shows the areas that must be coded in different ways.

You are not restricted to simply testing the common components of a typical personal computer system. If you purchase a new peripheral device for your system, such as a mouse, a different type of monitor, or an optical disk, you can write a new diagnostic module for that device and add it to this program. Chapter 8 covers techniques, information, and references for writing diagnostics for new devices.

TOOLS AND RESOURCES

Troubleshooting and repairing computers, or for that matter any electronic system, might seem like a difficult and dangerous task. However, most computer technicians go about their work systematically, trying out various aspects of the system to gather information about what the problem might be. They use this information to make logical deductions about what is malfunctioning. Then, they repair the problem by making mechanical adjustments and component replacements.

To perform such system troubleshooting and repair, computer technicians have specific knowledge and training about the particular systems they service. They have complete sets of pertinent documentation and schematics, use a good set of tools, have spare parts available to them, and have various information resources to draw upon.

Although you cannot expect to do as thorough a job as a computer technician without the same investment in training, tools, and parts, you can take some cues from them by learning some of their basic principles and techniques. Then, when your computer malfunctions or breaks down, you'll be able to analyze the problem and get it repaired with a minimum of effort and expense.

Documentation

Save all manuals, user guides, specifications, installation instructions, and any other technical documentation that comes with your system, peripherals, and

upgrades. This book provides general information, but your documentation will provide the necessary specifics regarding your particular computer. This is vital when installing and setting up a new device, or when something goes wrong.

If you have unwittingly lost or thrown out important documentation, you can probably obtain a copy from your local computer store, or send for one directly from the manufacturer. You can also check with your library and your user's group for documentation to borrow or buy.

Tools

Tools let you open, bend, cut, insert, and connect things in your computer. The following tables list required tools, optional tools, and cleaning supplies. The tools make the job of computer troubleshooting and repair possible. The cleaning supplies are vital, because cleaning is a major aspect of maintenance.

When you buy your tools, try to buy good-quality ones. This does not necessarily mean you should buy the most expensive tools on the market, but you do want tools that will not bend or break at a critical moment. In other words, buy dependable tools that will last a long time.

Table 1-1 lists required tools, and Fig. 1-5 illustrates them. Table 1-2 lists optional tools, and Fig. 1-6 provides their illustration. Table 1-3 lists cleaning supplies, and Fig. 1-7 illustrates them. The tables list the functions, uses and average costs for the tools and cleaning supplies.

Spares

Professional technicians usually have an inventory of spare parts associated with the equipment they work on. The spares help them in troubleshooting, when they replace parts to find out whether or not it is causing the problem. If the replacement solves the problem, then they know that the old part was bad.

In your case, however, maintaining your own inventory of spare parts for your personal computer can be very costly and inefficient. An alternative to this is to find someone (a friend, a colleague at work, an acquaintance in a user's group) who has a system similar to yours. Work out a deal with this person in which you can temporarily swap parts with each other when one of your systems is malfunctioning. You can use this person's system for troubleshooting by temporarily swapping suspect parts to find exactly where the problem is.

> **NOTE:** Swapping parts from someone else's system should be used as a last resort. Make sure both parties understand that there is a risk associated with swapping. There is always the possibility that the particular problem in your system could damage or destroy the swapped part.

Find out where you can get the best prices for computer parts. There might be a good discount computer store in your area that will have computer components cheaper than a computer retailer. The want ads in the back pages of most computer magazines are good sources for used parts, as are flea markets and computer swap meets.

Table 1-1. Required Tools, Use, and Cost.

Required Tools	*Use*	*Average Cost*
1/4" flat-blade screwdriver	to tighten and untighten 1/4" slotted screws for opening the system unit and keyboard covers	$3.50
1/8" flat-blade screwdriver	to tighten and untighten 1/8" slotted screws on interface cables	$2.00
small Phillips screwdriver	to tighten and untighten small cross-slotted screws used for holding bolts down	$3.00
needle nose pliers	to grip and/or bend small objects as an extension or a "helping hand"	$5-12.00
pliers	to grip and/or bend small objects, to hold larger nuts	$5.00
wire strippers/cutters	to cut wire, to strip away plastic insulation	$5-12.00
non-conductive screwdriver	to tighten and untighten screws in an electrically charged environment such as the monitor and power supply	$2.00

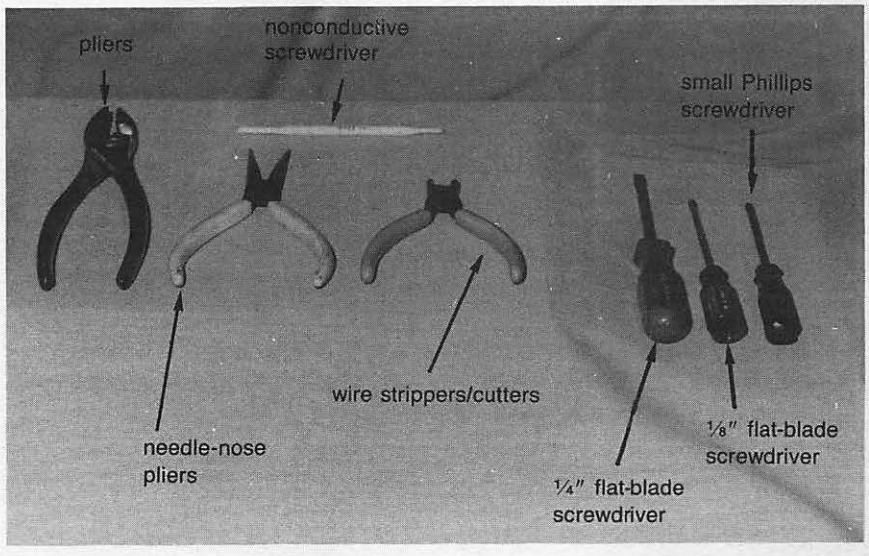


Fig. 1-5. Required tools.

Table 1-2. Optional Tools, Use, and Cost.

Optional Tools	*Use*	*Average Cost*
socket wrench set	to fit over hex head screws of various sizes to tighten or untighten	$15-25.00
small metal flat file	to smooth or grind down a plastic surface such as a keycap	$3.50
digital voltmeter or multimeter	to measure volts, ohms, and amps in a power supply or interface board circuit	$50-200
fine-tip pencil-type soldering iron	to melt and apply solder in order to join or patch metal surfaces such as cable wires, IC leads, components	$20-25
fine robin core solder	the metal alloy used in soldering for joining metal surfaces	$4-5.00
desoldering tool	to remove solder from metal surfaces to disengage them	$5-15.00
solder tip sponge	to clean up excess solder	$1.50

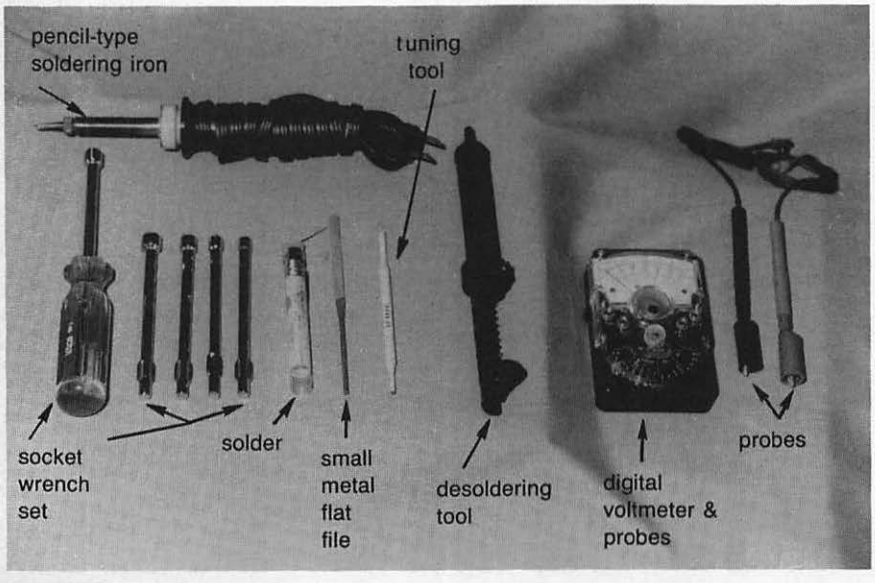


Fig. 1-6. Optional tools.

Table 1-3. Cleaning Supplies, Use, and Cost.

Cleaning Supplies	Use	Average Cost
compressed air with nozzle and extension	pressurized air used to blow out dust and dirt from the keyboard and printer	$5.00
6-oz can freon or contact cleaner	solution used to clean contacts and switches on PC board components	$5.00
16-oz bottle isopropyl alcohol	for general cleaning of electronic hardware floppy disk drive heads, and print wheels	$1.00
cotton swabs	for cleaning of hardware and components in tight places, such as disk drive heads	$2.00
sponges	for cleaning hardware components, disk drive heads, and print heads with isopropyl alcohol	$1.00

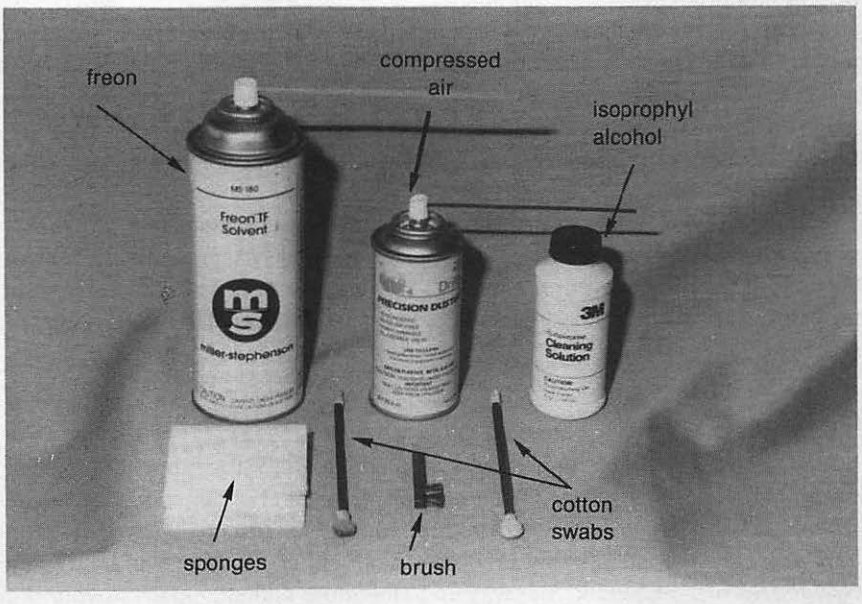


Fig. 1-7. Cleaning supplies.

User Networking

Computer resources become more and more important, the more you "get into" your computer. It becomes less important to know everything if you have trusted resources that can provide you with good information.

If, after doing a bit of troubleshooting, you still can't find the problem, call your local computer dealer. Most stores will give you valuable information if you can ask specific questions.

That friend of yours with the similar computer system might know things you don't.

User's groups are excellent repositories of information and experience. You can find a local user's group through your computer dealer, in your newspaper's computer page or social events page, or from listings in computer magazines. The people you'll meet at a user's group can help you answer questions, solve problems, and find other resources.

Read the computer magazines pertinent to your system. They often provide good advice as well.

General Troubleshooting and Repair Guidelines

This book is broken down in chapters according to major computer subsystems. If you're having trouble with your keyboard, for example, turn to Chapter 2. This chapter explains possible keyboard problems, and how to troubleshoot and repair them. Part of this troubleshooting process includes running one of the diagnostic tests. The chapter goes on to tell you how to run the diagnostic module and what results to expect. With the diagnostic modules, troubleshooting is easy and automatic.

Once you run the diagnostic test and discover what the problem is, the general repair guidelines offer suggestions on how to have the problem fixed. Many of these suggestions have you doing the repairs. Even if you have never been mechanically or technically "inclined," you should have little problem troubleshooting your computer, and perhaps even repairing some of the diagnosed problems.

If you run one of the diagnostic tests and the results indicate a problem, it is a good practice to test the diagnostic on an identical system, particularly if you have modified the program in any way. You want to ensure that the diagnostic program is providing accurate results before spending any time and money on repair.

Limits to Tinkering

Never try to fix something for which you do not have the proper knowledge, documentation, tools, or parts. A little knowledge can be dangerous, especially when you know just enough to get yourself in trouble. You might make the problem worse. Never attempt any procedure you feel you cannot handle. Do the easy repairs first, and work your way up to the more difficult repairs as your skill and confidence grows. When in doubt, let the professionals do it. Don't put your computer in jeopardy.

Also, remember that a good technician does not tinker needlessly:

If it's not broken, don't fix it!

GENERAL PREVENTIVE MAINTENANCE AND TROUBLESHOOTING TECHNIQUES

A good computer technician performs general care and preventive maintenance on computers to minimize "downtime." Good working habits and preventive maintenance are the keys to system and component longevity. Many system problems can be avoided if a few good habits are established.

General Care

Everyday working habits are instrumental to the health of your computer. Such measures as using a dust cover, an anti-static mat, and a surge protector can go a long way in preventing system problems.

Dust Cover. Buy a dust cover to protect the system unit, monitor, keyboard, disk drive, and printer. Dirt and moisture are your system's worst enemies. Dust settles and collects on the *contacts*—the exposed wires, or "legs", of the integrated circuit (IC) chips on the system's printed circuit boards. These contacts have electric current flowing through them, and they become ionized and actually attract particles of dust. The dust in turn attracts moisture out of the air. As this cycle continues, the dust and moisture eventually cause electrical shorts and corrosion of the electronic parts, which can then cause system malfunctions.

Spending $10 or $20 for a dust cover is preferable to buying a brand new printed circuit board to replace one that has been ruined by corrosion caused by dust. Get into the habit of covering your system when it is not in use.

Static Electricity. Static electricity is another computer foe. The IC chips throughout your system are made of a fragile material and contain minute electronic circuits that can be easily damaged. Naturally, the most valuable ones, like the CPU and the memory chips, are particularly vulnerable to static electricity. If you live in a dry, cool climate, your computer is especially susceptible.

There are several ways to protect your computer from getting "zapped." Spray the floor or carpet around the system with an anti-static solution, or lay down an anti-static mat under the computer. You can get anti-static spray as well as the mat at an office or computer supply store. The least expensive solution is to simply touch something metal that's not part of your system, perhaps your desk lamp or stapler. This grounds you and keeps you from transferring static electricity to your computer.

Power Surges. The electricity coming out of the wall sockets is supposed to be 110/120 volts at 60 Hertz (cycles per second). Unfortunately that's just an average; it's not always what you actually get. There are times when you get surges of electricity at a good deal higher than 110/120 volts, and times when you get sags in voltage a good deal lower than the average.

You've probably experienced drops in electricity when your lights dim slightly.

This "brown-out" effect might cause your keyboard to temporarily lock up, and you might lose some data.

The more serious problem is the voltage surge. A *power surge* can pass through your power supply's filter circuits by giving it more peak voltage than it can effectively suppress. Your computer's power supply takes the 110/120 volts ac and converts it down to 5, 12, or 15 volts dc. These are the voltages with which the interface boards on your system are designed to operate. When the computer power supply gets a large power surge, the power supply is not designed to handle it, and this instantaneous *spike* is sent through to the boards. This can damage or even destroy the chips (ICs) on your boards.

Boards and many chips are expensive to repair and replace, so you want to protect your power supply from these indiscriminate surges. Hence, the surge protector. A surge protector isolates the voltage spikes to provide your system no more than a constant 110/120-volt supply. You can obtain a surge protector at a computer or electronics supply store. Using a surge protector with your computer will contribute toward extending the life of its boards.

Food and Drink. Don't eat or drink close to your computer. If you spill or dribble something into your keyboard, for example, you'll have a huge cleaning job on your hands, or even the instant need for a new keyboard.

Cookie crumbs, sandwich particles, and dinner bits can all end up inside your keyboard, and maybe even other parts of the system, if you eat while working. This can destroy the keyboard contacts and render your keyboard useless.

Preventive Maintenance

You should perform the following preventive maintenance procedures for your computer at least every three months. Such procedures keep your system clean and in adjustment, thereby warding off possible problems.

Vacuum/Compressed Air. Use a hand-held vacuum cleaner or can of compressed air (like camera lens cleaner), to vacuum or blow around components such as the keyboard keys (Fig. 1-8), or the printer mechanism. This clears out any dust and dirt that gets into the workings in spite of the dust cover.

If you have a choice between the two, it's preferable to vacuum dust up rather than blow it around. Blowing dust can possibly send it deeper into the system where it might cause even more trouble.

Cleaning. Certain parts of your system can become dirty through continuous use. The floppy disk drive head, the daisy print wheel, or the dot-matrix print head can accumulate a layer of oxidation or dried ink.

There are disk drive head cleaning kits available at most computer stores, and they cost approximately six dollars. You insert a special diskette permeated with a cleaning solution, and the heads are cleaned in this way.

A gauze cloth soaked in isopropyl alcohol (not rubbing alcohol) can be used to clean daisy wheels and dot-matrix print heads. Isopropyl alcohol can also be used to clean the exterior or interior of your system where dirt and dust have accumulated. Use cotton swabs soaked in isopropyl alcohol to clean tight spots.

Loose Cables. There are several cables connected to your system unit at the rear. These cables connect interface boards to the peripheral devices. You plug

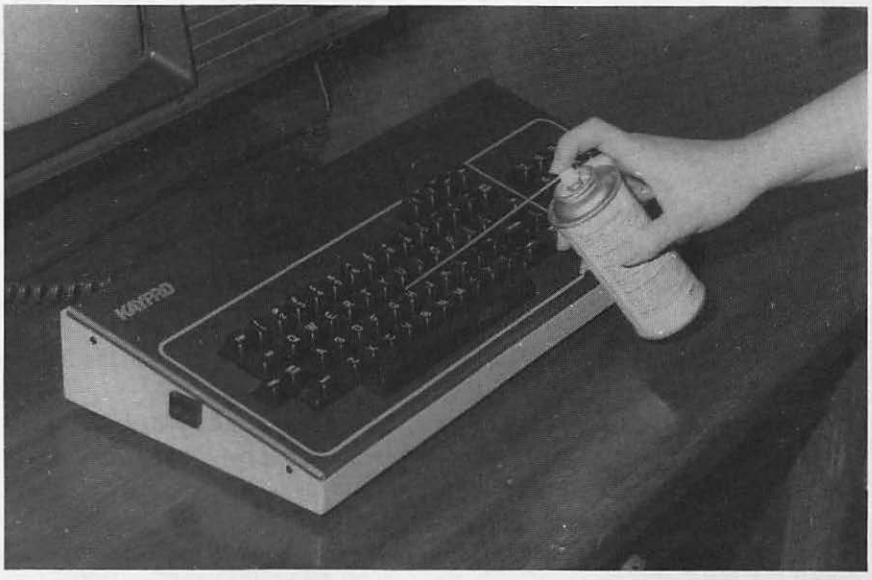


Fig. 1-8. Dusting with compressed air.

the cable connectors into the system unit's receptacles, and then tighten the screws. These cables can get worked loose when, for example, you move the printer or bump the modem. Suddenly your peripheral device stops working because you're not getting the proper connection.

The solution to this is to use your small screwdriver and secure the cables into their receptacles (Fig. 1-9).

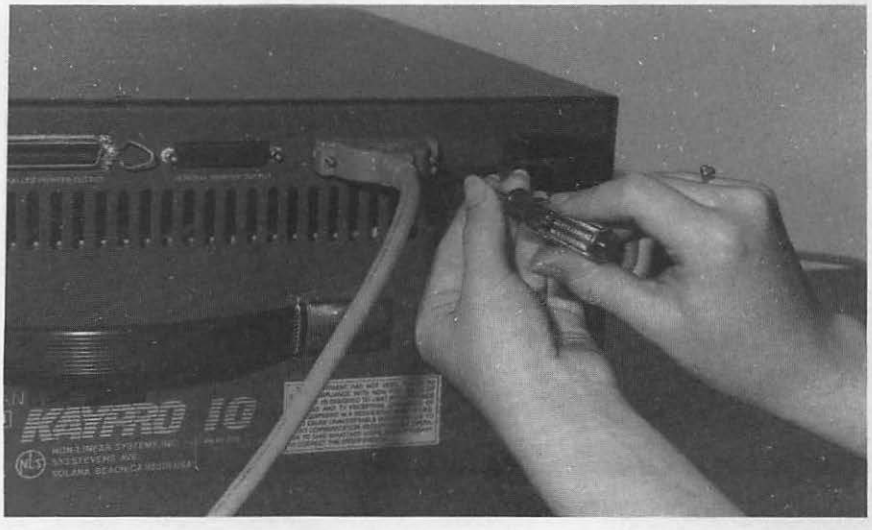


Fig. 1-9. Tightening cable screws.

Troubleshooting Techniques

"Troubleshooting" is another word for problem-solving. You have a computer malfunction, and you want to systematically go about discovering what caused it so you can effect a repair.

Start with the easy things first:

- Does it have power?
- Are all the connections made correctly?
- Is everything clean and unobstructed by dirt and dust?

Each chapter that follows lists several more specific solutions to try in the course of troubleshooting a problem.

When you've tried everything and the problem still is not fixed, the interface board for the particular subsystem is often involved. But before swapping chips, buying new boards, or packing the system up and taking it in for servicing, there are two more "home remedies" you can try.

Erasing Oxidation. Oxidation can form over a period of time on the metal edge connectors of interface boards. This manifests itself as either a dull white or black film, and it can cause a bad contact which could be the source of the system malfunction.

To eliminate oxidation, start by turning the computer system off and opening up the system unit (Fig. 1-10). Remove the cable connected to the faulty interface board.

Fig. 1-10. Opening the system unit.

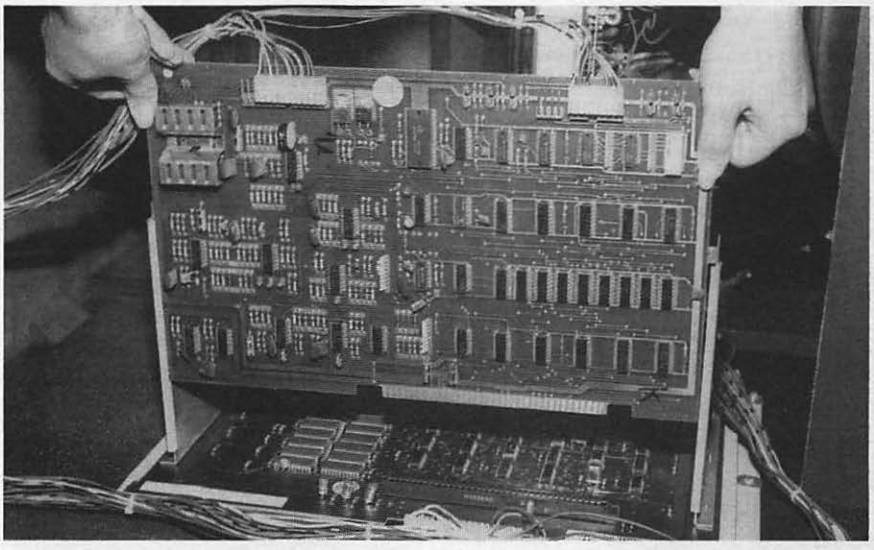


Fig. 1-11. Removing the board from its slot.

All the boards inside your system unit face the same direction. The component sides of each board face one way, while the soldered sides of each board all face the other way. Be careful when you handle these boards, because the soldered sides also have the ends of the IC pins exposed. Remove the faulty board from its slot, but handle it by its edges so you don't cut yourself (Fig. 1-11).

Now take a pencil eraser and erase both edges of the board that fit into the slot (Fig. 1-12). You'll see that the eraser removes the black or white film from the edge connectors.

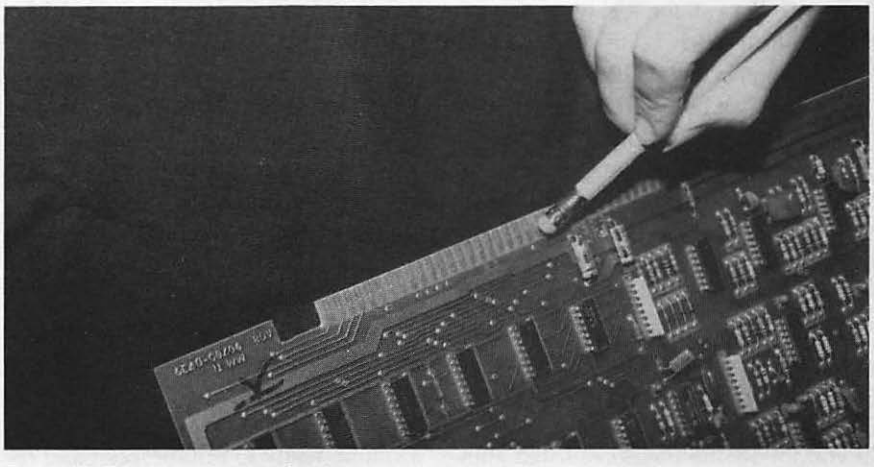


Fig. 1-12. Erasing oxidation from the edge connector.

Place the board back into its slot. When doing this, make sure it's facing the right direction, just like all the others. Line the board up to its slot, and then use your thumbs to press the board back into place.

Reconnect the cable to the board and turn the system on. Check the system to see if the malfunction still exists. If it does not, then you have solved the problem. If the problem is still there, try the following procedure.

Testing Other Slots. If you have some empty printed circuit board slots within your system unit, you can "test other slots." To begin this procedure, turn the computer system off, open the system unit, remove the cable from the faulty board, and remove the board from its slot.

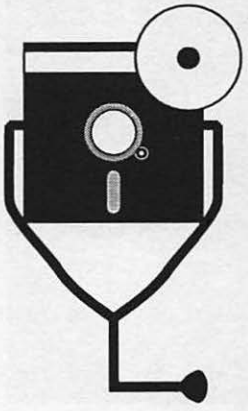
Unscrew the silver plug blocking the spare slot. Insert the board into the spare slot. Reconnect the cable, turn the system on, and check for the problem. If the problem no longer exists, it means that the slot was defective, but the interface board is working fine.

WHAT TO DO FIRST

To get yourself going on general care of your computer, preventive maintenance, troubleshooting problems, and repairing them, do the following:

- Read through this book to familiarize yourself with the concepts and organization of the material. In this way, you can get started immediately with preventive maintenance, and, when a problem arises, you'll know right where to look.
- Make a backup of the System Diagnostic Program. If you did not purchase the program diskette, type in all the diagnostic programs now, before any problems arise. Then, if something does go wrong, you'll already be prepared to troubleshoot.
- Gather at least the basic tools and cleaning supplies into a computer maintenance kit.
- Collect all the technical documentation you have and find a central place for it, such as an area at your desk or bookcase, or in a binder.
- Contact that special someone who has a similar computer system and agree to loan and borrow components when needed for troubleshooting. Start establishing your "computer network".
- Try out the diagnostics now, before you have a problem. Know how the results are supposed to appear under healthy circumstances.
- Code any necessary modifications into the System Diagnostic Program. These modifications might have to do with accommodating another peripheral you have, or a different kind of system, printer, or keyboard than what is allowed in the program. Test the modified program thoroughly, and when you're satisfied that it is working as it should be, make a backup copy.

If you follow the measures listed above, you'll be ready to troubleshoot and repair at the first sign of a suspected computer malfunction.



Chapter 2

The Keyboard

No matter how technologically advanced the computer is, it is just an idle box until it receives data and specific commands. The keyboard provides the means for data and command input, and is your main link to the computer.

DESCRIPTION AND FUNCTION OF THE KEYBOARD

The keyboard is used to enter data into your application software, whether that data includes numbers in a spreadsheet, words in a word processor, or records in a database. You also use the keyboard to give commands and make selections. The keyboard is your personal computer's primary input device.

There are two types of keyboards: *integral* and *unattached.* An integral keyboard is one that is built into the system unit. Examples of systems with integral keyboards include the Commodore 64/128, Apple II series, and the Atari ST (Fig. 2-1).

An unattached keyboard connects to the system unit with an interface cable. Examples of systems with unattached keyboards include the IBM PC series, IBM clones, and the Apple Macintosh (Fig. 2-2).

The key set found on all standard keyboards consists of the "QWERTY" typewriter keys (named for the first six alphabetic characters found on a typewriter), arrow cursor keys, and BREAK, CTRL (Control), DELETE, and ESCAPE keys. More "deluxe" keyboards also include a ten-key pad, additional cursor control keys such as HOME, an ALT function key, and anywhere from two to twenty-four special function keys (Fig. 2-3).

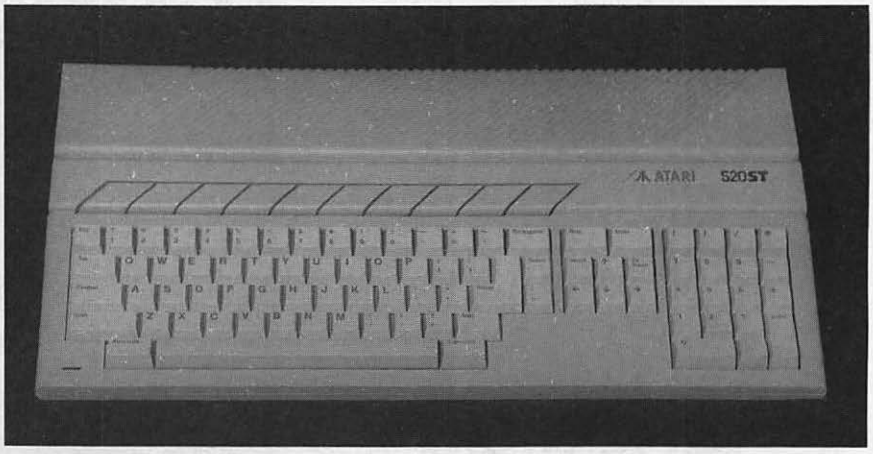


Fig. 2-1. Atari keyboard.

Fig. 2-2. Macintosh keyboard.

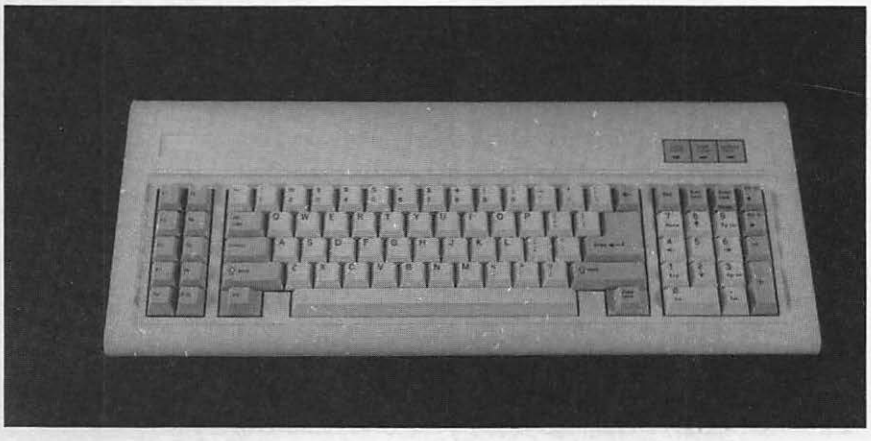


Fig. 2-3. Deluxe IBM AT keyboard.

How the Keyboard Functions

These different keyboards all work according to the same basic principle: pressing a key sends a code to the system unit for translation and processing.

This is similar to throwing a light switch or pressing a button on your television to change channels. The key makes contact and causes a *code*, which uniquely identifies what key or key combination has been pressed, to be placed into a keyboard buffer.

Further processing, usually from the operating system (i.e., MS-DOS or CP/M), takes the character from the keyboard buffer and transfers it to the monitor's screen memory, thus displaying the character on the screen.

TROUBLESHOOTING AND REPAIR GUIDELINES

There are a few problems your keyboard could experience. This section describes these problems, what causes them, and how to fix them.

Problem: The Keyboard Doesn't Work At All

If you press keys on the keyboard, and nothing happens on the screen, the keyboard is not making contact. The character signals are not being sent through the system unit to be displayed on the screen.

SOLUTIONS

- ☐ Check the keyboard cable that runs between the keyboard and the system unit. Make sure it is firmly seated in the keyboard plug.
- ☐ If you have an early model IBM PC that uses a cassette tape interface, check that the keyboard is plugged into the keyboard receptacle, and not the cassette tape interface receptacle.
- ☐ Take the keyboard to a friend or computer store and plug it in to another system like yours. If the keyboard works, then your system unit needs repair and you should take it in for servicing. If the keyboard does not work when you swap it, take it in for servicing.

Problem: The Keyboard Functions Intermittently

Perhaps your problem is that the keyboard sometimes sends a character to the monitor, and sometimes it doesn't. This could be a result of dropping the keyboard, or jarring it in some way. Sometimes you find you must "pound" on a key to make it appear on the screen.

SOLUTION

- ☐ This indicates a loose board in the keyboard assembly. Remove the screws from the bottom of the keyboard and take off the bottom cover to expose the assembly's circuit board (Fig. 2-4). Check the board for cracks. If you find any, take it in for servicing. Tighten the screws that hold the board

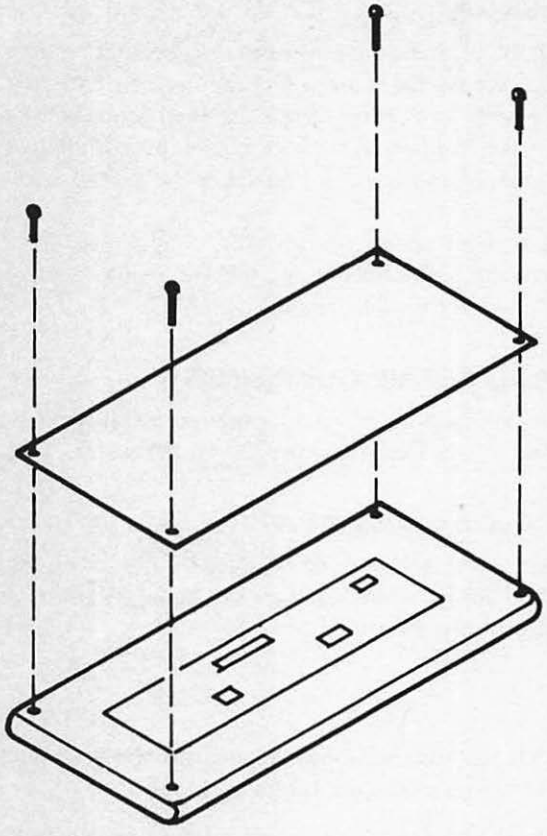


Fig. 2-4. Opening keyboard.

in place. Replace the keyboard bottom cover and run the Keyboard Diagnostic Module to confirm that the key works.

Problem: Particular Keys Do Not Work

You might be typing along and notice what looks like a "typo." Perhaps you see an "a" that was left out of a word. You go back to the word and type the "a" where it's supposed to be. But nothing happens; the "a" simply is not working while all the other keys are working fine.

SOLUTIONS

- ☐ First, run the Keyboard Diagnostic Module and confirm that the key does not work.
- ☐ If it does not work, use a flat-blade screwdriver to pry the key from the keyboard (Fig. 2-5). Spray contact cleaner (available at most electronics stores) on the key post (Fig. 2-6). Replace the keycap on the post and try

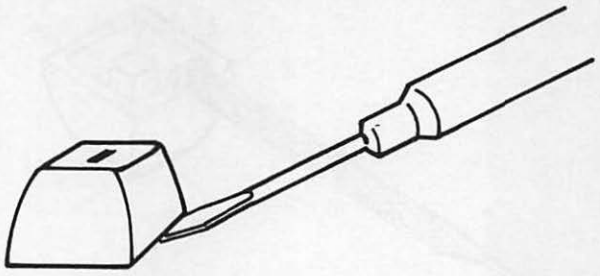


Fig. 2-5. Lifting key from key post.

it again. If the key still does not work, the keyboard might have a more serious problem; take it in for servicing.

Problem: Keys Stick When Pressed

You might press a particular key, and find that it sticks. Its character fills up the whole screen. You end up having to pry the key up to make it stop. There are three remedies you can try for this problem.

SOLUTIONS

- ☐ Lift the key off and spray contact cleaner on the key post as described for the previous problem. This clears off any dust or dirt that might be obstructing the free movement of the key.
- ☐ Remove the keyboard cover from the bottom. Take a flat file and slightly file the cutout in the case where the key is binding. This makes the receptacle larger, so the key doesn't fit so tightly (Fig. 2-7).

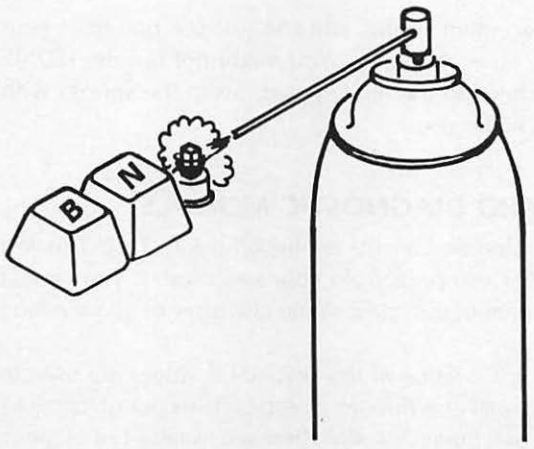


Fig. 2-6. Spraying contact cleaner on key post.

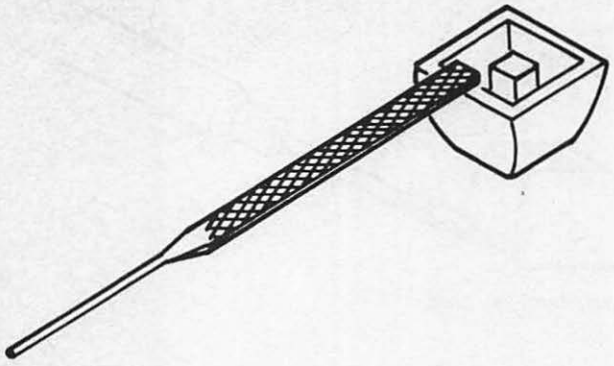


Fig. 2-7. Filing the key cutout.

- ☐ If the key was not binding on the case, pop the key off and lightly file the edges of the key itself. This should loosen the fit between the key and key case so that it can move more freely and no longer cause the key to stick.

Problem: A Key Doesn't Feel Right

As a keyboard ages, some of the more frequently used keys will begin to feel "mushy," or at least not as solid as the other keys. This has to do with the key springs.

SOLUTIONS

- ☐ Call around to your computer retailers to find keyboard springs for your specific keyboard brand. Pry off the key cap and then remove the spring from its post (Fig. 2-8). Replace the old spring with the new, and pop the key cap back on.
- ☐ If you can't find a replacement spring, you can just use one from your keyboard that you don't use very often. You might not use the HOME key, or F7, or the numbers on the ten-key pad. Swap the springs with one of these keys you never use.

RUNNING THE KEYBOARD DIAGNOSTIC MODULE

The Keyboard Diagnostic Module consists of the Echo Key Test. This test displays, or echoes, any key that you press from your keyboard. If you press a special character or key combination, the name of the character or combination is displayed.

This test also displays the ASCII value of the key. ASCII values are used to represent the display characters, such as A through Z, and certain control functions like cursor-left and clear-screen as a numeric code. There are usually 128 numeric codes represented in a standard system. Most systems, however, recognize 256

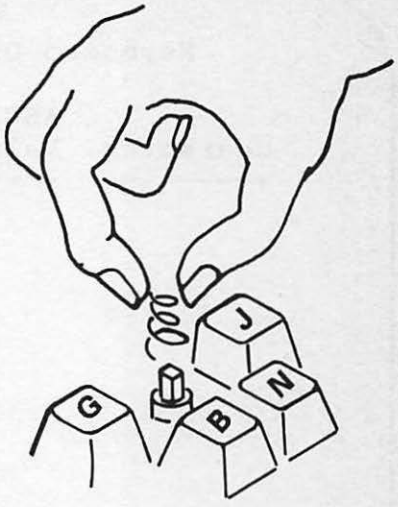


Fig. 2-8. Removing a spring from the key post.

ASCII values. The additional 128 codes represent special graphic characters, foreign character sets, or math symbols. Because these characters are not normally accessible from the keyboard, the ASCII code must be used to access them in programs. For example, the command PRINT CHR$(1) displays the smile-face on the IBM PC. CHR$ is a BASIC function that treats the numeric operand in parenthesis as a character.

By echoing the pressed key, you can check that the keyboard is working, if a particular key is working, or if the keys are sending the proper code to the computer to display the proper character. It also checks key combinations, such as CTRL A and CTRL R.

To run the Keyboard Diagnostic Module, follow the instructions provided in "Running the Diagnostic Program" in Chapter 1. Then press K to initiate the Keyboard Diagnostic Module. The screen clears and several lines of instruction are displayed (Fig. 2-9).

Following these instructions, press any key you wish to test. You can test any key on the keyboard: alphanumeric keys, cursor keys, function keys, and special CTRL character keys. The key you press is displayed along with its ASCII value and any applicable key combination (Fig. 2-10).

If a character other than the one you pressed is displayed, or if no ASCII value is displayed, then you have accomplished the first troubleshooting step: there is something wrong with your keyboard. Refer to the "Troubleshooting and Repair Guidelines" section earlier in this chapter for help in getting the keyboard working right again.

When you finish testing, press the ESC key, and the program will return to the System Diagnostic Main Menu. If this does not happen, the ESC key is failing. If this occurs, press CTRL BREAK to end the program.

HOW THE KEYBOARD MODULE WORKS

The Keyboard Diagnostic Module begins at Line 220 with a REMARK

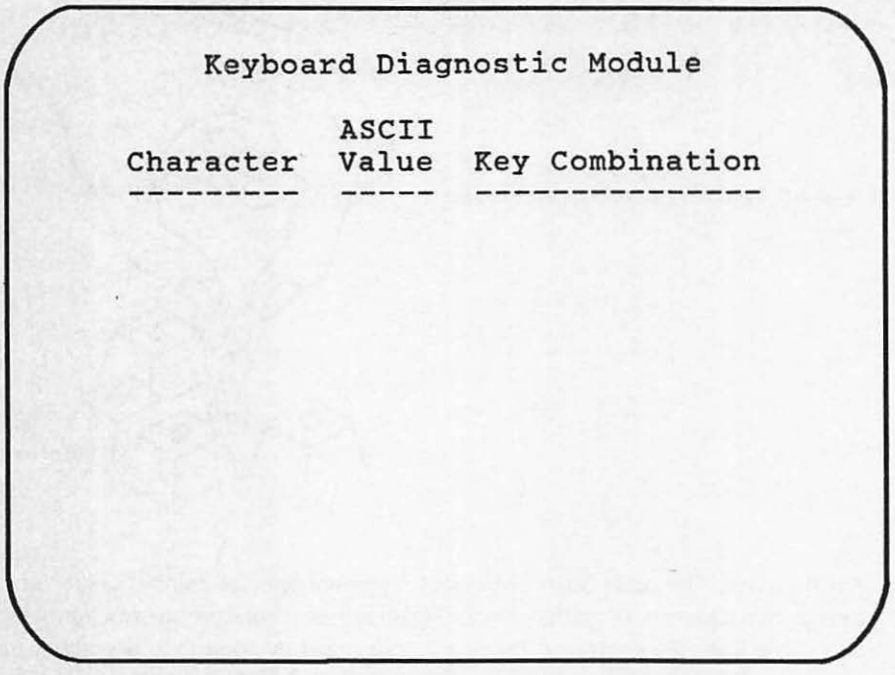


Fig. 2-9. Keyboard diagnostic module.

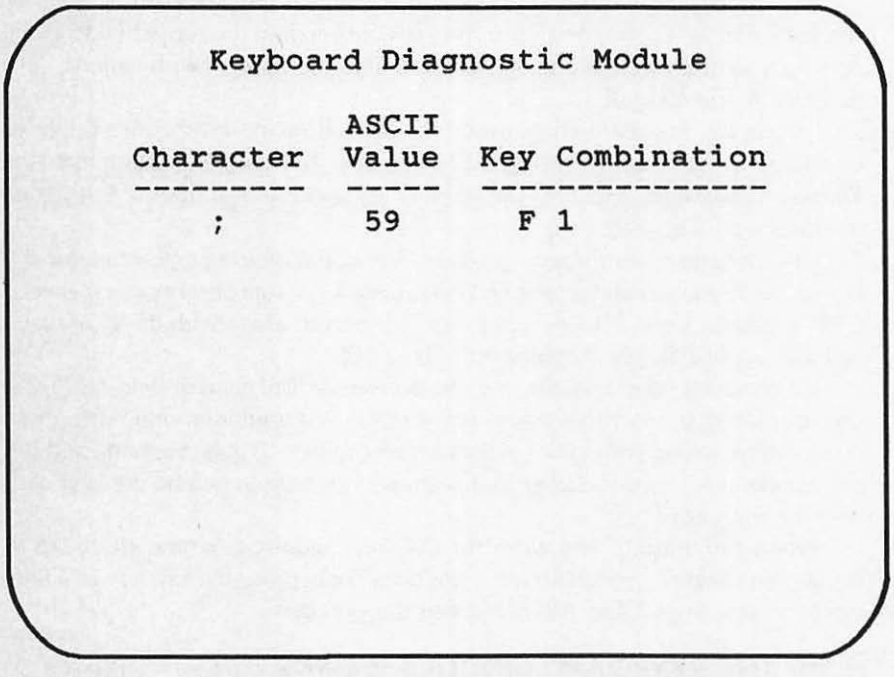


Fig. 2-10. Keyboard test.

statement indicating the beginning of the module (Fig. 2-11):

```
220   REM: Keyboard Diagnostic Module
230   FOR I=1 TO 10:KEY I," ":NEXT I:KEY OFF:REM Disable Function Keys
```

Lines 240 and 250 clear the screen and display the Keyboard Test instruction prompts:

```
240   CLS:PRINT TAB(7) "Keyboard Diagnostic Module":PRINT
250   PRINT TAB(14) "ASCII":PRINT TAB(3) "Character Value Key
      Combination":PRINT TAB(3) "--------- ----- ---------------"
```

Line 260 calls subroutine 2930, which checks whether a key has been pressed. Using the INKEY$ instruction and a loop, this lets you check for any key without having to enter the RETURN key.

```
260   GOSUB 2930
```

Line 270 sets variable N to the numeric ASCII value of the keystroke that has been pressed. This will be used later in the routine to see whether a function key, ALT key, BREAK key, or a CTRL key combination has been pressed:

```
270   N=ASC (A$)
```

```
220 REM: Keyboard Diagnostic Module
230 FOR I=1 TO 10:KEY I,"":NEXT I:KEY OFF:REM Disable
    Function Keys
240 CLS:PRINT TAB(7) "Keyboard Diagnostic Module":PRINT
250 PRINT TAB(14) "ASCII":PRINT TAB(3) "Character  Value
    Key Combination":PRINT TAB(3)"---------  -----
    ---------------"
260 GOSUB 2930
270 N=ASC(A$)
280 IF ASC(A$)=27 THEN 40
290 LOCATE 6,6:PRINT SPACE$(40);
300 A=LEN(A$):IF A>= 2 THEN 370
310 IF N>28 THEN 320 ELSE KY$="Ctrl":KY2$=CHR$(N+96):
    GOTO 730
320 KY$="Ctrl"
330 IF N=30 THEN KY2$="6":GOTO 770
340 IF N=31 THEN KY2$="-":GOTO 770
350 LOCATE 6,6 :PRINT AN$;:PRINT TAB(14) ASC(A$);:GOTO 260
360 REM: Check for Function Keys
```

Fig. 2-11. Keyboard diagnostic module listing.

```
370 KY$="":KY2$="":A=ASC(RIGHT$(A$,1))
380 IF A>3 THEN 400
390 KY$="Ctrl":KY2$="2":GOTO 730
400 IF A=15 THEN KY$="Shift Tab":GOTO 730
410 IF A>=16 OR A<=50 THEN KY$="Alt"
420 IF A>25 THEN 430 ELSE KY2$=A1$((A-16)+1):GOTO 730
430 IF A>38 THEN 440 ELSE KY2$=A2$((A-30)+1):GOTO 730
440 IF A>50 THEN 450 ELSE KY2$=A3$((A-44)+1):GOTO 730
450 IF A >68 THEN GOTO 470
460 KY$="F"+STR$((A-59)+1):GOTO 730
470 IF A=71 THEN KY$="Home":GOTO 730
480 IF A=72 THEN KY$="Cursor Up":GOTO 730
490 IF A=73 THEN KY$="Pg Up":GOTO 730
500 IF A=75 THEN KY$="Cursor Left":GOTO 730
510 IF A=77 THEN KY$="Cursor Right":GOTO 730
520 IF A=79 THEN KY$="End":GOTO 730
530 IF A=80 THEN KY$="Cursor Down":GOTO 730
540 IF A=81 THEN KY$="Pg Dn":GOTO 730
550 IF A=82 THEN KY$="Ins":GOTO 730
560 IF A=83 THEN KY$="Del":GOTO 730
570 IF A>113 THEN GOTO 590
580 IF A<=93 THEN KY2$="F"+STR$((A-84)+1):KY$="Shift":
    GOTO 730
590 IF A>103 THEN GOTO 610
600 KY$="Cntrl":KY2$="F"+STR$((A-94)+1):GOTO 730
610 IF A<104 OR A>113 THEN GOTO 630
620 KY$="Alt":KY2$="F"+STR$((A-104)+1):GOTO 730
630 IF A>113 AND A<120 THEN KY$="Ctrl"
640 IF A=114 THEN KY2$="PrtSc":GOTO 760
650 IF A=115 THEN KY2$="Cursor Left":GOTO 760
660 IF A=116 THEN KY2$="Cursor Right":GOTO 760
670 IF A=117 THEN KY2$="End":GOTO 760
680 IF A=118 THEN KY2$="Pg Dn":GOTO 760
690 IF A=119 THEN KY2$="Home":GOTO 760
700 IF A>131 THEN 720
710 KY$="Alt":KY2$=A4$((A-120)+1):GOTO 760
720 IF A=132 THEN KY$="Ctrl":KY2$="Pg Up"
730 IF N>=9 AND N<=13 THEN 770
740 IF A=30 OR A=31 THEN 770
750 REM
760 LOCATE 6,6:PRINT A$;:PRINT TAB(14) ASC(RIGHT$(A$,1));
    :PRINT TAB(23) KY$+" "+KY2$:GOTO 260
770 LOCATE 6,6:PRINT TAB(14) ASC(RIGHT$(A$,1));:PRINT TAB
    (23) KY$+" "+KY2$:GOTO 260
780 GOTO 40
```

Line 280 checks to see if the ESC key (or the CLEAR key on the Macintosh) has been pressed. The ASCII value for this key is 27, and would appear in A$. If it has been pressed, the program branches back to line 40, then returns to the System Diagnostic Main Menu:

```
280 IF ASC(A$)=27 THEN 40
```

Line 290 clears the sixth line of the display by printing a line of 40 spaces. This is done so that the Keyboard Test title, menu, and instructions do not have to be redisplayed each time a key is pressed:

```
290    LOCATE 6,6:PRINT SPACE$(40);
```

CP/M USERS: This does not hold true for the CP/M version of this program. Because the LOCATE instruction is invalid for CP/M, the entire display must be erased and redisplayed for every subsequent keystroke.

Line 300 checks to see if a function key has been pressed. It does this by checking the length (LEN) of variable A$. If the length of A$ is greater than or equal to two characters, then the program knows that a function or special key has been pressed. In this case, the program branches to line 370:

```
300    A=LEN(A$):IF A>= 2 THEN 370
```

The program proceeds to line 310 through line 350 if a regular alphanumeric key is pressed. If an alphanumeric key is pressed in combination with the CTRL key, this code loads the proper key combination into the display line. The program then branches to appropriate display instruction.

If a regular alphanumeric key is pressed, the program falls through to line 310. Line 310 checks to see if N, which was set in line 270, is greater than 28:

```
310    IF N>28 THEN 320 ELSE KY$="Ctrl":KY2$=CHR$(N+96):
       GOTO 730
```

If it is, then the program falls through to Line 320. If it is not greater than 28, the ELSE statement sets a value called KY$, which is used to display the value of the word "Ctrl" to indicate that it is a control sequence. It then sets KY2$ to the character string plus 96, hence displaying the actual key that was pressed. For some of the lower CTRL sequences, it will be a value less than 96. Adding 96 to the resulting value gives you the actual graphic key, such as a, b, or c, that has been pressed. The program then branches to line 730.

```
320    KY$="Ctrl"
```

Line 320 also sets KY$ to the value "Ctrl." Again, this is set if N is greater than 28.

Line 330 checks to see if N is equal to 30. If so, it sets KY2$ to six and then branches down to line 770:

```
330   IF N=30 THEN KY2$="6":GOTO 770
```

Line 340 checks to see if N is equal to 31. If this is true, then K2$ is equal to the "-" character, and the program branches to line 770:

```
340   IF N=31 THEN KY2$="-":GOTO 770
```

If N is not equal to 30 or 31, the program falls through to line 350. The cursor is positioned at line 6, and the key that was pressed is displayed. The cursor moves over 14 spaces, and then the ASCII value of A$ is printed. Finally, the program branches back to line 260, looking for the next key to be pressed:

```
350   LOCATE 6,6 :PRINT A$;:PRINT TAB(14) ASC(A$);:GOTO 260
```

Line 350 is essentially where the program falls through if a CTRL sequence or function key has not been selected.

Line 360 is a REMARK statement stating that this is where function key checking is done. Function key checking is not valid for Macintosh and CP/M systems, and these particular lines of code do not exist in those versions of the module:

```
360   REM: Check for Function Keys
```

Lines 370-730 check to see which function key or CTRL sequence has been pressed. It loads the appropriate description into the key combination fields and then displays it.

Line 370 sets KY$ and KY2$ to null. A, the ASCII code, is assigned to the second byte of A$:

```
370   KY$=
```

If a function key is pressed, the program branches to line 370. If the length of the character string is two characters, the program knows it is a function key. At this point, the program is only concerned with the second byte of A$.

Line 380 checks to see if A is greater than three. If it is, then the program branches to Line 400:

```
380    IF A>3 THEN 400
390   KY$="Ctrl":KY2$="2":GOTO 730
```

If A is less than three, then it is a CTRL sequence, and KY$ is set to CTRL. If KY2$ is equal to two, the program branches to line 730 to be displayed. If A

is greater than three, the program falls through to line 400. Then the program looks for specific cases:

```
400   IF A=15 THEN KY$="Shift Tab":GOTO 730
```

If A=15, KY$ is equal to SHIFT TAB, meaning that the SHIFT TAB keys were pressed. At this point, the program branches to line 730.

Line 410 checks to see if A is between 16 and 50. If it is, KY$ is equal to the ALT key, meaning that the ALT key was pressed:

```
410   IF A>=16 OR A<=50 THEN KY$="Alt"
```

If the statement in Line 410 is not true, the program falls through to line 420, checking to see if A is greater than 25. If it is, the program goes on to line 430, which makes sure that it gets the proper keystroke that the ALT function relates to. This also holds true for lines 430 and 440. These are used to place the correct key combination into the display:

```
420   IF A>25 THEN 430 ELSE KY2$=A1$((A-16)+1):GOTO 730
430   IF A>38 THEN 440 ELSE KY2$=A2$((A-30)+1):GOTO 730
440   IF A>50 THEN 450 ELSE KY2$=A3$((A-44)+1):GOTO 730
```

If A is greater than 68 in line 450, then the program goes on to Line 470, which indicates that the key pressed was a special function key such as the Home or Cursor Left key:

```
450   IF A >68 THEN GOTO 470
```

If the statement in line 450 is not true, the program falls through to line 460, which indicates that a function key was pressed. The first part of this statement provides a number between 1 to 10, for function keys F1 through F10, and then it goes on to line 730 to display the value, ASCII code, and key combination:

```
460   KY$="F"+STR$((A-59)+1):GOTO 730
```

Lines 470 through 560 check for specific values such as 71, which is the Home key; 72, which is the Cursor Up arrow key; 83, which is the Del key, and so forth. If one of these values is encountered, the appropriate value is loaded into KY$ and the program branches to Line 730:

```
470   IF A=71 THEN KY$="Home":GOTO 730
480   IF A=72 THEN KY$="Cursor Up":GOTO 730
490   IF A=73 THEN KY$="Pg Up":GOTO 730
500   IF A=75 THEN KY$="Cursor Left":GOTO 730
510   IF A=77 THEN KY$="Cursor Right":GOTO 730
```

```
520   IF A=79 THEN KY$="End":GOTO 730
530   IF A=80 THEN KY$="Cursor Down":GOTO 730
540   IF A=81 THEN KY$="Pg Dn":GOTO 730
550   IF A=82 THEN KY$="Ins":GOTO 730
560   IF A=83 THEN KY$="Del":GOTO 730
```

If the ASCII code of the key pressed does not fall within these values, the program falls through to line 570. This checks to see if A is greater than 113. If so, the program branches to Line 590:

```
570   IF A>113 THEN GOTO 590
580   IF A<=93 THEN KY2$="F"+STR$((A-84)+1):KY$="Shift":
      GOTO 730
```

If A is less than 113, the program falls through to line 580, where it checks to see if A is less than or equal to 93. If so, it indicates that a shifted function key was pressed. The appropriate value is set for the key combination in KY2$ and KY$. The program then branches to line 730.

Line 590 checks to see if A is greater than 103. If it is, the program branches to line 610. If not, it falls through to line 600, indicating that a CTRL function key was pressed. The appropriate values are set up in KY2$ and KY$. The program then branches to line 730:

```
590   IF A>103 THEN GOTO 610
600   KY$="CNTRL":KY2$="F"+STR$((A-94)+1):GOTO 730
```

Line 610 checks to see if A is less than 104 or greater than 113. If either of these is true, the program goes through to Line 630. Otherwise, it indicates that the ALT function key was pressed. "Alt" is loaded into KY$, and the function key number is loaded into KY2$. The program then branches to line 730:

```
610   IF A<104 OR A>113 THEN GOTO 630
620   KY$="Alt":KY2$="F"+STR$((A-104)+1):GOTO 730
```

Line 630 checks to see if A is greater than the value 113 and less than the value 120. If so, KY$ is made equal to "Ctrl."

```
630   IF A>113 AND A<120 THEN KY$="Ctrl"
```

Lines 640 through 650 place the appropriate value into KY2$ so that the key combination displays the name of the combination pressed, such as "Ctrl PrtSc," or "Ctrl Cursor Left":

```
640   IF A=114 THEN KY2$="PrtSc":GOTO 760
650   IF A=115 THEN KY2$="Cursor Left":GOTO 760
660   IF A=116 THEN KY2$="Cursor Right":GOTO 760
```

```
670   IF A=117 THEN KY2$="End":GOTO 760
680   IF A=118 THEN KY2$="Pg Dn":GOTO 760
690   IF A=119 THEN KY2$="Home":GOTO 760
```

The program then branches to line 760. If A in line 630 is not between 113 and 120, the program proceeds to line 700. Line 700 checks to see whether A is greater than 131. If it is, the program branches to line 720:

```
700   IF A>131 THEN 720
```

If A is not greater than 131, the program falls through to line 710, which sets KY$ equal to "Alt," and KY2$ equal to the appropriate key value. Then the program branches to line 760:

```
710   KY$="Alt":KY2$=A4$((A-120)+1):GOTO 760
```

Line 720 checks to see if A is equal to 132 and then sets the KY$ to "Ctrl" and KY2$ to "Pg Up," which is a special function key:

```
720   IF A=132 THEN KY$="Ctrl":KY2$="Pg Up"
```

If A is not equal to 132, the program falls through to line 730. Line 730 checks to see if N is between 9 and 13:

```
730   IF N>=9 AND N<=13 THEN 770
```

If it is, the program branches to line 770.

Line 740 checks to see if A is equal to 30 or 31:

```
740   IF A=30 OR A=31 THEN 770
```

If so, the program branches to line 770.

Certain CTRL sequences can affect the screen display by clearing the screen or feeding lines. Because this can confuse the diagnostic program, line 760 ensures that those particular character sequences are not displayed with the ASCII code and key combination.

Line 760 locates the cursor on line 6, position 6 of the monitor. It prints the actual character, the ASCII value of the character, and the values of KY$ and KY2$, which is the key combination:

```
760   LOCATE 6,6:PRINT A$;:PRINT TAB(14) ASC(RIGHT$(A$,1));
      :PRINT TAB(23) KY$+" "+KY2$:GOTO 260
```

The program then branches to line 260 to check for more keystrokes.

Line 770 again locates the cursor at line 6, position 6 on the monitor display. This time, however, the keystroke value is not printed. The lines that branch to

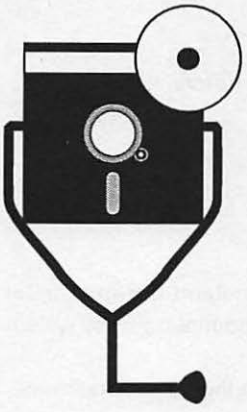
line 770 filter out keystrokes that would adversely affect the diagnostic program by clearing the screen, or display, such as keystrokes that move the cursor around:

```
770   LOCATE 6,6:PRINT TAB(14) ASC(RIGHT$(A$,1));:PRINT TAB
      (23) KY$+" "+KY2$:GOTO 260
```

Line 770 tabs past AN$ without printing it. But it does print the ASCII value as well as the value of KY$ and KY2$ for the key combination. This calls up the routine that checks for another key to be pressed. Once this is done, the program branches to line 260.

If you are interested in seeing a chart of the ASCII display codes, refer to your system's BASIC manual.

If you do any of your own programming, this program can also help you discern how to include certain special characters in your program. Use the PRINT CHR$(xx) instruction, where xx is the ASCII code that displays next to the echoed character. When you run your program, the appropriate graphic character will be displayed.



Chapter 3

The Monitor

When working with a computer, you usually need constant visual feedback, or *output*. You need to see your input as you enter it, and you need to see processing results as the computer produces them. Just as the keyboard is the primary input device, your computer uses the monitor to provide the primary output.

DESCRIPTION AND FUNCTION OF THE MONITOR

The monitor is also referred to as the cathode ray tube (CRT), or video display. It is the screen on which you see the status of the program you're currently running.

Types of Monitors

There are several different types of monitors available. You might have an internal monitor, an external monitor, or your monitor might be a television set. You might also have a monochrome monitor, or a high-resolution color monitor.

The Internal Monitor. In some computer systems, such as the Kaypro, the monitor and system unit are built together as one integral unit (Fig. 3-1).

The Television Set Monitor. Many of the less expensive home computers are designed to use a television set as the monitor. The Commodore 64, Apple II, and the Coleco Adam are examples of such systems. If desired, the IBM PC and many of the PC clones can use a television set as the monitor. These computers are supplied with an external radio frequency (rf) converter that attaches to the

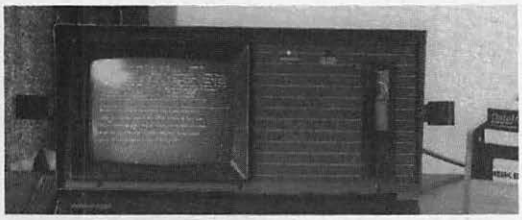


Fig. 3-1. Kaypro monitor.

television via the cable connector or antenna leads.

The External Monitor. The IBM and most compatible systems use an external monitor. The external monitor is physically a separate component from the system unit and keyboard.

There are three basic variations on the external monitor theme: monochrome, high-resolution graphics, and color monitors.

The Monochrome Monitor. The monochrome monitor operates only in the character mode, meaning it displays the A-Z alphabet, as well as the standard special characters such as @, #, $, and %. This is in contrast to a monitor that can operate in a graphic mode, which is one that can display special graphic characters (see High-Resolution Graphics Monitor, below). The monochrome monitor has a character set of about 256 characters. It can accommodate 80 characters on each of 24 or 25 lines at a time for approximately 2000 on-screen character locations (Fig. 3-2).

The High-Resolution Graphics Monitor. The high-resolution graphics monitor can display the same character set as the monochrome monitor. What makes the graphics monitor special is that every little speck of light, or *pixel* picture

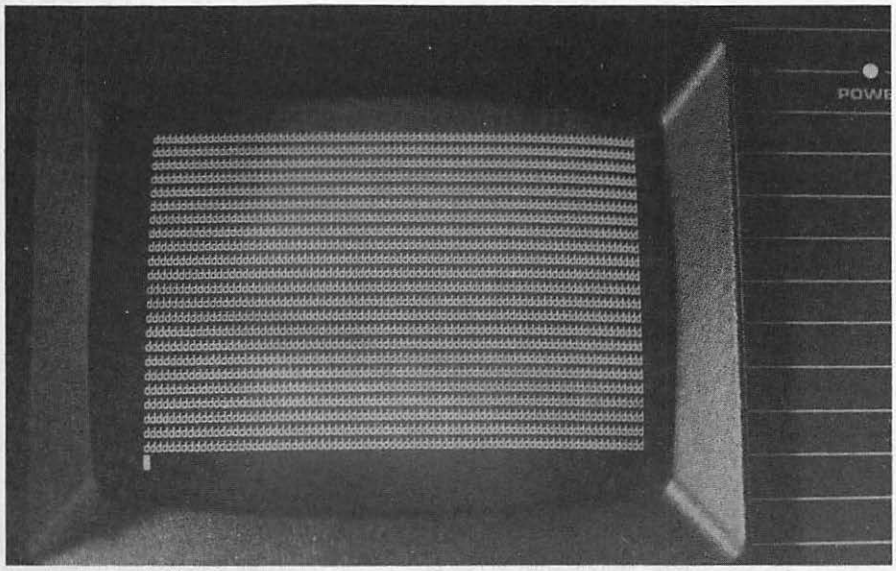


Fig. 3-2. 80 by 25 character monitor.

element, can be illuminated on the screen in different combinations to form pictures. These pixels allow you to create and display computer drawings and graphics on the screen. The more pixels there are on your screen, the better the screen resolution. The better the screen resolution, the sharper the displayed image.

The different types of high-resolution monitors are classified by the actual number of horizontal and vertical pixels that can be accommodated on the screen at one time. Customarily, this number is approximately 380 by 740 pixels. The higher these numbers, the sharper the monitor's resolution (Fig. 3-3).

The Color Monitor. The color monitor is a high-resolution monitor that can display the pixels in different colors. This might seem to be a luxury feature, but often it is in fact a necessity when doing business graphics or design work with your computer.

The type of video interface board you have in your system unit determines which of these three monitors can be attached to a given external monitor computer system. If you have the standard monochrome video interface board, then your system can support only a monochrome monitor. However, you can install and use a graphics or color monitor on your system if you also install a video board designed to accommodate graphics or color.

An alternative to buying a new monitor is to obtain a "monochromatic graphics board" such as the Hercules graphics board. This board allows you to display graphics on a standard monochrome character-only monitor. Although the screen remains monochromatic, you can create pie charts, bar graphs, and other kinds of drawings, using different gray-tone shading and black-and-white patterns instead of color.

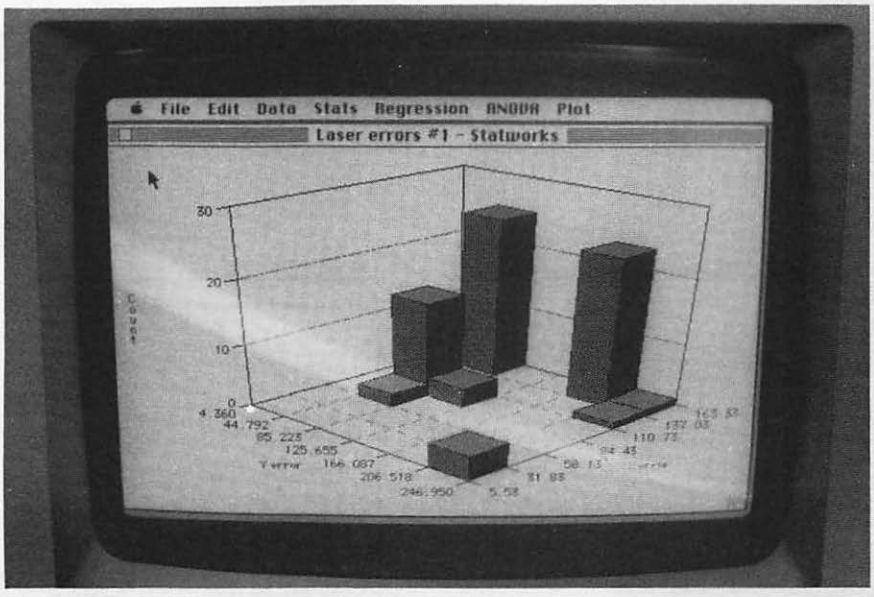


Fig. 3-3. High-resolution monitor.

How the Monitor Functions

The monitor consists of the cathode ray tube, the power supply, and the contrast, brightness, and vertical/horizontal hold adjustments. The CRT is a large vacuum tube with a viewing face similar to a television screen. The electron beam is focused and controlled to shoot against the back of the picture tube, which is coated with a material called phosphor. As the electrons strike the phosphor atoms, the atoms are excited to the point where they glow. This reaction serves to form a small, clearly defined light spot (Fig. 3-4).

A number of these light spots are the dots that form the characters. On a monochrome monitor, about seven of these light spots make up one character. On a high-resolution monitor, these light spots are the pixels which can densely form the characters, graphic elements, and drawings.

When you press a character on the keyboard to be displayed on the monitor, or when a particular application prints processing results, the program instructs the monitor to paint a certain pattern. This pattern is dependent on the type of display matrix your monitor has.

The monochromatic monitor display works on a character matrix. The display is divided into 24 to 25 horizontal rows. These rows are intersected by 40 to 80 vertical columns. The position where a column and row intersect is one character location (Fig. 3-5).

The high-resolution or color display is a matrix as well, but a much more precise matrix, working at a pixel level rather than a character level. Because an individual pixel is much smaller than a character, and because there is a greater density of pixels, there is more flexibility and precision represented on the display. This creates a sharper picture with higher resolution.

You might notice that some monitors are able to display certain graphic characters when others cannot. You might also notice that you have certain keys on your keyboard that your monitor might not be able to represent. This is because the monochrome and alphanumeric displays of most computers have a particular piece of hardware, usually an EPROM (Eraseable Programmable Read-Only Mem-

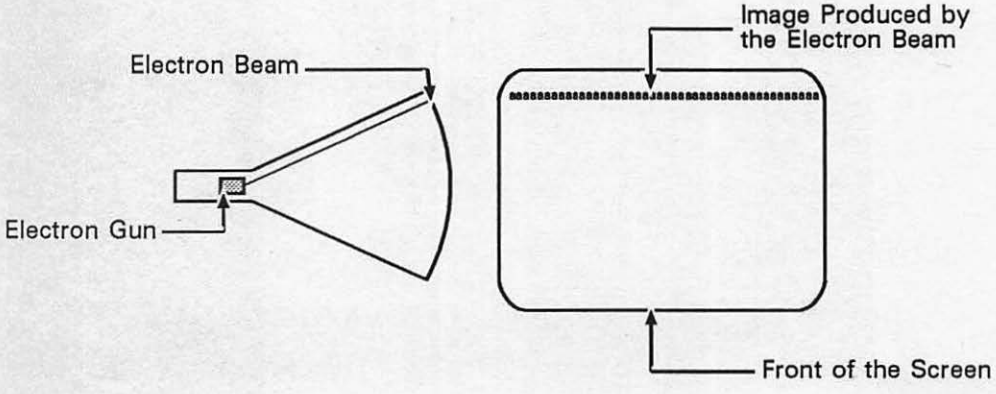


Fig. 3-4. Monitor and electron beam.

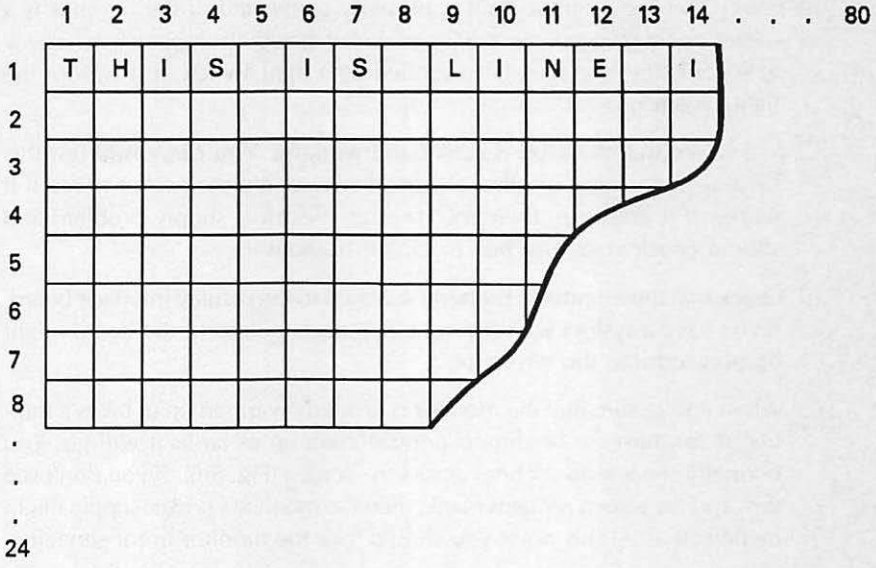


Fig. 3-5. Column/row matrix.

ory chip), called a "character generator." The character generator stores the character sets you have available with your monitor. It is responsible for placing the individual dots that make up a character on the screen. The character sets that the character generator can create vary from monitor to monitor, and computer to computer.

TROUBLESHOOTING AND REPAIR GUIDELINES

This section describes many of the possible monitor problems, and how to troubleshoot and fix them. Many of the troubleshooting steps include running the Monitor Diagnostic Module. Further instructions and explanations of this module are provided in the next section.

Problem: Nothing Displays on the Monitor

The first major problem is when the monitor does not work at all; when nothing displays on the screen at all.

SOLUTIONS

- ☐ Some computers have a "screen-saver" program that automatically blanks the screen after several minutes of running idle. Usually all you need to do to get the screen to display again is to press any key.
- ☐ See if the brightness and contrast knobs have been turned down. Another user might have turned the controls down to prevent the screen from etching (see Etching in the next section).

- ☐ Check that the monitor itself is properly connected. If the monitor is a television set plugged into a different socket than the computer itself, check to see whether the plug is controlled by a light switch. If it is, turn the light switch on.
- ☐ Make sure that the plug is active and working. You can easily test this by plugging a lamp or other electrical device into the socket to see if it works. If it does not, then you have an electrical supply problem and should check your fuse box or circuit breaker.
- ☐ Check that the monitor is properly attached to the display interface board. If you have a system with several different output ports, the board might be plugged into the wrong port.
- ☐ When you're sure that the monitor is properly warmed up (it takes a minute or so), turn the brightness control knob up as far as it will go. You normally see a series of lines across the screen (Fig. 3-6). If you don't see this, and the screen remains blank, then the monitor's power supply might be defective. At this point you should take the monitor in for servicing.
- ☐ If there is a problem with the display interface board, it will manifest itself in a different way. You might hear the monitor make a crackling noise. If you run your hand next to the screen, the hairs on the back of your hand might become attracted to it. No cursor is displayed on the screen. You might try swapping the display interface board with an identical system. If the monitor works with the swapped board, then you know you need to purchase a new interface board.

Problem: Etching

"Etching," or "ghosting," takes place on the screen when one particular format, such as your text editor menus or your database screen, is left sitting on the screen for long periods of time without changing. If you leave the computer running with the same image displayed for many hours a day, or many days a week, the constant bombardment of the electrons on the phosphor of the picture tube can permanently burn or "etch" that image into the screen. At this point, even when you display something else, you still see a ghost image of the etched

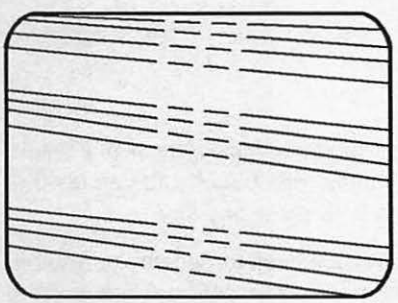


Fig. 3-6. High brightness lines on monitor.

display. At the same time, the display you want to see is fainter, washed out.

There isn't anything you can do to make an image that is etched into your display go away once it's there. In other words, you can't fix etching, unless you buy a new monitor. The key in this case is prevention. To prevent etching, either turn off the display or turn down the brightness when the computer is not in use.

Incidentally, if you have a Macintosh, you'll never need to worry about etching. The Macintosh screen works in reverse video. That is, instead of green or white letters on a black background, the Macintosh screen displays black letters on a white background. The majority of the pixels are always on, therefore a single image cannot be etched into the phosphor.

Problem: Blank Character Location

Depending on the type of monitor you have, characters are displayed either in a character or a pixel matrix. One possible problem is when one of these locations in the matrix does not display anything; you type a character to go into a certain location, but the character does not display.

The high-resolution or color display is based on the pixel matrix, with the display divided by a large number of pixels, giving the screen its high-resolution attributes. This creates a very fine matrix, so rather than the character being the smallest unit that can display on the monitor, the pixel is the smallest unit.

You can have a problem with pixels not displaying. You might see certain characters missing, which happens when the computer has lost a memory location. You might also see a part of a character missing, or even intermittent spots on the display, looking like pin pricks poked into the screen.

SOLUTIONS

- ☐ To troubleshoot this problem, run the Display Test of the Monitor Diagnostic Module. Press a character on the keyboard, and the entire screen is filled with this character at every available character location. If you see that a particular character is missing, this indicates a problem in the display interface board. Each character represents a video memory location. If a character does not display, it means that its memory location is probably bad.
- ☐ If you know what you're doing, you can swap the chips out of the video interface board yourself. However, unless you're experienced at doing this, it's probably best to just take the board in for servicing and have a professional take care of it.

Problem: Misaligned Display

Because the monitor is very similar to a television set, with many of the same qualities as a television, it is subject to the same types of problems. For example, the picture can become skewed, or it can physically compress. It can generally go out of alignment (Fig. 3-7).

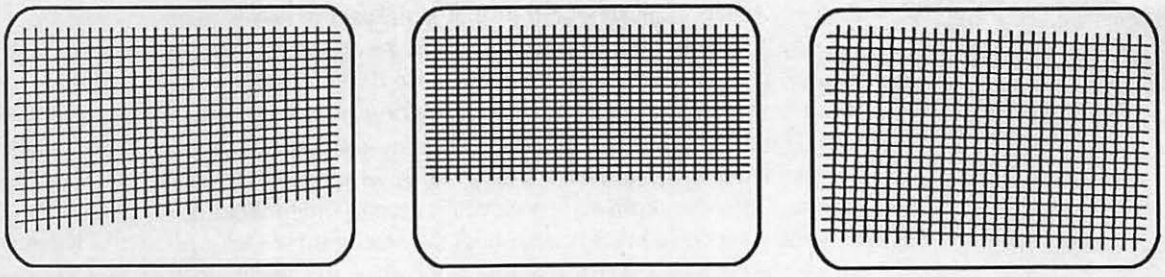


Fig. 3-7. Misaligned monitor displays.

In many cases it is more difficult to perceive such alignment problems when you have nothing but text on the screen. To check for proper alignment, follow these suggestions.

SOLUTIONS

- ☐ Run the Alignment Test. If the grid does not appear square and symmetrical on the screen, the monitor needs adjustment. There are adjustment potentiometers on the back of most monitors. Use your tuning wand to gently turn these potentiometers ("pots") either clockwise or counterclockwise, to bring the screen back into alignment.
- ☐ If adjusting these pots does not bring the monitor back into the proper alignment, take the monitor in for servicing and have it adjusted.
- ☐ If your monitor is a television set, you can simply adjust the horizontal and vertical hold controls until the alignment grid looks straight.

Problem: A Particular Character Does Not Display

Suppose you have a certain character available on the keyboard. You press the key, but you don't see it displayed on the monitor. The key itself might be functioning right, but your monitor's character generator does not support that particular character.

SOLUTION

- ☐ If you run the All Characters Display Test and do not see all the characters that you expect, or if you get some kind of "garbage" on the screen, then you have a problem with the interface itself, and you should take the board in for servicing.

Intensity Problems

On many systems, such as the IBM PC and the Macintosh, various text intensities can be displayed. You can specify that the text be displayed in a higher or lower intensity than normal. You can also make the text flash or blink, or

Table 3-1. Intensity Mode Table.

Text Attribute Modes

	IBM PC	Macintosh
Normal	COLOR 7	CALL TEXTFACE(0)
Reverse Video	COLOR 0,7	CALL TEXTFACE(3)
Blink	COLOR 31	N/A
Underscore	COLOR 7,1	CALL TEXTFACE(4)
Bold/High Intensity	COLOR 15,0	CALL TEXTFACE(1)
Italic	N/A	CALL TEXTFACE(2)
Invisible	COLOR 0	N/A

underscore it.On the Macintosh, you can change the text to display in a washed-out or ghost-type image.

If the contrast or brightness knob is not adjusted properly, these intensities could be masked. Two very different intensities might look identical. Text in the lighter intensities might not display at all.

SOLUTION

☐ Run the Intensity Test. In this test, all available intensities are displayed at once so you can see if any of them are being displayed incorrectly. If they are, adjust the screen for contrast or brightness until they all display as they should.

CP/M USERS: The Intensity Test is not available on CP/M systems.

Scrolling Problems

When your screen is completely filled with text and you display another character on the next line, the entire screen moves itself up, pushing the top line off the screen, to make a place for the next line. This is called *scrolling.*

If the display does not move up, but rather fills one screen and then overwrites at the bottom of the screen, the monitor has a scrolling problem.

SOLUTION

☐ To check this, run the Scroll Test. Enter enough repetitions (at least 26) so that the printed lines will more than fill up the screen. If the display does not scroll properly, then there is a problem with the display interface board, and it should be taken in for servicing.

Twenty-fifth Line Display Problems

Most monitors display either 24 or 25 horizontal lines. Sometimes you have 25 lines, but one of those lines is used to display system or program status. On many systems you can change the system programming to allow you to display your own information on all 25 available lines.

NOTE: All MS-DOS systems have the 25th status line feature available. The Macintosh does not have the 25th status line feature. Whether or not a particular CP/M system has this feature depends on the manufacturer.

SOLUTIONS

- ☐ If the Status Line Test does not display anything on the 25th line, check your system documentation. Your BASIC programming manual might be a good source to find out whether or not a status line is indeed available. If it is available, check to see if there is any special programming you need to do in order to display your own data in this line.
- ☐ As noted, MS-DOS systems do support the 25th status line feature. If you do not get the expected pattern in the 25th status line on these kinds of systems, then you have a problem with the display interface board and it should be taken in for servicing.

RUNNING THE MONITOR DIAGNOSTIC MODULE

To run the Monitor Diagnostic Module, follow the instructions provided in "Running the Diagnostic Program" in Chapter 1. Then press M. The screen clears, and the Monitor Diagnostic Menu is displayed (Fig. 3-8).

To activate one of these tests, press the corresponding number. To end the Monitor Diagnostic Module, press the ESC key and the Diagnostic Main Menu is displayed again.

Display Test

To run the Display Test, press 1. The screen clears and then gives the test instruction (Fig. 3-9).

Press a character on the keyboard. (Function keys and the cursor arrow keys do not work for this test, because they do not have a particular graphic display.) The character fills the entire screen at every available character location, including the status line.

At this point, any character locations that do not display the character, or any pixels that are "out," are suddenly very obvious (Fig. 3-10).

After displaying the character at every location, the program waits for you to press another character. You may enter another character at this point, or press ESC (or CLEAR on the Macintosh) to return to the Monitor Diagnostic Menu.

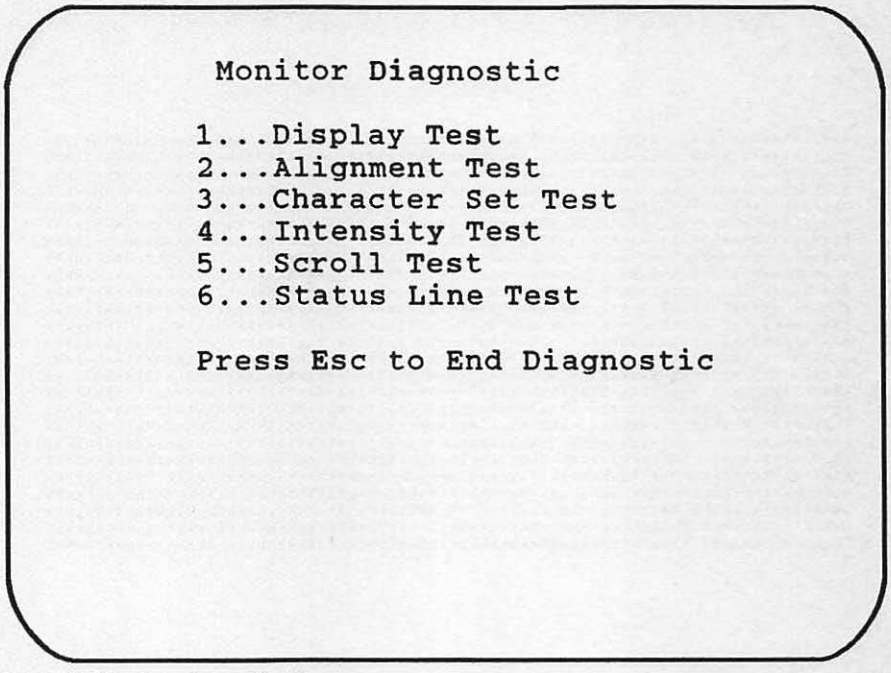


Fig. 3-8. Monitor diagnostic menu.

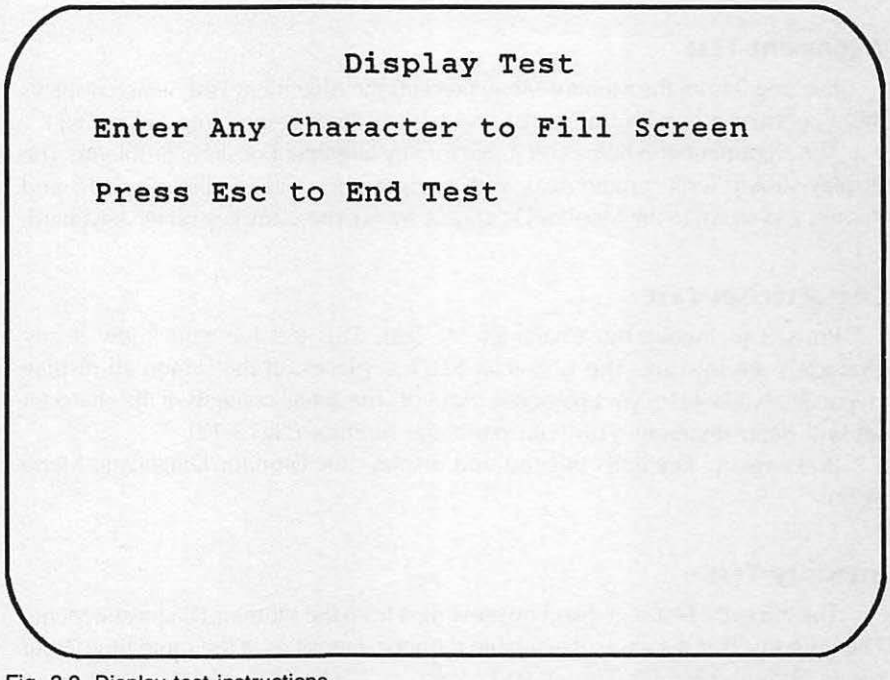


Fig. 3-9. Display test instructions.

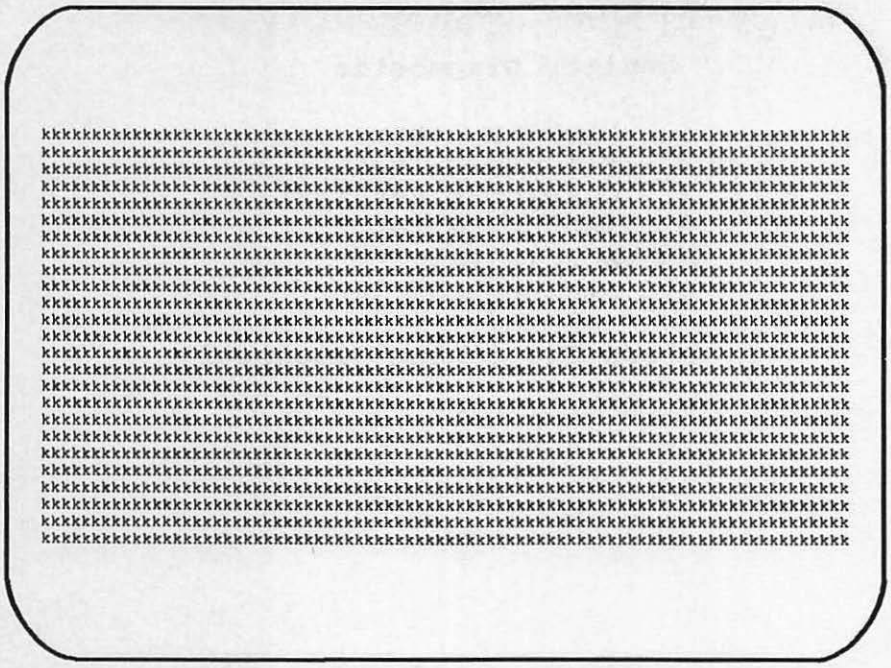


Fig. 3-10. Display test results.

Alignment Test

Pressing 2 from the Monitor Menu presents the Alignment Test, which displays the alignment box with horizontal and vertical lines resembling Figure 3-11.

The alignment grid helps you judge for any alignment or skew problems. The display should look symmetrical, with straight lines and no distortion. To end this test and return to the Monitor Diagnostic Menu, press any key on the keyboard.

Character Set Test

Press 3 to invoke the Character Set Test. This test lets you know if any characters are missing. The Character Set Test places on the screen all display characters available for your particular monitor. The actual contents of the character set will differ depending on your particular monitor (Fig. 3-12).

Pressing any key ends this test and displays the Monitor Diagnostic Menu again.

Intensity Test

The Intensity Test is invoked by pressing 4 from the Monitor Diagnostic Menu. The Intensity Test displays all available monitor intensities at the same time. Your results should resemble Figure 3-13.

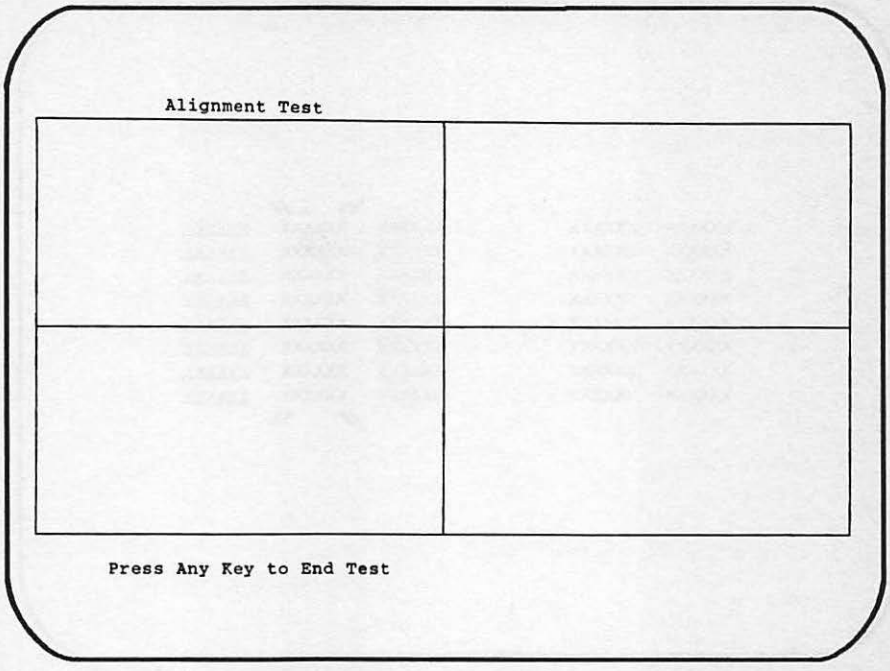


Fig. 3-11. Alignment grid.

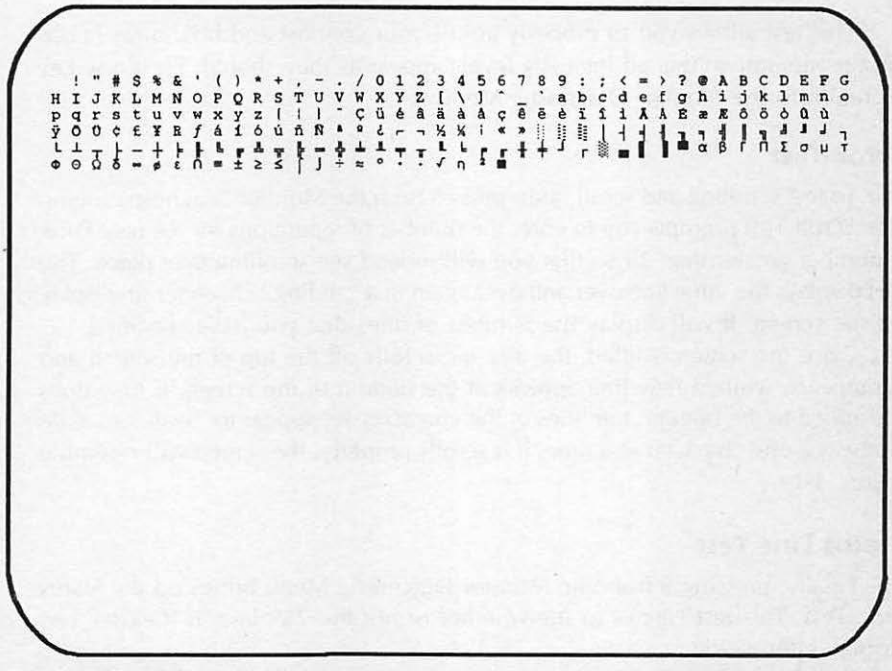


Fig. 3-12. Character set test.

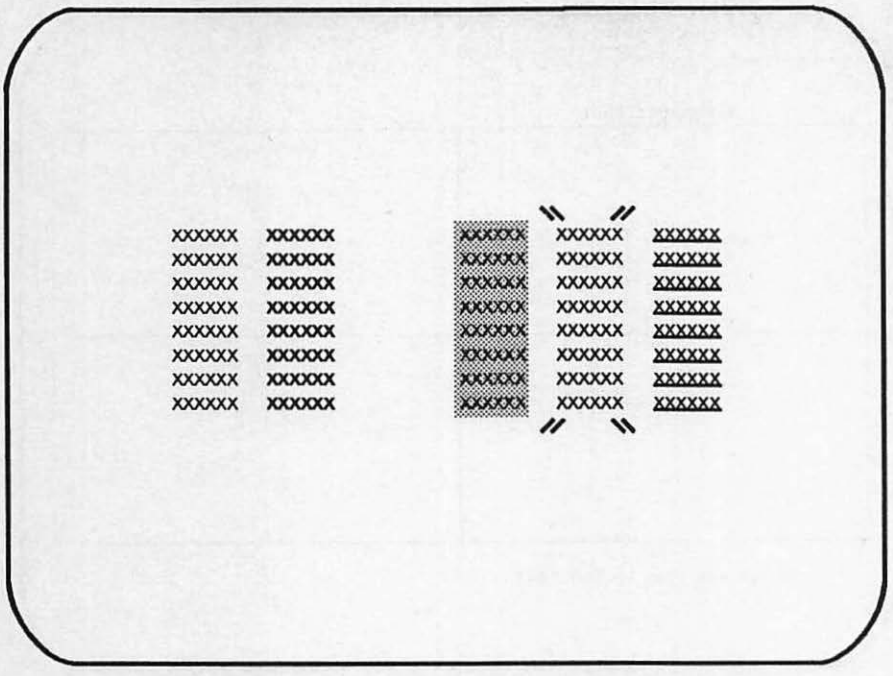


Fig. 3-13. Intensity test results.

This test allows you to properly adjust your contrast and brightness knobs on the monitor so that all intensity levels appear as they should. Press any key to return to the Monitor Diagnostic Menu.

Scroll Test

To test scrolling and scroll rates, press 5 from the Monitor Diagnostic Menu. The Scroll Test prompts you to enter the number of repetitions for the test. Enter a number greater than 25 so that you will indeed see scrolling take place. The test displays the same line over and over again in a "sliding" character set display on the screen. It will display the number of lines that you have specified.

Once the screen is filled, the top line scrolls off the top of the screen and disappears, while a new line appears at the bottom of the screen. If new lines are added to the bottom, the lines of the character set appear to "walk" or slide sideways, one character at a time. If it scrolls properly, the screen will resemble Figure 3-14.

Status Line Test

Finally, pressing 6 from the Monitor Diagnostic Menu brings up the Status Line Test. This test checks to see whether or not the 25th line, if it exists, can display information.

You are prompted to enter a character (function and cursor keys do not work

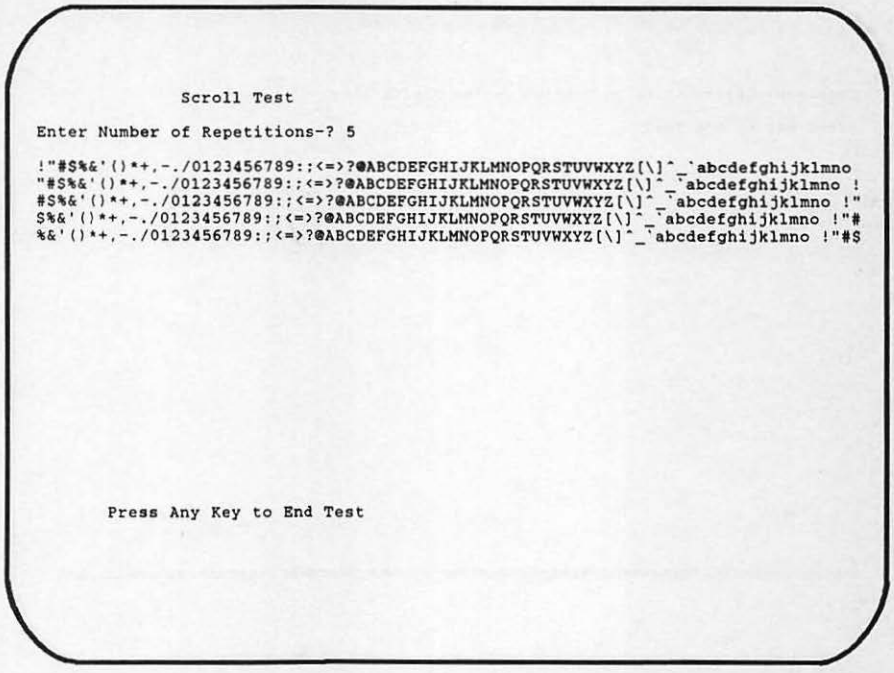


Fig. 3-14. Scroll test results.

for this test). If you have a 25th status line, the character completely fills this line (Fig. 3-15). You may continue the test by entering another key. Press ESC to return to the Monitor Diagnostic Menu.

When you have completed your monitor testing, press ESC to return to the System Diagnostic Main Menu.

HOW THE MONITOR MODULE WORKS

Selecting M (or m) from the System Diagnostic Main Menu causes the program to branch to line 790, which is the start of the Monitor Diagnostic Module (Fig. 3-16).

Six tests are included in this module. Lines 790-810 display the Monitor Diagnostic Menu:

```
790   CLS:REM Monitor Diagnostic Module
800   PRINT TAB(6) "Monitor Diagnostic":PRINT:PRINT TAB(5)
      "1"...Display Test":PRINT TAB(5) "2...Alignment Test":
      PRINT TAB(5) "3...Character Set Test":PRINT TAB(5)
      "4...Intensity Test":PRINT TAB(5) "5...Scroll Test":
      PRINT TAB(5) "6...Status Line Test":PRINT
810   PRINT TAB(5) "Press Esc to End Diagnostic"
```

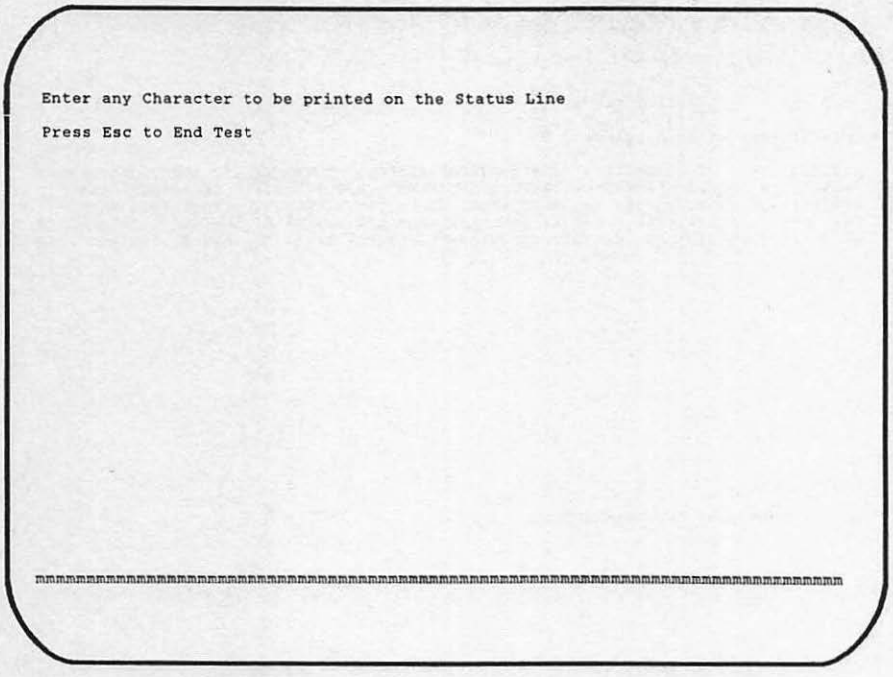


Fig. 3-15. Status line test results.

Line 820 returns to the keyboard input subroutine. The program waits for a keystroke to be entered and places it into A$:

```
820 GOSUB 2930:IF ASC(A$)=27 THEN 40
```

The IF statement checks to see if the ESC key is pressed. If it is, the program branches to line 40, which displays the System Diagnostic Main Menu.

Lines 830 to 880 check for valid keyboard entries (1, 2, 3, 4, 5, 6, ESC). When one of these options is entered, the program branches to the appropriate subtest:

```
830 IF A$="1" THEN 910
840 IF A$="2" THEN 980
850 IF A$="3" THEN 1060
860 IF A$="4" THEN 1140
870 IF A$="5" THEN 1190
880 IF A$="6" THEN 1310
890 BEEP:GOTO 820
```

For example, if A$ is equal to one, the program branches to the Display Test routine starting at line 910. If there is an invalid entry—less than one or greater than six—the routine proceeds to line 890, which causes the beep to sound and returns to line 820 to await another keystroke.

```
 790 CLS:REM Monitor Diagnostic Module
 800 PRINT TAB(6) "Monitor Diagnostic":PRINT:PRINT TAB(5)
     "1"...Display Test":PRINT TAB(5) "2...Alignment Test":
     PRINT TAB(5) "3...Character Set Test":PRINT TAB(5)
     "4...Intensity Test":PRINT TAB(5) "5...Scroll Test":
     PRINT TAB(5) "6...Status Line Test":PRINT
 810 PRINT TAB(5) "Press Esc to End Diagnostic"
 820 GOSUB 2930:IF ASC(A$)=27 THEN 40
 830 IF A$="1" THEN 910
 840 IF A$="2" THEN 980
 850 IF A$="3" THEN 1060
 860 IF A$="4" THEN 1140
 870 IF A$="5" THEN 1190
 880 IF A$="6" THEN 1310
 890 BEEP:GOTO 820
 900 GOTO 40
 910 CLS:PRINT TAB(14) "Display Test":PRINT:PRINT "Enter Any
     Character to Fill Screen":PRINT:PRINT "Press Esc to End
     Test":DEF SEG=&HB000
 920 GOSUB 2930 IF LEN(A$)>1 THEN 920
 930 IF A$=CHR$(27) THEN 960
 940 FOR I=0 TO 3998 STEP 2
 950 POKE I,ASC(A$):NEXT I:IF EX=0 THEN GOTO 920
 960 DEF SEG:IF EX=1 THEN GOTO 1070
 970 GOTO 790
 980 CLS:REM Alignment Test
 990 PRINT TAB(13) "Alignment Test":PRINT CHR$(218);:FOR I=1
     TO 35 :PRINT CHR$(196);:NEXT I:PRINT CHR$(194);:FOR I=1
     TO 35:PRINT CHR$(196);:NEXT I:PRINT CHR$(191)
1000 FOR I=1 TO 10:PRINT CHR$(179),TAB(37) CHR$(179),TAB(73)
     CHR$(179):NEXT I
1010 PRINT CHR$(195);:FOR I=1 TO 35:PRINT CHR$(196);:NEXT
     I:PRINT CHR$(197);:FOR I=1 TO 35:PRINT CHR$(196);:NEXT
     I:PRINT CHR$(180)
1020 FOR I=1 TO 10:PRINT CHR$(179),TAB(37) CHR$(179),TAB(73)
     CHR$(179):NEXT I
1030 PRINT CHR$(192);:FOR I=1 TO 35:PRINT CHR$(196);:NEXT I:
     PRINT CHR$(193);:FOR I=1 TO 35:PRINT CHR$(196);:NEXT I:
     PRINT CHR$(217)
1040 GOSUB 2940:GOTO 790
1050 REM Character Set Test
1060 CLS:PRINT TAB(11) "Character Set Test":PRINT
1070 FOR I=1 TO 255
1080 IF I=7 OR I=9 OR I=10 OR I=11 OR I=12 OR I=13 OR I=28
     OR I=29 OR I=30 OR I=31 THEN GOTO 1100
```

Fig. 3-16. Monitor diagnostic module listing.

```
1090 PRINT CHR$(I)+" ";
1100 NEXT I
1110 IF EX=1 THEN GOTO 1200
1120 GOSUB 2940:GOTO 790
1130 REM Intensity Test
1140 CLS:PRINT "Intensity Test":PRINT:A$="XXXXXX"
1150 FOR I=1 TO 20
1160 PRINT TAB(10) A$;:PRINT "  ";:COLOR 15:PRINT A$;:COLOR
     7:PRINT "  ";:COLOR 0:PRINT A$;:COLOR 0,7:PRINT A$;:
     COLOR 7:PRINT "  ";:COLOR 23,0:PRINT A$;:COLOR 7: PRINT
      "  ";:COLOR 1:PRINT A$:COLOR 7:NEXT I
1170 GOSUB 2940:GOTO 790
1180 REM Scroll Test
1190 CLS:PRINT TAB(15)"Scroll Test":PRINT:INPUT "Enter
     Number of Repetitions-";A:PRINT
1200 N=32
1210 FOR I=1 TO A
1220 N=N+1:IF N>111 THEN N=32
1230 FOR X=1 TO 80
1240 PRINT CHR$(N);:N=N+1
1250 IF N>111 THEN N=32
1260 NEXT X
1270 NEXT I
1280 IF EX=1 THEN RETURN
1290 GOSUB 2940:GOTO 790
1300 REM Status Line Test
1310 CLS:PRINT "Status Line Test":LOCATE 12,1:PRINT "Enter
     any Character to be printed on the Status Line":PRINT:
     PRINT "Press Esc to End Test"
1320 GOSUB 2930:IF A$=CHR$(27) THEN 1350
1330 IF LEN(A$)>1 THEN GOTO 1320
1340 LOCATE 25,1:FOR I=1 TO 80:PRINT A$;:NEXT I:GOTO 1320
1350 GOTO 790
```

Fig. 3-16. Continued.

The Display Test resides in lines 910 through 970. Line 910 clears the screen and prompts you to enter any character to display, or press ESC to end the test. This routine writes directly into the display memory of the MS-DOS systems monochrome display adapter. (The code for CP/M and Macintosh systems directly display the characters using PRINT commands.):

```
910  CLS:PRINT TAB(14) "Display Test":PRINT:PRINT "Enter
     Any Character to Fill Screen":PRINT:PRINT "Press Esc to
     End Test":DEF SEG=&HB000
```

The last command on line 910, DEF SEG=&HB000, points the program to the

beginning memory location of the display map. This command will vary with your type of monitor. Check your documentation.

Line 920 calls up the keyboard input subroutine. Function keys and many of the key combinations are not valid in this test. The statement IF LEN(A$)>1 THEN 920 filters out these invalid functions:

```
920 GOSUB 2930 IF LEN(A$)>1 THEN 920
930 IF A$=CHR$(27) THEN 960
```

Line 930 ends the test if the ESC key is pressed.

Lines 940 through 950 place the valid keyboard entry directly into the display's memory. All 2000 locations (80 columns multiplied by 25 lines) are filled within this loop. The display memory actually has 4000 locations. The odd bytes control the *character attributes* of the corresponding even bytes. Character attributes refer to the different modes in which a character can be displayed. For example, on many monitors, characters can be displayed at high or normal intensity, blinking, underscored, or reverse video (black on white as opposed to white on black):

```
940 FOR I=0 TO 3998 STEP 2
950 POKE I,ASC(A$):NEXT I:IF EX=0 THEN GOTO 920
```

The loop steps in increments of 2, so the program writes only to the even bytes of memory. As a character is written into memory, it displays on the screen.

Line 960 restores the segment pointer to its original data area. Variable EX is checked to see if it is equal to one. If it is, this means the Exerciser Module is in control, and the program branches to the Character Set Test at line 1070:

```
960 DEF SEG:IF EX=1 THEN GOTO 1070
970 GOTO 790
```

If EX is equal to 0, the program proceeds to line 970 and then branches back to the Monitor Diagnostic Menu.

Pressing 2 from the Monitor Diagnostic Menu causes the program to branch to the Alignment Test. Line 980 clears the screen. Lines 990 through 1030 draw the alignment grid on the screen:

```
 980   CLS:REM Alignment Test
 990   PRINT TAB(13) "Alignment Test":PRINT CHR$(218);:FOR I=1
       TO 35 :PRINT CHR$(196);:NEXT I:PRINT CHR$(194);:FOR I=1
       TO 35:PRINT CHR$(196);:NEXT I:PRINT CHR$(191)
1000   FOR I=1 TO 10:PRINT CHR$(179),TAB(37) CHR$(179),
       TAB(73)CHR$(179):NEXT I
1010   PRINT CHR$(195);:FOR I=1 TO 35:PRINT CHR$(196);:NEXT
       I:PRINT CHR$(197);:FOR I=1 TO 35:PRINT CHR$(196);:NEXT
       I:PRINT CHR$(180)
```

```
1020    FOR I=1 TO 10:PRINT CHR$(179),TAB(37) CHR$(179),TAB(73)
        CHR$(179):NEXT I
1030    PRINT CHR$(192);:FOR I=1 TO 35:PRINT CHR$(196);:NEXT I:
        PRINT CHR$(193);:FOR I=1 TO 35:PRINT CHR$(196);:NEXT I:
        PRINT CHR$(217)
```

Notice that the PRINT statements use the CHR$ function instead of actual characters. The numeric values correspond to the ASCII values of the special characters that make up the grid.

CP/M AND MAC USERS: The CP/M version must use different CHR$ values, which are machine-dependent. Refer to your MBASIC programming manual, or your CP/M technical manual for the appropriate values. The Macintosh version actually draws, or plots, the grid with LINE instructions. Refer to Appendix B to see the actual code.

The GOSUB 2940 statement displays a message that prompts you to press any character to end the test. Once you press a key, the routine branches back to display the Monitor Diagnostic Menu:

```
1040 GOSUB 2940:GOTO 790
```

Pressing 3 from the Monitor Diagnostic Menu starts the Character Set Test. This immediately provides a display of the full graphic character set available through your computer's character generator. (However, if you have software that can change fonts and create special characters, these will not be displayed with this test.):

```
1050 REM Character Set Test
1060 CLS:PRINT TAB(11) "Character Set Test":PRINT
1070 FOR I=1 TO 255
```

Most systems can display at least 128 characters, while others, such as the IBM PC, can have as many as 256. After line 1060 clears the screen, line 1070 sets up to loop for 255 repetitions.

Because certain characters within the character set can issue a beep from the computer's speaker, or control the screen in ways you don't want during this test, line 1080 filters out undesirable characters for the purpose of this test:

```
1080    IF I=7 OR I=9 OR I=10 OR I=11 OR I=12 OR I=13 OR I=28
        OR I=29 OR I=30 OR I=31 THEN GOTO 1100
```

Line 1090 takes the FOR loop counter I, and applies it to a CHR$ function. This displays the graphic character to which the number in CHR$ corresponds:

```
1090 PRINT CHR$(I)+" ";
1100 NEXT I
```

The PRINT statement also concatenates a blank beside the character for even spacing.

Line 1110 checks for EX again, to see if it is equal to one. If it is, then the Exerciser Module causes the program to branch to the Scroll Test at line 1200:

```
1110 IF EX=1 THEN GOTO 1200
```

If EX is equal to zero, then the program proceeds to line 1120.

Line 1120 displays the "Press Any Character to End Test" message on line 25, and returns to the Monitor Diagnostic Menu once a key is pressed:

```
1120 GOSUB 2940:GOTO 790
```

If you want to test the monitor's intensity modes, enter 4 from the Monitor Diagnostic Menu. The Intensity Test begins at line 1140. (If you do not have a computer that supports different intensity modes, then you can delete this routine from your diagnostic module, or just ignore it):

```
1130    REM Intensity Test
1140    CLS:PRINT "Intensity Test":PRINT:A$="XXXXXX"
```

Line 1140 clears the screen and displays the test title on the screen. Also note that A$ is set to a value of six Xs.

A FOR-NEXT loop is set up in line 1150 that will display 20 lines of Xs in the different intensity modes. The COLOR statements establish the different modes of display:

```
1150    FOR I=1 TO 20
1160    PRINT TAB(10) A$;:PRINT " ";:COLOR 15:PRINT A$;:COLOR
        7:PRINT " ";:COLOR 0:PRINT A$;:COLOR 0,7:PRINT A$;:
        COLOR 7:PRINT " ";:COLOR 23,0:PRINT A$;:COLOR 7: PRINT
        " " ;:COLOR 1:PRINT A$:COLOR 7:NEXT I
```

If you have a Macintosh, the TEXTMODE instruction is used in place of the COLOR instruction.

Line 1170 displays the "Press Any Character to End Test" message and waits for you to press a key.

```
1170 GOSUB 2940:GOTO 790
```

Once you do this, the program returns to the Monitor Diagnostic Menu.

The Scroll Test is initiated by pressing 5 from the Monitor Diagnostic Menu. The routine starts at line 1190 of the program. This line clears the screen, displays the test title, and prompts you for the number of times that you want the line to repeat. You must enter at least 25 repetitions in order for the screen to scroll at least once. Your entry goes into an INPUT statement.

```
1180    REM Scroll Test
1190    CLS:PRINT TAB(15)"Scroll Test":PRINT:INPUT "Enter
        Number of Repetitions-";A:PRINT
1200    N=32
```

Then, it tests whether the N is equal to 32 in line 1200. N is the CHR$ instruction operand used to display the sliding character line. The number 32 is the ASCII code for a blank space.

The remainder of the code for this test consists of a set of FOR-NEXT loops. The job of the first outer loop is to repeat the line for the number of repetitions that you entered.

```
1210 FOR I=1 TO A
1220 N=N+1:IF N>111 THEN N=32
1230 FOR X=1 TO 80
1240 PRINT CHR$(N);:N=N+1
1250 IF N>111 THEN N=32
1260 NEXT X
1270 NEXT I
```

The second, inner FOR-NEXT loop prints the actual line, "sliding" the display one character over from the previous line.

Line 1280 once again checks to see if the Exerciser Module is in control by seeing if EX is equal to one. If so, the program returns to the Exerciser Module at this point:

```
1280 IF EX=1 THEN RETURN
```

A zero in variable EX causes the program to proceed to line 1290.

Once again, the "Press Any Character to End Test" message is displayed, and the program waits for you to hit a key to return to the Monitor Diagnostic Menu:

```
1290 GOSUB 2940:GOTO 790
```

The last selection on the Monitor Diagnostic Menu tests the status line. Some systems do not support a status line. If that is the case with your system, this routine should be deleted. (From BASIC, type in the line numbers of the lines you wish to delete, then press RETURN.):

```
1300    REM Status Line Test
1310    CLS:PRINT "Status Line Test":LOCATE 12,1:PRINT "Enter
        any Character to be printed on the Status Line":PRINT
        PRINT "Press Esc to End Test"
```

Line 1310 marks the beginning of this test. This line clears the screen and prompts you to enter any character to be displayed in the status line, or an ESC if you want to end the test.

Line 1320 calls the keyboard input subroutine. Then it compares A$ to CHR$(27) to check whether you have pressed the ESC key:

```
1320 GOSUB 2930:IF A$=CHR$(27) THEN 790
```

If you have, the program returns to the Monitor Diagnostic Menu.

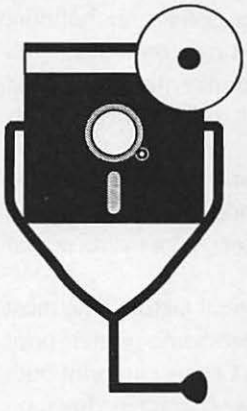
This routine blocks the entry of function and cursor keys. If you enter one of these invalid keys, Line 1330 returns the program to Line 1320 where you can try again:

```
1330 IF LEN(A$)>1 THEN GOTO 1320
```

Line 1340 positions the cursor at the beginning of line 25 and fills the line with the selected character in A$:

```
1340   LOCATE 25,1:FOR I=1 TO 80:PRINT A$;:NEXT I:GOTO 1320
```

Once the display is complete, it loops back to line 1320 to await another character or an ESC to return to the System Diagnostic Main Menu.



Chapter 4

The Printer

Although you get visual computer output from your monitor, you usually need this output in a physical hardcopy form in order to work with the information in more detail, or to present it to others. The printer is the means by which you can make a paper copy of your computer processing results.

DESCRIPTION AND FUNCTION OF THE PRINTER

The printer is like a typewriter without the keyboard. It is ordinarily an optional device to your computer, because the computer does not need a printer in order to run programs and process data. But, if you want to see results on paper instead of just the monitor screen, you must have a printer.

Types of Printers

Printers available on the market today include the dot-matrix printer, the letter-quality printer, the ink-jet printer, and the laser printer. The two most popular printers used with personal computers are the dot-matrix and the letter-quality printers. This being the case, these are the two types of printers covered in this chapter.

The Dot-Matrix Printer. If you were to magnify a character on the monitor screen, you would see that, instead of being made up of solid lines, the character is in fact composed of a series of light dots. Each character is typically composed

of seven dots. The series of dots used to build characters is called a *dot matrix*.

The dot-matrix printer uses the same method as the screen for building characters to print letters on paper. In the printer there is a single print head with an array of small wires. When a character is sent to the printer, this print head punches out the shape of the character through the ink ribbon to create its impression on the paper (Fig. 4-1).

The dot-matrix printer provides output very quickly, and is the least expensive printer available. Speed is measured in characters per second, or *cps*. This speed refers to how fast the printer can lay down characters on paper. A dot-matrix printer can range in speed from 30 to 200 cps, possibly more.

The wide range of possible speeds is attributed to several factors. The most obvious is how fast the computer can feed data to the printer. Some printers print only left to right. These are called *unidirectional* printers. Others can print both left to right and right to left. These *bidirectional* printers lay characters down on a page much faster, because no time is wasted sending the print head back to the "beginning" of a new line.

Some dot-matrix printers offer several "qualities" of print. These might include draft, near-letter-quality, and letter-quality modes. The draft mode might print a minimum number of dots for a character, thus taking the least amount of time. The other modes use multiple print passes to bleed the dots together to make a more solid-looking character. The trade-off for higher quality printing is speed.

In addition to the advantages of speed, the dot-matrix printer is attractive because of its ability to produce an unlimited number of character font styles and point sizes. Certain dot-matrix printers, called graphic printers, can render graphic characters. A graphic printer can address just as many ink points (which correspond to the monitor's pixels) as your monitor resolution can display. This type of printer can reproduce graphic characters, as well as drawings and business graphics. Any shape can be created with a series of dots. This makes the dot-matrix printer highly flexible as to the output that can be created (Fig. 4-2).

The Letter-Quality Printer. Another very popular printer is the letter-quality printer, sometimes referred to as a *daisy-wheel* printer. On such a printer, the output looks as though it were typed on a typewriter.

The letter-quality printer is often preferred by companies and individuals who

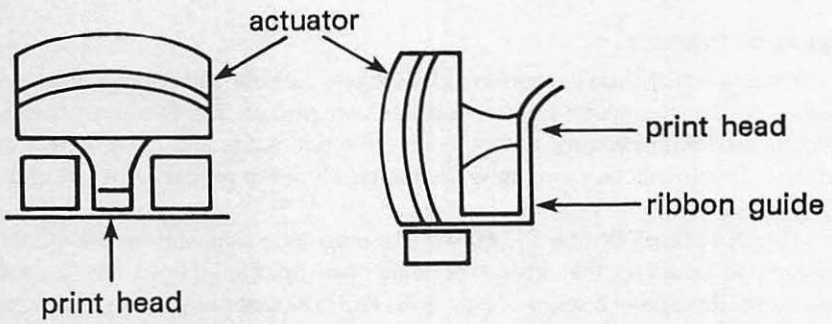


Fig. 4-1. Dot-matrix print head.

```
! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G
H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~
```

```
! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G
H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~
```

```
! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G
H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~
```

Fig. 4-2. Various dot-matrix outputs.

use their computers to do mass mailings. Letters printed on a letter-quality printer appear as though they were individually typed on a typewriter, rather than mass-produced with a computer. In fact, any document that a company or individual wants to look particularly handsome is best done with a letter-quality printer. Letter-quality output simply looks cleaner, with sharper characters than dot-matrix output.

The letter-quality printer uses a "daisy wheel" to create the characters. On the end of each "petal" of the daisy wheel is a character (Fig. 4-3). The daisy wheel includes the entire character set for a given font family and point size. Instead of being formed by a single print head with a series of dots, these characters are individually and solidly embossed into the daisy wheel. A printer motor spins the daisy wheel around. When the desired character spins into position, the print hammer strikes it against the ink ribbon to form the character on the paper. It works on a similar principle to the typewriter, except that the motor-driven printer is much faster.

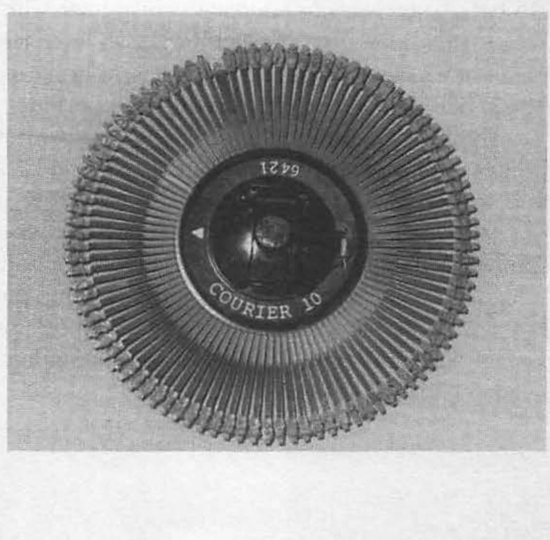


Fig. 4-3. Daisy wheel.

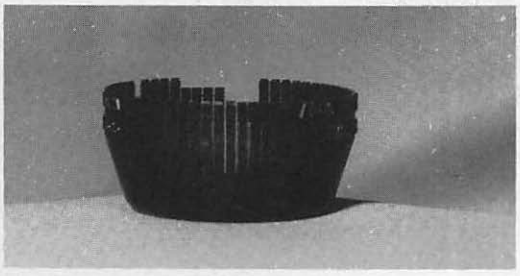


Fig. 4-4. Character cup.

Although the daisy wheel is the most common print element used in letter-quality printers, character "thimbles" and "cups" are also used in some printers (Fig. 4-4). They work on the same principle as the daisy wheel.

The Laser Printer. The laser printer is becoming more and more popular, especially with the advent of desktop publishing applications. The Apple Macintosh offers the Laserwriter printer as an option, and Hewlett-Packard offers the Laserjet.

Using lasers and photographic principles to create a printed image, the laser printer produces output at a resolution of 300 dots per inch. Like the dot-matrix printer, the laser printer uses dots to build characters and graphics, but does it with the dots at a much higher density. The dots are so numerous and close together that the letters look as though they were solidly formed by a typesetter. In fact, the print quality is so much better than letter-quality output that it is often referred to as being "near-typeset" quality.

In addition, the laser printer produces output at a much higher rate of speed. While the speed of dot-matrix and letter-quality printers is rated in characters per second, that of a laser printer is rated in pages per minute.

Printer Interfaces

Your printer is physically connected, or *interfaced*, to your computer through some type of interface board or receptacle which accepts the printer's cable. There are two major ways to interface your printer with your computer: *parallel* and *serial*. The difference between these two interfaces is the manner in which the data is transferred. A parallel interface pushes 7 or 8 bits of data in one transfer. The serial interface sends data one bit at a time, in a "series" (Fig. 4-5).

The parallel interface is generally (though not always) faster. The physical printer itself is rated at a particular speed. In this case, it doesn't matter how fast you send data; it will get there as fast or as slow as the physical printer is rated.

Most computers have two different input/output (I/O) ports available: a Centronix parallel port and a serial RS-232 port. The monochrome monitor interface board contains an integral Centronix parallel interface (Fig. 4-6).

The other method for attaching the printer to your computer is through the serial RS-232 I/O port (Fig. 4-7). Many computers have integral serial I/O ports available on the system.

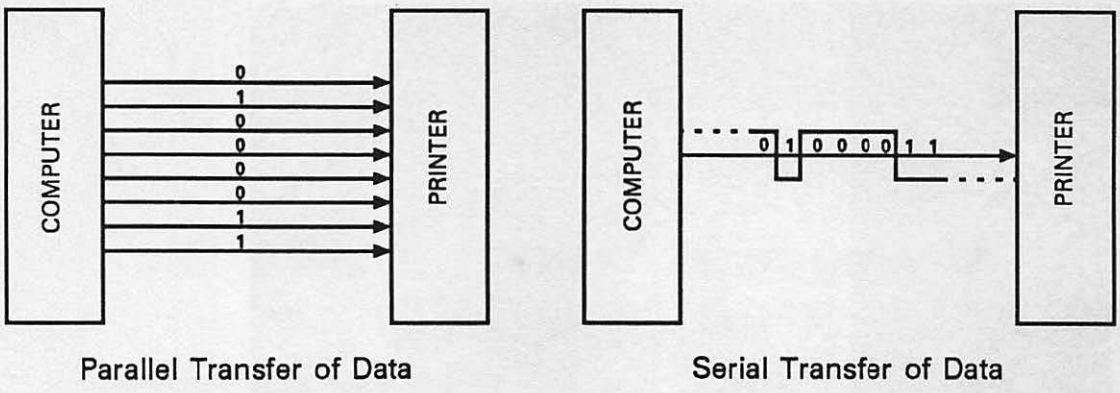


Fig. 4-5. Parallel and serial data transfer.

TROUBLESHOOTING AND REPAIR GUIDELINES

You might encounter a number of problems with your printer, whether it is a dot-matrix or letter-quality printer. The following sections describe these potential problems, indicate how to diagnose them, and describe steps to repair them.

Problem: Printer Does Not Work At All

Suppose you've just bought a new printer, or you've moved your system. Everything is in place, you issue a print command from your application software, but nothing happens at all. There can be several problems associated with this.

☑ SOLUTIONS

☐ First check that power is supplied to the printer. Your printer probably has a power indicator light on the top or front of the chassis. If the printer's

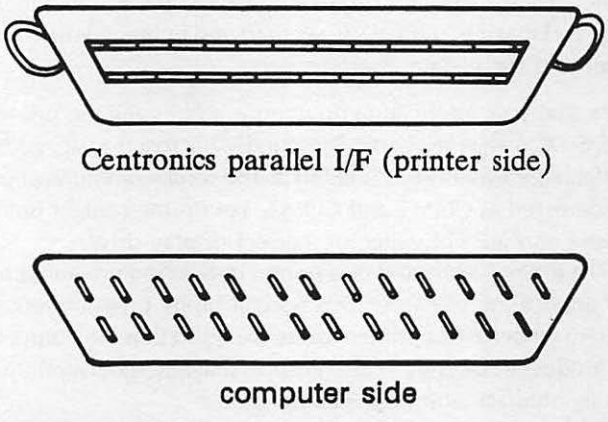


Fig. 4-6. Centronix parallel interface.

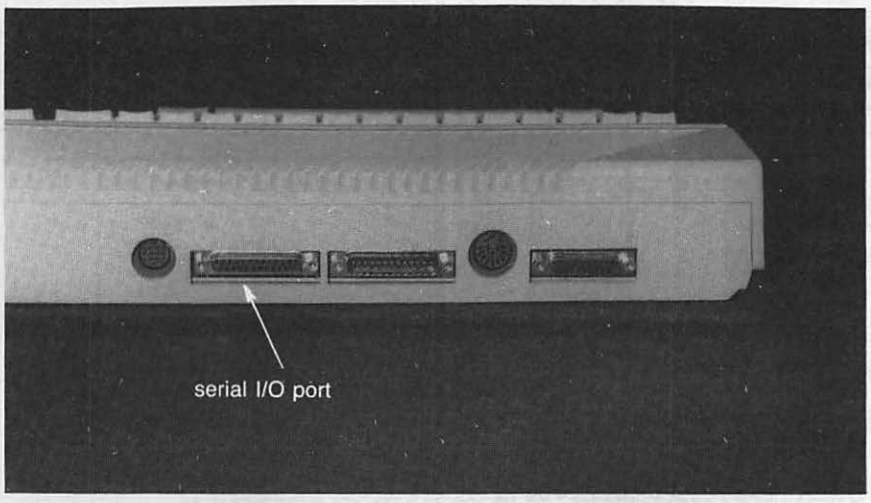


Fig. 4-7. Serial I/O port.

ON/OFF switch is in the ON position, but the indicator is not lit, make sure the printer is properly plugged in. If it's plugged in but still not working, try using a different electrical plug. See whether the plug you're using is controlled by a light switch.

- ☐ Make sure the printer is in the proper mode. Most printers have an "online" mode to indicate that they are ready to receive data from the computer. They might also have a "local" mode, allowing you to physically manipulate the printer: roll the paper, prepare the printer for a different form, or adjust the carriage. To get the printer operating again, switch it back to the "online" mode.

- ☐ Check the cabling. Make sure you have the right kind of cable for the parallel or serial interface. Make sure the cable is inserted into the proper plugs, and that it is sufficiently secured and tightened down between the printer and the system interface port.

- ☐ Check that your application program is addressing the printer correctly. The MS-DOS systems, particularly under Microsoft BASIC, address the two parallel interfaces as LPT1 and LPT2. The serial communication interfaces are addressed as COM1 and COM2. Your printer might not be working because you are not using the correct display driver.

  The driver is activated by a printer installation program provided with your application software. This special utility program lets you tell the program what type of printer you're using, so that the output to the printer is produced correctly. It also ensures that any special printing functions such as boldface and italic can be used.

  The printer installation program provides numerous models from which to choose your own. Even off-brand printers always have an

equivalent to one of these choices. If you do not run your application's printer installation program, the driver might not be installed properly, so that the printer would not work.

- ☐ Certain option switch settings within the printer might be making it impossible to print. Such switch settings might control parameters such as "baud rate" and "word length." Refer to the printer documentation for more information about these switch settings. For more information about what the terms baud rate and word length mean, refer to Chapter 6.

  MS-DOS printer commands default to the parallel printer port, which is addressed as LPT1. If you wish to use the serial printer port, you must use the MODE command to change the printer mode so that the printer commands work correctly in the Printer Diagnostic Module. To do this, place the DOS system diskette in the active disk drive, and at the DOS prompt (A>), enter MODE LPT1=COM1. You might need to set up communication parameters as in this command: MODE COM1: 9600,N,8,1,P. These codes signify a baud rate of 9600, no parity, 8-bit word length, one stop bit, and continuous retry. Refer to your DOS manual for details about this command.

  Rather than typing these lines in every time you boot up your system, key these commands into the file named AUTOEXEC.BAT on your operating system diskette. Then, whenever you boot up the system, these lines are automatically executed for you.

  The Macintosh and CP/M systems usually only have a parallel or a serial printer port, so they do not require these commands.

Problem: Poor Print Quality

One problem you might notice is that the print image quality is poor. While the characters are recognizable, they look blurred, light, washed out. A related problem is that you're printing on a multiple copy form, and while the top copy prints well, subsequent copies print poorly or not at all.

SOLUTIONS

- ☐ Run the Sliding Alpha or Display Print Character Test to obtain sample output. Poor print quality is usually a mechanical rather than a software problem.
- ☐ Check the printer ribbon. It might be jammed or spent. When this happens, the print heads strike the same place on the ribbon, and the ribbon is unable to advance to fresh ink. Unjam the ribbon, or install a new ribbon cartridge.
- ☐ Some printers have a forms adjustment. This adjustment causes the print head to be brought further back from the carriage to accommodate paper of heavier thicknesses, or multi-part forms. Adjust the head so that it is closer to the carriage. Run a print sample again, and adjust the head until the print image is clear and sharp again.

- ☐ Clean the letter-quality daisy wheel or the dot-matrix print head with isopropyl alcohol.
- ☐ If the print quality is still unacceptable after changing the ribbon, adjusting the print head, and cleaning the element, take the printer in for servicing.

Problem: A Horizontal Tab Is Not Working

If you are unable to tab over to a certain position along the carriage, run the Horizontal Tab Test. The asterisks should print in a diagonal across the paper starting at position 1.

SOLUTIONS

- ☐ If it does not print in this diagonal, check whether something is obstructing the platen or the print head. Often simply blowing through the print head, or pulling out a bit of paper hung on it, will solve the problem.
- ☐ If this is not the case, make sure the proper carriage width, 80 or 132, has been designated in the Printer Setup routine.
- ☐ If neither of these solutions take care of the problem, then your printer is due for a service call.

Problem: Paper Does Not Feed Properly

The printer might not feed the paper correctly. Certain commands issued from your application program to the printer can cause the paper to feed several lines, like carriage returns on a typewriter. When you work with special forms like checks and invoices, your application program helps you set up the line spacing to occur in a specific pattern that is more complex than simple single- or double-spacing. In addition, you might establish a *form feed* command at a certain defined point, causing the printer to roll up to the next form.

Problems can occur when the specified vertical spacing or the form feed control do not work as expected. You might not see the specified number of lines between text, or the paper might bunch up and jam as it's rolling to the next page. These types of problems usually occur upon initial printer setup.

SOLUTIONS

- ☐ Run the Line Feed Test to check the carriage control. This test prints lines with variable line spacing and then it feeds the paper up to the next page.
- ☐ If the test does not take place as expected, check that the paper has been fed into the printer correctly.
- ☐ If you're working with continuous-feed paper on a forms tractor, check that the paper is not pulled too tight or left too loose in the tractor clamps (Fig. 4-8).

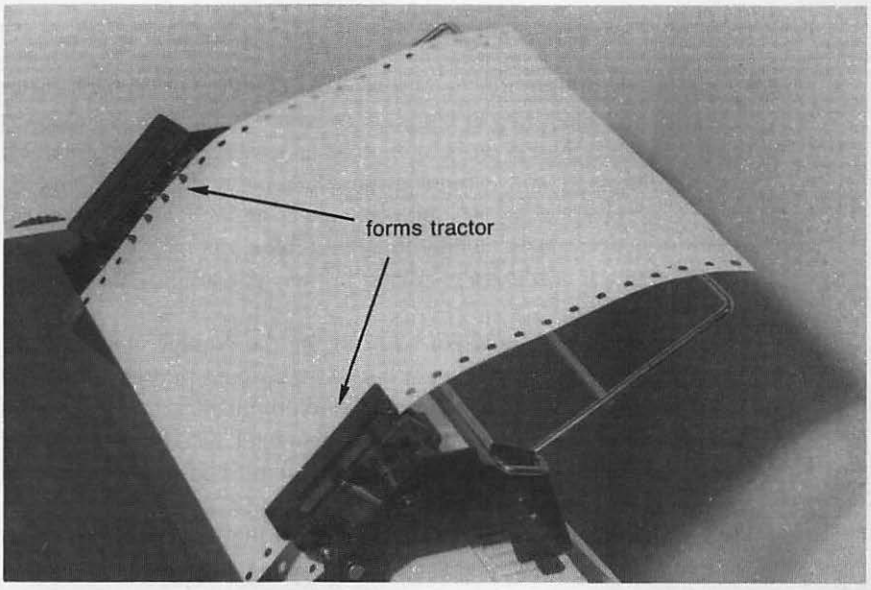


Fig. 4-8. Paper in forms tractor.

- ☐ Check the platen adjustment. The *platen* is the cylinder around which the paper rolls. There is a lever that loosens the platen when rolling the paper in. In order for the paper to automatically feed with the forms tractor, you need to have the platen at its locked position.
- ☐ Check for a scrap of paper or other foreign object that might have lodged into the platen or into the area where the paper feeds.
- ☐ If you suspect that the printer is inserting extra spaces between lines, run the Sliding Alpha Test. The output should appear single-spaced. If there are blank line spaces between the printed lines, there might be a faulty option switch setting that's causing the extra line feeds. Check your printer documentation for more information about the switch settings.
- ☐ If none of these suggestions repair the line spacing problem, take your printer in for adjustment.

Problem: Certain Characters Do Not Print

Sometimes a certain character does not print. The character is there on the screen, and you know it's available within your printer's character set. However, when you request printer output, the character does not form properly, or you see a black box or blank space where the character is supposed to be. This means different things, depending on whether you have a dot-matrix printer or a letter-quality printer.

SOLUTIONS

- ☐ **For a dot-matrix printer**, run the Display Character Print or Echo Character Test. If you see a black box or blank space where a character is supposed to be, then your printer is probably not set up to accommodate graphics.

  As we discussed earlier, there are several types of dot-matrix printers. One type accommodates graphic characters as well as the standard alphanumeric characters, while the other type operates only in the character mode. If you use a character printer, you are not able to print graphic characters.

  It might be confusing when you can see the graphic characters on your monitor display, but can't get it to print on your printer. Just as the different brands of computers support different character sets, different brands of printers also support different character sets. Some character sets are more limited than others. As such, you must be sure to send only valid characters to your printer.

  You might have an alternate character set that is activated with an ESC sequence. These ESC sequences need to be conveyed to your printer to tell it when to use this alternate character set. Refer to your printer documentation for information on what the ESC sequences are, and how to activate them to your printer.

- ☐ **With a letter-quality printer**, the daisy wheels are limited in size and can usually only accommodate about 100 characters, upper- and lower-case. Because of this, it is most likely that your standard daisy wheel does not include any special graphic characters. (However, you can get special graphic daisy wheels consisting of such characters.)

- ☐ If it's not graphic characters, but regular alphanumeric characters you're having trouble with on your letter-quality printer, your problem is of an entirely different nature.

  For example, if the letter "a" does not appear wherever it's supposed to, run the Sliding Alpha or Echo Character Test to see if you're missing both the upper-case "A" and the lower-case "a." If so, you know that the physical character position on your daisy wheel or cup has broken off its stem. Daisy wheels and cups are fairly fragile and characters can break after a period of time (Fig. 4-9). Remove the daisy wheel and examine it to confirm that it is indeed damaged. Go out and buy yourself a new daisy wheel, and you're in business again.

Problem: Certain Characters Do Not Print Completely

A potential dot-matrix printer problem is that while all the characters do print, they do not print completely. In this case, it would not be just one or two individual characters that have this problem, but the entire character set. You might see part of the "a," part of the "r," part of the "m," with a segment from each character missing in the same location, such as the upper left or lower right corner (Fig. 4-10).

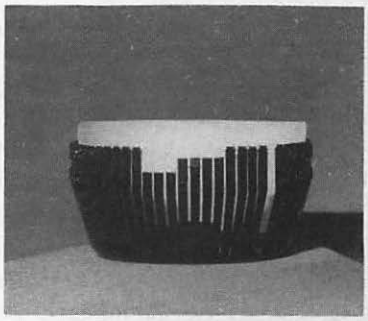


Fig. 4-9. Broken character print cup.

SOLUTIONS

- ☐ First check the ribbon. It might be folded down or spent. Replace the ribbon and run the Sliding Alpha or Display Character Print Test to print sample output.
- ☐ If the ribbon is in good condition, then there is a problem in the print head. One of the solenoid motors that drives the dot matrix pins in the head has probably malfunctioned or worn out, and that's why you're seeing partially formed characters. You'll need to take the printer in for servicing to have this solenoid motor repaired or replaced.

Installing a New Printer

The majority of printer problems occur upon initial printer installation. This is because there are several variables that must be taken into account when installing a new printer.

For instance, suppose you have been producing hardcopies of your software processing results on an IBM dot-matrix printer with no problem whatsoever. The day comes when you decide to upgrade to a letter-quality printer, for example, an NEC Pinwriter. You take the old printer out and plug the new printer into your computer, and think you're finished. But when you run your application and attempt to print your results, you discover that either the printer does not print at all, or you get unexpected results. This happens because, as far as your computer system is concerned, the new printer is a completely foreign device. As such, the computer and new printer do not know how to effectively communicate with one another.

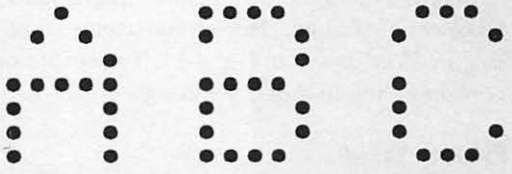


Fig. 4-10. Partial characters printed.

Particular rules, or *protocols*, must be followed in order for the various system components to communicate and send data back and forth. The protocol that the old printer understood might be quite different from the protocol that your new printer understands.

To help alleviate this problem, many of the application programs you use will have a program called a "device driver" that is established for a particular printer. A device driver is a program that recognizes a special peripheral device such as a letter-quality printer. Many applications, especially word processors, need to know the exact printer model so that special functions such as underlining and boldface can be handled correctly.

The device driver lets you specify which printer is being used, thus enabling the program to communicate correctly with the printer. Refer to your application software user manual for more information on configuring the device driver.

When connecting your printer to your computer, the printer interface board must be properly configured for your particular printer. You must also ensure that any software program that uses the printer is set up to the proper interface. This information should be provided with your various user documentation; the computer system, the printer, and the software documentation are all vital for configuration.

Another consideration when installing a new printer is the interface port. This can be one of the most confusing aspects about installing a new printer. As mentioned in the beginning of this chapter, most computer systems have two different interface ports available: the parallel port and the serial port. Some printers can only be attached to parallel ports; others only to serial ports. Many can be attached to either type. The type you can use depends on the type of interface you have in your system. For example, if you have the IBM monochrome display interface with the integral parallel printer interface, the printer can be plugged into the parallel interface. If you're using a multifunction board that has several parallel ports, several serial ports, and additional memory on the board, be sure the printer is plugged into the correct interface. Check your system technical manual and printer documentation to find out which port you should use.

You might encounter other problems when setting up a new printer. Again, read the printer documentation thoroughly before installing. It might help to consult other documentation, including the computer system manual section on printers and interfaces, and the user guide for any software programs with which you're going to use the printer.

RUNNING THE PRINTER DIAGNOSTIC MODULE

To run the Printer Diagnostic Module, follow the instructions provided in "Running the Diagnostic Program" in Chapter 1. Then press P to initiate the Printer Diagnostic Module. The screen clears, and the Printer Diagnostic Menu is displayed as shown in Fig. 4-11. To activate one of these printer tests, press the corresponding number, 1 through 6.

Printer Setup

There are many types of printers, and different printers have different

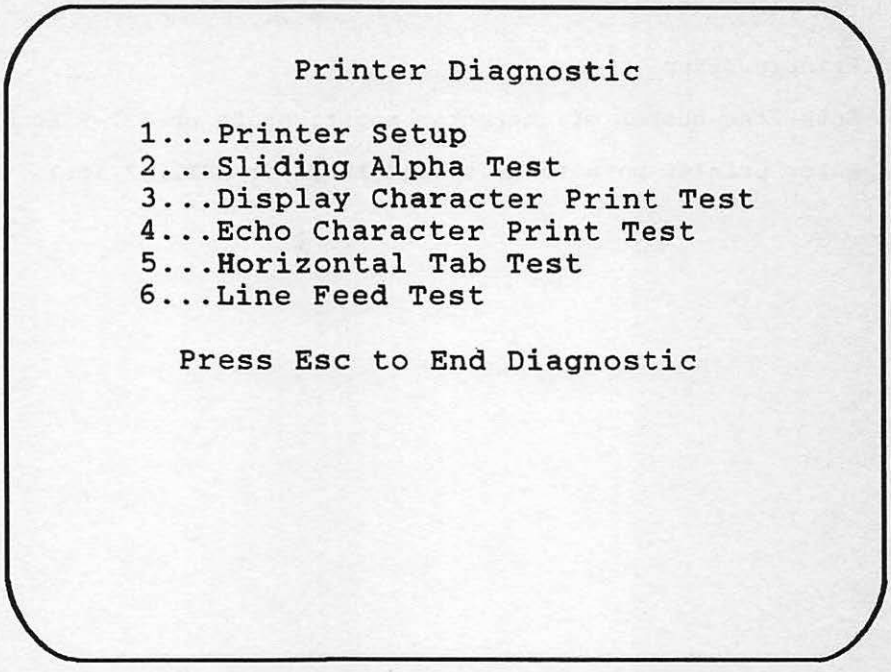


Fig. 4-11. Printer diagnostic menu.

characteristics. Your printer might print 80 characters across the page, or it might have an extended carriage with the ability to accommodate 132 characters in one line.

In order to produce accurate results, the tests in the Printer Diagnostic Module need to know whether the data is being sent in a parallel or serial manner from the computer to the printer. And if there is more than one interface port available on the system, the printer needs to know which port the printer is on.

To inform the Printer Diagnostic Module what your printer variables are, press 1 to begin the Printer Setup routine and define your particular printer as to its carriage length and its interface port. Do this before trying any of the printer tests. You will be able to specify the parameters of your own printer, thus allowing you to "customize" the Printer Diagnostic Module.

The Printer Setup routine displays two questions about your printer (Fig. 4-12). The first question asks you to specify your printer's line length. Your printer might support an 80-column print mode on 8½-inch wide paper, or you might have more flexibility with a printer that can output 132 characters across on wider paper. The Printer Diagnostic setup default is for an 80-character line length. If you have a 132-character printer carriage, enter 132 and press RETURN. If you enter any number other than 132, the system automatically defaults to 80.

The second question asks you to specify which port, or interface board, your printer is connected to. In the standard IBM configuration, the printer is attached to the monochrome monitor interface board. This board includes a parallel printer port. The operating system program code addresses this printer port by the name

```
Printer Setup

Enter the number of character positions 80 or 132-? 80

Enter printer port to be tested (LPT1 or LPT2)-? lpt1
```

Fig. 4-12. Printer setup routine.

"LPT1." You should recognize these port addresses from when you first installed the printer. Check your system documentation.

Instead of having it connected to a monochrome monitor interface, you might have your printer attached to a multi-function interface board which supports an "LPT2" or "LPT3" designation.

Other systems might use the serial communication port to drive a printer. The serial communication port is the same one used to connect a modem to a computer system. The program code addresses this port by the "COM1" or "COM2" designation. If yours is such a system, be sure to run the MS-DOS MODE routine described in "Troubleshooting and Repair Guidelines" earlier in this chapter. This causes the system to read a "COM1" designation as "LPT1," for example.

The port designation default in the Printer Setup routine is LPT1. If you have a different port designation, enter it in at this point and press RETURN. The Printer Diagnostic Module adapts accordingly.

If your printer has an 80-character carriage width, and is attached to the first parallel interface of LPT1, then you never need to set printer parameters for the Printer Diagnostic Module, because these are the defaults. However, if your printer is different from these defaults, you must run the Printer Setup routine each time you wish to use the Printer Diagnostic Module.

After you answer the setup questions and press RETURN, the Printer Diagnostic Menu is displayed once again. At this point, the module is set up for your printer. You can go on to use any of the following printer tests.

Sliding Alpha Test

The printer Sliding Alpha Test is similar to the sliding alpha routine used in the Scroll Test of the Monitor Diagnostic Module. On the printer this test serves two functions: to fully exercise the printer, and to get a quick look at the full alphanumeric display character set available for your printer.

When you press 2, you are prompted to enter the number of lines you wish to run. Enter the desired number and press RETURN. The printer begins printing a sliding alpha pattern. It prints a line full of as many different characters of the character set as will fit on a line. It then feeds up another line, slides one character to the left, and prints the same line of characters again. At the carriage limit, it again feeds another line, slides another character to the left, and prints the line of characters. The program continues to do this for your specified number of lines (Fig. 4-13).

If nothing happens when you try to run this test, or if you get results that do not look similar to Fig. 4-13, refer to the "Troubleshooting and Repair Guidelines" section of this chapter.

Display Character Print Test

The Display Character Print Test displays all characters available within your printer's character set, and is used to check whether all characters are printing satisfactorily.

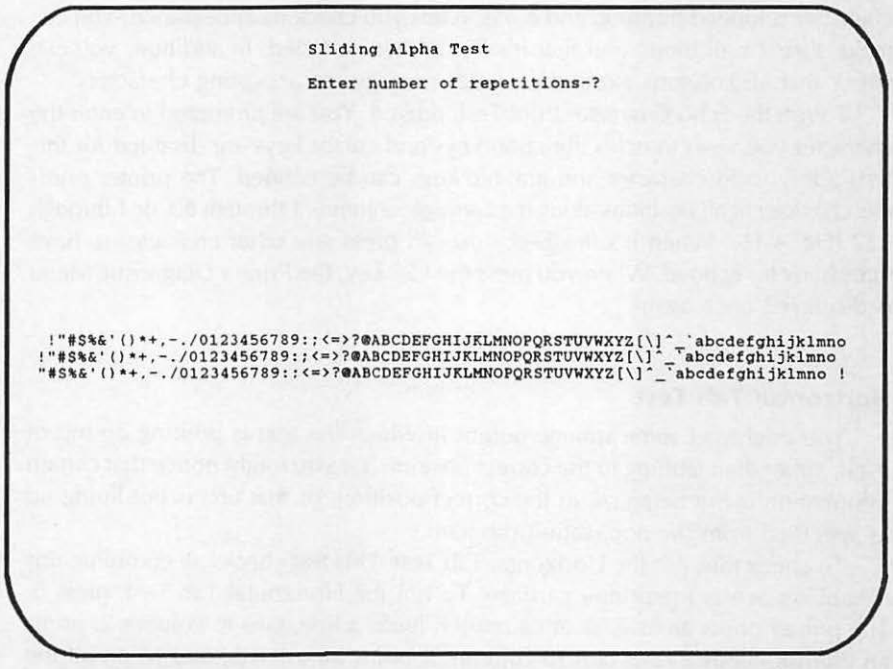


Fig. 4-13. Sliding alpha test results.

This particular test is highly printer-dependent, and currently supports the character set of the IBM generic character printer. If you use a different printer, you will probably need to modify this test's program code in order for it to work. The section in this chapter entitled "How the Printer Module Works" points out the particular line numbers in the code that would need to be modified.

Graphics characters are not supported in the Display Character Print Test. If you have a graphics printer, refer to the printer documentation to learn about the ESC sequences used to access your printer's special graphic characters. Then follow the guidelines within the Printer Diagnostic Module's code to program these ESC sequences into the module.

To run the Display Character Print Test, simply press 3. All characters in the character set will be printed (Fig. 4-14). After printing the character set, the Printer Diagnostic Menu is displayed once again.

```
 ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G
H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _ ` a b c d e f g h i j k l m n
p q r s t u v w x y z { | } ~
```

Fig. 4-14. Display character print test.

Echo Character Print Test

The Echo Character Print Test lets you enter a character and have it immediately printed, or *echoed*, on the printer. You can echo a given character across all 80 or 132 columns of your printer. This test lets you ensure that a certain character is indeed printing, and if it is, it lets you check its appearance. You can make sure it's all there, and that it's not broken or faded. In addition, you can check that all columns along the printer carriage are accepting characters.

To run the Echo Character Print Test, press 4. You are prompted to enter the character you wish to echo (function keys and cursor keys are disabled for this test). Only valid character and graphic keys can be echoed. The printer prints this character at all positions along the carriage, columns 1 through 80, or 1 through 132 (Fig. 4-15). When it's finished, you can press any other character to have that character echoed. When you press the ESC key, the Printer Diagnostic Menu is displayed once again.

Horizontal Tab Test

You might get some strange output in which the text is printing on top of itself, rather than tabbing to the correct position. Or you might notice that certain information is not being set in the correct position, or that text is not lining up as specified from the application program.

To check this, run the Horizontal Tab Test. This test checks all combinations of tabbing across the printer carriage. To run the Horizontal Tab Test, press 5. The printer prints an asterisk at column 1, feeds a line, tabs to column 2, prints an asterisk, feeds a line, tabs to column 3, prints an asterisk, and so on all the way through to column 80 or 132. In this way, every horizontal tab position is

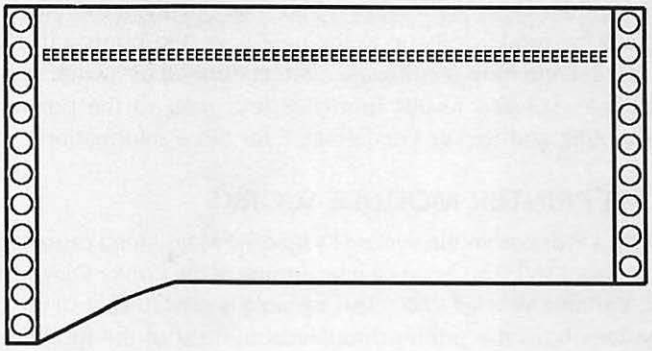


Fig. 4-15. Echo character print test results.

checked. Your printed result will be a diagonal line of asterisks running from the upper left corner to the bottom right corner of your paper (Fig. 4-16).

If you do not get such a result, there might be something wrong with your printer mechanism, or your printer might need a switch setting adjusted. Refer to "General Troubleshooting and Repair Guidelines" earlier in this chapter for further information.

Line Feed Test

When you have line feed problems, you might see text printing on top of previous lines because the printer failed to feed a line where it was supposed to. The printer might jam whenever it is supposed to feed a number of lines, or go to the next page with a top-of-form command.

The Line Feed Test checks the printer's vertical line spacing, and is selected by pressing 6 from the Printer Diagnostic Menu. Like the Horizontal Tab Test, this test needs no additional parameters entered, and runs as soon as you select it. The printer types the first line. Then it gives two line feeds, types the second line, gives three line feeds, types a third line, gives four line feeds, types a fourth

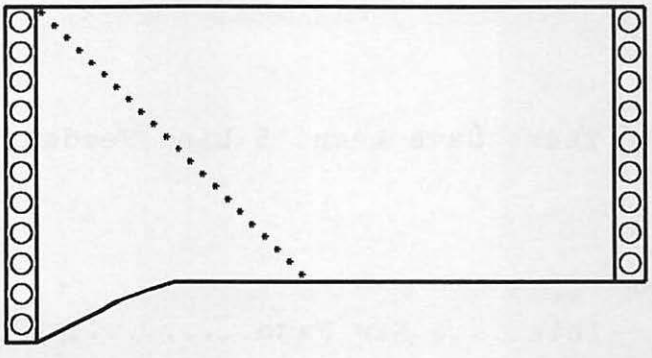


Fig. 4-16. Horizontal tab test results.

line, gives five line feeds, then types a fifth line. Finally, a form-feed command is issued, and the printer rolls up to the next page and prints a line (Fig. 4-17).

Line spacing and form feeding are often controlled by switch settings. If you do not get the expected results from this test, refer to the previous section, "Troubleshooting and Repair Guidelines," for more information.

HOW THE PRINTER MODULE WORKS

Selecting a P or p from the System Diagnostic Main Menu causes the program to branch to line 1360. This line is the beginning of the Printer Diagnostic Module (Fig. 4-18). Variable W is set to 80. This variable is used to control the line length, or carriage length, of the printer throughout the rest of the module.

Line 1370 clears the screen, and together with line 1380 displays the Printer Diagnostic Menu:

```
1370    CLS:PRINT TAB(11) "Printer Diagnostic":PRINT:PRINT TAB(3)
        "1...Printer Setup":PRINT TAB(3) "2...Sliding Alpha Test":PRINT
        TAB(3) "3...Display Character Print Test":PRINT TAB(3) "4...Echo
        Character Print Test"
1380    PRINT TAB(3) "5...Horizontal Tab Test":PRINT TAB(3) "6...Line Feed
        Test":PRINT:PRINT TAB(5) "Press Esc to End Diagnostic"
```

```
This is the First Line.........

There have been  2 Line Feeds


There have been  3 Line Feeds


There have been  4 Line Feeds



There have been  5 Line Feeds



This is a New Page...........
```

Fig. 4-17. Line feed test results.

```
1360 W=80:REM Printer Diagnostic Module
1370 CLS:PRINT TAB(11) "Printer Diagnostic":PRINT:PRINT
     TAB(3) "1...Printer Setup":PRINT TAB(3) "2...Sliding
     Alpha Test":PRINT TAB(3) "3...Display Character Print
     Test":PRINT TAB(3) "4...Echo Character Print Test"
1380 PRINT TAB(3) "5...Horizontal Tab Test":PRINT TAB(3)
     "6...Line Feed Test":PRINT:PRINT TAB(5) "Press Esc to
     End Diagnostic"
1390 GOSUB 2930:IF ASC(A$)<>27 THEN 1410
1400 CLOSE #1:GOTO 40
1410 IF A$="1" THEN 1490
1420 IF A$="2" THEN 1580
1430 IF A$="3" THEN 1700
1440 IF A$="4" THEN 1800
1450 IF A$="5" THEN 1890
1460 IF A$="6" THEN 1950
1470 BEEP:GOTO 1390
1480 REM Printer Setup Routine
1490 CLS:PRINT "Printer Setup":PRINT
1500 INPUT "Enter the number of character positions, 80 or
     132-";W
1510 IF W<>80 OR W<>132 THEN W=80
1520 PRINT:INPUT "Enter printer port to be tested (LPT1 or
     LPT2)-";PR$
1530 IF PR$<>"LPT2" or PR$<>"lpt2" THEN PR$="lpt1"
1540 PR$=PR$+":"
1550 OPEN PR$ AS #1
1560 WIDTH #1,W:GOTO 1370
1570 REM Sliding Alpha Test
1580 CLS:PRINT "Sliding Alpha Test":PRINT:INPUT "Enter
     number of repetitions-";X
1590 N=31
1600 FOR I=1 TO X
1610 L$="":N=N+1:IF N>111 THEN N=32
1620 FOR A=1 TO W
1630 L$=L$+CHR$(N):N=N+1
1640 IF N>111 THEN N=32
1650 NEXT A
1660 PRINT #1,L$;:NEXT I
1670 IF EX=1 THEN GOTO 1700
1680 GOTO 1360
1690 REM Display Character Print Test
1700 CLS:PRINT "Display Character Print Test":L$="";N=1
1710 REM Enter the pertinent ESC Sequence for the Selection
```

Fig. 4-18. Printer diagnostic module listing.

```
     of Graphics Here Example: PRINT #1, CHR$(28); "I";
     CHR$(1); for the NEC Pinwriter
1720 FOR I= 32 TO 126
1730 IF N=W THEN PRINT #1,L$:N=1:L$="":GOTO 1750
1740 L$=L$+CHR$(I)+" ":N=N+1
1750 NEXT I
1760 IF L$<>"" THEN PRINT #1,L$
1770 IF EX=1 THEN GOTO 1820
1780 GOTO 1360
1790 REM Echo Character Print Test
1800 CLS: PRINT "Echo Character Print Test":PRINT:PRINT
     "Enter Character to Echo":PRINT:PRINT "Press Esc to End
     Test"
1810 GOSUB 2930:IF ASC(A$)=27 THEN GOTO 1360
1820 L$=""
1830 IF LEN(A$)>1 THEN GOTO 1810
1840 FOR I=1 TO W
1850 L$=L$+A$:NEXT I
1860 PRINT #1,L$:IF EX=1 THEN GOTO 1890
1870 GOTO 1810
1880 REM Horizontal Tab Test
1890 CLS:PRINT "Horizontal Tab Test"
1900 FOR I=1 TO W
1910 PRINT #1,TAB(I) "*":NEXT I
1920 IF EX=1 THEN GOTO 1950
1930 GOTO 1360
1940 REM Line Feed Test
1950 CLS:PRINT "Line Feed Test"
1960 PRINT #1 "This is the First Line.....";
1970 FOR I=2 TO 5
1980 FOR A=1 TO I
1990 PRINT #1,CHR$(10);:NEXT A
2000 N$=STR$(I)
2010 PRINT #1,"There have been "+N$+" Line Feeds";:NEXT I
2020 PRINT #1,CHR$(12);:PRINT #1 "This is a New Page........"
2030 IF EX=1 THEN RETURN
2040 GOTO 1360
```

Fig. 4-18. Continued.

Line 1390 calls up the keyboard input subroutine, and waits for you to press a key:

```
1390     GOSUB 2930:IF ASC(A$)< >27 THEN 1410
```

If you press the ESC key (ASCII value 27), the program proceeds to line 1400,

which closes the printer and returns to the System Diagnostic Main Menu:

```
1400    CLOSE #1:GOTO 40
```

When you press a key other than ESC, the program branches to lines 1410 through 1460:

```
1410    IF A$="1" THEN 1490
1420    IF A$="2" THEN 1580
1430    IF A$="3" THEN 1700
1440    IF A$="4" THEN 1800
1450    IF A$="5" THEN 1890
1460    IF A$="6" THEN 1950
```

These particular instructions check to see if you have a value of from one to six in variable A$. If you pressed 1, for example, the program branches to line 1490, the beginning of the Printer Setup routine.

CP/M and MAC USERS: The Macintosh and CP/M versions of this program test only the LPT1 port. The LPRINT command is used rather than the PRINT #1 L$ command.

If you press something other than keys one through six, the program falls through to line 1470, sounding the beep to alert you to the fact that you have entered an invalid key.

```
1470    BEEP:GOTO 1390
```

The program returns to line 1390, at which point you can try again with a valid key.

When you press 1 at line 1390, the program branches to line 1490:

```
1480    REM Printer Setup Routine
1490    CLS:PRINT "Printer Setup":PRINT
```

Line 1490 is the beginning of the Printer Setup Routine. This line clears the screen and displays the test title.

Line 1500 is an INPUT statement asking you to enter the number of character positions for your printer:

```
1500    INPUT "Enter the number of character positions 80 or 132-";W
1510    IF W< >80 OR W< >132 THEN W=80
```

The program accepts a value for the variable W. Line 1510 ensures that W defaults to 80. If you have a printer with a 132-position carriage, then enter 132 and press RETURN.

The next setup parameter checked is for the printer port that you wish to test. Line 1520 asks which printer port, LPT1 or LPT2, is to be tested.

CP/M and MAC USERS: This test is not incorporated in the Macintosh or CP/M versions of the System Diagnostic Program, because they typically support only the LPT1 printer port:

```
1520    PRINT:INPUT "Enter printer port to be tested (LPT1 or LPT2)-";PR$
```

Line 1530 ensures that only LPT1 or LPT2 are entered into variable PR$. The default is LPT1:

```
1530    IF PR$< >"LPT2" or PR$< >"lpt2" THEN PR$="lpt1"
```

Line 1540 adds the required colon (:) to the variable string PR$ for use in the OPEN statement in line 1550:

```
1540    PR$=PR$+":"
```

Line 1550 opens variable PR$, which is the printer, LPT1 or LPT2, as File #1:

```
1550    OPEN PR$ AS #1
```

CP/M and MAC USERS: The Macintosh and CP/M versions of the code do not use line 1550.

Line 1560 sets the width of LPT1 or LPT2 to the desired carriage size, W:

```
1560    WIDTH #1,W:GOTO 1370
```

If you entered 132, the width of LPT1 or LPT2 is set to 132 for the rest of the diagnostic. After completing this, the program branches back to Line 1370.

If the program had returned to Line 1360, it would have reset W to the default of 80 columns, thus negating what was just done in this section.

When you press 2, the program branches to line 1580 for the Sliding Alpha Test:

```
1570    REM Sliding Alpha Test
1580    CLS:PRINT "Sliding Alpha Test":PRINT:INPUT "Enter number of
        repetitions-";X
```

Line 1580 clears the screen and displays the Sliding Alpha Test title. The INPUT statement asks for the number of repetitions to print. This number goes into variable X.

Variable N holds the ASCII value of the sliding alpha routine, and in line 1590, N is set to 31. This is one less than 32, the ASCII value for a blank space. Line 1600 is the beginning of a FOR-NEXT loop:

```
1600    FOR I=1 TO X
```

This determines the number of lines to be printed, so it is written as FOR 1 TO X; X being the number of repetitions.

Within this FOR-NEXT loop, L$ is set to null, and 1 is added to N:

```
1610    L$=" ":N=N+1:IF N>111 THEN N=32
```

N starts out as 31, and then becomes 32, the ASCII code for the blank space character. If N is greater than 111, N is reset back to 32. The characters with ASCII values between 32 and 111 are printed. Once 112 is reached, the IF statement causes N to return to 32. This prevents invalid characters, such as ESC sequences, from being printed, and also causes the string to slide.

Another FOR-NEXT loop begins at line 1620:

```
1620    FOR A=1 TO W
1630    L$=L$+CHR$(N):N=N+1
```

This builds the actual line to be printed. A character determined by N is added to L$. This character starts out again as ASCII value 32, the blank space character. Then one is added to N to get the ASCII value for the next character.

Line 1640 begins a test similar to the test done back in line 1610:

```
1640    IF N>111 THEN N=32
1650    NEXT A
```

Again, if N is greater than 111, N is reset to ASCII value 32, the blank space character. NEXT A causes the characters to fill an entire line.

Once all the characters have built an entire line based on W, the width, the program proceeds to line 1660:

```
1660    PRINT #1,L$;:NEXT I
```

This causes the output of L$ to go to the printer instead of to the monitor. The semicolon prevents a line feed. NEXT I means that one of the repetitions has been carried out. This routine prints the lines for the specified number of repetitions.

The number of repetitions entered into the INPUT statement for X is satisfied in this manner. When complete, the program goes to line 1670, which checks to see if variable EX is equal to one. If it is, then the Exerciser Module is in control, and the program branches to line 1700. If EX is equal to zero, then the program proceeds to line 1680, which causes the program to branch back to display the Printer Diagnostic Menu:

```
1670    IF EX=1 THEN GOTO 1700
1680    GOTO 1360
```

When 3 is pressed, the program goes to the Display Character Print Test starting at line 1690:

```
1690    REM Display Character Print Test
1700    CLS:PRINT "Display Character Print Test":L$=" ";N=1
```

Line 1700 clears the screen and then displays the test title. As before, L$ is used to build the print line, and is set to null. N is set to one.

Line 1710 is a REMARK statement that shows how to set up code for accessing an alternate character set that your printer might be able to render. The commands will be specific to your printer:

```
1710    REM Enter the pertinent ESC Sequence for the Selection of Graphics
        Here Example: PRINT #1,CHR$(28),"I"; CHR$(1); for the NEC
        Pinwriter
```

If your printer supports a graphic character set, the code for sending its particular ESC sequence should be inserted into this line.

Line 1720 is a FOR-TO loop that cycles through values 32 to 126. These are the values for the standard ASCII display codes (blank space, A through Z, 1 through 0, etc.):

```
1720    FOR I= 32 TO 126
```

Line 1730 checks to see if variable N is equal to the carriage width W:

```
1730    IF N=W THEN PRINT #1,L$:N=1:L$=" ":GOTO 1750
1740    L$=L$+CHR$(I)+" ":N=N+1
1750    NEXT I
```

If N=W, a full line has been built, and L$ is output to the printer. In this case, the line feed is not suppressed as in the previous test, so the test is double-spaced. N is reset to one. The program skips to line 1750, the NEXT I statement.

If N is not equal to W at line 1130, it falls through to 1740, which concatenates the next character onto the L$ and adds one to N.

This loop continues until all 128 positions have been printed. A full line is not built for the part of the character set, so L$ must be emptied after the FOR-NEXT loop. Line 1760 accomplishes this:

```
1760    IF L$< >" " THEN PRINT #1,L$
```

Line 1770 checks to see if EX is equal to one, which means the Exerciser Module is in effect. If it is equal to zero, the program proceeds to line 1780:

```
1770    IF EX=1 THEN GOTO 1820
```

At line 1780, the program branches back to the Printer Diagnostic Menu:

```
1780    GOTO 1360
```

When you press 4 from the Printer Diagnostic Menu, the program branches

to line 1800 for the Echo Character Print Test:

```
1790    REM Echo Character Print Test
1800    CLS: PRINT "Echo Character Print Test":PRINT:PRINT "Enter
        Character to Echo":PRINT:PRINT "Press Esc to End Test"
```

At 1800, the screen is cleared, the test name is printed, and you are prompted to enter the character to be echoed on the printer. You are also reminded to press the ESC key when you wish to end the test.

Line 1810 calls up the keyboard input subroutine at line 2930:

```
1810    GOSUB 2930:IF ASC(A$)=27 THEN GOTO 1360
```

As before, if the value of A$ is equal to ASCII value 27 (the ESC key), it branches back to the Printer Diagnostic Menu.

Line 1820 sets L$ to a null value:

```
1820    L$=" "
```

Line 1830 is activated when you press a key to be echoed, and it checks to make sure it is a valid alphanumeric key:

```
1830    IF LEN(A$)>1 THEN GOTO 1810
```

Function keys, cursor keys, and other non-graphic characters are suppressed for this test. These special keys have a two-character length in A$, while valid keys have a one-character length. The length check ensures that A$ contains a valid key. If it does not, the program branches back to 1810 to let you try again.

Once you press a valid key to be echoed, the program falls to 1840, which is a FOR-NEXT loop:

```
1840    FOR I=1 TO W
```

As before, W, the length of the carriage, is used as the parameter for how far the character is to be printed.

Line 1850 takes the character you have entered and joins it with L$ for the length of the carriage:

```
1850    L$=L$+A$:NEXT I
```

Once all positions in the carriage width are filled, the program proceeds to line 1860:

```
1860    PRINT #1,L$:IF EX=1 THEN GOTO 1890
```

Line 1860 outputs L$ to the printer. The balance of the line checks once again to see if the Exerciser Module is in control. If it is not, the program proceeds to line 1870:

```
1870    GOTO 1810
```

Line 1870 returns the program to line 1810, where you can enter another character to be echoed, or you can press ESC to end this test and return to the Printer Diagnostic Menu.

Pressing 5 from the Printer Diagnostic Menu branches the program to the Horizontal Tab Test starting at line 1890:

```
1880    REM Horizontal Tab Test
1890    CLS:PRINT "Horizontal Tab Test"
```

The screen is cleared and the test title is displayed on the screen.

Line 1900 begins another FOR-NEXT loop for the length of W, the printer carriage length:

```
1900    FOR I=1 TO W
```

Line 1910 prints a series of asterisks using the TAB statement, which causes horizontal tabbing:

```
1910    PRINT #1,TAB(I) "*":NEXT I
```

The TAB statement uses the FOR-NEXT loop's variable I as the tab value. It prints an asterisk and then loops back to do it again for the next value of variable I. This causes the asterisks to be printed in the stairstepped row.

After printing this diagonal row of asterisks, the program proceeds to line 1920, which checks for variable EX being equal to one. If it is not, the program proceeds to line 1930, which returns the program to the Printer Diagnostic Menu:

```
1920    IF EX=1 THEN GOTO 1950
1930    GOTO 1360
```

The final test is activated by pressing 6. This takes the program to Line 1950 for the Line Feed Test:

```
1940    REM Line Feed Test
1950    CLS:PRINT "Line Feed Test"
```

The screen is cleared, and the test title is displayed.

Line 1960 prints the first-line message:

```
1960    PRINT #1 "This is the First Line.....";
```

Line 1970 begins one of two FOR-NEXT loops:

```
1970     FOR I=2 TO 5
```

Line 1970 causes vertical line spacing with lines 2 through 5.

Line 1980 begins a FOR-NEXT loop that has the printer producing blank lines: simple carriage returns:

```
1980     FOR A=1 TO I
1990     PRINT #1,CHR$(10);:NEXT A
```

This causes the printing to space the desired number of lines. On the first iteration, a 2 will be printed, followed by two line feeds.

The program falls through to Line 1345, which sets N$ equal to the string value of I:

```
2000     N$=STR$(I)
```

This creates orderly output for Line 2010, which is the message indicating the number of lines you have tabbed:

```
2010     PRINT #1,"There have been "+N$+" Line Feeds";:NEXT I
```

The program then goes to the NEXT statement. This sequence repeats four more times.

When I is equal to five, the program proceeds to Line 2020:

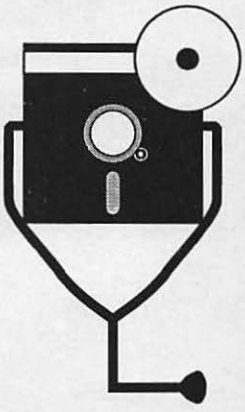
```
2020     PRINT #1,CHR$(12);:PRINT #1 "This is a New Page....."
```

CHR$(12) in line 2020 causes the printer to issue a form feed. The words "This is a New Page" are printed at the top of the next page of the printer.

After this is completed, line 2030 checks to see if EX is equal to one. If it is not, the program proceeds to Line 2040:

```
2030     IF EX=1 THEN RETURN
2040     GOTO 1360
```

This causes the program to branch back to line 805, returning the program to the Printer Diagnostic Menu.



Chapter 5
The Disk Drive

Every computer system unit is rated for a certain capacity of data it can store at one time. This capacity might be 64, 128, 256, 512, 640 or 1000 kilobytes of data. This capacity is not large enough to hold software programs and all of your data. To provide additional storage for these purposes, you have the disk drive.

DESCRIPTION AND FUNCTION OF THE DISK DRIVE

The disk drive provides you with the means for auxiliary high-speed storage for programs and data, and as such is one of the most useful devices you can have on your computer. With the disk drive, you can quickly store and access programs such as spreadsheets, database, and word processing applications. The disk drive also lets you store the data on which these programs operate: the numbers and table data in the spreadsheet, the field and record data in the database, and the words in the word processor (Fig. 5-1).

Throughout this chapter, the terms "disk," "diskette," "disk drive," and "drive" are used. "Disk" is the generic term for the disk media on which the data is written. This can be either a floppy diskette or a hard disk. "Diskette" refers to the floppy diskette media. "Disk drive" and "drive" are both generic terms for the disk drive, whether it is a floppy or hard disk drive.

Types of Disk Drives

The floppy disk drive might be an integral part of the system unit, built into the system and sharing its power supply. This is the case with the IBM PC and

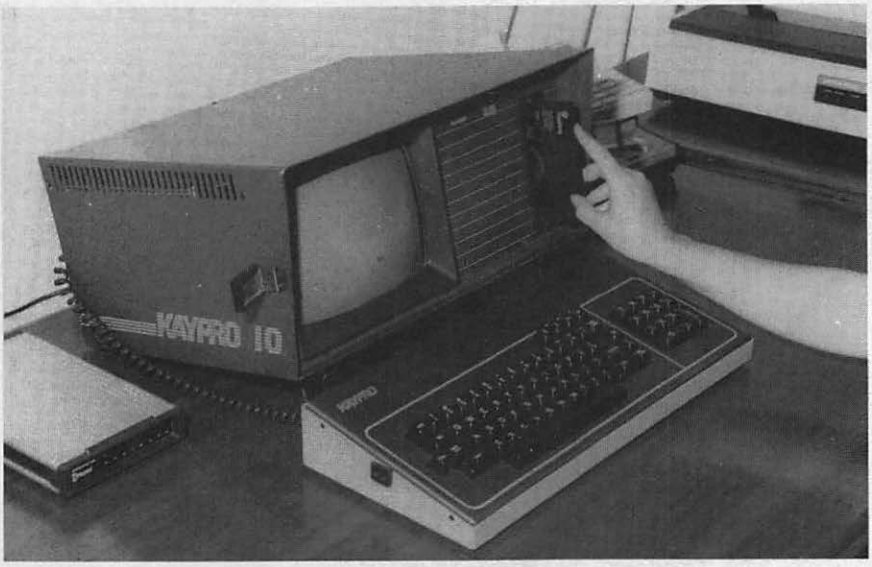


Fig. 5-1. User inserting diskette into disk drive.

the Macintosh. Or, you might have an external disk drive, which resides in its own housing and uses its own power supply (Fig. 5-2).

In addition to integral and external disk drives, disk drives differ in terms of their size and the technology used. A brief history and explanation of disk drives

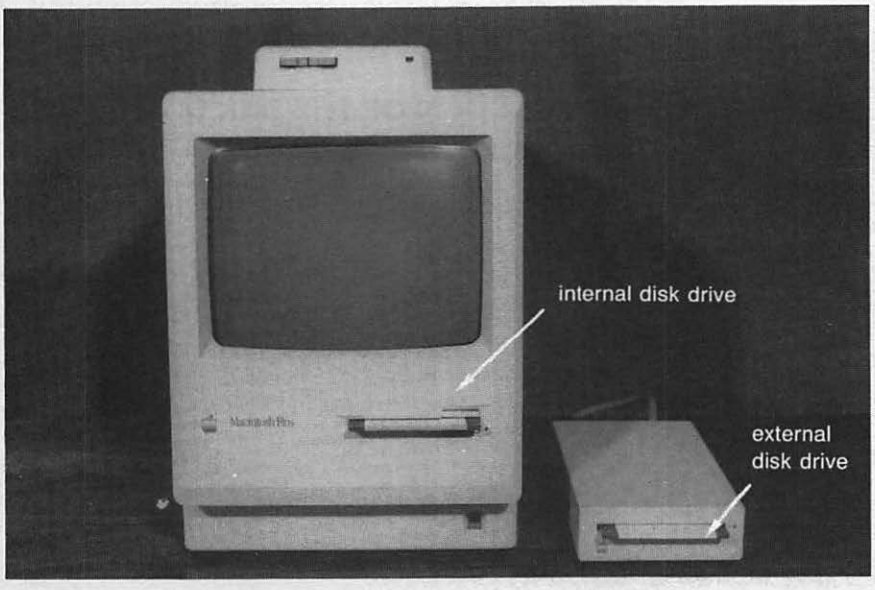


Fig. 5-2. Internal and external disk drives.

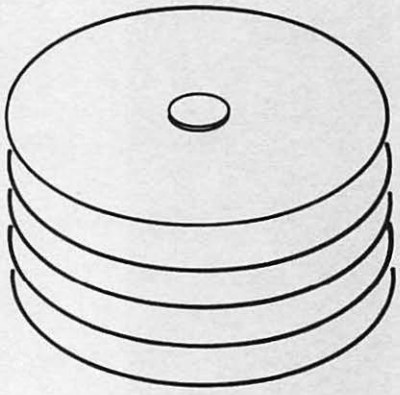


Fig. 5-3. Fourteen-inch mass storage platters.

and diskettes might help you understand the background behind the various physical types of disk drives.

The first computers with disk drives were the mainframe computers used in large corporations. The storage media used were 14-inch metal platters (Fig. 5-3). These rigid platters, or disks, were stacked like phonograph records in the mainframe's disk drive.

When the personal computer was invented in the seventies, a new kind of storage disk was introduced: the eight-inch floppy diskette (Fig. 5-4). They were called "floppy," because, unlike their metal predecessors, the recording media consisted of a smaller platter made of plastic and enclosed in a protective cardboard casing. The microcomputer had miniaturized everything about the computer, including the disk drive.

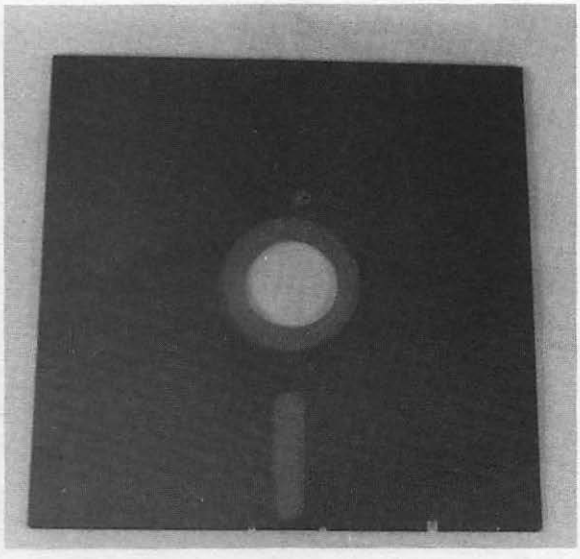


Fig. 5-4. Eight-inch floppy diskette.

Fig. 5-5. 5¼-inch floppy diskette.

As the microcomputer's components became more powerful, they also became more compact. Soon the 5¼-inch floppy diskette with its 5¼-inch disk drive came into vogue (Fig. 5-5). These diskettes were smaller and easier to use than the more unwieldy eight-inch versions. A more recent phenomenon made popular by Apple Computer has been the 3½-inch diskette (Fig. 5-6). Instead of a cardboard enclosure for protection, this diskette has a hard plastic covering. It is a more durable diskette so the contents of the diskette are reliable for a longer period of time.

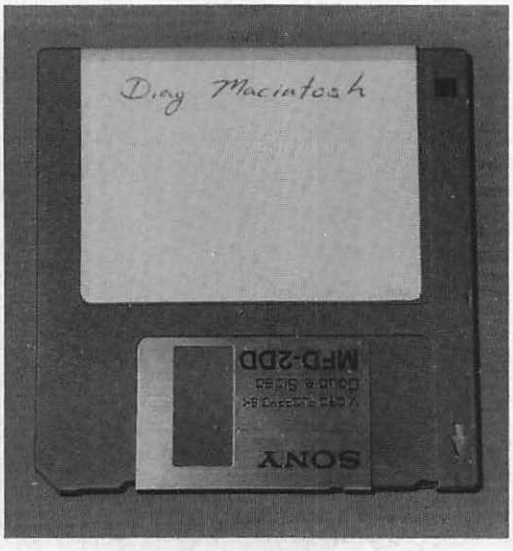


Fig. 5-6. 3½-inch floppy diskette.

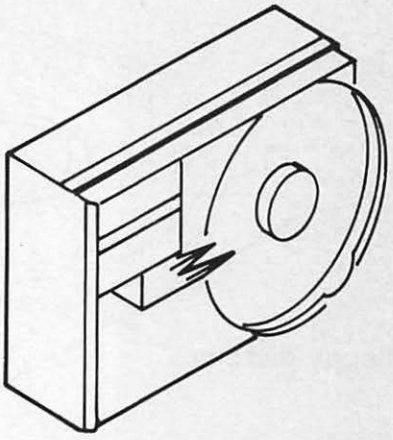


Fig. 5-7. Hard disk.

The hard disk is another type of disk drive that has become extremely popular, especially in applications requiring a lot of memory. The hard disk, also known as the winchester disk, is similar to the mainframe's rigid platter disk on a smaller scale. The hard disk is completely enclosed and the platters are sealed as a separate assembly within the system unit enclosure (Fig. 5-7). This prevents dirt from getting into the mechanical parts and physical surfaces of the disk and drive. This is very important, because a spec of dust or dirt can cause severe malfunctions and loss of data.

Another major advantage of the hard disk is its capacity. One of the original eight-inch diskettes could hold a maximum of 100,000 bytes of information, one byte being approximately one character. The 5¼-inch diskettes can hold between 100,000 bytes and 1,000,000 bytes, depending on how advanced the system is, and whether the disk drive can write on one or two sides of the diskette. The 3½-inch diskette can hold between 400,000 bytes and 800,000 bytes.

Contrasting with the floppy diskette capacities, the hard disk can hold between five million bytes (five megabytes) and 120 million bytes of information (120 megabytes).

How a Disk Drive Functions

The disk drive is attached to the computer system unit by means of the disk interface board. If a system includes a floppy disk drive as well as a hard disk drive, they will each have separate controller interface boards. Each controller board can support from one to four disk drives.

The disk drive includes a motor which spins the disk at a particular rate of speed. Floppy disk drives spin the diskette at 300 to 400 revolutions per minute (rpm). Hard disks, on the other hand, rotate between 2,400 and 4,700 rpm, thus accounting for their higher access speeds. The disk drive also contains a head-arm assembly which serves to locate, read, and write data to and from the disk according to the computer's program instructions (Fig. 5-8).

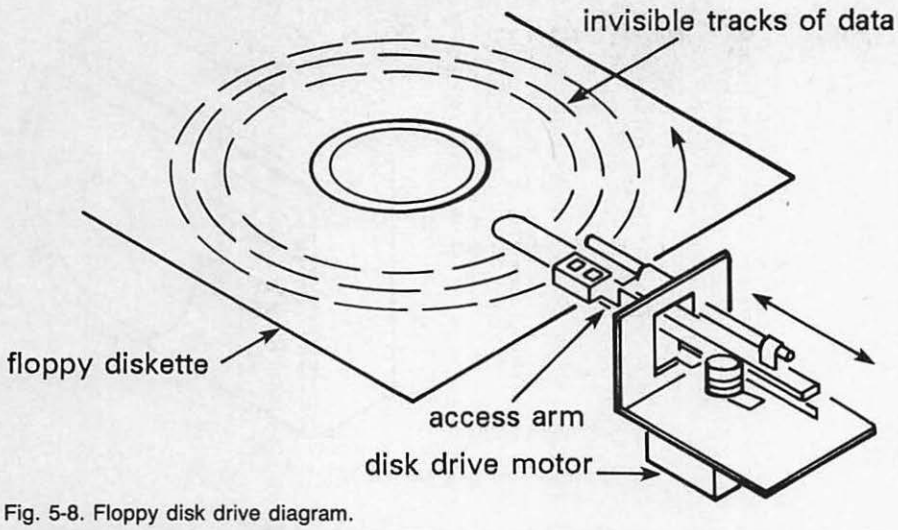


Fig. 5-8. Floppy disk drive diagram.

A diskette is a magnetic recording media made with iron oxide, similar to the material used on cassette tapes for your stereo's cassette player. The magnetic material completely covers the surface of the diskette like an emulsion, or film. It is into this material that data are written, stored, and read magnetically (Fig. 5-9).

The floppy diskette always has a means for "write-protection." The 5¼-inch diskette has a notch in its side called the "write-protect notch." As long as the

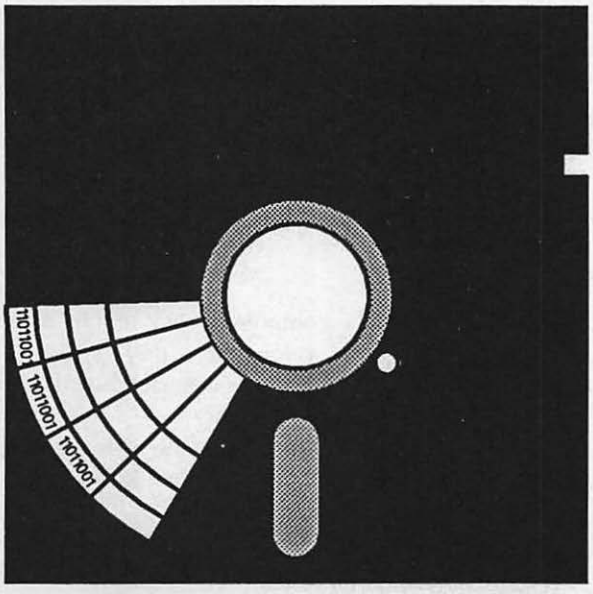


Fig. 5-9. Cross-section of diskette.

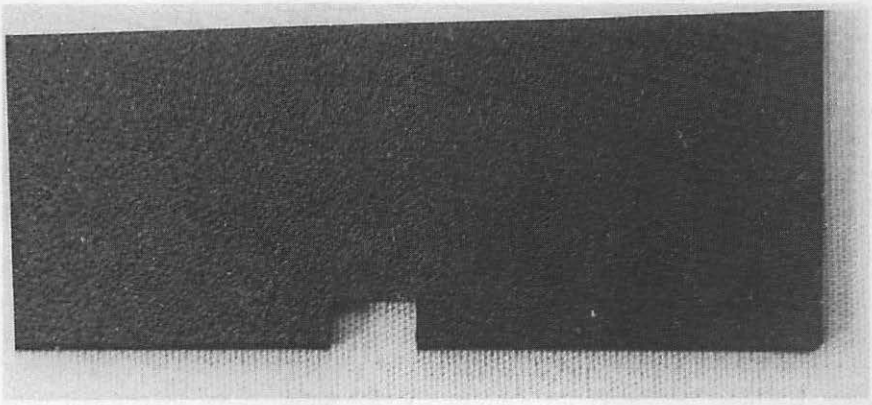


Fig. 5-10. 5¼-inch diskette write-protect notch.

write-protect notch is exposed, the disk drive is able to write information onto the disk as well as read from it (Fig. 5-10). You'll notice that when you buy a new box of diskettes, it includes a package of adhesive foil tapes. When you place one of these tapes over a diskette's write-protect notch, you "turn off" the disk drive's ability to write information onto the diskette. The tape blocks the write-logic sensor in the disk drive, and no data can be written onto this diskette.

A form of write-protection used on the 3½-inch diskette is a sliding tab located in the upper left corner of the diskette (Fig. 5-11). It works on the same principle as the write-protect notch. When the tab is down, data can be written on the diskette. When the tab is up, the diskette is write-protected. Write-protection is a means for protecting the information already on the diskette.

When you insert your diskette into the disk drive and close the drive latch, a switch inside the drive is automatically thrown, telling the system that a diskette has been inserted. The drive motor comes up to speed and spins the diskette.

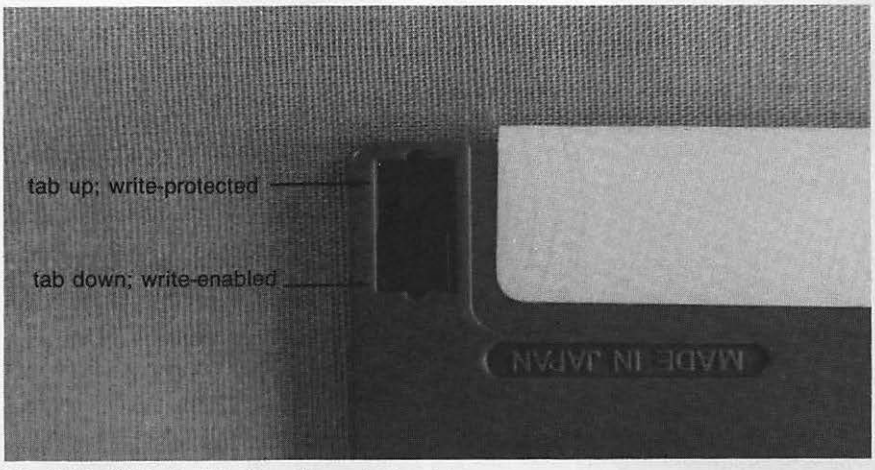


Fig. 5-11. 3½-inch diskette write-protect notch.

If you were to look near the center of the diskette where it fits onto the hub of the disk drive, you would see a small hole. This hole is used for timing and synchronization. The diskette must be spinning at about 300 rpm to be in a ready state for reading or writing data. This timing and synchronization hole is used as a means for sensing when the platter is spinning at the correct speed. This hole is also used as a reference point, marking the start point for information to be written on the diskette. When the disk drive comes up to speed, which takes about a second, it is ready to read or write data.

Within the disk drive unit is a mechanism referred to as the *head*. The head is a magnetic device on the end of a mechanical access arm. The head can magnetize the diskette emulsion, thereby writing new data onto the diskette. It can also sense a magnetic pattern already in the emulsion, thereby reading existing data. The head itself never physically touches the surface of the diskette as it reads or writes. Instead, it rides a few microns above it on a cushion of air. Software and hardware logic is used to position the head at the proper location for reading or writing.

The diskette, like a phonograph record or a compact laser disk, stores specific information in specific areas. Just as the laser disk might have different songs on twelve different tracks, for example, a diskette has information stored on various "tracks" and "sectors."

The diskette is divided into a number of concentric circles going in toward its hub (Fig. 5-12). Each concentric circle is a *track*. There are 40 tracks on a 5¼-inch disk drive, and 80 tracks on a 400 kilobyte 3½-inch disk drive. Each track is divided into *sectors* (Fig. 5-13).

Exactly one sector's worth of information is transferred in any given read or write operation. Your program might need to manipulate only 20 characters of that sector, but the computer always transfers all the information in the entire

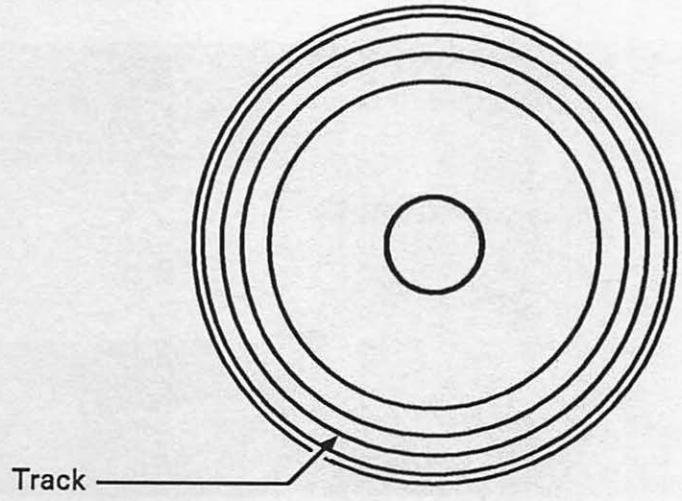


Fig. 5-12. Diskette tracks.

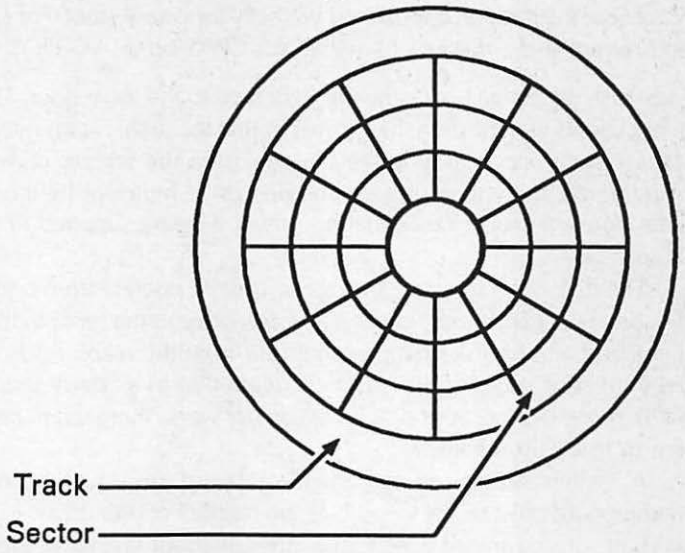


Fig. 5-13. Tracks and sectors.

sector. One sector on the IBM PC contains 512 characters. A typical CP/M sector contains 128 or 256 characters.

When a read command is issued by the operating system or the application program, it tells the disk drive to position the head at the proper track and sector. The head copies the data from the sector and transfers it to a disk buffer. The program logic extracts the record needed, and it is used as directed by the program.

A write command points to a disk buffer that contains the data to be written. The head is positioned over the proper track and sector, and then the data are transferred onto the diskette.

TROUBLESHOOTING AND REPAIR GUIDELINES

Problems with your disk drives are dependent upon the type of disk drive you have, as well as on the particular configuration of your disk drives.

The Disk Drive Does Not Work

You might insert a diskette into a disk drive and try to read from or write to it, but nothing happens.

SOLUTIONS

- ☐ Check how you inserted the diskette. If it's inserted wrong, the drive cannot read from or write to it. For horizontal disk drives, the diskette label should be face up. For vertical disk drives, the diskette label should face toward the left.

- ☐ Make sure the diskette is formatted correctly for your system. For example, you cannot use a diskette formatted for CP/M on an MS-DOS system.

- ☐ Most disk drives have an indicator light next to the drive door. This light either shows that the drive has power or that the drive is currently active. If the system boots up, you see the cursor on the screen, perhaps you hear the fan, but you do not see the disk drive indicator light come on, then there's a good chance that no power is being supplied to the disk drive.

  The disk drive interface board can usually operate from one to four disk drives. A cable goes from the first disk drive to the second disk drive. If you add a third disk drive, a cable runs from the second disk drive to the third. This series of disk drives is referred to as a "daisy chain" (Fig. 5-14). If one or all of your disk drives do not work, there might be a problem in this daisy chain.

  A switch setting on the interface board or on the computer's motherboard must be set to indicate the number of disk drives in the daisy chain. If you install a new disk drive without changing this switch setting, the computer might not be aware that the disk drive is there. Information on changing these switch settings can be found in your disk drive or system unit technical documentation.

- ☐ Regardless of the number of disk drives you have, the computer needs a mechanism that indicates the last disk drive in the daisy chain. This is done with a device called a "terminator," and it serves to "close" an otherwise open circuit. The terminator is a strip consisting of resistors. It plugs into the daisy chain port of the last disk drive (Fig. 5-15). The terminator provides this signal, and the computer stops looking for any more disk drives.

  A problem can arise if the terminator is not connected, or if it is not connected to the last disk drive. Perhaps the last disk drive, the one with the terminator, is broken and you've taken it in for repair. If you try to use the remaining disk drive(s) on your system, and you forgot to plug the terminator into the disk drive which is now the last one, the system

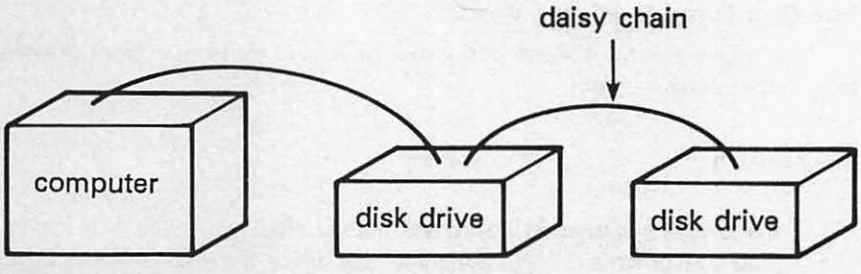


Fig. 5-14. Disk drive daisy chain.

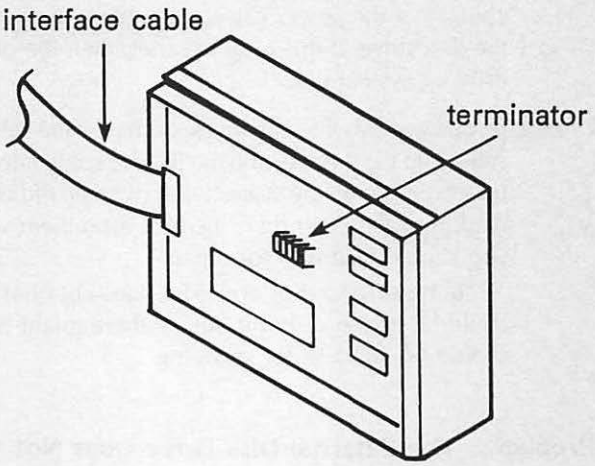


Fig. 5-15. Terminator strip in daisy chain port.

will not be able to sense the end of the daisy chain, and it will probably just "hang," or freeze up.

The same thing can happen if you've installed a replacement disk drive, and forgotten to plug the terminator into it. If you install a new additional drive onto your system, the system does not know it's there until you move the terminator to indicate it as the last disk drive. The system still stops wherever the terminator is plugged in, thinking it's found the end of the daisy chain.

Problem: The Integral Disk Drive Does Not Work

You might find that, although the system itself is functioning, no legitimate power is getting through to the internal disk drive or to the hard disk itself. The screen lights up, and the system prompt is displayed, but you are unable to access the disk drive. The disk drive power indicator light does not light when you insert a diskette into the disk drive or access your hard drive. If this is the case, you have a problem that can be caused by several different factors.

SOLUTIONS

- ☐ If you've added a new board to your system, or have done any other work or repair to the disk drive itself, you might have forgotten to plug the interface cable back into the disk drive or onto the interface board. The problem might also be in the internal power cable that connects the disk drive to the computer's main power supply. Remove the computer cover and check that all the cables are securely in place.
- ☐ If the cables are in order and the disk drive still doesn't work, then there is probably a physical problem with the disk drive or the power supply. You know that the power supply is bad when nothing works on the system at all. In this case, take the power supply in for repair. On the other hand, if everything else in the system is working, open up the computer and

check that the power cable is plugged into both the power supply and the disk drive. If this is all in order, then the problem lies with the disk drive or system unit.

- ☐ If you have two integral drives, exchange the cabling. Connect the A-drive cable into the B-drive, and the B-drive cable into the A-drive. If the system now boots up on the suspect disk, when it did not before, but you cannot read from the other disk, then it's a problem with the system unit, and you should take it in for repair.

  If the suspect disk drive still does not boot up, the disk drive motor could be frozen or burnt out, or there might be a bad relay. The drive should be taken in for servicing.

Problem: The External Disk Drive Does Not Work

The external disk drive has its own power supply. First make sure that power is being supplied to the drive. Do the normal power checks. Make sure the computer is properly plugged in. If it is, try using a different electrical outlet. See whether the outlet you're using for the computer is controlled by a light switch.

SOLUTIONS

- ☐ If all this checks out and the drive's indicator still does not light, there is probably a problem with the drive's power supply. Check the power supply's fuse, if you can get to it. It's usually located near the computer's power switch. If this is not available, or not the problem, take the drive in for service.
- ☐ If the operating system does not boot up, check that the disk drive interface cable is properly attached between the drive and the interface board.

Problem: Cannot Read or Write to the Diskette

The two most common problems with any type of disk drive is that you cannot read data from or write data to a diskette. In this case, the first thing to check is the diskette itself.

SOLUTIONS

- ☐ If you're using a new diskette and the system is not writing to it, the problem might be that the diskette is not formatted. There are many manufacturers of diskettes, and the same diskettes can be used with a number of different computer types. A diskette is made to work on your particular computer by inserting the diskette into the disk drive and running the operating system's format program. This must be done to each new diskette in order for it to work with your computer.

The format program places operating system information such as sector location and a disk directory program onto the new diskette. You can choose to format a new diskette as a "system diskette." This puts additional system information and utility programs on the diskette, making it more versatile, but also using more of the available storage.

The format program also runs a test to see if any of the diskette's sectors are bad. If any bad sectors are found, the format program blocks them out. This maintains the usability of the diskette, but prevents valuable data from being written onto bad areas.

- ☐ If you find that you can read from a diskette, but cannot write to it, or if you've run the Disk Drive Diagnostic Module and got the "Disk is Write Protected" error message, look at the diskette and see whether it is write-protected.

- ☐ If you've been using a particular diskette, and now find that it no longer lets you read the data or write to it, it might have just worn out. Diskettes are fragile, and after a great deal of use they can actually wear out, much the same way that a phonograph record or cassette tape can wear out after a number of plays. This is one reason why keeping a current backup copy of each of your diskettes is so important. If a diskette wears out, and you get the message "Disk Media Error—Try Another Diskette," you can simply use the backup diskette without a great deal of inconvenience.

  If you try these diskette solutions to the problem of not being able to read from or write to a diskette, and none of them solve the problem, then it's time to take a look at the disk drive and the disk drive controller board.

- ☐ Run the Disk Drive Diagnostic Module against the suspected disk drive. The diagnostic writes a known pattern to the disk, reads it back, and compares the two to make certain they match. This checks the read/write logic.

  When running the test, if you get two or three error messages, you probably have a "bad spot" on the disk. The disk can be salvaged with reformatting. The spot apparently went bad sometime after formatting, or it would have been caught and blocked out at that time. Just remember that when you reformat a disk, you erase all the data on it. Reformatting lets you start over with a "fresh" disk that has any bad sectors blocked out.

If you run the diagnostic and get a long series of read errors, this indicates that the disk drive or the controller board is faulty. To check this, boot up the system and load the program from that disk drive. If it boots up and loads satisfactorily, then the read logic in the controller board is fine, and the problem lies in the disk drive.

Problem: Cannot Load the Operating System or Read a Data File

Suppose you are trying to boot your system up, but you keep getting an error message such as "Press A to abort, R to retry, I to ignore." If you press R to retry the boot, and still nothing happens, then there is probably a problem with the circuitry responsible for reading data from the disk.

SOLUTIONS

- ☐ If you have a second disk drive, place the operating system diskette in that drive, and try booting the system up from there. If you still cannot boot up from the other drive, this indicates a bad system diskette. You should be able to boot up with your backup diskette.

  If the operating system does boot up from the other drive, load BASIC and the Disk Drive Diagnostic Module. Run the diagnostic against the "bad" drive, the one that would not boot up the operating system. If the test runs successfully, and you get the successful test completion message returned from the diagnostic, then nothing was wrong with the disk drive. It was probably just a problem with the operating system diskette. Try a different diskette, and it should work. If the diagnostic test fails, then you know that the disk drive is bad and you need to take it in for servicing.

- ☐ When you first turn your computer on and start to load the operating system, an internal system unit self-test program is run. This diagnostic resides in read-only memory on the motherboard of the system unit. The system diagnostic checks memory and then reads in the operating system. This program, itself, might have a problem that prevents you from being able to load the operating system. If this is the case, you should take your system unit in for repair.

- ☐ If you cannot boot up, read data from or write data to any of your disk drives, then your problem almost certainly resides on the disk drive interface board, and not on the disk drive. At this point, you should take the board in for repair.

Problem: Head Crash

The disk drive head travels microns above the surface of the diskette to read and write data. A *head crash* is the result of the head coming into contact with the diskette. This can happen if the head is out of adjustment, which can happen if the disk drive has been shaken or dropped. A head crash can also be caused by contamination on the disk or in the drive (Fig. 5-16).

If the head crashes, you will hear a grinding noise, and the disk will be scratched. This can be catastrophic to your disk and its data.

A head crash can be caused, not only by a head being out of adjustment, but also by a warped diskette. Diskettes, like records and tapes, can become warped and damaged when exposed to heat or direct sunlight. If you place a

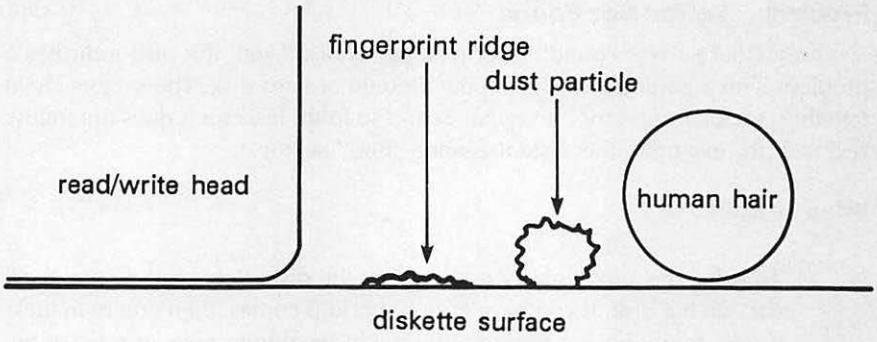


Fig. 5-16. Disk contamination.

warped diskette into the drive, it can scrape up against the head and cause a head crash. Don't try to use a warped diskette, because if it does cause a head crash, your disk drive will go out of adjustment and you'll have to take it in for servicing.

You'll know you have a head crash situation if you cannot read from, write to, or format the disk. Another indication is a scratchy hissing sound heard when you insert a diskette in the drive.

SOLUTION:

- ☐ When the head crashes, you need to take the drive in for repair. Discard any diskettes that have come into contact with the drive head after the crash.

Problem: Read Data Is Garbled

In addition to wearing out, diskettes can be damaged and the data on them garbled if not handled properly. Diskettes are a magnetic media, so magnetism can adversely affect the data on a diskette. If you place a diskette next to a power supply, or near an appliance with a motor, the magnetic field could garble the data.

Although the 5¼-inch diskette is enclosed in the cardboard, there is a small window at the hub of the diskette where the diskette is exposed for the drive head access. If the diskette is handled roughly, or if this exposed area is contaminated in some way, the emulsion can be altered, and the data could be garbled.

Problem: Bad Address Mark

If you run the Disk Drive Diagnostic Module and the "Bad Address Mark" error message is displayed, this indicates that a sector or track on the disk has an address that is either out of order or beyond the capacity of the drive. This is caused by a defective disk. Use your backup copy at this point. The disk itself can be salvaged by reformatting.

Problem: Sector Not Found

If the "Sector Not Found" error message is displayed, this also indicates a problem with a particular sector on the diskette or hard disk. The sectors chain together, sector 1 to sector 2 to sector 3, and so forth. If sector 3 does not follow sector 2, for example, the system cannot "find" sector 3.

SOLUTIONS

- ☐ To solve this problem, try reformatting the disk. Reformatting erases all data on the disk. If you have current backup copies, then you're in luck. If you do not have a backup, you might be able to recover most of the data if the disk drive lets you read up to the point of the bad sector. If you reformat your hard disk, anticipate a big job reinstalling your software and data files.
- ☐ If reformatting the *hard disk* does not solve the "Sector Not Found" problem, take it in for servicing. If reformatting the *floppy diskette* does not solve this problem, you should probably just discard the diskette as bad.

Problem: Device Timeout

The error message "Device Timeout—Check Power to Drive" indicates that the disk drive has not come up to its correct speed within a given period of time. The disk must be up to a certain speed (in rpms) in order for the mechanism to work correctly. Check the following four solutions to solve this problem.

SOLUTIONS

- ☐ Make sure you have inserted a diskette into the drive.
- ☐ Check that the disk drive door is closed.
- ☐ Go through the checks as described under "The Disk Drive Does Not Work" section above.
- ☐ If none of these solutions clear the error message, then the drive is probably in need of repair.

Problem: Device Fault

When the disk drive interface board does not receive an expected signal, such as "Drive Ready," you might get the error message "Device Fault—Drive May be Faulty." This indicates a problem with the disk drive (not the interface board).

SOLUTION

- ☐ Use compressed air or a vacuum cleaner to remove any dust or dirt from the disk drive. A sensor might be obscured by dust. If this does not solve the problem, take the disk drive in for repair.

Problem: Internal Error

If you get the error message "Internal Error—Problem With System Unit," there is probably a serious problem, not so much with the disk drive or interface board, but with the system unit itself. Something in the logic or memory might have caused a fatal error, which causes the entire system to stop operating.

SOLUTIONS

- ☐ This might be a controller bus problem. If you can switch the disk drive interface board to another slot in the system unit, this might solve the problem. Refer to "Troubleshooting Techniques" in Chapter 1 for the procedure on swapping slots.
- ☐ If swapping slots does not solve the problem, or if you cannot swap slots on your system, take your system unit in for servicing.

Problem: Cannot Find File

The error message "Cannot Find File—Problem With Disk Media" is displayed when the Disk Drive Diagnostic Module creates a test file, closes it, and opens it in order to read back what it wrote. If the system cannot find this file, there is likely to be a problem with the disk media itself, or a problem in the directory listing the disk contents.

SOLUTIONS

- ☐ Replace the diskette with another one, and run the diagnostic again.
- ☐ If this problem occurs on your hard disk, make sure you have complete backups of all programs and data stored there. Reformat the hard disk and then reinstall all the information.

Problem: Device I/O Error

If the error message "Device I/O Error—Check Drive and Interface" is displayed, try the following solutions.

SOLUTIONS

- ☐ Check the cable on the disk drive interface board. It might be loose or making only intermittent contact.
- ☐ Move the disk drive interface board to a different slot and try running the diagnostic again.

Problem: Attempt to Read Past EOF

The error message "Attempt to Read Past EOF—Problem With Disk Media" indicates a problem with a faulty diskette. Replace it with another diskette and run the diagnostic again.

Problem: Disk Not Available

The error message "Disk Not Available—Check Power or Problem With Interface" indicates either a power or interface board problem.

SOLUTIONS

- ☐ Do the power checks as described under the section titled, "The Disk Drive Does Not Work."
- ☐ Check the interface board cables and make sure they're properly tightened down.
- ☐ If it's a new disk drive, check to see that the proper number of drives is set on the motherboard switch. Refer to your disk drive or system technical manual for more information about this switch setting.

Problem: Disk Is Write-Protected

The error message "Disk is Write Protected" means just that. You're attempting to run the diagnostic against a diskette that is write-protected. Remove the write-protect tab from the diskette, and the diagnostic should run correctly. However, be sure that you really want to do this. Chances are you write-protected this diskette for a reason.

Problem: Disk Not Ready

If the error message "Disk Not Ready—Door is Open on Drive or No Power to Drive" is displayed, try the following solutions.

SOLUTIONS

- ☐ Check the disk drive door and make sure it's closed.
- ☐ Do the power checks as described under "The Disk Drive Does Not Work."

Problem: Disk Media Error

The error message "Disk Media Error—Try Another Diskette" indicates a problem with the diskette. Replace it with another one and retry the diagnostic.

Problem: Test Has Ended with Fatal Error

If you get the error message "Test Has Ended With Fatal Error!," it means that an error has occurred that will not allow the read/write test to continue. Something must be done at this point to fix the problem before the test can continue. If there are one or two errors, there is probably a problem with the disk media. Reformat the diskette or try a new one. Several errors indicate a problem with the drive rather than the disk.

PREVENTIVE MAINTENANCE

Disk drives, particularly floppy disk drives, require a degree of preventive maintenance. Make sure that the disk drive doors are closed when not in use. This prevents dirt and dust from collecting on the disk drive mechanism.

Dust can be a big enemy to the disk drive motors. Dirt can collect on the head mechanism and cause head crashes. Covering your system with a dust cover protects integral disk drives from dust and dirt. If you have an external disk drive, place a dust cover over it as well.

When moving a system with a hard disk drive, the heads might need to be bolted down. This is done with a bolt mechanism that tightens the heads down to let you safely transport the system. On some systems, this has to be done manually with a nylon screw accessed through the bottom of the drive housing; on other systems it can be done through a "lockdown" or "safety" utility program. If this is not done, the head mechanism, and possibly even the disk platter itself could be damaged. On some systems, it is recommended that the heads be locked down before turning the system off, whether you're planning to move it or not. This prevents damage that can occur when you power the system down while the heads are still moving. Newer hard drives automatically lock the heads down as soon as you turn the computer off.

When moving a computer with floppy disk drives, insert the cardboard "diskettes" into the drives and close the doors. These protect the head mechanism from being jostled, thereby preventing it from going out of alignment during transport.

RUNNING THE DISK DRIVE DIAGNOSTIC MODULE

To run the Disk Drive Diagnostic Module, follow the instructions provided in "Running the Diagnostic Program" in Chapter 1. Then press D to initiate the Disk Drive Diagnostic Module (Fig. 5-17).

This diagnostic does a very thorough check of the disk drive and the disk media. It also tests the two main functions of the disk drive: reading from and writing to a disk. It exercises the entire disk, checking for many types of problems. If a problem is encountered, the appropriate error message is displayed on the screen.

The Disk Drive Diagnostic Module can run on any make of disk drive, and can test either a floppy or hard disk drive. When running the test on a floppy disk drive, use a freshly formatted, empty diskette. The diagnostic module tests the data that it writes onto the disk to fill it up, but it does not test data already stored on the diskette. This would include the directory or any system files that might exist on a system diskette.

The same holds true if you run this test against a hard disk or a diskette that contains program or data files. The information already residing on the disk is not tested, and cannot be damaged or wiped out. However, as always, make sure you always have full current backups of programs and files residing on any disk you test, especially a hard disk.

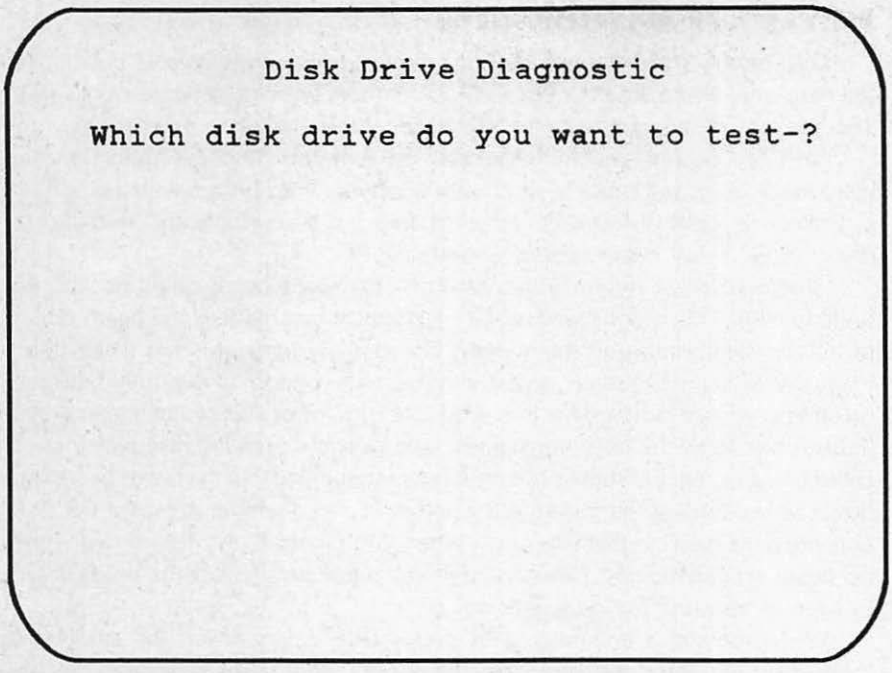


Fig. 5-17. Disk drive diagnostic instructions.

In response to the prompt, indicate the drive to be tested: A, B, or C. The diagnostic creates a file called "TEST," and makes four distinct passes through the disk.

The first pass writes a series of 128-byte records that contain a 128-character string of ASCII value 170. ASCII value 170 is a character that has every other bit turned on (10101010). The diagnostic writes these records throughout the entire disk. Once the disk is full, the file is closed.

The diagnostic immediately begins the second pass through the disk, which consists of reopening the file and rereading the data that were written during the first pass. The read data are compared to ASCII value 170. If there is a discrepancy between what was written and what is now being read, an error message is displayed on the screen.

When the second pass of the test is completed, the file is deleted. The third pass consists of recreating the file, this time with a series of 128-character records of ASCII value 85, which has every other bit turned off (01010101). It completely fills the disk up again with this alternate character record, just as it did before with ASCII value 170. When the disk is full, the file is closed.

As mentioned above, ASCII 170 has every odd bit turned off, and every even bit turned on. ASCII 85 has the opposite characteristics, with every odd bit turned off and every even bit turned on. This creates an alternating bit pattern. Testing for alternating bit patterns ensures that all possible bits on the disk are written to. If the test wrote a random pattern, or all ASCII values, the disk media could

be damaged, and if the bits were, in fact, set, it would go ahead and pass the diagnostic, even if a problem existed.

Just as in the second pass, the fourth pass through the disk opens the file and rereads the data written during the third pass. The read data are compared to ASCII 85. An error message is displayed if there is a discrepancy between what was written and what is now being read. Then the file is erased, just as it was after the second pass of the test.

HOW THE DISK DRIVE MODULE WORKS

When you press D or d from the System Diagnostic Main Menu, the program branches to Line 2050. A REMARK statement in the program marks the beginning of the Disk Drive Diagnostic Module (Fig. 5-18).

Lines 2060 and 2070 initialize two variables, HVAL$, meaning high values, and LVAL$, meaning low values. HVAL$ builds a string of the high ASCII value of 170. The loop continues 128 times, essentially building a 128-character record:

```
2060    HVAL$=" ":FOR I=1 TO 128:HVAL$=HVAL$+CHR$(170):NEXT
        I
2070    HVAL$=" ":FOR I=1 TO 128:LVAL$=LVAL$+CHR$(85):NEXT I:
        IF EX=1 THEN GOTO 2090
```

Line 2070 does the same thing, building a string of the low ASCII value of 85 128 times. In addition, the latter part of Line 2070 checks to see if the Exerciser Module is in control of the program. If it is, the program branches to Line 2090, thus bypassing the setup procedure. If EX is not equal to one, the program proceeds to Line 2080.

Line 2080 clears the screen and displays the diagnostic title. A prompt is displayed to enter the disk drive to be tested. You would enter a value such as A, designating the A drive, B designating the B drive, and so forth. A hard disk drive might be designated as the C drive on some systems, and as the A drive on others:

```
2080    CLS:PRINT TAB(10)"Disk Drive Diagnostic":PRINT:
        INPUT "Which disk drive do you want to test-";
        D$:D$=D$+":TEST"
```

MAC USERS: The Macintosh handles disk identification in a different manner. It does not use an A, B, or C designator. Instead, a name is assigned to the diskette upon formatting. This name can be found under the disk icons on the Macintosh "desktop." Pressing RETURN causes the test to take place on the currently active drive.

The last part of the instruction builds the string that creates the disk test file name. It does this by joining the drive designation with the colon and the word

```
2050 REM: Disk Drive Diagnostic Module
2060 HVAL$="":FOR I=1 TO 128:HVAL$=HVAL$+CHR$(170):NEXT I
2070 LVAL$="":FOR I=1 TO 128:LVAL$=LVAL$+CHR$(85):NEXT I:
     IF EX=1 THEN GOTO 2090
2080 CLS:PRINT TAB(10)"Disk Drive Diagnostic":PRINT:
     INPUT "Which disk drive do you want to test-";
     D$:D$=D$+":TEST"
2090 BUF$=HVAL$:NM$="High Values"
2100 ON ERROR GOTO 2270
2110 FOR I=1 TO 2
2120 OPEN D$ FOR OUTPUT AS #1
2130 PRINT:PRINT "Writing" ";NM$;" to Disk"
2140 ON ERROR GOTO 2270
2150 PRINT #1,BUF$:GOTO 2150
2160 CLOSE #1
2170 OPEN D$ FOR INPUT AS #1
2180 PRINT "Reading ";NM$;" from Disk":PRINT
2190 INPUT #1,BUF1$
2200 IF EOF(1) GOTO 2230
2210 IF BUF1$<>BUF$ THEN PRINT "Error Reading ";NM$:E=E+1
2220 GOTO 2190
2230 CLOSE #1:KILL D$:BUF$=LVAL$:NM$="Low Values":NEXT I
2240 PRINT "There were ";E:" Errors Encountered During This
     Test"
2245 ON ERROR GOTO 0
2250 IF EX=1 THEN RETURN
2260 GOSUB 2940:ON ERROR GOTO 0:GOTO 40
2270 IF ERR=61 THEN CLOSE #1:RESUME 2160
2280 IF ERR=24 THEN PRINT "Device Timeout - Check Power to
     Drive"
2290 IF ERR=25 THEN PRINT "Device Fault - Drive may be
     Faulty"
2300 IF ERR=51 THEN PRINT "Internal Error - Problem with
     System Unit"
2310 IF ERR=53 THEN PRINT "Cannot Find File - Problem with
     Disk Media"
2320 IF ERR=57 THEN PRINT "Device I/O Error - Check Drive
     and Interface"
2330 IF ERR=62 THEN PRINT "Attempt to Read Past EOF -
     Problem with Disk Media"
2340 IF ERR=68 THEN PRINT "Disk Not Available - Check Power
     or Problem with Interface"
2350 IF ERR=70 THEN PRINT "Disk is Write-Protected"
2360 IF ERR=71 THEN PRINT "Disk Not Ready - Door is Open on
     Drive or No Power to Drive"
```

Fig. 5-18. Disk drive diagnostic module listing.

```
2370 IF ERR=72 THEN PRINT "Disk Media Error - Try Another
     Diskette"
2380 CLOSE #1
2390 PRINT:PRINT "Test has Ended with Fatal Error!":
     GOSUB 2940:RESUME 2080
```

Fig. 5-18. Continued.

"TEST." So when testing the A drive, a file called "TEST" is created on the A drive. Line 2090 sets the variable BUF$ equal to HVAL$. The name variable NM$ is initialized to the character literal "High Values:"

```
2090 BUF$=HVAL$:NM$="High Values"
```

When the diagnostic is running, the words "High Values" are displayed on the screen, indicating that it is currently writing the ASCII 170 character to disk.

Line 2100 bypasses the standard error-checking used by BASIC. If an error is encountered, the program branches to line 2270, which does the error-checking:

```
2100 ON ERROR GOTO 2270
```

In this way, more specific and helpful error messages are displayed, instead of the cryptic BASIC error messages.

Line 2110 is the beginning of a FOR-TO loop. The test runs through two passes: one for the high values of ASCII value 170, and one for the low values of ASCII value 85:

```
2110 FOR I=1 TO 2
```

So, rather than unnecessarily duplicating code, a FOR-TO loop allows the same code to be used in both cases, simply plugging in the different variables as appropriate.

The program proceeds to line 1465. D$, the file name for the output to be written to disk, is opened and assigned as file identifier #1:

```
2120 OPEN D$ FOR OUTPUT AS #1
```

Line 2130 prints a blank line and then displays an informational message that says it is writing NM$ to disk. NM$ was initialized in Line 2090 to "High Values:"

```
2130 PRINT:PRINT "Writing" ";NM$;" to Disk"
2140 ON ERROR GOTO 2270
```

Line 2150 is where the actual output to disk takes place. The PRINT #1

statement from variable BUF$ writes the high values out to the disk. The remainder of the line consists of a GOTO statement, in which the line goes back on itself for a continuous loop, causing the high values to be written to the disk over and over again, thereby filling up the disk:

```
2150 PRINT #1,BUF$:GOTO 2150
```

To prevent this continuous loop from going on infinitely, line 2100 causes the program to branch to line 2270 when it encounters an error of some kind. When the disk is finally filled to capacity with Line 2150, the disk returns BASIC error code 61, which says the disk is full and no more data can be written to it. When this error (or any other error, for that matter) is encountered, the continuous loop is broken, and the program branches to line 2270. As described in more detail below, line 2270 closes the file and then resumes the program at the following statement, line 2170.

Line 2170 reopens D$, which was the name of the output file in the previous test:

```
2170 OPEN D$ FOR INPUT AS #1
```

However, for this phase of the test, D$ is an input file.

Line 2180 displays the informational message that says the high values of NM$ are being read from the disk:

```
2180 PRINT "Reading ";NM$;" from Disk":PRINT
```

Line 2190 begins the input of the data being read from the disk. It is read into a variable called BUF1$:

```
2190 INPUT #1,BUF1$
```

Line 2200 checks to see if the end of the disk file has been reached:

```
2200 IF EOF(1) GOTO 2230
```

When all the data have been read, the program branches to line 2230.

If the end-of-file has not yet been reached, the program falls through to line 2210, which compares variable BUF1$ to variable BUF$. BUF1$ contains the high values read from the disk in line 2190, while BUF$ contains the high values written into the disk in line 2150:

```
2210 IF BUF1$> <BUF$ THEN PRINT "Error Reading ";NM$:E=E+1
```

In other words, this statement checks that it's reading the same data that it just wrote. If there is a discrepancy, a message is displayed, saying that there was an error reading the high values.

The latter part of the statement, E=E+1, increments an error counter. This counter keeps track of the number of read-errors encountered during the test. After this, the program proceeds to line 2220, a GOTO statement which loops the program back to line 2190, which is the statement that causes the data to be read from the disk:

```
2220 GOTO 2190
```

Line 2200 contains an end-of-file trap. When reading data file #1, if the program encounters the end of the file, the program branches to line 2230. The first thing that line 2230 does is close the file:

```
2230 CLOSE #1:KILL D$:BUF$=LVAL$:NM$="Low Values":NEXT I
```

The KILL statement erases the file, clearing it from the disk. Then variable BUF$ is initialized to LVAL$. This resets the test to check for the alternate bit pattern, ASCII value 85. In addition, NM$ is also initialized to the low value, where in the previous test it was set to the high value. The last portion of this statement is the NEXT statement in the FOR-TO loop. It causes the program to branch back to line 2120, at which point the entire process begins again. The difference is that instead of writing and reading the high ASCII value of 170, the low value of ASCII value 85 is being written, read, and compared.

Once the FOR-NEXT loop is completed, the program proceeds to line 2240. This displays a summary of the number of read-errors encountered during the test, such as "There were 2 Errors Encountered During this Test," or better yet, "There were 0 Errors Encountered During this Test."

```
2240 PRINT "There were ";E:" Errors Encountered During this Test"
```

If there were read-errors, it's probably a problem with the disk medium itself; in other words, you have a bad disk. However, it is conceivable that errors could indicate a problem with the disk drive or with the disk drive controller board. But if this is the case, you'll usually see some other error message.

Line 2245 resets normal error handling for BASIC:

```
2245 ON ERROR GOTO 0
```

Line 2250 checks again to see if EX is equal to one. If it is, then the disk testing is finished, and the program returns to the Exerciser Module:

```
2250 IF EX=1 THEN RETURN
```

After the read-error message is displayed on the screen, the program falls through to Line 2260. This calls subroutine 2940, which displays the prompt to

press any character to end the test and return to the System Diagnostic Main Menu:

```
2260 GOSUB 2940:ON ERROR GOTO 0:GOTO 40
```

The GOTO 40 statement returns the program to line 40, which is the beginning of the System Diagnostic Main Menu routine, and signals the end of the Disk Drive Diagnostic Module.

The Disk Drive Diagnostic Module's error-handling routine runs from line 2270 through line 2390. This routine looks at the various disk drive error codes that can be returned from BASIC. These BASIC error-handling routines are overridden with the use of ON ERROR traps, as seen in line 2100. Any error code encountered is captured and placed in a reserved variable called ERR. The program examines the value of ERR and takes further action automatically, instead of merely halting. If any of these errors are encountered, refer to "TROUBLESHOOTING AND REPAIR GUIDELINES" earlier in this chapter to correct the error.

Line 2700 starts with an IF statement, and is the first statement that looks for variable ERR. If the value of ERR is 61, it indicates that the disk is full:

```
2270 IF ERR=61 THEN CLOSE #1:RESUME 2170
```

For the purposes of this diagnostic, this isn't really an error, but it's necessary for the program to get out of its continuous loop at Line 1480. When this "error" is encountered, file #1 is closed, and the program resumes at line 2170.

Line 2280 is the first indication of a real disk drive problem. Error 24 displays a message which states that there has been a device timeout, and that the disk drive power should be checked:

```
2280 IF ERR=24 THEN PRINT "Device Timeout - Check Power to Drive"
```

Line 2290 checks for error code 25, which typically signifies a hardware failure:

```
2290 IF ERR=25 THEN PRINT "Device Fault - Drive may be Faulty
```

In this case, it would be a disk drive malfunction.

If error 51 is encountered, line 2300 displays the message indicating an internal error:

```
2300 IF ERR=51 THEN PRINT "Internal Error - Problem with System Unit"
```

If the program encounters error 53, line 2310 prints the message indicating that the system cannot find the file:

```
2310 IF ERR=53 THEN PRINT "Cannot Find File - Problem with Disk
     Media"
```

Line 2320 checks for error 57, a device I/O error:

```
2320 IF ERR=57 THEN PRINT "Device I/O Error - Check Drive and
     Interface"
```

Line 2330 checks for error 62 and shows an attempt to read past the end-of-file mark:

```
2330 IF ERR=62 THEN PRINT "Attempt to Read Past EOF - Problem with
     Disk Media"
```

Line 2340 checks for error 68, indicating that a disk is not available:

```
2340 IF ERR=68 THEN PRINT "Disk Not Available - Check Power or Prob-
     lem with Interface"
```

Line 2350 checks for error code 70:

```
2350 IF ERR=70 THEN PRINT "Disk is Write-Protected"
```

Line 2360 checks for error 71, indicating that the disk is not ready because the disk drive door is open:

```
2360 IF ERR=71 THEN PRINT "Disk Not Ready - Door is Open on Drive
     or No Power to Drive"
```

Line 2370 checks for error 72, a disk media error:

```
2370 IF ERR=72 THEN PRINT "Disk Media Error - Try Another Diskette"
```

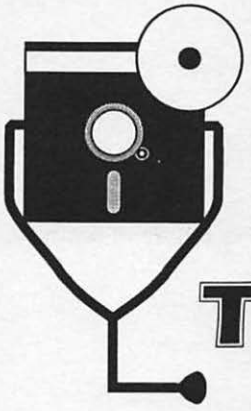
Line 2380 closes the disk file:

```
2380 CLOSE #1
```

Finally, line 2390 marks the end of the special error-handling routine. It indicates that the test has ended with a fatal error:

```
2390 PRINT;PRINT "Test has Ended with Fatal Error!":
     GOSUB 2940:RESUME 2080
```

When any of these errors are encountered, other than the disk full "error" 61 at line 2270, the Disk Drive Diagnostic Module halts. Then, the subroutine at line 2940 is called up to display the message "Press Any Key to End Test." The program then resumes at line 2080, which is the beginning of the Disk Drive Diagnostic Module. At this point, you can either exit out of the module to respond to an error message or to use another diagnostic module.



Chapter 6

THE SERIAL COMMUNICATION INTERFACE

The computer need not be an island unto itself. With the proper equipment, your computer can "talk" to many other devices: a modem, printer, mouse, . . . even another computer. Data and information can be sent and received to and from these devices. Such communication among different devices requires the services of the *serial communication interface*.

DESCRIPTION AND FUNCTION OF THE SERIAL COMMUNICATION INTERFACE

A serial communication interface is an interface board in your computer's system unit (Fig. 6-1). Although this is an add-on option for MS-DOS systems, some computers have a serial communication port built in as a standard feature. The Macintosh and Atari ST are examples of such a system.

The interface board reaches the outside world by means of a plug and cable which connects to the other device with which communication is to take place. It is through this cable that the appropriate functional signals and data are passed to and from the other device.

Types of Serial Communication Devices

The serial communication interface can be used to let your computer communicate with devices such as the modem, printer, mouse, and another computer.

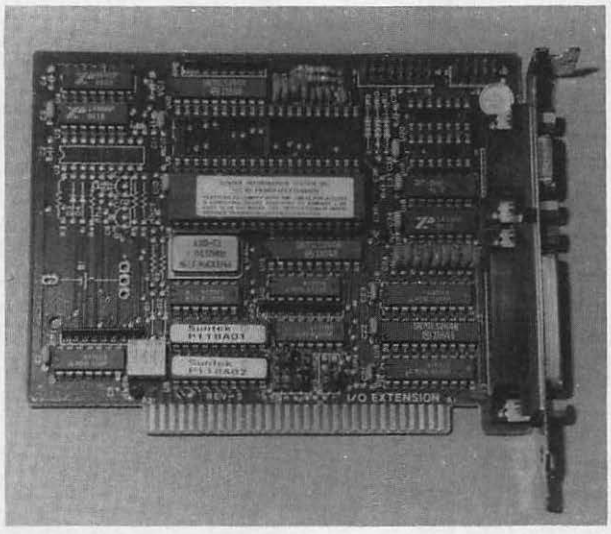


Fig. 6-1. Serial communication interface board.

Interfacing with a Modem. The modem is a device widely used for serial communication. It lets you communicate computer data across long distances using telephone lines. The word "modem" is derived from the term "modulator-demodulator." The computer produces data and control signals in the form of electrical impulses, also known as *digital* signals. These digital data exist in the form of ONs and OFFs, or ones and zeros. So a bit of digital data might look like 01010101.

The modem converts this digital signal into sound waves, or *analog noise,* for transmission over the telephone line. This conversion is referred to as *modulation.*

The modem at the receiving end then converts this analog signal back into a digital signal, and this conversion is referred to as *demodulation.* Because the information is once again in the form of ones and zeros, the computer connected to the receiving modem can read it (Fig. 6-2).

Modems operate in pairs. To let two computers communicate with one another, each computer must be connected to a modem that has matching configurations in terms of communication speed, data size, and other parameters that are discussed later in this chapter.

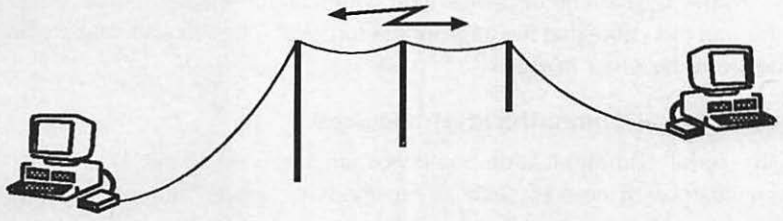


Fig. 6-2. Modems sending and receiving data.

"Online services" such as CompuServe, DELPHI, Dow Jones News, The Source, and Computext make it easy for people with computers and modems to "link up" to their centers of information and communicate. By coupling the modem to your telephone, then dialing up a given telephone number, which is often a local or toll-free number, your computer can be patched into a network's mainframe computer. You can then use the network service to research a wide range of information, as well as communicate with other users. Online services can also allow you to bank, shop, and make airline ticket reservations with your computer through the use of your modem (Fig. 6-3).

Interfacing with a Printer. Chapter 4 discussed the two types of printer interfaces, serial and parallel. The parallel interface is most commonly used for the printer, although most printers can be used with either interface. You might want to use the serial interface for the printer if the parallel interface is being used for another device. If you have two printers, with one being connected to the parallel interface, you'll want to use the serial interface for the second printer.

Just like the modem, the printer must be configured with the same data speed and size parameters that are set in your computer. This is so that the printer knows what to expect from the computer, and knows how to handle it.

The printer is typically attached to your computer with an interface cable, or *hard wire*. In some cases, however, the printer can even be used in conjunction with the modem to provide long-range printing. A central computer from a remote site can send information via a communication line, and have it print out on this local printer.

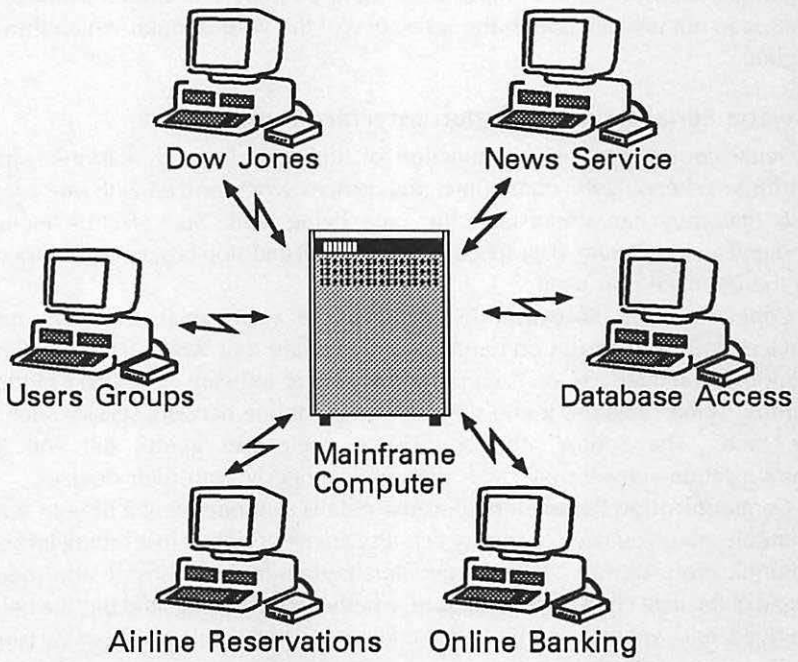


Fig. 6-3. Using online services.

Interfacing with a Mouse. There are many specialty devices that can be interfaced with the computer. Manufacturers often design their devices to use the serial communication port, because serial communication adheres to the RS-232 standard.

One very popular specialty device using the serial communication port is the mouse. A *mouse* is a hand-held pointing device used as an alternative to the arrow keys for positioning the cursor on the screen.

The mouse is attached to the computer via a cable. The mouse has a light-sensing lens or mechanical ball which senses the direction and distance that it is being moved. It also has at least one button that the user presses to indicate a choice of some kind. The mouse transmits this direction, distance, and selection information via serial communication to the computer.

Interfacing Directly with Another Computer. If you use your computer in an environment in which there are computers in relatively close proximity (less than 50 feet), these computers can be hard-wired together simply with an interface. In other words, they can be connected with a cable through their serial communication ports to exchange data just as they would if they were hooked up through modems. The difference is that telephone lines are not needed in a hard-wire connection.

When computers are hard-wired together, the speed of data being sent can be much faster. As always, the communication parameters between the two computers must match in order for them to understand and process the information being sent and received.

Hard wiring works over short distances of less than 50 feet. Communication over longer distances requires modems or signal amplifiers, because the electrical pulses fade out and die due to the resistance of the wire through which they're traveling.

How the Serial Communication Interface Functions

Serial communication is a function of timing and speed. Various factors contribute to keeping the communicating devices synchronized with one another so that they can understand the data being sent. Such factors include communication software, data speed, data size, start and stop bits, parity, protocols, and the interface port itself.

Communication Software. In order to use your serial communication interface, you need to run communication software that works from the serial communication interface. An example of this kind of software would be a modem communication program. If you subscribe to an online network service such as Dow Jones, The Source, or CompuServe, their user guides tell you the communication software packages that work properly with their devices.

Communication Parameters. Because data is sent one bit at a time in serial communication, you need to specify certain parameters when first setting up your communication software. These parameters pertain to data transmission speed, the size of the data characters being sent, whether start bits and stop bits are being employed, and whether "parity" error-checking is to take place. These variables

serve to coordinate the communication and ensure that the data transfer is successful. The particular variables used depend on the device with which you're communicating. If you subscribe to an online service, for example, their user documentation spells out these parameters.

The printer, modem, or other device with which your system is going to communicate must have the communication interface variables set identically on their end in order for this communication link to be established and maintained. These parameters are set either through the software or with switches on the communication interface board. Check your documentation for specific details about this.

When you run your communication for the first time, you can access a setup mode in which you can set up these communication parameters. Once these parameters are set, they can usually be stored on the program disk so that they are automatically there every time you use the program.

Baud Rate. The first parameter pertains to how fast the device sends information. The device receiving the data must know the speed at which the data is being sent in order for the two devices to be in sync with one another.

In serial communication, speed is expressed in a measurement called *baud.* Baud is roughly the number of bits that can be transmitted in a second. The differences in baud rate are analogous to the different ways a car horn might be sounded. Three short beeps can be heard much faster than three long blares. By the same token, eight bits of data traveling at 2400 baud are transmitted and received much faster than the same eight bits traveling at 300 baud. When the baud rate is set to a predetermined speed that both devices are aware of, the system knows how long each ON bit lasts, and how long each OFF bit lasts. Most communication interfaces for personal computers run at 300 or 1200 baud.

Higher speed communication in the range of 4800 and 9600 bits per second (baud) is available, but is used primarily for communicating with devices that are hard-wired to the system through an interface cable. A hard-wired setup does not need an intermediary such as a modem in order to establish communication. A printer, or even a computer that is hard-wired to another computer, runs at the faster speeds. There are modems available that can run at such high speeds, but they are prohibitively expensive, especially to the small business or home user.

Data Size. The data size parameter refers to the size of the data being sent: how many bits make up a standard "character" or "word." A standard character of data is made up of seven or eight bits, depending on the software. The software needs to know the "character length" or "word length" of the data being sent, so it knows what to expect (Fig. 6-4).

Start Bits and Stop Bits. In serial communication, the software adds additional bits to each character to help coordinate the communication transfer. These added bits are referred to as "start bits" and "stop bits." When the software adds the start bit and stop bit, then nine or ten bits are actually sent for each data character.

The *start bit* signals the beginning of a character, and captures the receiving device's attention. It tells the device to expect seven or eight more bits of data for processing. Likewise, the *stop bit* signifies the end of a character.

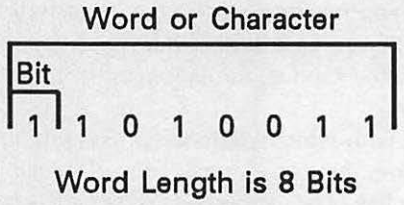


Fig. 6-4. Bits, characters, and words.

Both ends of the communication line need to know whether start bits and stop bits are being employed, and if so, how many there are—one or two of each (Fig. 6-5).

Parity. Another variable that many systems use is *parity,* which is an error-checking scheme used to ensure accurate data transmission. With parity, the last bit in a data word is set so that the sum of the number of bits is either an odd number, for "odd parity," or an even number, for "even parity."

If you're set up with odd parity from your modem to your computer, for example, and you receive a character or word with an even number of bits, parity is not correct. This might mean that one of the character bits have dropped off during transmission. When parity is not met, an error in transmission is indicated, and your communication software either prompts you to try the data transmission again, or restarts it automatically.

Directionality. Depending on the device with which your computer is communicating, the data might be sent in one direction only, or it might be sent in both directions—to and from both devices. Printers are typically "unidirectional," meaning that they only receive data; they do not transmit it back to the computer. When two computers are linked up through modems, on the other hand, the data is said to be "bidirectional," since you can send and receive data to and from both systems.

Whether the data is unidirectional or bidirectional depends on the type of device to which the data is sending, as well as the software being used for communication. You might need to set some parameters in the support software, and there might be switches to set on the serial communication board. This would be detailed in the device documentation.

Emulation. Different types of computers have different keyboard control sequences, keyboard mapping, screen format display, text handling, and special functions. These aspects of unique computer design can hinder effective communication between two different computers.

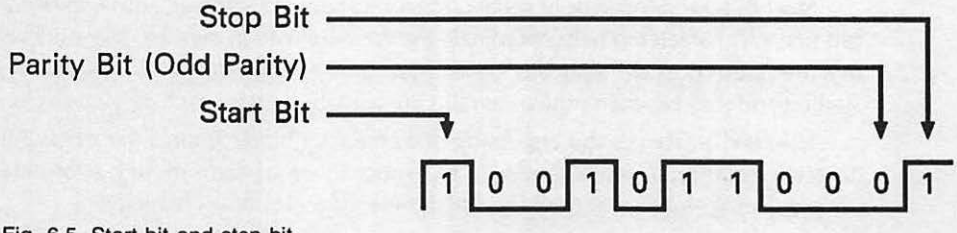


Fig. 6-5. Start bit and stop bit.

To solve this problem, an emulation program in your communication software can disguise your computer and make it "pretend" that it is another computer of a different design. The DEC VT100 is a common emulation mode, in which your IBM PC or Macintosh can emulate the VT100 terminal. When both computers are set up in the same emulation mode, then all code sequences for the keyboard and monitor function as expected.

Protocols. The "XON/XOFF" protocol works in conjunction with the emulator, and is set with the other communication software parameters. You set whether or not you wish this protocol to take effect. Even though the baud rates might be compatible, if the receiving system cannot process the data fast enough, some data could get lost. The XON/XOFF protocol places transmission "on hold" to prevent the communication buffers from overflowing, and then turns it on again to make sure that data is completely transferred.

For example, suppose your computer has a 2000 byte communication buffer. In the process of receiving a file that may be, say, 16,000 bytes long, the communication buffer is quickly filled up. When this happens, your system sends a "suspend" command to the other system, temporarily stopping communication. This suspension is the XOFF command. When the buffer is emptied through computer processing, such as displaying characters on the screen, your system sends a "resume" command, which is XON. The transfer then continues where it left off. This process continues throughout the file in 2000 byte chunks until all 16,000 bytes are successfully transferred.

There is another protocol used by modems to establish communication. This is called the *handshaking* protocol. The handshaking protocol is analogous to human greeting and communication protocols. When two people meet in a hallway, for example, they ordinarily greet one another in the following manner:

"Hello, how are you doing?"
"Hello, I'm doing fine. How are you?"
"Fine. Nice weather we're having today, isn't it?"

Then they might go on their way, or they might continue talking in another vein. The important thing is that they have satisfied the human protocol of the greeting.

People use such protocols in order to enhance their personal communication, and to make sure that they understand one another. If one person talks much faster than the other, there might be a breakdown in communication: words might be missed, and the thought might be misunderstood.

The manner in which protocols are used is important as well. If two people meet, and they follow the protocols but in the wrong order, they might still have a communication failure, as in this example:

"Good-bye. Nice weather we're having today."
"I'm fine, and you?"

When a person says "good-bye," our protocol implies that this is the end of the

conversation. If more is said after this end-of-conversation message, then there might be confusion and lack of understanding.

The modem handshaking protocol operates under the same principles. Many signals are used to establish modem communication. The following is a list of the handshaking protocol used to establish and maintain communication:

1. Run the program.
   It brings up TR (Terminal Ready).
2. Dial the phone.
   Switch into data mode, DSR (Data Set Ready).
3. The other system answers and then sends the carrier signal.
   The CD (Carrier Detect) light is illuminated on the modem panel.
4. Press a key to send a character code to the other system.
   The RTS (Request to Send) signal is sent to the modem and the appropriate light is illuminated.
5. The modem tells your system that it is prepared to accept data for transmission.
   The CTS (Clear to Send) signal is sent to your system.
6. The data is sent.
   The SD (Send Data) light is illuminated on the modem.

Interface Port. The serial communication interface board has a physical connection, or port, into which a cable is attached. This physical connection is where the functional signals and data are passed across the interface, one bit at a time, between the two communicating devices.

As explained in Chapter 4, the serial communication interface sends seven or eight bits of data out "single file." The functional signals ensure that both machines are in sync, and are prepared to receive or send data. They also help detect breaks in the communication line and other communication errors, and send back error signals to the programs when necessary.

The RS-232 serial communication standard determines what these functional signals consist of, and which cable wires carry which type of signal or data.

TROUBLESHOOTING AND REPAIR GUIDELINES

Establishing communication between two devices requires a delicate balance of software and hardware requirements. The software parameters and hardware components must be set up and functioning properly on both ends of the communication line in order for data to be sent, received, and read correctly.

The communication interface board is located within the system unit. It derives its power from the system unit's power supply. Because of this, you would never

have the usual problem of the communication board simply not working, unless of course your system unit as a whole is not working.

There are two possible problems you can experience with serial communication. The first is that communication is not established. The second is that the results of communication are garbled or incorrect in some way.

Problem: Communication Cannot Be Established

Make sure that the serial communication board is properly configured. The majority of communication interface problems are caused by a poorly configured system. When you initialize the communication interface board, particularly when it's new and you're using it for the first time, it needs to be configured to determine its address. This address essentially names the interface. Its name might be "COM1" or "COM2."

If you have bought yourself a combination board like the AST SixPakPlus, additional memory, a clock, a parallel printer port, and one or two serial communication ports are provided. You'll need to configure one of the serial ports with the "COM1" address, and the other port with the "COM2" address.

If you do not assign an address to the interface, you might never be able to establish communication, because the communication software cannot find the interface. Refer to the serial communication interface documentation for an explanation on how to determine and assign your interface address.

SOLUTIONS

- ☐ Check that your cable is properly connected to the system unit. Make sure that any loose cable screws are tightened on both ends of the connector. Refer to the section titled "Preventive Maintenance" in Chapter 1 for more information about these cable screws.

- ☐ Make sure that the proper interface cable is connected to the serial communication interface. The serial communication interface is usually a male connector, meaning its pins are exposed.

- ☐ Run the Serial Communication Diagnostic Module (described in the following section of this chapter). If the test fails, and you do not see the character you press echoed on the screen, try the test against another serial connector if you have one.

  Move the cable to the other connector and run the test again with the same address. If the test checks out here, it shows that you merely have an addressing problem within the software. Refer to the first solution explained above to fix this problem.

  If you have only one serial connector, enter a different address from the Serial Communication Diagnostic Module, for example, try COM2 if COM1 didn't work.

- ☐ If the test still fails with the other serial connector, you probably have

a problem with the interface board. Exchange the board with one from a similar system. If the board works on the other system, there is a problem with the system unit, and it should be taken in for repair. Another possibility is that there might be a conflict with another communication port. Check the devices you have installed on your system to see if there is another communication port being used. If this is the case, configure the system in a different way and retry the test.

- ☐ If the Serial Communication Diagnostic Module checks out, reload the communication program and make sure that the parameters are set correctly. Is the baud rate set correctly? Have you specified the appropriate character or word size? Is the other system expecting a parity bit to be enabled?

- ☐ Check the device with which you're trying to communicate. If it's another computer like yours, run the diagnostic program on it to see whether the communication interface is working correctly.

- ☐ Check the interface or system technical manual to see if the interface cable has any special requirements. RS-232 is a relatively loose standard. Some cable manufacturers use a different scheme for their RS-232 cable. These cables might expect data from different wires, look for other signals, or have signals tied together. If this is the case, you'll need to either modify your cable or purchase another one.

- ☐ If you're sending data through a modem, you have some extra help in troubleshooting. Modems usually have a row of red indicator lights across their front status panels (Fig. 6-6).

  If you do indeed have an indicator panel, then you can check that certain signals are functioning. For instance, when the modem is properly cabled into your computer, and the communication package is being run, the lights over TR and DSR are illuminated. This light indicates that the computer and the modem itself are ready and that the interface signals are turned on and ready to talk to each other. If these indicator lights are not on, there might be a problem with the interface cable connection or with the modem.

  When data are being sent, the indicator lights over RD, denoting "receive data," or TD, "transmit data" are illuminated. If you were to press

Fig. 6-6. Modem status panel.

a key, you would see this light illuminate, indicating that the data for the key code are indeed being sent.

- ☐ The modem might also have a test mode (check your modem documentation to see if this is the case). This test mode loops a data pattern back through the modem. In other words, it takes a pattern of characters from the computer, sends it through the modem back to itself, and then returns it to the computer. This internal loopback test checks the modem itself to make sure that the sending and receiving is taking place as it should. Use this test mode if you cannot establish communication. It helps you troubleshoot one suspected modem problem at a time.

- ☐ A *carrier* is a tone that the modem expects before it can send data. It's rather like saying "Hello" when your phone rings and you pick up the receiver. After saying "Hello," communication can be established, and information from the caller can be transmitted.

  You can hear this carrier if you have a "smart" modem that dials the phone for you. First you hear the phone ring through the modem speaker, then you hear a high-pitched tone. This audible signal is the carrier, and it tells your system that the modem is ready to accept data, either from your system or from the other system. You should see the CD (Carrier Detect) light illuminated. This indicates that the other system has answered and been given the carrier. If you do not get the carrier signal, this could indicate a wrong number, a busy signal, or some other problem at the other end.

Problem: Communication Results Are Garbled

Garbled data indicates that communication has been established, but there's a problem with the data coming over the lines. This usually has to do with one of the parameter settings such as the word size or parity. Access your communication software parameter setup mode again, and check all the settings.

If you get a mixture of garbled characters and good characters, for example, the word "login" was sent, but "loxxn" is received, the problem is usually a poor telephone connection. In this case, you should hang up and redial.

Problem: Printer Does Not Work

If your serial communication interface connects with a printer, make sure that the printer's internal switches are set up to conform to your configuration. Most printers can communicate across either a serial or a parallel interface.

SOLUTIONS

- ☐ Printers ordinarily default to the parallel interface, and you need to set a switch to use the serial communication line. Refer to your printer documentation.

- ☐ The printer also needs to know the usual communication parameters, such as the baud rate you're running, the character length, the parity, and so forth. This is set up with a DIP-switch in the printer interface board. The printer documentation details how to make the necessary settings.
- ☐ Check the interface cables and plugs. You might have plugged the cable into the wrong interface. Parallel and serial cables are physically different. Although you cannot plug a serial cable into a parallel interface, if you have a combination board, you might have gotten confused and tried it.
- ☐ MS-DOS printer commands default to the parallel printer port, which is addressed as LPT1. You must use the MODE command in order for printer commands to work with the serial printer port. The MODE command is explained in Chapter 4.

RUNNING THE SERIAL COMMUNICATION INTERFACE DIAGNOSTIC MODULE

The Serial Communication Diagnostic Module initializes any of the serial communication interfaces that you choose to test. It sends a character through the interface, loops it around, and displays it again on the screen. It checks that the interface address is working properly, that the interface itself is working properly, and that the system is receiving the same data that it is sending through the communication loop.

To run the Serial Communication Diagnostic Module, follow the instructions provided in "Running the Diagnostic Program" in Chapter 1. Then press S to initiate the Serial Communication Diagnostic Module.

Take the end of the cable that would go into the device with which you want to communicate. In the connector at this end, attach a wire between pins 2 and 3 (Fig. 6-7).

Pin 2 is the Receive data line, while pin 3 is the Transmit data line. Placing a wire bridge or jumper across these two pins causes the data that is sent out to be "looped back" and received. In other words, any character that you press

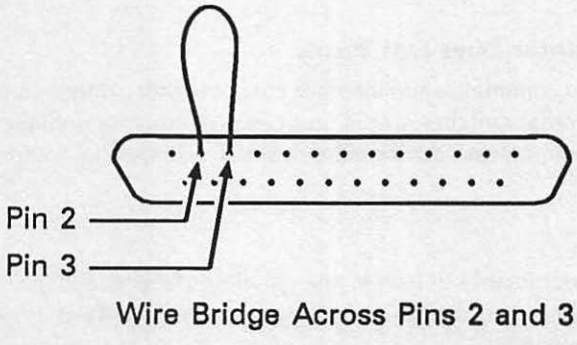


Fig. 6-7. Wire bridge across pins 2 and 3.

from your computer is sent out to the device, immediately received in through these pins and sent back to the computer, and you see the character echoed back on your screen (Fig. 6-8).

The serial communication Receive and Transmit lines on the Macintosh are pins 5 and 9. In this case, wire those two pins together, instead of pins 2 and 3. If you have a cable that has a standard 25-pin plug, jumper pins 2 and 3 on the plug itself, on the end of the cable.

There are plugs available that already have pins 2 and 3 wired together. You can purchase one of these loopback test plugs if you don't want to wire the pins together yourself. You can obtain one of these plugs at most computer supply stores. The plug goes directly into your serial port, and accomplishes the same loopback test.

This diagnostic is also a good method for checking the wires on which the transmit and receive data are going out. Suppose you're working with a device that you think transmits and receives data over pins 5 and 6, but you're not quite sure. Attach the interface cable to your serial communication port, wire pins 5 and 6 together, and run the Serial Communication Diagnostic Module. If the keystrokes you enter as part of the test echo back onto the screen as expected, then you guessed correctly about pins 5 and 6. If the test does not work, however, try another set of pins to find your transmit and receive data lines. You are probably using a special cable that routes the signal from pin 2 on the interface to pin 5 in the cable. The test will indeed work with such a cable.

The instructions to the Serial Communication Diagnostic Module appear on the screen, giving you the choice of Test 1 or Test 2. The first test is the Display Characters Loopback Test. The second test is the ASCII Values Loopback Test (Fig. 6-9).

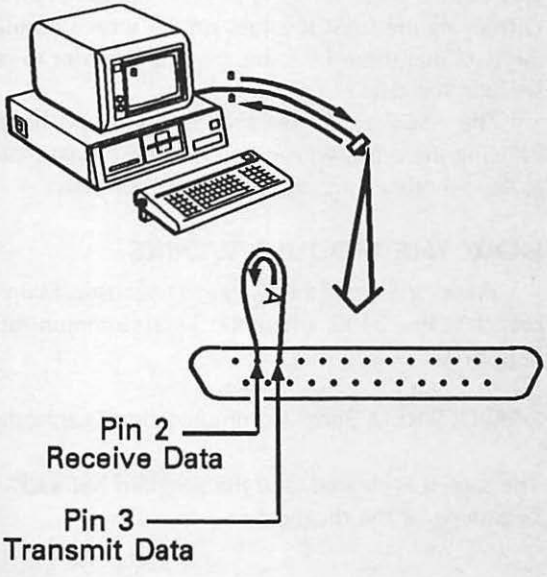


Fig. 6-8. Loopback diagram.

```
Serial Communications Diagnostic

        1...Display Character Loopback Test
        2...ASCII Values Loop Back Test

Press Esc to End Diagnostic



Wire Pins 2 and 3 together

Press Esc to return to the Serial Communications

Diagnostic Menu Enter the communications

interface to test: COM1 or COM2-?
```

Fig. 6-9. Serial communications test instructions.

The Display Characters Loopback Test lets you enter a character from the keyboard and then see it display on your screen. When you press a key, that character is sent through the communication loop, and immediately sent back and received as output on your monitor screen. This checks the serial communication interface by transmitting and receiving data.

The ASCII Values Loopback Test tests the characters that you send out for their actual ASCII values. This is particularly useful when you want to test function keys such as the carriage return, tab key, cursor arrow keys, and so forth. By their very nature, these function keys do not generate any kind of graphic character. Displaying their ASCII values on the screen during this test is a way of trapping the data that these keys are sending in order to see whether they are correctly sending the data.

The ASCII value test also works if you have programmed function keys. Pressing them lets you see all the ASCII data that are being sent over the lines to see whether they are being sent correctly.

HOW THE MODULE WORKS

Pressing S from the System Diagnostic Main Menu causes the program to branch to line 2400, where the Serial Communication Diagnostic Module begins (Fig. 6-10):

```
2400 CLS:REM Serial Communication Diagnostic Module
```

The screen is cleared, and the program has a REMARK statement indicating the beginning of the diagnostic.

```
2400 CLS:REM Serial Communication Diagnostic Module
2410 PRINT TAB(4) "Serial Communication Diagnostic":PRINT
     PRINT TAB(11) "1...Display Character Loopback Test":
     PRINT TAB(11) "2...ASCII Values Loopback Test" PRINT:
     PRINT TAB(4) "Press Esc to End Diagnostic"
2420 GOSUB 2930:IF ASC(A$)=27 THEN GOTO 40
2430 IF A$="1" THEN SW=0:GOTO 2460
2440 IF A$="2" THEN SW=1:GOTO 2460
2450 BEEP:GOTO 2420
2460 CLS:PRINT "Wire Pins 2 and 3 together":PRINT:PRINT
     "Press Esc to return to the Serial Communication
     Diagnostic Menu"
2470 PRINT:INPUT "Enter the Communication Interface to Test:
     COM1 or COM2-";C$
2480 IF C$="" THEN C$="COM1"
2490 CM$=C$+":,,,,RS,DS0"
2500 OPEN CM$ AS #1
2510 OPEN "scrn:" FOR OUTPUT AS #2
2520 CLS:LOCATE ,,1
2530 GOSUB 2930:IF ASC(A$)=27 THEN CLOSE #1:CLOSE #2:
     GOTO 2400
2540 IF A$<>"" THEN PRINT #1,A$
2550 IF EOF(1) THEN 2530
2560 IF SW=1 THEN 2580
2570 B$=INPUT$(LOC(1),#1):PRINT B$;:GOTO 2530
2580 B$=INPUT$(LOC(1),#1):PRINT ASC(B$);" ";:GOTO 2530
```

Fig. 6-10. Serial communication interface diagnostic module listing.

Line 2410 prints the instruction menu with two possibilities:

```
2410 PRINT TAB(4) "Serial Communication Diagnostic":PRINT
     PRINT TAB(11) "1...Display Character Loopback Test":
     PRINT TAB(11) "2...ASCII Values Loopback Test" PRINT:
     PRINT TAB(4) "Press Esc to End Diagnostic"
```

Test 1 is a loopback test with standard graphic display characters. Test 2 is a loopback test that displays the characters in their ASCII values. The instructions also state that pressing the ESC key ends the diagnostic.

Line 2420 calls up subroutine 2930, which looks for keyboard input:

```
2420 GOSUB 2930:IF ASC(A$)=27 THEN GOTO 40
```

In addition, if the ASCII value of A$ is 27, the program branches back to line 40, the System Diagnostic Main Menu.

Line 2430 checks to see if A$ is equal to one. If it is, it sets software switch variable SW to zero:

```
2430        IF A$="1" THEN SW=0:GOTO 2460
```

It branches to line 2460, where processing begins.

Line 2440 checks to see if A$ is equal to two. If it is, variable SW is set to one instead of zero, and the program branches to line 2460:

```
2440        IF A$="2" THEN SW=1:GOTO 2460
```

Line 2450 beeps and returns to line 2420 if you have entered an invalid keystroke:

```
2450        BEEP:GOTO 2420
```

One, two and ESC are the only valid keystrokes at this point.

Line 2460 begins the actual processing code. The screen is cleared again, and then prints a message reminding you to wire pins 2 and 3 together (or pins 5 and 9 on the Macintosh):

```
2460        CLS:PRINT "Wire Pins 2 and 3 together":PRINT:PRINT
            "Press Esc to return to the Serial Communication
            Diagnostic Menu"
```

After this message is displayed, the program goes on to line 2470. This skips a line on the display and then issues an INPUT statement:

```
2470 PRINT:INPUT "Enter the Communication Interface to Test: COM1 or
     COM2-";C$
```

You can have two serial communication interfaces on a system, and these are usually addressed as COM1 or COM2. The INPUT statement asks if you want to test COM1 or COM2. Your answer goes into a variable called C$.

Line 2480 checks to see if RETURN has been pressed. If it has, C$ is automatically assigned to the default device of COM1:

```
2480 IF C$="" THEN C$="COM1"
```

Line 2490 builds the parameters for the communication device's OPEN statement. Variable C$, which contains the value of either COM1 or COM2, is assigned to variable CM$. The four commas are blank fields that pertain to testing for signals that are not available in a loopback test. Such signals are more commonly used when working with a printer or modem:

```
2490 CM$=C$+":,,,,RS,DS0"
```

This line also contains two other variables. RS suppresses the Request to Send signal, and DS0 disables the Data Set Ready signal. These are modem- or printer-related signals which cause this diagnostic to fail unless they are not disabled.

Line 2500 opens the communication device, and the parameters that were just set are in variable CM$ as file #1. The output device in this case is the screen. The display is opened as a file for output in line 1830, using an OPEN statement:

```
2500 OPEN CM$ AS #1
2510 OPEN "scrn:" FOR OUTPUT AS #2
```

"SCRN" designates that the monitor display is being used for output, and is assigned as file #2.

Line 2520 clears the screen along with all the instructions, places the cursor at position 1,1, and turns the cursor on:

```
2520 CLS:LOCATE ,,1
```

Line 2530 branches to the keyboard input subroutine at line 1930. If ESC is pressed, the two files are closed and the program returns to line 2400. Line 2400 clears the screen and then displays the Serial Communication Diagnostic Menu:

```
2530 GOSUB 2930:IF ASC(A$) =27 THEN CLOSE #1:CLOSE #2: GOTO
     2400
```

If a key other than ESC is pressed, the program proceeds to line 2540:

```
2540 IF A$< >"" THEN PRINT #1,A$
```

This checks if A$ is not equal to null, meaning no key has been pressed. When a valid key is pressed, the program prints file #1, writing the character to the communication buffer. The program sends A$, the character you pressed, over the communication line. It is followed by a semicolon, which kills the carriage-return line feed.

Line 2550 checks for an end-of-file condition on file #1:

```
2550 IF EOF(1) THEN 2530
```

If it is indeed the end of the file, the program returns to line 2530 to check for another keystroke. This statement prevents a communication buffer overflow, which would otherwise occur after 255 characters.

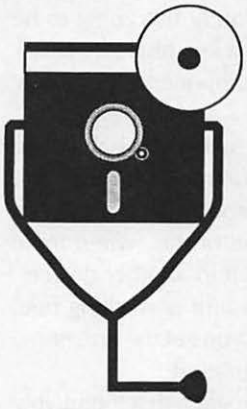
Line 2560 checks variable SW for a value of one. Recall that SW indicates which of the two tests is being run, either the graphic character display, or the ASCII value display. If SW is equal to one, the program branches to line 2580. If SW is not equal to one, the program falls through to line 2570:

```
2560 IF SW=1 THEN 2580
2570 B$=INPUT$(LOC(1),#1):PRINT B$;:GOTO 2530
2580 B$=INPUT$(LOC(1),#1):PRINT ASC(B$);" ";:GOTO 2530
```

At line 2570, the string input of location 1 of the COM buffer is assigned to variable B$. The variable B$ is printed on the display, again with the semicolon which kills the carriage-return line feed.

The program returns to line 2530 for another character input. Line 2530 receives the character input and checks for ESC being pressed. Line 2540 sends the character through the communication interface, and lines 2570 and 2580 intercept the character and redisplay it on the screen. This routine works out the circuitry of the serial communication board.

If variable SW is equal to one at line 2560, the program proceeds to line 2580. Again, the character in the COM buffer is assigned to variable B$. Instead of printing the character, however, the ASCII value of the character is displayed on the screen. The semicolon kills the carriage-return line feed, and adds a blank space to prepare for the next character. Once this is done, the program returns to line 2530, which lets you either send another character, or press ESC to return to the Serial Communication Diagnostic Menu.



Chapter 7
Post-Repair Test and Burn-In

The previous chapters have covered individual components of your personal computer system. Problems that can occur with a particular subsystem have been examined and treated. This chapter looks at the computer system as a whole, and provides a diagnostic module that exercises the entire system to check for problems.

If something does go wrong with your computer and you repair it or take it in for professional servicing, the first thing you want to do when you get it back is check it out, i.e., take it for a "test drive." This chapter describes and provides methods for testing your repaired system to make sure everything works as it should.

WHY ARE TESTING AND BURN-IN NECESSARY?

When you pick up your computer from servicing, treat it like you would an automobile repair. If you had your car in for servicing with a mechanic, you wouldn't just pay your bill and drive off. You would probably want to test drive it, or have the mechanic show you what work was done. If you had a brake problem, you'd check to see that the brakes now worked properly. If oil was leaking, you'd want to see that the leak was stopped. Once you were satisfied that the problem was fixed, you would pay your bill and drive away.

You should do the same when you pick up your computer from being repaired. Have the service technician describe what was done, explain why it was done,

and demonstrate that the problem has been fixed. Keep in mind, however, that even while checking the component at the repair shop, you're not going to be able to check all system functions. You'll probably just see a few tests that check the specific problem, showing you that the particular component is working correctly.

For example, if your system unit had a problem, you would probably take just the system unit in for servicing, leaving the printer, external disk drive, and modem at home. But your computer as a whole is a system, consisting of a number of devices and peripherals that all work together. Because of this, when there is a problem in one device, it can sometimes manifest itself in another device.

Even when the technician shows you that the system unit is working fine, therefore, you won't know the status of the big picture until you get the unit home and assembled with the entire system in its usual configuration.

The final diagnostic module provided in this book is a program that thoroughly exercises your system when it's new, when you've changed its configuration, or when you get it back from service. This Exerciser Module checks various components of your system and leaves it running in a continuous mode so that you could let it burn in for several hours, even overnight.

Burn-in fully exercises the system and should be done for 12 to 24 hours. If there is a problem with the computer at this point, it's better to discover it during burn-in than while entering and processing valuable data. This is a good practice widely used in industrial manufacturing and product testing, and it serves to test the system thoroughly.

THE FUNCTION OF THE EXERCISER

The System Exerciser Module calls on diagnostic tests that are already part of individual component diagnostic modules elsewhere in the System Diagnostic Program. Specifically, it branches back to routines in the Monitor Diagnostic Module, in the Printer Diagnostic Module, and in the Disk Drive Diagnostic Module.

When you first call up the System Exerciser Module, a menu is displayed. This menu asks you to specify the devices you wish to test, with the choices of the monitor, printer, and disk drives. If you have only one disk drive, you can specify that. If you do not own a printer, or do not wish to exercise your printer, you can leave it out of the Exerciser run. You are able to set up these component parameters in whatever combination is convenient and appropriate for your system.

Once configured according to what you wish to exercise, the System Exerciser Module immediately begins to perform the continuous test on the specified devices for as long as you wish the test to run.

If you test your monitor, the Sliding Alpha Scroll Test, the Character Set Test, and the Fill Screen Display Test are run. For more information on these tests, refer to Chapter 3.

If you test your printer, the Sliding Alpha Test is run for ten lines, the Echo Character Print Test is run on one line, and the Horizontal Tab Test and Line Space

Test are each run through once. For more information on these tests, refer to Chapter 4.

If you test your disk drive, the read/write test is run, with records filling your disk and then being read back. The disk drives include some of the most delicate components of your entire system, and should be tested after repair. For more information on the read/write test, refer to Chapter 5.

The System Exerciser begins with the monitor diagnostics, then goes to the printer diagnostics, and finally the disk drives. When this is complete, it cycles back through to the monitor to start the process again. This continues in a round-robin fashion until you press the appropriate function key to interrupt the Exerciser.

Benefits of Test and Burn-In

The Exerciser tests the three major components of your system: the monitor, disk drive(s), and printer. The Exerciser shows you almost immediately if there is a problem with your printer interface. If there's a marginal or intermittent problem of some sort anywhere else in the system, it will show up within a 24-hour burn-in run of the Exerciser.

The System Exerciser Module runs under BASIC, just like the rest of the System Diagnostic Program. Because of this, it is testing out an extensive section of your system unit's machine instruction set.

RUNNING THE SYSTEM EXERCISER

When you're ready to run the Exerciser on your system, make sure that the components you wish to test are ready for hours of testing. If you plan to test your printer, make sure it is loaded with enough continuous form paper and that there is enough ribbon to last throughout the duration of the test. Blank, formatted diskettes should be in each of the drives being tested. You might want to dim your monitor, but there is no danger from etching (see Chapter 3 for more information about etching), because the display is continually changing.

To run the System Exerciser, follow the instructions provided in the section titled "Running the Diagnostic Program" in Chapter 1. Then, from the System Diagnostic Main Menu, press E to initiate the Exerciser Module.

The screen clears and you are asked if you wish to test the monitor, yes or no. Enter y or n, in either uppercase or lowercase, and press RETURN.

Then you are asked if you wish to test the printer, yes or no. Again, enter y or n. If you enter y, you are asked to enter the carriage length of the printer you want to test, either 80 or 132 columns.

Finally, you are asked if you wish to test your disk drives. Enter either y or n. If you answer yes, you are then asked for the number of drives you would like to test. You can test all disk drives on your system at one time. Enter 1, 2, or 3. Then, you are asked to enter the drive IDs for each drive being tested. This would be the disk drive designator, such as A for the A-drive, B for the B-drive, or C for the hard disk drive on the IBM system (Fig. 7-1).

You can choose any combination of parameters for exercising that you choose. Once you've entered these parameters, the program begins exercising the system.

```
Test Monitor (Y or N)-? y
Test Printer (Y or N)-? y
Enter the Printer Carriage Length (80 or 132)-? 80
Enter the printer port to test (LPT1 or LPT2)-? lpt1
Test Disk Drive(s) (Y or N)-? y
How many drives to test-? 2
Enter Drive ID (A,B,C)-? a
Enter Drive ID (A,B,C)-? c



Press F1 to End Test
```

Fig. 7-1. Exerciser parameter setup.

If you've included the monitor in the Exerciser, the screen is filled with the Fill Screen Display Test, then clears and runs through the Character Set Test, and then the Sliding Alpha Scroll Test for 50 repetitions.

If you have included the printer in the exerciser, once the monitor test is complete, your printer starts up and prints ten repetitions of the Sliding Alpha Test. It then runs through the Display Character Print Test. Then, it prints a line of Es for the Echo Character Print Test. The Horizontal Tab Test and Line Space Test are also run, so the printer is thoroughly tested.

When the printer tests are completed, and if you have selected the disk drives to be included in the Exerciser, the Disk Drive Diagnostic is run on each disk drive you have specified.

When this is completed, the program loops back and tests the monitor again, then the printer, and so forth. The Exerciser runs in a continuous loop like this, running the tests over and over again. To halt the Exerciser, press F1 on an MS-DOS system, (the ✤ Cloverleaf key) followed by a period on a Macintosh system, or CTRL C on a CP/M system. This ends the System Exerciser Module, and a message indicates the number of passes the program has cycled through. You are prompted to enter any character to end the diagnostic. Once this is done, the System Diagnostic Program returns to display the System Diagnostic Main Menu.

NOTE: When you press CTRL C to interrupt the Exerciser on the CP/M system, the System Diagnostic Program is completely halted, and the

MBASIC OK prompt is displayed. Enter GOTO 2000. This resumes the diagnostic at the point where it displays the Exerciser program results.

If you interrupt the Exerciser while it's running the disk drive test, you might see a file called "TEST" in your directory. You should remove this file. Normally the diagnostic removes this file after it completes its run.

If one of the tests fails, the Exerciser halts. So, if your printer runs out of paper, or if something happens to the disk drive, the program stops at the point at which it had the problem. The particular diagnostic module that the Exerciser was running at the time that problem took place will display an error message describing the problem. Take the appropriate action.

HOW THE SYSTEM EXERCISER MODULE WORKS

The code for the System Exerciser begins at Line 2600 (Fig. 7-2). When E is pressed from the System Diagnostic Main Menu, the program branches to line 2600.

Line 2600 clears the screen and sets parameters PS and EX. PS is a counter variable that keeps track of the number of passes the Exerciser runs through. At the beginning, variable PS is set to zero. This ensures that if you start the Exerciser several times in a row, the number of passes from these different runs are not cumulative:

```
2600 CLS:PS=0:EX=1:PRINT TAB (10) "Exerciser Diagnostic":
     PRINT
```

Next, the flag EX is set to one. This flag is found throughout the other modules that the Exerciser uses. It lets the program know that the Exerciser is running, causing the tests to run automatically without waiting for interactive user input. When running the Monitor Diagnostic interactively, for example, EX is set to zero, and the user is prompted to supply certain inputs. On the other hand, when running the Monitor Diagnostic through the Exerciser Module, EX is set to one, telling the program not to prompt and wait for keyboard entry. This allows the Exerciser to run continuously and automatically until interrupted.

The last thing line 2600 does is display the diagnostic title on the screen. Finally, a blank line is skipped.

Line 2610 activates function key F1 so that the Exerciser can be halted. Line 2620 displays a message to this effect on the last line of the screen:

```
2610 KEY(1) ON:ON KEY(1) GOSUB 2900
2620 LOCATE 25,5: PRINT "Press F1 to End Test":LOCATE 1,1
```

GOSUB 2900 is a trap statement. It sets the program up so that if F1 is pressed, it goes to the subroutine at line 2900 which interrupts the Exerciser.

NOTE: While IBM-type systems are stopped with F1, the Macintosh version

```
2590 REM Exerciser Diagnostic Module
2600 CLS:PS=0:EX=1:PRINT TAB(10)"Exerciser Diagnostic":
     PRINT
2610 KEY(1) ON:ON KEY(1) GOSUB 2900
2620 LOCATE 25,5:PRINT "Press F1 to End Test":LOCATE 1,1
2630 INPUT "Test Monitor (Y or N)-";MN$:PRINT
2640 INPUT "Test Printer (Y or N)-";PT$
2650 IF PT$="N" OR PT$="n" THEN GOTO 2700
2660 PRINT:INPUT "Enter the Printer Carriage Length (80 or
     132)-";WP
2670 IF WP<>80 OR WP<>132 THEN WP=80
2680 PRINT:INPUT "Enter the printer port to test (LPT1 or
     LPT2)-";PR$
2690 IF PR$<>"lpt2" OR PR$<>"LPT2" THEN PR$="lpt1":
     PR$=PR$+":"
2695 WIDTH PR$,WP
2700 PRINT:INPUT "Test Disk Drive(s) (Y or N)-";DD$(0)
2720 IF DD$(0)="N" OR DD$(0)="n" THEN GOTO 2760
2730 PRINT:INPUT "How many drives to test-";DD
2740 FOR I=1 TO DD
2750 PRINT:INPUT "Enter Drive ID (A, B, C)-";DD$(I):NEXT I
2760 IF MN$="N" OR MN$="n" THEN GOTO 2790
2770 A$="X":A=50:DEF SEG=&HB000:GOSUB 930
2780 LOCATE 25,5:PRINT "Press F1 to End Test":LOCATE 1,1
2790 IF PT$="N" OR PT$="n" THEN GOTO 2830
2800 OPEN PR$ AS #1
2810 A$="E":W=WP:X=10:GOSUB 1590
2820 CLOSE #1
2830 IF DD$(0)="N" OR DD$(0)="n" THEN GOTO 2890
2840 LOCATE 25,5:PRINT "Press F1 to End Test":LOCATE 1,1
2850 FOR I=1 TO DD
2860 D$=DD$(I)+":TEST":CLS
2870 GOSUB 2060
2880 NEXT I
2890 PS=PS+1:GOTO 2760
2900 CLS:CLOSE #1:PRINT TAB(10)"Exerciser Diagnostic":PRINT
2910 PRINT TAB(10)"Total Passes through Exerciser-";PS
2920 GOSUB 2940:EX=0:KEY(1) OFF:ON ERROR GOTO 0:GOTO 40
```

Fig. 7-2. System exerciser diagnostic module listing.

is stopped by pressing ✣. (CLOVERLEAF-period), and the CP/M version is stopped by pressing CTRL C, then GOTO 2000.

Line 2630 is an INPUT statement that asks whether or not the monitor is to be tested:

```
2630 INPUT "Test Monitor (Y or N)-";MN$:PRINT
```

The y or n value entered is placed into variable MN$.

Line 2640 is another INPUT statement asking whether or not the printer is to be tested:

```
2640 INPUT "Test Printer (Y or N)-";PT$
```

The y or n input goes into variable PT$.

Line 2650 checks to see if the printer is to be tested. If it is not, there is no need to ask for the carriage length, so the program branches to Line 2700:

```
2650 IF PT$="N" OR PT$="n" THEN GOTO 2700
```

If the printer is to be tested, and has been selected with a y, the program proceeds to Line 2660. Line 2660 is an INPUT statement asking for the carriage length of the printer, which could be either 80 or 132 columns wide. This value goes into variable WP:

```
2660 PRINT:INPUT "Enter the Printer Carriage Length (80 or
     132)-";WP
```

Line 2670 ensures that only 80 or 132 is entered. A value other than 132 defaults to 80:

```
2670 IF WP< >80 OR WP< >132 THEN WP=80
```

Line 2680 checks to see which printer port is to be tested, LPT1 or LPT2:

```
2680 PRINT:INPUT "Enter the printer port to test (LPT1 or
     LPT2)-";PR$
```

Line 2690 ensures that only LPT1 or LPT2 is entered. Anything other than LPT2 defaults to LPT1:

```
2690 IF PR$ < > "lpt2" OR PR$< >"LPT2" THEN PR$="lpt1":
     PR$=PR$+":"
```

The width of LPT1, the printer, is then also set according to the carriage length input.

Line 2700 is an INPUT statement which asks whether or not the disk drive is to be tested:

```
2700 PRINT:INPUT "Test Disk Drive(s) (Y or N)-";DD$(0)
```

The input goes into DD$(0), which is the first entry of an array.

Line 2720 checks to see if the disk drive is not to be included in the Exerciser:

```
2720  IF DD$(0)="N" OR DD$(0)="n" THEN GOTO 2760
```

If the disk drive is not included, the program branches to line 2760. If the disk drives are to be tested, line 2730 provides an INPUT statement that asks how many disk drives are to be tested:

```
2730  INPUT "How many drives to test-";DD
```

This information is placed into variable DD.

Line 2740 is the beginning of a FOR-NEXT loop:

```
2740  FOR I=1 TO DD
```

The loop lasts for the value of DD, which is the number of drives specified to be tested. In other words, if 2 is input into line 2730, there will be two iterations of the loop.

Line 2750 is an INPUT statement asking for the drive ID, A, B, C, etc. This input is placed into DD$(I). The first pass through the loop goes into DD$(1), the second pass goes into DD$(2), and so forth:

```
2750  PRINT:INPUT "Enter Drive ID (A, B or C)-";DD$(I):NEXT I
```

The NEXT I statement at the end of line 2750 corresponds to the FOR in line 2740.

Line 2760 begins the body of the Exerciser, where the program starts checking which options have been selected and acts upon them. Line 2760 is an IF statement, checking to see if whether the monitor, variable MN$, was selected to be included in the Exerciser. If it was not, the program branches to line 2760 to bypass the monitor routine:

```
2760  IF MN$="N" OR MN$="n" THEN GOTO 2790
2770  A$="X":A=50:DEF SEG=&HB000:GOSUB 930
```

If the monitor was selected, the program falls through to line 2780:

```
2780  LOCATE 25,5:PRINT "Press F1 to End Test":LOCATE 1,1
```

Line 2780 redisplays the "Press F1 to End Test" message at the bottom of the screen, because certain tests might erase this message during processing.

The Exerciser runs in an unattended, automatic mode, so certain information needs to be passed to the Monitor Diagnostic Module when used as part of the Exerciser in order to satisfy certain inputs and variables. Line 2770 does this by setting variable A$ equal to X. The character X is therefore used in the Display Test to fill the screen:

```
2770  A$="X":A=50:DEF SEG=&HB000:GOSUB 930
```

Another variable used in the Monitor Diagnostic is A, used in the Sliding Alpha Scroll Test. Variable A tells the program how many times to display the sliding alpha line. For the Exerciser, A is set to 50, to display the line 50 times.

Also needed is to define the segment at &HB000. This is a housekeeping function for the Fill Screen Display Test, and is explained in Chapter 3. The end of line 2770 is the GOSUB statement to line 930.

If you look at the complete program listing, you'll see that line 930 is where the Monitor Diagnostic Module resides. Starting at line 930 serves to bypass the interactive questions that have already been taken care of earlier in the process.

When the Monitor Diagnostic tests are completed, and the RETURN statement in Line 1280 is reached, the program returns to line 2790. This line checks to see whether you have chosen the printer to be included in the Exerciser. If you have not, the program branches to line 2830:

```
2790  IF PT$="N" OR PT$="n" THEN GOTO 2830
```

If the printer is included, the program falls through to line 2800:

```
2800  OPEN #1 PR$ AS #1
```

Line 2800 opens the printer for future write statements.

Line 2810 sets variables for the Printer Diagnostic Module in much the same way that the monitor variables were set in line 2770. A$ is set to E as the character to be echoed in the Echo Character Print Test:

```
2810  A$="E":W=WP:X=10:GOSUB 1590
```

Earlier, in line 2660, variable WP was set to the length of your printer's carriage. Here in line 2810, the value in WP is applied to variable W for the Sliding Alpha and the Display Character Print tests in the Printer Diagnostic Module.

X is a variable used in the Sliding Alpha Test to contain the number of lines to print. Line 2810 sets X equal to 10, so ten lines are printed in sliding alpha. The end of line 2810 is the GOSUB statement branching the program back to line 1590. This is close to the beginning of the Printer Diagnostic Module. It runs through the designated printer tests of the module, and then when it gets to line 2030, the RETURN statement brings the program back to line 2820, which closes the printer:

```
2820  CLOSE #1
```

Line 2830 checks to see if the disk drive was selected to be tested as part of the Exerciser. If it was not selected, the program branches to line 2890:

```
2830  IF DD$(0)="N" OR DD$(0)="n" THEN GOTO 2890
```

If it was selected, the program falls through to line 2840. This line makes sure

that the "Press F1 to End Test" message is on the screen:

```
2840 LOCATE 25,5:PRINT "Press F1 to End Test":LOCATE 1,1
```

Line 2850 is a FOR-TO loop. Variable DD is the number of disk drives to be tested:

```
2850 FOR I=1 TO DD
```

If 2 had been entered at the INPUT statement in line 2730, for example, this FOR-TO loop would cause the disk drive test to loop through twice in order to test both drives.

Variable D$ is used in an OPEN statement in the Disk Drive Diagnostic Module. Line 2860 sets D$ to DD$(I) which starts at one, being the identification of the first disk drive. On the second pass through the FOR loop, it will be two, the identification of the second disk drive, and so forth:

```
2860 D$=DD$(I)+":TEST":CLS
```

Once D$ is set equal to DD$(I), it is joined with a colon and the word "TEST." This means that a new file named "TEST" is being created onto the disk drive being tested. The end of line 2860 clears the screen, getting rid of leftover displays from the printer or monitor tests.

Line 2870 is the GOSUB to line 2060 where the Disk Drive Diagnostic Module resides:

```
2870 GOSUB 2060
```

It runs through this module until it hits the RETURN statement at line 2250. At this point, the program returns to line 2880, which is the NEXT statement corresponding to the FOR-TO loop in line 2850:

```
2880 NEXT I
```

If you are testing a second or third disk drive, line 2880 sends the program back through the loop again to do this.

Once the FOR-TO loop is satisfied, the program falls through to line 2890. This adds one to the variable PS. PS keeps track of the number of passes the exerciser runs through:

```
2890 PS=PS+1:GOTO 2110
```

If you leave it running all night, for example, when you interrupt the program, a message is displayed indicating this final number of Exerciser runs.

Then the program branches to line 2760, which starts the Exerciser routine

over again. This is a continuous loop until you press F1. When you press F1 to interrupt the Exerciser, it goes to line 2900, which clears the screen:

```
2900 CLS:CLOSE #1:PRINT TAB(10)"Exerciser Diagnostic":PRINT
```

If you're interrupting the Exerciser in the middle of the Disk Drive Test, it closes File #1. The "Exerciser Diagnostic" title is printed on the screen.

Then, in line 2910, the value in variable PS described in line 2890 is printed:

```
2910 PRINT TAB(10)"Total Passes through Exerciser-";PS
```

This indicates the total number of runs the Exerciser has gone through this time.

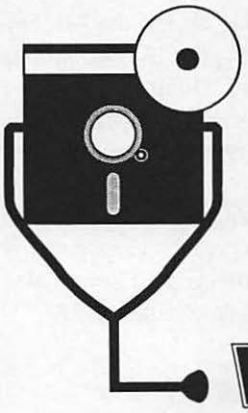
The program then proceeds to line 2920, which goes to the press-any-key-to-end test subroutine at line 2920:

```
2920 GOSUB 40100:EX=0:KEY(1) OFF:ON ERROR GOTO 0:GOTO 40
```

As soon as you press a key to end the Exerciser, the program sets variable EX back to zero. EX, you remember, is the Exerciser flag that was set to one throughout the Exerciser Module. When the Exerciser is set to zero, you can interactively run the individual diagnostic modules within themselves. Then, F1 is turned off. This prevents the F1 trap from taking place in another test, for example, in the middle of the Keyboard Test.

The ON ERROR GOTO 0 message turns off the error trap that was set in the Disk Drive Test.

Finally, the GOTO statement sends the program back to the System Diagnostic Main Menu to let you choose another module or to exit from the program.



Chapter 8
Writing Diagnostics for Other Peripherals

From your experience with computers and from reading the previous chapters, you are probably convinced that computers are mere machines with parts that get dirty, go out of alignment, wear out, or go bad. When this happens, and you have a problem somewhere in your system, you want to be able to isolate the problem, identify it, and perform the necessary repairs. The System Diagnostic Program presented in this book provides you with the means for doing this with your keyboard, monitor, printer, disk drives, and serial communication interface.

However, if you have other peripherals, or when you buy additional peripherals not covered by this program, you need to know how to write new diagnostic modules for them. In this way, the System Diagnostic Program can grow with your computer, and new components are not left out of the diagnostic loop that benefits the rest of your standard system.

Using the mouse as an example, this chapter demonstrates and explains how to identify the basic functions of a new device, the types of problems it might experience, how to find the information needed to develop the diagnostic for testing the device functions, and how to integrate the new diagnostic module into the System Diagnostic Program. This assumes that you are conversant in the BASIC programming language in which the System Diagnostic Program is written.

If you are not currently familiar with BASIC and would like to be, obtain one of the many excellent books on learning BASIC. There are also BASIC classes available at many computer stores, community colleges, and adult education programs.

TYPES OF MICROCOMPUTER PERIPHERALS

There are numerous other peripheral devices available on the market. For example, you can buy a game paddle controller board for your IBM PC which allows you to attach joysticks for certain interactive games. There are also new, sophisticated, video display boards that give your monitor extremely high resolution to allow for fancy graphic capabilities. Other types of input devices include the bar code reader, light pen, and mouse.

There are many other interesting peripherals available for personal computers, such as digitizer tablets, voice recognition devices that can recognize voice patterns and respond to verbal commands, voice synthesizers, cameras, and laser disks. And more and more peripherals are constantly being made available.

DEVELOPING A NEW DIAGNOSTIC MODULE

Start thinking about developing a new diagnostic module for your new peripheral as soon as "the new wears off." This occurs after you've acquired a new device, attached it to your system, played with it, worked with it, and are now familiar with how it operates and what it can do for you. You now consider it an integral part of your permanent system, and you would not want to have to do without it. It is at this point that you should develop a diagnostic.

Learn the Functions of the Device

The first thing to do when developing a new diagnostic module for a new peripheral is to learn all you can about how the device functions. Find out specifically what the device is intended and designed to do. Why did you buy it? What does it do for you? Experiment with the device. Explore its possibilities. Try to do real work with it.

Read the Device Manual

Read the device documentation to learn about capabilities you might not have been aware of. You should do this as soon as you obtain the new peripheral. The manual typically includes two major sections: one covering actual operation of the device; the other providing technical information you'll find useful in the actual development and programming of the diagnostic module. NEVER throw away any manuals or other documentation that come with a device.

The operation section describes functions and uses of the device, its special features, how to use it most efficiently, how to program it, and how to integrate it with the other system components and software.

The technical section provides installation instructions, describes the various options that are available, explains how to set these options, and helps you set up your existing hardware and software in order for the new device to work.

Another section you'll probably find in the manual is a troubleshooting guide. This guide outlines problems that can occur and provides solutions in terms of actions to take in order to alleviate the problems. Be sure to look the

troubleshooting guide over in the beginning, so you can get a good idea of typical problems that you can test for. You want to develop your diagnostic *before* trouble strikes and you have to refer to the troubleshooting guide. Don't wait until there is a problem before looking at the troubleshooting guide.

In addition, the troubleshooting guide provides information only about the particular device. It does not take into account the software you're running, your particular hardware configuration, or your interface type. In other words, the troubleshooting guide doesn't know your particular system. It would be impossible for any troubleshooting guide to consider all these items, simply because there are infinite possibilities and combinations. So, by its very nature, the troubleshooting guide cannot specifically address your particular system setup.

Because of this, it is more beneficial for you to develop a troubleshooting diagnostic of your own for the device as it works with your particular system. You can then integrate it into the System Diagnostic Program as a new module.

Decide What to Test For

Once you know the functions of the device, you can simply create tests for the inverse of these functions. In other words, if a device is supposed to perform a particular function, test for the possibility that it is not performing this function.

You will need to do further research, either in your system technical manual, the device documentation, or your BASIC programming manual to find out how to program checks for particular functions.

Create a Program Flowchart

Once you know what functions and problems you would like to test, you need to develop the program flowchart. This flowchart becomes your program design and the "road map" to writing the actual diagnostic code.

You want your flowchart to break down the different test routines that will exist within the module, as well as any GOTO statements and FOR-TO loops. You will also want to detail in your flowchart any new program functions and commands you have learned through your research on the particular device. Be as detailed as you can in your flowchart. This will make coding a lot easier.

Flowcharting is the method by which the process or "flow" of a program can be charted graphically. Arrows connect process and decision boxes together to represent the direction (or any change of direction) in which a program or segment of a program will flow. Although there are a great number of flowcharting symbols, only a small set of these is needed to accurately represent the program flow.

You should create your flowchart in two stages. In the first stage, express your ideas in English. Figure 8-1 is a flowchart for a segment of a program asking if a beep is to be sounded or not.

In the second stage, translate your English into BASIC code (Fig. 8-2). Now follow the arrows and add the line numbers to the code to write the program shown in Fig. 8-3.

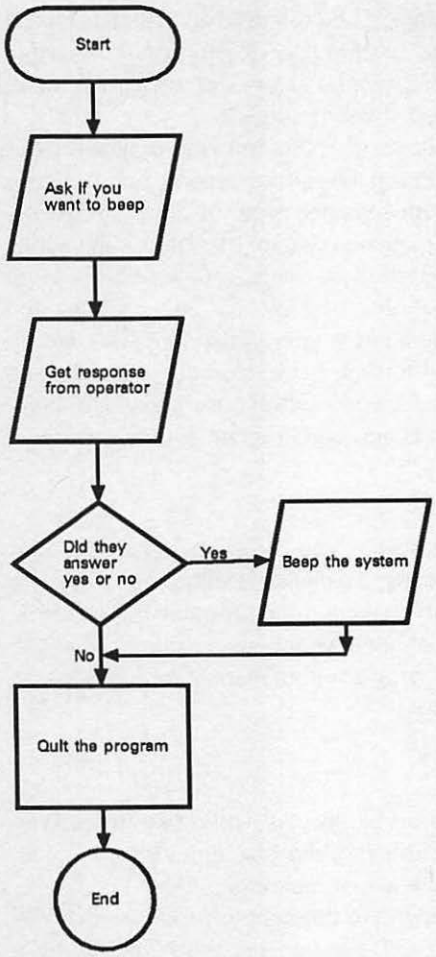


Fig. 8-1. Flowcharting, stage 1.

Flowcharts can be as detailed or sketchy as you like. The objective is to establish the program's flow and logic to the point where you can more easily write the program.

Write the Diagnostic Module

When you've diagrammed your program flowchart, you're finally ready to begin the actual coding. The first thing to remember when you're developing a new test is to make sure your system is healthy. Do not develop a test on a malfunctioning system, or with a peripheral that is experiencing a problem. If you were to develop a test on a malfunctioning system, as soon as you had the problem fixed, the results of the test might be null and void.

When exploring new BASIC instructions for your device, follow examples presented in your manual to test them out. Get a good feel for what these instructions do before "hacking" your System Diagnostic Program. After creating

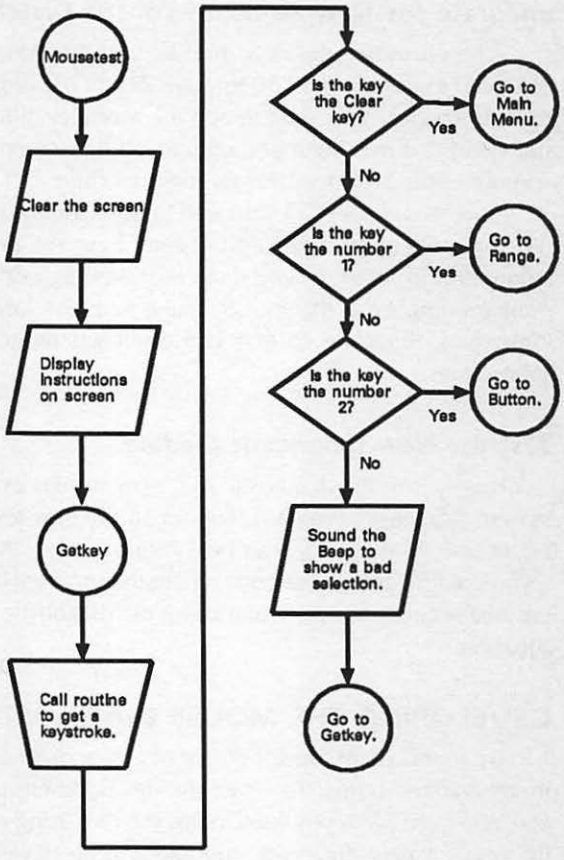


Fig. 8-2. Flowcharting, stage 2.

a flowchart of your ideas and converting them to BASIC code, write a stand-alone version of your diagnostic module (a version that does not involve any other programs). Test it by itself before you integrate it into the System Diagnostic Program. Refer to the code of existing modules in Appendices A, B, or C to familiarize yourself with the proper techniques.

```
10 CLS
20 INPUT "Do you want to hear a beep";A$
30 IF A$<>"y" THEN GOTO 50
40 BEEP
50 END
```

Fig. 8-3. Writing the program to the flowchart.

Integrate the New Module with the System Diagnostic Program

When inserting the new module into the System Diagnostic Program, you first need to modify lines 50 through 210 of the program, where the Main Menu Module resides. You need to add the module's title and identifier to the PRINT statement that lists the menu options on the screen. You also need to insert the appropriate IF statement for the new module.

Once this is done, all you need to do is add the code to the program. If you're programming an IBM or CP/M system, have the line numbers of the new code begin around 3000, or wherever the last diagnostic module stopped. If you're programming on a Macintosh, make sure the labels you're using are unique throughout the code so that you don't get program branching to the wrong subroutines.

Test the New Diagnostic Module

Finally, test the diagnostic as it now resides as an integrated module of the System Diagnostic Program. Try out all the new tests with as many possibilities as you can think of. Try your best to find "bugs" that will "break" the program. If you do, find and fix the code where the problem occurred. This testing process ensures accurate results when using the diagnostic in an actual troubleshooting situation.

DEVELOPING THE MOUSE DIAGNOSTIC MODULE

Up to this point, development of a new diagnostic module has been treated in generalized terms. To make the development process more clear to you, a concrete example is provided in this section, using a mouse as the new peripheral for which a new diagnostic module will be developed.

You are taken step-by-step through the development of the Mouse Diagnostic Module, from researching the functions of the mouse, to writing the new code for the Mouse Diagnostic, through testing the new module as integrated with the System Diagnostic Module as a whole.

With the general principles outlined in the previous section and the concrete examples described in this section, you should be able to gain a good understanding of how you can go about developing a diagnostic module for any peripheral that you add to your system.

Throughout this book, the IBM PC has been used as a model, using BASICA and GWBASIC for writing the diagnostic program. In this case, however, because it already uses a mouse as a standard device, the Macintosh is used as the model for developing the Mouse Diagnostic. Therefore, the program code is presented as written with the Macintosh version of Microsoft BASIC.

Functions of the Mouse

The mouse is a pointing device. It is used to position the cursor on the screen, to make selections, and to give commands.

Tracking the Cursor. The cursor on the screen moves according to how you move the mouse on the pad. When you move the mouse to the right, the cursor on the screen tracks to the right. If you move the mouse to the left, the cursor tracks to the left. If you move the mouse diagonally, the cursor likewise tracks diagonally. The mouse causes the cursor on the screen to follow its movements.

With the mouse, you can position the cursor where you need it: at the point where you wish to enter text or begin a drawing, on text or other object that you wish to select for further action, on a particular menu, or on a certain command.

Clicking the Button. In addition to pointing, the mouse has at least one button. You can position the cursor in a certain place, and then press the button, informing the computer that you wish to take a certain action. By clicking the mouse button, the computer knows you wish to do something with the item on the screen where the cursor is pointing.

Clicking the button can let you select an object on the screen for further action, choose a command to act upon a selected object, dismiss a confirmer message, and so forth.

Double-Clicking the Button. Sometimes, clicking the mouse twice, or "double-clicking," on a command or other object causes a particular action or event to take place. Double-clicking is often used as a shortcut command. For example, when in the Macintosh "desktop" environment, double-clicking on a file or application icon opens up that file or application. Alternatively, you could select the file by clicking on the icon once, then pulling down the appropriate menu and clicking on the "Open" command. But double-clicking is much faster, so is used as a shorthand command.

Dragging the Mouse. A final use of the mouse is called "dragging." You can hold the button down while moving the mouse from one place to another, with the cursor moving with it on the screen. This action can mean a variety of things depending on the software you're using. It could mean that you're selecting a sizable area within a file; that you're pulling down a command menu; or that you're increasing or decreasing the size of a window.

Information about the Mouse

The Macintosh system manual provides a good deal of information about the mouse. The first part of the manual describes how the mouse operates and details its features. A latter section of the manual includes a section on how to take care of the mouse, as well as what steps to take if anything goes wrong.

These sections provide you with much of the information needed to develop a new diagnostic module. They might also open your eyes to features as well as potential problems that you might not have been aware of from your hands-on experimentation with the mouse.

In addition to knowing how the mouse functions, how to troubleshoot, and how to repair it, you also need to know what programming facilities you have available to test the device. To get this kind of information, read the BASIC programming manual, in this case, the Macintosh Microsoft BASIC manual. This particular manual has a section set aside specifically for programming the mouse.

Read every reference the manual has on the mouse.

Learn how to write code that determines when the cursor is moving, when the button has been clicked, when the button has been double-clicked, and when it has been dragged. This information will provide you with the specifics needed to write the code for the Mouse Diagnostic Module.

Troubleshooting and Repair Guidelines

The mouse is primarily a mechanical device, with two basic components: the tracking mechanism and the button mechanism.

The Cursor Does Not Track Properly. The roller ball on the underside of the mouse is the tracking mechanism. The mouse might stick or catch on the pad instead of rolling smoothly. The cursor on the screen might not track evenly with the mouse movement, but instead jerk erratically on the screen, undershooting or overshooting the desired placement. Either of these symptoms indicate that the mouse roller ball is dirty. Clean the roller ball with isopropyl alcohol, and this should take care of the tracking problem.

The Mouse Button Does Not Make Contact. The mouse button is a similar mechanism to other appliance pushbuttons. You've probably had a pushbutton television set or radio on which one of the buttons wears out and you can no longer access a particular station. The mouse button can wear out just as easily. In this case, you might click or drag the mouse button, but there is no corresponding action with the cursor. If you experience such button problems, try the following remedies:

1. See whether there is a bad cable connection. Make sure the mouse cord is properly plugged in.
2. Remove the outer cover of the mouse and spray contact cleaner under the button.
3. If this still doesn't make the mouse work, swap another mouse onto your system. If this still doesn't make the mouse work, then you probably have a problem in the system unit.

MOUSE TESTS

From reading about the mouse in the Macintosh system manual and Microsoft BASIC manual, and from playing with the mouse, you now know that it has four basic functions: cursor tracking, clicking, double-clicking, and dragging. You now have done enough research and have enough knowledge to be able to determine the kinds of tests you want the Mouse Diagnostic Module to include. Because you know what the mouse is supposed to do, you can start making logical guesses as to the kinds of problems it can experience.

If one of the mouse's functions is to place the cursor on the screen, you can deduce that a possible malfunction is that it does not do this. The cursor does not track with the mouse. You move the mouse to the right, and the cursor does not move, or it does move, but erratically.

Another mouse function is clicking the button and having this signal registered in the computer in some way. If this does not happen, there is another

potential problem. If you use the mouse to position the cursor on a menu command, for example, but when you click the button nothing happens, then you know that the mouse button signal is not being registered by the system, and there is a problem there.

The third mouse function is double-clicking. It is conceivable that while the single click works fine, the double-clicking signal is having problems, and is not being seen by the system.

The fourth and final mouse function is dragging. If you've used the mouse to position the cursor, pressed the button, held it down while you dragged the mouse to another location, but the cursor does not follow, then you have a problem with the drag mechanism.

Now that you know all functions of the mouse, simply by taking the inverse of these operations, you know the problems the mouse can experience when it does not perform these functions upon request. At this point, you can take these four functions, with their four inverses that indicate four possible problems, and create four tests to be performed by the Mouse Diagnostic Module.

The first test would diagnose a cursor tracking problem. The second test would diagnose a single-click problem. The third test would diagnose a double-click problem. And the fourth test would diagnose a dragging problem. So, within the Mouse Diagnostic Module, you will have four submodules, or four tests.

The Cursor Tracking Test

Testing the tracking of the cursor to the mouse involves measuring distances and ranges. Cursor tracking is not a matter of something being on or off, the way the mouse button is. When testing cursor tracking, you want to measure the range of the mouse movement, and make sure that it matches the manner in which the cursor is moving.

The cursor has constrained parameters in which it operates, and this makes your testing task easier. The actual window in which the cursor moves has finite dimensions: there is a left side, a right side, a top, and a bottom.

The mouse, on the other hand, is not so constrained and limited. Its only real constraint is the length of its cord. If you wanted to, you could roll the mouse all over your desk, and the cursor would track within its limits.

To test the correlation of the mouse movements to the cursor movement, you need to create a map, or scale. This scale can help you measure the range of the mouse to the cursor, and will constrain the mouse in a similar way that the cursor is constrained.

To do this, bring up a standard window on the screen. Place a piece of paper underneath the mouse, then use the mouse to position the cursor on the far left side of the window. When this is done, mark the position of the left edge of the mouse on the paper.

Without lifting the mouse from the table, roll the mouse to the right until the cursor is positioned on the far right side of the window. Mark the position of the right edge of the mouse on the paper (Fig. 8-4).

Follow the same procedure to mark the top edge and bottom edge of the mouse to cursor tracking on the range scale paper.

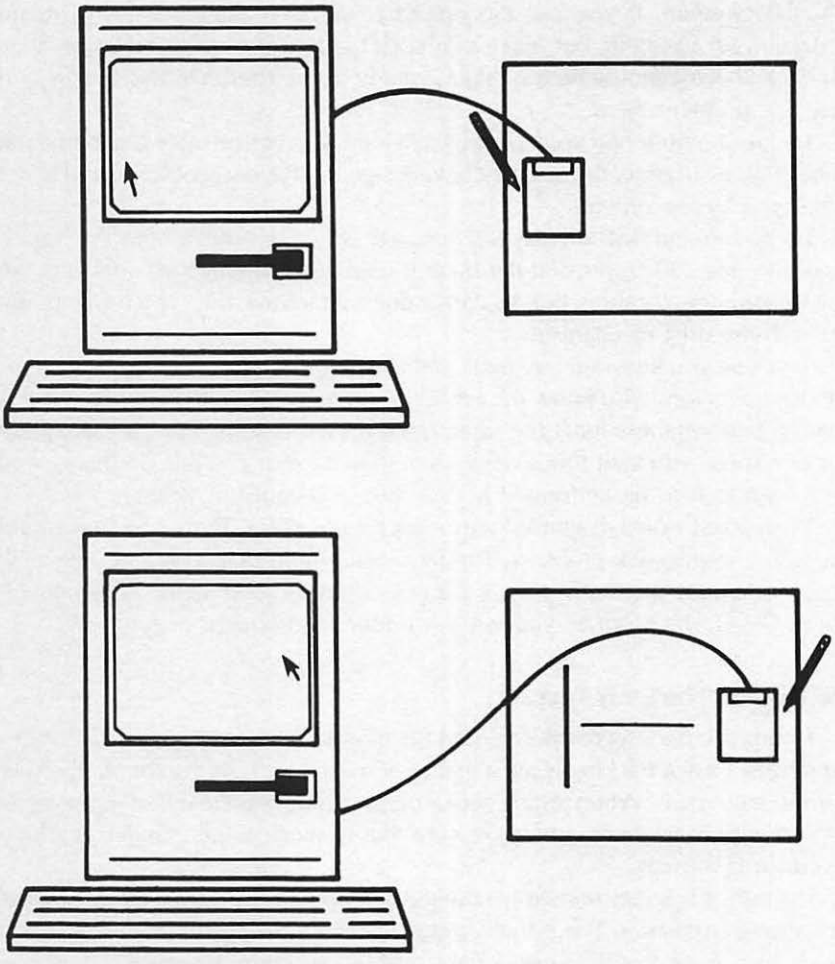


Fig. 8-4. Drawing the mouse range template.

There, on paper, you will have two vertical lines and two horizontal lines with a measurable distance between them. This distance measures the cursor tracking. If you ever suspect the integrity of mouse-to-cursor tracking, you can lay this range scale down again, position the cursor on the left edge of the screen with the mouse on the left line, and move it over to the right until the mouse hits the right line and the cursor hits the right edge of the window. It the two do not match, you know you have a problem with your cursor tracking.

You can use this range scale as a guide for your first BASIC test routine for the Mouse Diagnostic Module. The test would prompt the user to position the cursor on the left side of the window, place the mouse with its left edge of the range scale template, slowly move the mouse to the right until its right edge touches the right edge of the template, and then click the button when finished.

Clicking the button places a little square marker on the screen where the user finished moving the mouse. This marker would show whether or not the cursor is indeed tracking within the correct parameters.

If the cursor falls short of the right edge by an inch or so, a problem with the mouse's tracking facility is indicated. The cursor might also end up outside the window, and you would see no square marker at all. This would mean that the cursor is overshooting; the cursor reaches the right edge before the mouse does. This also points to a tracking problem.

The computer system troubleshooting guide tells you that if you have a problem with cursor tracking, to clean the mouse. It would probably provide instructions on removing the ball from the mouse, cleaning it, replacing it, and testing it.

After you've done whatever servicing necessary, you can then run the test again to see if you get the proper results. If the mouse matches the cursor this time, you can conclude that the device is fixed.

The Single-Click Test

The test routines involving the mouse button merely check to see whether or not an event has occurred. It's rather like a light switch: either the light is on, or it's off.

The single-click test would prompt the user to click the mouse button once. When this is done, and the program successfully detects the click, the program would display a message saying that a click has been detected.

If this message is not displayed, it would mean that the single click had not been detected, and that there is a problem with the mouse button, particularly with clicking.

The Double-Click Test

The third test of the Mouse Diagnostic Module would be the double-click test, which would be very similar to the single-click test. Instead of detecting a single click, the program would prompt and look to detect a double click.

When the user double-clicked the mouse button and the program successfully detected it, a message would be displayed saying that a double-button click had been detected.

If this message were not displayed, it would mean that the double click had not been detected, and that there was a problem with the mouse button, particularly with double-clicking.

The Drag Test

The final test of the Mouse Diagnostic would be the drag test, another mouse button test. The program would prompt for the user to press the mouse button, drag the mouse over a few inches, then let up on the button. When the user dragged the mouse and the program successfully detected it, a message would be displayed saying that the mouse is dragging properly.

If the program did not detect the mouse being dragged, a message would eventually be displayed saying that no drag had taken place. Again, this would indicate a problem with the mouse button, particularly when holding it down to drag.

THE MOUSE PROGRAM FLOWCHART

Now that you have determined what functions you wish to test as part of the Mouse Diagnostic, and have worked out many of the testing details, the next step is to develop the road map for writing the actual code. This road map is done in the form of a program flowchart.

This flowchart shows the commands that will be used in writing the actual code for the diagnostic module. To write it, you need your BASIC programming manual at your fingertips. You'll need to do some reading to see whether there are any specialized instructions used to access commands for the mouse.

You would find that there is such a command, the MOUSE command. This MOUSE command is much like the INKEY$ command, in that it is a function that will return a value to a numeric variable. This variable indicates whether or not the function has taken place. In this case, the function would indicate whether or not the mouse button has been pressed, and if so, whether it has been clicked once, twice, or has been dragged.

Once you know how the MOUSE function works, you're ready to diagram the module in the flowchart. This flowchart will reflect what you've learned about the functions of the mouse, and will particularly detail the four tests you've decided to develop for this module.

WRITING THE MOUSE DIAGNOSTIC MODULE

Once the program flowchart is written, you have your road map for writing the actual code.

Notice that the code for the Macintosh version of Microsoft BASIC looks different from the BASIC code seen on IBM and CP/M systems. Instead of using line numbers, the Macintosh BASIC uses labels such as *test, range,* and *waiter* as line and section identifiers.

The Mouse Diagnostic Module Listing

Based on your flowchart, your program code might look similar to the listing in Fig. 8-5.

How the Mouse Diagnostic Module Works

To maintain consistency within the System Diagnostic Program, the first things that need to be done are to clear the screen, identify which module this is, and display the options available within the module:

```
mousetest:
CLS
PRINT TAB(10) "Mouse Diagnostic Module"
PRINT:PRINT TAB(5) "1 . . . Range Test":PRINT TAB(5) "2 . . . Button Test"
```

```
mousetest:
CLS
PRINT TAB(10) "Mouse Diagnostic Module"
PRINT:PRINT TAB(5) "1...Range Test":PRINT TAB(5) "2...Button
Test"
PRINT:PRINT TAB(5)"Enter Test Number or Press Clear to End
Test":GOTO test
kbd:
a$=INKEY$:IF a$="" THEN GOTO kbd
RETURN
test:
GOSUB kbd
IF ASC(a$)=27 THEN mainmenu
IF a$="1" THEN GOTO range
IF a$="2" THEN GOTO btn
BEEP:GOTO test
range:
CLS
PRINT "Position the left edge of the Mouse on the left edge
of the Screen ":PRINT "and Click the button and hold it
down."
gmouse:
a:MOUSE(0):IF a=0 THEN gmouse
PRINT:PRINT "Now place the Mouse on left edge of the
template.":PRINT "Roll the Mouse slowly to the right until
the right edge of the
PRINT "Mouse is touching the right edge of the template then
let up on the button."
gmouse1:
b=MOUSE(0):IF b<=0 THEN gmouse1
x=MOUSE(1):y=MOUSE(2)
FOR i=1 TO 2
PSET (x,y),33:x=x+1:NEXT i
y=y+1:x=x-2
FOR i=1 TO 2
PSET (x,y),33:x=x+1:NEXT i
PRINT:PRINT "You should see a black box where the Mouse is
positioned on the screen."
PRINT "This should be near the right edge of the screen."
PRINT:INPUT "Any Character to Exit Test-";a$:GOTO mousetest
btn:
CLS:PRINT TAB(10)"Button Test"
PRINT:PRINT "Click the Mouse Button Once"
mlook:
a=MOUSE(0):IF a=0 THEN GOTO mlook
PRINT "Single Click is Okay"
PRINT:PRINT "Double-Click the Mouse Button."
dclick:
a=MOUSE(0):IF a<>2 THEN GOTO dclick
PRINT "Double Click is Okay":PRINT
```

Fig. 8-5. Mouse diagnostic module listing.

```
PRINT "Now hold the Mouse Button down and drag Mouse a few
inches.":PRINT "Let up on the Button."
dragm:
a=MOUSE(0):IF a=0 THEN dragm
x=MOUSE(1):y=MOUSE(2)
dragm1:
b=MOUSE(0):IF B<0 THEN dragm1
x1=MOUSE(1):y1=MOUSE(2)
IF x=x1 AND y=y1 THEN PRINT "No Drag Occurred":GOTO waiter
PRINT "Drag Test is Okay"
waiter:
LOCATE 19,20:PRINT "Press Any Key to End Test";
GOSUB kbd
RETURN
kbd:
a$=" ":a$=INKEY$:IF LEN(a$)=0 THEN kbd
RETURN
```

Fig. 8-5. Mouse diagnostic module listing. (Continued from page 161.)

The PRINT statements indicate that the Mouse Diagnostic Module consists of two basic tests: the Range Test and the Button Test.

The Range Test checks that the mouse covers the particular distance that has been measured out from the screen onto the range scale template.

The Button Test consists of three other tests, the Single-Click Test, the Double-Click Test, and the Drag Test. Each of these tests a different function of the mouse button.

Still remaining consistent with the rest of the System Diagnostic Program, another PRINT statement prompts the user to either enter the number for the desired test, or press the CLEAR (or ESC) key to exit from the module:

```
PRINT:PRINT TAB(5)"Enter Test Number or Press Clear to End Test":
GOTO test
```

Next, the process loop of the module begins. One of the first things that every module in the System Diagnostic Program does is to go to the subroutine that checks for a test selection being made from the keyboard. In IBM code, this is the keyboard input subroutine at line 2930. In the Macintosh version, this subroutine is called kbd, denoting "keyboard."

This subroutine is already part of the System Diagnostic Program, so it does not need to be rewritten. At the beginning of this process loop, which might be called something like "test," there is a GOSUB to the keyboard routine:

```
kbd:
a$=INKEY$:IF a$=" " THEN GOTO kbd
RETURN
test:
GOSUB kbd
```

The prompt has already been displayed that says if the CLEAR key is pressed, the Mouse Diagnostic Module ends and the program returns to the System Diagnostic Main Menu. So, after the GOSUB statement, an IF statement looks for the ASCII code value of A$, which contains the ASCII value of the key the user presses. If it's equal to 27, which is an ESC for the IBM and CP/M, and a CLEAR for the Macintosh, then the program returns to the System Diagnostic Main Menu. In this case, pressing CLEAR ends the program:

```
IF ASC(a$)=27 THEN mainmenu
```

If A$ is not ASCII value 27, the program falls through to the next line, which checks for the two test options: 1 for the Range Test, and 2 for the Button Test:

```
IF a$="1" THEN GOTO range
IF a$="2" THEN GOTO btn
```

If the value entered is not 1, 2, or CLR, the program falls through to the next line, which registers a beep, telling the user that an invalid key has been pressed. The program then loops back to test, the beginning of the process loop, to let the user press a valid key:

```
BEEP:GOTO test
```

At this point, the actual routines for performing the tests begin.

Earlier, when you were determining what tests the Mouse Diagnostic Module would consist of, you created the range scale template for the Range Test. The code for the Range Test begins with the label range, and is the point to which the program branches when you enter 1. The screen is cleared with the CLS instruction:

```
range:
CLS
```

The PRINT statement provides the prompt that tells the user what to do to perform the Range Test.

```
PRINT "Position the left edge of the Mouse on the left edge of the Screen
":PRINT"and Click the button and hold it down."
```

The gmouse label begins a routine that sets variable A equal to the MOUSE(0) function:

```
gmouse:
a:MOUSE(0):IF a=0 THEN gmouse
```

You would have read in the Macintosh Microsoft BASIC programming manual that the MOUSE(0) function returns a nonzero value if the button is clicked.

So, at this point, you set variable A equal to MOUSE(0). Then the program checks to see whether A is actually equal to zero. If it is, the button has not been clicked yet, and the user is either not doing anything, or the mouse is being moved into position. As long as A is equal to zero, the program loops back to the gmouse label.

As soon as the button is clicked, the variable A becomes a nonzero value. When this happens, two more PRINT statements are issued to provide further prompts for the Range Test:

```
PRINT:PRINT "Now place the mouse on left edge of the template."
PRINT "Roll the Mouse slowly to the right until the right edge of the"
PRINT "Mouse is touching the right edge of the template then let up on the
button."
```

Once the PRINT statements have issued their prompts, the program falls down to another routine entitled gmouse1. As with the gmouse routine, the program looks for a value and uses the gmouse1 label as a looping reference point:

```
gmouse1:
b=MOUSE(0):IF b< =0 THEN gmouse1
```

Variable B is set equal to MOUSE(0). Then, if B is less than or equal to zero, the program loops back to gmouse1. And just as before, the program is looking for the mouse button to be clicked.

Once the button is clicked, the program proceeds to the next statement, which sets an xy coordinate. This coordinate places a special graphic on the screen to show where the user placed the cursor.

Your BASIC manual tells you that the pixels on the screen are addressed as xy coordinates. The MOUSE(1) function returns the cursor's current x coordinate, while the MOUSE(2) function returns the cursor's y coordinate. This means that you can set an x variable equal to a MOUSE(1) function, and a y coordinate with MOUSE(2):

```
x=MOUSE(1):y=MOUSE(2)
```

This statement indicates that the xy coordinate of the cursor location is recorded into variables x and y.

After marking the cursor placement, the program draws a box by turning on four individual pixels. This is done with a FOR-TO statement, and because there are only two lines involved, the statement reads FOR i=1 TO 2:

```
FOR i=1 TO 2
PSET (x,y),33:x=x+1:NEXT i
```

PSET turns on a pixel at x,y, which is the current cursor location. The 33 tells the monitor to display black pixels on a white background. Then, one is added to the x variable, so a second pixel is turned on next to the first pixel. The NEXT

i statement sends the program back to the FOR statement for the second round of the loop. This routine turns on a pixel at variable x at the top of the display, and turns on another pixel to the right of the display, corresponding to the cursor location.

The next line sets variable y equal to y+1, thereby incrementing the y coordinate of the cursor. This places the cursor on the next line of pixels:

```
y=y+1:x=x-2
```

Then, variable x is set equal to x-2, returning it to its original value before it was incremented in the previous PSET statement described above. This places the cursor directly below the previous line.

Another FOR-TO loop does the same thing that the identical line in the previous FOR-TO routine did, drawing two more pixels under the first two, thereby finishing the box. It turns on black pixels on a white background at coordinates x,y, increments x by one, and loops through once again to satisfy the FOR-TO loop:

```
FOR i=1 TO 2
PSET (x,y),33:x=x+1:NEXT i
```

These two groups of FOR-TO statements serve to draw a black box graphic to mark the location of the cursor on the screen.

Then the program falls through to two informational PRINT statements:

```
PRINT:PRINT "You should see a black box where the Mouse is positioned on
the screen."
PRINT "This should be near the right edge of the screen."
```

If your mouse is hanging up, sticking, or skipping, this black box might not be close to the right or bottom edge of the screen display, as you expect. This would tell you that there is a problem with the cursor tracking and the mouse might need some cleaning and servicing.

Then, another PRINT statement prompts the user to press any key to exit the Range Test:

```
PRINT:INPUT "Any Character to Exit Test-";a$:GOTO mousetest
```

This routine performs the same function as line 2940 does in the IBM and CP/M versions of the System Diagnostic Program. In the Macintosh version, line 2940 has its equivalent in the form of a routine labeled waiter. The waiter routine prompts the user to enter any character to exit the test and waits for a keystroke to be pressed. Once this is done, the program branches back to mousetest, and the Mouse Diagnostic Module menu is displayed on the screen.

When the user presses 2 to choose the Button Test, the program branches to the btn label to begin the Button Test routine:

```
btn:
```

The screen is cleared, and a PRINT statement identifies the test name:

```
CLS:PRINT TAB(10)"Button Test"
```

The program falls through to the PRINT statement, prompting the user to click the mouse button once:

```
PRINT:PRINT "Click the Mouse Button once."
```

The next line is the mlook label, which begins the Single-Click Test routine:

```
mlook:
```

The next line sets variable A equal to MOUSE(0). As long as this is true, and the mouse button is not clicked, this routine loops:

```
a=MOUSE(0):IF a=0 THEN GOTO mlook
```

As soon as the mouse button is clicked, A becomes a nonzero number, and the loop is halted. The program falls through to the next line that states that the single button click worked successfully:

```
PRINT "Single click is Okay"
```

The next line begins the Double-Click Test routine. This PRINT statement provides the prompt instructing the user to double-click the mouse button:

```
PRINT:PRINT "Double-Click the Mouse Button."
```

The next line is a new label indicating the start of a routine serving a similar purpose to the mlook routine:

```
dclick:
a=MOUSE(0):IF a< >2 THEN GOTO dclick
```

Variable A is set equal to MOUSE(0), and the routine loops until the mouse button is double-clicked. A single-click causes nothing to happen; the program continues to loop until a double-click is registered. Your BASIC manual indicates that a double-click is signified by a return of two from the MOUSE(0) function call. So as long as A is not equal to two, the program loops. When the user double-clicks the mouse, two is returned into variable A, and the loop stops.

The program falls through to the next line, which states that the double button click test was successful:

```
PRINT "Double Click is Okay":PRINT
```

The final phase of the Button Test is the Drag Test. A PRINT statement prompts the user to drag the mouse:

```
PRINT "Now hold the Mouse Button down and drag Mouse a few inches.":PRINT"Let up on the Button."
```

The dragm label begins the Drag Test routine, which, like the other button test routines, uses the MOUSE function:

```
dragm:
a=MOUSE(0):IF a=0 THEN dragm
```

Variable A is set equal to MOUSE(0) again. As long as the mouse button is not clicked up, this statement remains in the loop. As soon as the button is pressed, A is no longer equal to zero, the loop halts, and the program falls through to the next line.

The MOUSE(1) and MOUSE(2) functions are used to set the xy coordinate of the current cursor location:

```
x=MOUSE(1):y=MOUSE(2)
```

Now that the program knows the mouse button has been pressed down, and it knows the coordinates of the cursor where this took place, it looks to see whether the mouse button has been let up, and where that has taken place. This is done by setting variable B equal to MOUSE(0). If B is less than zero, the program loops back to dragm1. This loop continues until the mouse button is let up:

```
dragm1:
b=MOUSE(0):IF B<0 THEN dragm1
```

Your BASIC manual indicates that if the mouse's button is pressed down while being moved, which is called a drag, then the value returned by function MOUSE(0) is a negative number. So, if variable B is less than zero, this means the user is moving the mouse. The loop continues until the mouse button is let up. At this point, a positive value is returned to variable B.

When this happens, the program falls through to the next line, which calls up the MOUSE(1) and MOUSE(2) functions again for another set of coordinates indicating where the mouse drag stopped:

```
x1=MOUSE(1):y1=MOUSE(2)
```

Because x and y are already set and being used to indicate where the drag began, this line uses the variables x1 and y1, which are equal to MOUSE(1) and MOUSE(2), respectively.

Once these new coordinates x1 and y1 are set, the program falls through to

the next line. This statement indicates that if the beginning coordinates, x and y, are equal to the ending coordinates, x1 and y1, then no drag has taken place. What has taken place is just that the mouse button has been pressed down and let up immediately. This is a click, not a drag:

```
IF x=x1 AND y=y1 THEN PRINT "No Drag Occurred":GOTO waiter
```

If the user had actually dragged the mouse, but received this message, a problem with the mouse would be indicated.

If x is not equal to x1, and y is not equal to y1, then the program falls through to the PRINT statement which indicates that the Drag Test was successful:

```
PRINT "Drag Test is Okay"
```

When either of these messages is displayed, the waiter subroutine is called up:

```
waiter:
LOCATE 19,20:PRINT "Press Any Key to End Test";
GOSUB kbd
RETURN
kbd:
a$=" ":a$=INKEY$:IF LEN(a$)=0 THEN kbd
RETURN
```

When the user presses a character, the program branches back to mousetest, which is the beginning of the Mouse Diagnostic Module, and the Mouse Diagnostic menu is displayed again. At this point, the user can press 1 for the Range Test, 2 for the Button Test again, or CLEAR to exit from the Mouse Diagnostic and return to the System Diagnostic Main Menu.

INTEGRATING THE MOUSE MODULE WITH THE SYSTEM DIAGNOSTIC PROGRAM

Once you have written the code for the Mouse Diagnostic Module, you need to integrate it with the rest of the System Diagnostic Program (Fig. 8-6).

First, you need to modify the System Diagnostic Main Menu, which displays the diagnostic module options. You need to look at lines 10 through 100 of the program where the Main Menu Module resides. Here you need to add the Mouse Diagnostic identifier to the PRINT statement that lists the menu options on the screen. You also need to insert the appropriate IF statement for the Mouse Diagnostic identifier.

Using M would be the most logical and straightforward identifier, but it is already being used for the Monitor Diagnostic. You could use B to denote the mouse button, or you could use C to denote the cursor that the mouse tracks. Or you could change the monitor identifier to S for screen, if you like. This would create the need for additional program modifications elsewhere in the program. But the choice is entirely yours—it's your System Diagnostic Program.

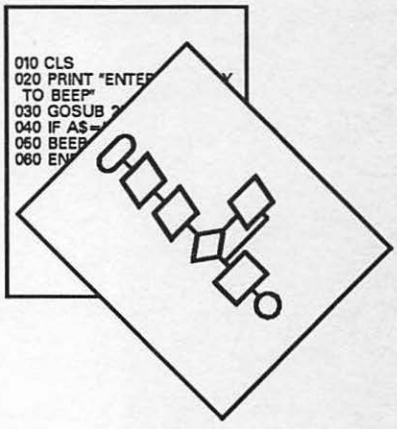


Fig. 8-6. Flowchart and code.

Once this is done, all you need to do is add the code to the program. If you're programming an IBM or CP/M system, have the line numbers of the new code begin around 3000. This starts the new module where the last diagnostic module stopped. If you're programming on a Macintosh, make sure the labels you're using are unique throughout the code so that you don't get program branches that you were not expecting.

TESTING THE MOUSE MODULE

The last thing left for you to do is to test the Mouse Diagnostic as it now resides as an integrated module of the System Diagnostic Program. Try out all four tests with as many possibilities as you can think of. Try your best to "break" the program, and if you do, find and fix the code where the problem occurred. Once this is done, your Mouse Diagnostic Module is ready for real work whenever you need to test the cursor tracking or the button functions.

From this example with the Mouse Diagnostic Module, you have probably gained a good understanding of how to go about researching a new peripheral device, identifying the types of problems that the device might experience, developing the diagnostic program based on these potential problems, and integrating this module into the System Diagnostic Program.

Appendices

Appendix A

MS-DOS BASIC System Diagnostic Program Listing

```
10 KEY OFF:FOR I=1 TO 10:KEY I,"":NEXT I:REM System
   Diagnostic Program
20 DATA "q","w","e","r","t","y","u","i","o","p"
   :DATA "a","s","d","f","g","h","j","k","l"
   :DATA "z","x","c","v","b","n","m"
   :DATA "1","2","3","4","5","6","7","8","9","0","-","=",
   :DIM A1$(10):DIM A2$(9):DIM A3$(7):DIM A4$(12)
30 FOR I=1 TO 10 :READ A1$(I):NEXT I:FOR I=1 TO 9:READ
   A2$(I):NEXT I:FOR I=1 TO 7:READ A3$(I):NEXT I:FOR I=1 TO
   12:READ A4$(I):NEXT I
40 REM: System Diagnostic Main Menu Module
50 CLS:PRINT TAB(12) "System Diagnostic":PRINT:PRINT TAB(3)
   "<K>eyboard Test  <S>erial Comm Test":PRINT TAB(3)
   "<M>onitor Test   <D>isk Drive Test":PRINT TAB(3)
   "<P>rinter Test   <E>xerciser":PRINT TAB(16) "<Q>uit":
   PRINT:PRINT TAB(12) "Enter Selection:"
60 AN$="":AN$=INKEY$:IF AN$="" THEN GOTO 60
70 IF AN$="K" THEN 220
80 IF AN$="k" THEN 220
90 IF AN$="M" THEN 790
100 IF AN$="m" THEN 790
110 IF AN$="P" THEN 1360
120 IF AN$="p" THEN 1360
130 IF AN$="D" THEN 2050
```

```
140 IF AN$="d" THEN 2050
150 IF AN$=S" THEN 2400
160 IF AN$=s" THEN 2400
170 IF AN$="E" THEN 2600
180 IF AN$="e" THEN 2600
190 IF AN$="Q" THEN END
200 IF AN$="q" THEN END
210 BEEP:GOTO 60
220 REM: Keyboard Diagnostic Module
230 FOR I=1 TO 10:KEY I,"":NEXT I:KEY OFF:REM Disable
    Function Keys
240 CLS:PRINT TAB(7) "Keyboard Diagnostic Module":PRINT
250 PRINT TAB(14) "ASCII":PRINT TAB(3) "Character  Value
    Key Combination":PRINT TAB(3)"---------  -----
    ---------------"
260 GOSUB 2930
270 N=ASC(A$)
280 IF ASC(A$)=27 THEN 40
290 LOCATE 6,6:PRINT SPACE$(40);
300 A=LEN(A$):IF A>= 2 THEN 370
310 IF N>28 THEN 320 ELSE KY$="Ctrl":KY2$=CHR$(N+96):
    GOTO 730
320 KY$="Ctrl"
330 IF N=30 THEN KY2$="6":GOTO 770
340 IF N=31 THEN KY2$="-":GOTO 770
350 LOCATE 6,6 :PRINT AN$;:PRINT TAB(14) ASC(A$);:GOTO 260
360 REM: Check for Function Keys
370 KY$="":KY2$="":A=ASC(RIGHT$(A$,1))
380 IF A>3 THEN 400
390 KY$="Ctrl":KY2$="2":GOTO 730
400 IF A=15 THEN KY$="Shift Tab":GOTO 730
410 IF A>=16 OR A<=50 THEN KY$="Alt"
420 IF A>25 THEN 430 ELSE KY2$=A1$((A-16)+1):GOTO 730
430 IF A>38 THEN 440 ELSE KY2$=A2$((A-30)+1):GOTO 730
440 IF A>50 THEN 450 ELSE KY2$=A3$((A-44)+1):GOTO 730
450 IF A >68 THEN GOTO 470
460 KY$="F"+STR$((A-59)+1):GOTO 730
470 IF A=71 THEN KY$="Home":GOTO 730
480 IF A=72 THEN KY$="Cursor Up":GOTO 730
490 IF A=73 THEN KY$="Pg Up":GOTO 730
500 IF A=75 THEN KY$="Cursor Left":GOTO 730
510 IF A=77 THEN KY$="Cursor Right":GOTO 730
520 IF A=79 THEN KY$="End":GOTO 730
530 IF A=80 THEN KY$="Cursor Down":GOTO 730
540 IF A=81 THEN KY$="Pg Dn":GOTO 730
550 IF A=82 THEN KY$="Ins":GOTO 730
560 IF A=83 THEN KY$="Del":GOTO 730
570 IF A>113 THEN GOTO 590
580 IF A<=93 THEN KY2$="F"+STR$((A-84)+1):KY$="Shift":
    GOTO 730
```

```
590 IF A>103 THEN GOTO 610
600 KY$="Cntrl":KY2$="F"+STR$((A-94)+1):GOTO 730
610 IF A<104 OR A>113 THEN GOTO 630
620 KY$="Alt":KY2$="F"+STR$((A-104)+1):GOTO 730
630 IF A>113 AND A<120 THEN KY$="Ctrl"
640 IF A=114 THEN KY2$="PrtSc":GOTO 760
650 IF A=115 THEN KY2$="Cursor Left":GOTO 760
660 IF A=116 THEN KY2$="Cursor Right":GOTO 760
670 IF A=117 THEN KY2$="End":GOTO 760
680 IF A=118 THEN KY2$="Pg Dn":GOTO 760
690 IF A=119 THEN KY2$="Home":GOTO 760
700 IF A>131 THEN 720
710 KY$="Alt":KY2$=A4$((A-120)+1):GOTO 760
720 IF A=132 THEN KY$="Ctrl":KY2$="Pg Up"
730 IF N>=9 AND N<=13 THEN 770
740 IF A=30 OR A=31 THEN 770
750 REM
760 LOCATE 6,6:PRINT A$;:PRINT TAB(14) ASC(RIGHT$(A$,1));
    :PRINT TAB(23) KY$+" "+KY2$:GOTO 260
770 LOCATE 6,6:PRINT TAB(14) ASC(RIGHT$(A$,1));:PRINT TAB
    (23) KY$+" "+KY2$:GOTO 260
780 GOTO 40
790 CLS:REM Monitor Diagnostic Module
800 PRINT TAB(6) "Monitor Diagnostic":PRINT:PRINT TAB(5)
    "1...Display Test":PRINT TAB(5) "2...Alignment Test":
    PRINT TAB(5) "3...Character Set Test":PRINT TAB(5)
    "4...Intensity Test":PRINT TAB(5) "5...Scroll Test":
    PRINT TAB(5) "6...Status Line Test":PRINT
810 PRINT TAB(5) "Press Esc to End Diagnostic"
820 GOSUB 2930:IF ASC(A$)=27 THEN 40
830 IF A$="1" THEN 910
840 IF A$="2" THEN 980
850 IF A$="3" THEN 1060
860 IF A$="4" THEN 1140
870 IF A$="5" THEN 1190
880 IF A$="6" THEN 1310
890 BEEP:GOTO 820
900 GOTO 40
910 CLS:PRINT TAB(14) "Display Test":PRINT:PRINT "Enter Any
    Character to Fill Screen":PRINT:PRINT "Press Esc to End
    Test":DEF SEG=&HB000
920 GOSUB 2930 IF LEN(A$)>1 THEN 920
930 IF A$=CHR$(27) THEN 960
940 FOR I=0 TO 3998 STEP 2
950 POKE I,ASC(A$):NEXT I:IF EX=0 THEN GOTO 920
960 DEF SEG:IF EX=1 THEN GOTO 1070
970 GOTO 790
980 CLS:REM Alignment Test
990 PRINT TAB(13) "Alignment Test":PRINT CHR$(218);:FOR I=1
    TO 35 :PRINT CHR$(196);:NEXT I:PRINT CHR$(194);:FOR I=1
```

```
    TO 35:PRINT CHR$(196);:NEXT I:PRINT CHR$(191)
1000 FOR I=1 TO 10:PRINT CHR$(179),TAB(37) CHR$(179),TAB(73)
     CHR$(179):NEXT I
1010 PRINT CHR$(195);:FOR I=1 TO 35:PRINT CHR$(196);:NEXT
     I:PRINT CHR$(197);:FOR I=1 TO 35:PRINT CHR$(196);:NEXT
     I:PRINT CHR$(180)
1020 FOR I=1 TO 10:PRINT CHR$(179),TAB(37) CHR$(179),TAB(73)
     CHR$(179):NEXT I
1030 PRINT CHR$(192);:FOR I=1 TO 35:PRINT CHR$(196);:NEXT I:
     PRINT CHR$(193);:FOR I=1 TO 35:PRINT CHR$(196);:NEXT I:
     PRINT CHR$(217)
1040 GOSUB 2940:GOTO 790
1050 REM Character Set Test
1060 CLS:PRINT TAB(11) "Character Set Test":PRINT
1070 FOR I=1 TO 255
1080 IF I=7 OR I=9 OR I=10 OR I=11 OR I=12 OR I=13 OR I=28
     OR I=29 OR I=30 OR I=31 THEN GOTO 1100
1090 PRINT CHR$(I)+" ";
1100 NEXT I
1110 IF EX=1 THEN GOTO 1200
1120 GOSUB 2940:GOTO 790
1130 REM Intensity Test
1140 CLS:PRINT "Intensity Test":PRINT:A$="XXXXXX"
1150 FOR I=1 TO 20
1160 PRINT TAB(10) A$;:PRINT "  ";:COLOR 15:PRINT A$;:COLOR
     7:PRINT "  ";:COLOR 0:PRINT A$;:COLOR 0,7:PRINT A$;:
     COLOR 7:PRINT "  ";:COLOR 23,0:PRINT A$;:COLOR 7: PRINT
     "  ";:COLOR 1:PRINT A$:COLOR 7:NEXT I
1170 GOSUB 2940:GOTO 790
1180 REM Scroll Test
1190 CLS:PRINT TAB(15)"Scroll Test":PRINT:INPUT "Enter
     Number of Repetitions-";A:PRINT
1200 N=32
1210 FOR I=1 TO A
1220 N=N+1:IF N>111 THEN N=32
1230 FOR X=1 TO 80
1240 PRINT CHR$(N);:N=N+1
1250 IF N>111 THEN N=32
1260 NEXT X
1270 NEXT I
1280 IF EX=1 THEN RETURN
1290 GOSUB 2940:GOTO 790
1300 REM Status Line Test
1310 CLS:PRINT "Status Line Test":LOCATE 12,1:PRINT "Enter
     any Character to be printed on the Status Line":PRINT:
     PRINT "Press Esc to End Test"
1320 GOSUB 2930:IF A$=CHR$(27) THEN 1350
1330 IF LEN(A$)>1 THEN GOTO 1320
1340 LOCATE 25,1:FOR I=1 TO 80:PRINT A$;:NEXT I:GOTO 1320
1350 GOTO 790
1360 W=80:REM Printer Diagnostic Module
```

```
1370 CLS:PRINT TAB(11) "Printer Diagnostic":PRINT:PRINT
     TAB(3) "1...Printer Setup":PRINT TAB(3) "2...Sliding
     Alpha Test":PRINT TAB(3) "3...Display Character Print
     Test":PRINT TAB(3) "4...Echo Character Print Test"
1380 PRINT TAB(3) "5...Horizontal Tab Test":PRINT TAB(3)
     "6...Line Feed Test":PRINT:PRINT TAB(5) "Press Esc to
     End Diagnostic"
1390 GOSUB 2930:IF ASC(A$)<>27 THEN 1410
1400 CLOSE #1:GOTO 40
1410 IF A$="1" THEN 1490
1420 IF A$="2" THEN 1580
1430 IF A$="3" THEN 1700
1440 IF A$="4" THEN 1800
1450 IF A$="5" THEN 1890
1460 IF A$="6" THEN 1950
1470 BEEP:GOTO 1390
1480 REM Printer Setup Routine
1490 CLS:PRINT "Printer Setup":PRINT
1500 INPUT "Enter the number of character positions, 80 or 132-";W
1510 IF W<>80 OR W<>132 THEN W=80
1520 PRINT:INPUT "Enter printer port to be tested (LPT1 or LPT2)-
     ";PR$
1530 IF PR$<> LPT2" or PR$<>"lpt2" THEN PR$="lpt11
1540 PR$=PR$+":"
1550 OPEN PR$ AS #1
1560 WIDTH #1,W:GOTO 1370
1570 REM Sliding Alpha Test
1580 CLS:PRINT "Sliding Alpha Test":PRINT:INPUT "Enter number of
     repetitions-";X
1590 N=31
1600 FOR I=1 TO X
1610 L$="":N=N+1:IF N>111 THEN N=32
1620 FOR A=1 TO W
1630 L$=L$+CHR$(N):N=N+1
1640 IF N>111 THEN N=32
1650 NEXT A
1660 PRINT #1,L$;:NEXT I
1670 IF EX=1 THEN GOTO 1700
1680 GOTO 1360
1690 REM Display Character Print Test
1700 CLS:PRINT "Display Character Print Test":L$="";N=1
1710 REM Enter the pertinent ESC Sequence for the Selection of
     Graphics Here Example: PRINT #1, CHR$(28); "I"; CHR$(1); for
     the NEC Pinwriter
1720 FOR I= 32 TO 126
1730 IF N=W THEN PRINT #1,L$:N=1:L$="":GOTO 1750
1740 L$=L$+CHR$(I)+" ":N=N+1
1750 NEXT I
1760 IF L$<>"" THEN PRINT #1,L$
1770 IF EX=1 THEN GOTO 1820
```

```
1780 GOTO 1360
1790 REM Echo Character Print Test
1800 CLS: PRINT "Echo Character Print Test":PRINT:PRINT "Enter
     Character to Echo":PRINT:PRINT "Press Esc to End Test"
1810 GOSUB 2930:IF ASC(A$)=27 THEN GOTO 1360
1820 L$=""
1830 IF LEN(A$)>1 THEN GOTO 1810
1840 FOR I=1 TO W
1850 L$=L$+A$:NEXT I
1860 PRINT #1,L$:IF EX=1 THEN GOTO 1890
1870 GOTO 1810
1880 REM Horizontal Tab Test
1890 CLS:PRINT "Horizontal Tab Test"
1900 FOR I=1 TO W
1910 PRINT #1,TAB(I) "*":NEXT I
1920 IF EX=1 THEN GOTO 1950
1930 GOTO 1360
1940 REM Line Feed Test
1950 CLS:PRINT "Line Feed Test"
1960 PRINT #1 "This is the First Line.....";
1970 FOR I=2 TO 5
1980 FOR A=1 TO I
1990 PRINT #1,CHR$(10);:NEXT A
2000 N$=STR$(I)
2010 PRINT #1,"There have been "+N$+" Line Feeds";:NEXT I
2020 PRINT #1,CHR$(12);:PRINT #1 "This is a New Page........"
2030 IF EX=1 THEN RETURN
2040 GOTO 1360
2050 REM: Disk Drive Diagnostic Module
2060 HVAL$="":FOR I=1 TO 128:HVAL$=HVAL$+CHR$(170):NEXT I
2070 LVAL$="":FOR I=1 TO 128:LVAL$=LVAL$+CHR$(85):NEXT I:IF EX=1
     THEN GOTO 2090
2080 CLS:PRINT TAB(10)"Disk Drive Diagnostic":PRINT:INPUT "Which
     disk drive do you want to test-";D$:D$=D$+":TEST"
2090 BUF$=HVAL$:NM$="High Values"
2100 ON ERROR GOTO 2270
2110 FOR I=1 TO 2
2120 OPEN D$ FOR OUTPUT AS #1
2130 PRINT:PRINT "Writing" ";NM$;" to Disk"
2140 ON ERROR GOTO 2270
2150 PRINT #1,BUF$:GOTO 2150
2160 CLOSE #1
2170 OPEN D$ FOR INPUT AS #1
2180 PRINT "Reading ";NM$;" from Disk":PRINT
2190 INPUT #1,BUF1$
2200 IF EOF(1) GOTO 2230
2210 IF BUF1$<>BUF$ THEN PRINT "Error Reading ";NM$:E=E+1
2220 GOTO 2190
2230 CLOSE #1:KILL D$:BUF$=LVAL$:NM$="Low Values":NEXT I
2240 PRINT "There were ";E:" Errors Encountered During This Test"
```

```
2245 ON ERROR GOTO 0
2250 IF EX=1 THEN RETURN
2260 GOSUB 2940:ON ERROR GOTO 0:GOTO 40
2270 IF ERR=61 THEN CLOSE #1:RESUME 2160
2280 IF ERR=24 THEN PRINT "Device Timeout - Check Power to Drive"
2290 IF ERR=25 THEN PRINT "Device Fault - Drive may be Faulty"
2300 IF ERR=51 THEN PRINT "Internal Error - Problem with System
     Unit"
2310 IF ERR=53 THEN PRINT "Cannot Find File - Problem with Disk
     Media"
2320 IF ERR=57 THEN PRINT "Device I/O Error - Check Drive and
     Interface"
2330 IF ERR=62 THEN PRINT "Attempt to Read Past EOF - Problem
     with Disk Media"
2340 IF ERR=68 THEN PRINT "Disk Not Available - Check Power or
     Problem with Interface"
2350 IF ERR=70 THEN PRINT "Disk is Write-Protected"
2360 IF ERR=71 THEN PRINT "Disk Not Ready - Door is Open on Drive
     or No Power to Drive"
2370 IF ERR=72 THEN PRINT "Disk Media Error - Try Another
     Diskette"
2380 CLOSE #1
2390 PRINT:PRINT "Test has Ended with Fatal Error!": GOSUB
     2940:RESUME 2080
2400 CLS:REM Serial Communication Diagnostic Module
2410 PRINT TAB(4) "Serial Communication Diagnostic":PRINT
     PRINT TAB(11) "1...Display Character Loopback Test":
     PRINT TAB(11) "2...ASCII Values Loopback Test" PRINT:
     PRINT TAB(4) "Press Esc to End Diagnostic"
2420 GOSUB 2930:IF ASC(A$)=27 THEN GOTO 40
2430 IF A$="1" THEN SW=0:GOTO 2460
2440 IF A$="2" THEN SW=1:GOTO 2460
2450 BEEP:GOTO 2420
2460 CLS:PRINT "Wire Pins 2 and 3 together":PRINT:PRINT
     "Press Esc to return to the Serial Communication
     Diagnostic Menu"
2470 PRINT:INPUT "Enter the Communication Interface to Test:
     COM1 or COM2-";C$
2480 IF C$="" THEN C$="COM1"
2490 CM$=C$+":,,,,RS,DS0"
2500 OPEN CM$ AS #1
2510 OPEN "scrn:" FOR OUTPUT AS #2
2520 CLS:LOCATE ,,1
2530 GOSUB 2930:IF ASC(A$)=27 THEN CLOSE #1:CLOSE #2:
     GOTO 2400
2540 IF A$<>"" THEN PRINT #1,A$
2550 IF EOF(1) THEN 2530
2560 IF SW=1 THEN 2580
2570 B$=INPUT$(LOC(1),#1):PRINT B$;:GOTO 2530
2580 B$=INPUT$(LOC(1),#1):PRINT ASC(B$);" ";:GOTO 2530
```

```
2590 REM Exerciser Diagnostic Module
2600 CLS:PS=0:EX=1:PRINT TAB(10)"Exerciser Diagnostic":PRINT
2610 KEY(1) ON:ON KEY(1) GOSUB 2900
2620 LOCATE 25,5:PRINT "Press F1 to End Test":LOCATE 1,1
2630 INPUT "Test Monitor (Y or N)-";MN$:PRINT
2640 INPUT "Test Printer (Y or N)-";PT$
2650 IF PT$="N" OR PT$="n" THEN GOTO 2700
2660 PRINT:INPUT "Enter the Printer Carriage Length (80 or
     132)-";WP
2670 IF WP<>80 OR WP<>132 THEN WP=80
2680 PRINT:INPUT "Enter the printer port to test (LPT1 or LPT2)-
     ";PR$
2690 IF PR$<>"lpt2" OR PR$<>"LPT2" THEN PR$="lpt1":
     PR$=PR$+":"
2695 WIDTH PR$,WP
2700 PRINT:INPUT "Test Disk Drive(s) (Y or N)-";DD$(0)
2720 IF DD$(0)="N" OR DD$(0)="n" THEN GOTO 2760
2730 PRINT:INPUT "How many drives to test-";DD
2740 FOR I=1 TO DD
2750 PRINT:INPUT "Enter Drive ID (A, B, C)-";DD$(I):NEXT I
2760 IF MN$="N" OR MN$="n" THEN GOTO 2790
2770 A$="X":A=50:DEF SEG=&HB000:GOSUB 930
2780 LOCATE 25,5:PRINT "Press F1 to End Test":LOCATE 1,1
2790 IF PT$="N" OR PT$="n" THEN GOTO 2830
2800 OPEN PR$ AS #1
2810 A$="E":W=WP:X=10:GOSUB 1590
2820 CLOSE #1
2830 IF DD$(0)="N" OR DD$(0)="n" THEN GOTO 2890
2840 LOCATE 25,5:PRINT "Press F1 to End Test":LOCATE 1,1
2850 FOR I=1 TO DD
2860 D$=DD$(I)+":TEST":CLS
2870 GOSUB 2060
2880 NEXT I
2890 PS=PS+1:GOTO 2760
2900 CLS:CLOSE #1:PRINT TAB(10)"Exerciser Diagnostic":PRINT
2910 PRINT TAB(10)"Total Passes through Exerciser-";PS
2920 GOSUB 2940:EX=0:KEY(1) OFF:ON ERROR GOTO 0:GOTO 40
2930 A$=INKEY$:IF A$="" THEN 2930 ELSE RETURN
2940 LOCATE 25,1:PRINT;:LOCATE 25,8:PRINT "Press Any Key to End
     Test";:GOSUB 2930:RETURN
```

Appendix B

Macintosh BASIC System Diagnostic Program Listing

```
'System Diagnostic
' by C. Morrison for TAB Books
'Main Menu Module
mainmenu:
CLS:PRINT TAB(12) "System Diagnostic":PRINT:PRINT TAB(3)
"<K>eyboard Test  <S>erial Comm Test"
PRINT TAB(3) "<M>onitor Test    <D>isk Drive Test"
PRINT TAB(3) "<P>rinter Test    <E>xerciser":PRINT TAB(3)
"<B>utton and Mouse Test":PRINT TAB(16) "<Q>uit":PRINT
PRINT TAB(12) "Enter Selection:"
'Look for Keystroke Here
GOSUB kbd
IF a$="K" THEN kbtest
IF a$="k" THEN kbtest
IF a$="M" THEN montest
IF a$="m" THEN montest
IF a$="P" THEN prtest
IF a$="p" THEN prtest
IF a$="D" THEN dsktest
IF a$="d" THEN dsktest
IF a$="S" THEN sertest
IF a$="s" THEN sertest
IF a$="E" THEN exercise
```

```
IF a$="e" THEN exercise
IF a$="b" THEN mousetest
IF a$="B" THEN mousetest
IF a$="Q" THEN END
IF a$="q" THEN END
BEEP:GOTO mainmenu
'keyboard module
kbtest:
sp$="Special"

CLS:PRINT TAB(12) "Keyboard Test":PRINT:PRINT TAB(11) "ASCII"
PRINT TAB(3) "Character Value Key Combination":PRINT TAB(3)
" _________ _____ ___ _____________"
keyget:
GOSUB kbd

IF ASC(A$)=27 THEN GOTO mainmenu
LOCATE 6,6:PRINT SPACE$(45);
n=ASC(a$):IF n=7 OR n=9 OR n=10 OR n=11 OR n=12 OR n=13 OR n=29
OR n=30 OR n=31 THEN GOTO nochars
IF n<32 THEN GOTO specprnt
IF n>127 THEN GOTO optionprt
LOCATE 6,6:PRINT a$;:PRINT TAB (11) ASC(a$);:GOTO keyget
nochars:
LOCATE 6,6:PRINT TAB(11) ASC(a$);:PRINT TAB(23) sp$:GOTO keyget
specprnt:
LOCATE 6,6:PRINT a$;:PRINT TAB(11) ASC(a$);:PRINT TAB(23)
sp$:GOTO keyget
optionprt:
LOCATE 6,6:PRINT a$;:PRINT TAB(11) ASC(a$);:PRINT TAB(23)
"Option":GOTO keyget
'Monitor Diagnostic Module
montest:
CLS
PRINT TAB(6) "Monitor Diagnostic":PRINT:PRINT TAB(5) "1...Display
Test":PRINT TAB(5) "2...Alignment Test":PRINT TAB(5)
"3...Character Set Test":PRINT TAB(5) "4...Intensity Test":PRINT
TAB(5) "5...Scroll Test":PRINT
PRINT TAB(5) "Press Clear to End Diagnostic"
monloop:
GOSUB kbd:IF ASC(a$)=27 THEN GOTO mainmenu
IF a$="1" THEN disp
IF a$="2" THEN alignment
IF a$="3" THEN charset
IF a$="4" THEN intensity
IF a$="5" THEN SCRL

BEEP:GOTO monloop
disp:
CLS:PRINT "Enter Any Character to Fill Screen":PRINT:PRINT
```

```
"Press Clear to End Test"
dtest:
IF ex=1 THEN GOTO mnex1
GOSUB kbd:IF LEN(a$)=0 THEN dtest
IF a$=CHR$(27) THEN montest
mnex1:
n=1
FOR i=1 TO 1843
PRINT a$;:n=n+1:IF n>97 THEN PRINT  " ":n=1
NEXT i:IF ex=1 THEN GOTO charset
GOTO dtest
alignment:
CLS:REM Alignment Test
LINE (3,3)-(490,280),,b
FOR i=1 TO 490 STEP 10
LINE (i,3)-(i,280):NEXT i
FOR i=1 TO 280 STEP 10
LINE (3,i)-(490,i):NEXT i
GOSUB waiter
GOTO montest
charset:
CLS:n=1:REM Character Set Display
FOR i=1 TO 255
IF i=7 OR i=8 OR i=9 OR i=10 OR i=11 OR i=12 OR i=13 OR i=28 OR
i=29 OR i=30 OR i=31 THEN GOTO next1
IF n<40 THEN GOTO cokay
PRINT CHR$(i):n=1:GOTO next1
cokay:
PRINT CHR$(i)+" ";:n=n+1
next1:
NEXT i
IF ex=1 THEN GOTO exck2
GOSUB waiter
GOTO montest
'Intensity Test
intensity:
CLS:a$="XXXXXX":PRINT TAB(20) "Intensity Test":PRINT
LINE (7,30)-(52,252),,bf
FOR i=1 TO 14
CALL TEXTFACE(0):REM Normal
CALL TEXTMODE(3):REM White on Black
PRINT TAB(2) a$+"   ";
PRINT a$+"   ";:REM Inhibited Display
CALL TEXTMODE(0):REM Return to Normal
PRINT a$+"   ";
CALL TEXTFACE(1):REM Bold
PRINT a$+"   ";
CALL TEXTFACE(2):REM Italic
PRINT a$+"   ";
CALL TEXTFACE(4):REM Underlined
```

```
PRINT a$:TEXTFACE(0):REM Reset to Normal before Return
NEXT i
GOSUB waiter
GOTO montest
'Scroll Test
scrl:
CLS:INPUT "Enter Numbner of Repetitions-";a
exck2:
n=31
FOR i=1 TO a
n=n+1:IF n>111 THEN n=32
FOR x=1 TO 79
PRINT CHR$(n);:n=n+1
IF n>111 THEN n=32
NEXT x
n=n+1:IF n>111 THEN n=32
PRINT CHR$(n)
NEXT i:IF ex=1 THEN RETURN
GOSUB waiter
GOTO montest
'Printer Diagnostic Module
prtest:
CLS:PRINT TAB(11)"Printer Diagnostic":PRINT:PRINT TAB(3)
"1...Printer Setup":PRINT TAB(3)"2...Sliding Alpha Test":PRINT
TAB(3) "3...Display Character Print Test":PRINT TAB(3) "4...Echo
Character Print Test"
PRINT TAB(3)"5...Horizontal Tab Test":PRINT TAB(3) "6...Line
Space Test":PRINT:PRINT TAB(5) "Press Clear to End Diagnostic"
prtloop:
GOSUB kbd
IF ASC(a$)=27 THEN GOTO mainmenu
IF a$="1" THEN GOTO setup
IF a$="2" TEHN GOTO slide
IF a$="3" THEN GOTO charprt
IF a$="4" THEN GOTO echoprt
IF a$="5" THEN GOTO htab
IF a$="6" THEN GOTO vtab
BEEP:GOTO prtloop
setup:
CLS:PRINT "Printer Setup":PRINT
INPUT "Enter the number of character positions, 80 or 132-";w
IF w<>132 THEN w=80
WIDTH "LPT1:",w
GOTO prtest
'Sliding Alpha Test
slide:
CLS:PRINT "Sliding Alpha Test":PRINT:INPUT "Enter number of
repetitions-";x
prex:
n=31
```

```
FOR i=1 TO x
l$="":n=n+1:IF n>111 THEN n=32
FOR a=1 TO w
l$=L$+CHR$(n):n=n+1
IF n>111 THEN n=32
NEXT a
LPRINT l$;:NEXT i
LPRINT
IF ex=1 THEN GOTO charprt
GOTO prtest
'Display Character Print Test
charprt:
CLS:PRINT "DisplaY Character Print Test":l$="":n=1
FOR i=32 TO 128
IF i=7 OR i=9 OR i=10 OR i=11 OR i=12 OR i=13 OR i=14 OR i=28 OR
i=29 OR i=30 OR i=31 THEN GOTO nextpass
IF n=w THEN LPRINT l$;:n=1
l$=l$+CHR$(i)+" ":n=n+2
nextpass:
NEXT i:
IF l$>"" THEN LPRINT L$
IF ex=1 THEN GOTO prex1
GOTO prtest
'Echo Character Print Test
echoprt:
CLS:PRINT "Echo Character Print Test":PRINT:PRINT "Enter
Character to Echo":PRINT:PRINT "Press Clear to End Test"

gkey:

GOSUB kbd
IF ASC(a$)=27 THEN GOTO prtest
IF ASC(a$)<32 THEN GOTO gkey
IF ASC(a$)>127 THEN GOTO gkey
prex1:
l$=""
FOR i=1 TO w
l$=l$+a$:NEXT i
LPRINT l$:IF ex=1 THEN GOTO htab
GOTO gkey
'Horizontal Tab Test
htab:
CLS:PRINT "Horizontal Tab Test"
FOR i=1 TO w
LPRINT TAB(i) "*"
NEXT i:IF ex=1 THEN GOTO vtab
GOTO prtest
'Line Feed Test
vtab:
CLS:PRINT "Line Space Test"
```

```
LPRINT "This is the First line.................................";
FOR i=2 TO 5
FOR a=1 TO i
LPRINT CHR$(10);:NEXT a
n$=STR$(i)
LPRINT "There have been "+n$+" Line Feeds";:NEXT i
LPRINT CHR$(12);:LPRINT "This is a New Page..................."
IF ex=1 THEN RETURN
GOTO prtest
'Disk Drive Diagnostic Module
dsktest:
hval$="":lval$=""
FOR i=1 TO 128:hval$=hval$+CHR$(170):NEXT i
FOR i=1 TO 128:lval$=lval$+CHR$(85):NEXT i
IF ex=1 THEN GOTO dex
dmenu:
CLS:PRINT TAB(10)"Disk Drive Diagnostic":PRINT:INPUT "Which disk
drive do you want to test-";d$:d$=d$+"Test"
dex:
buf$=hval$:nm$="High Values"
e=0
FOR i=1 TO 2
OPEN "o",1,d$
PRINT:PRINT "Writing ";nm$;" to Disk"
ON ERROR GOTO errrtn
dwrite:
PRINT #1,buf$:GOTO dwrite
finwrite:
CLOSE 1
OPEN "i",1,d$
PRINT "Reading ";nm$;" from Disk":PRINT
dread:
INPUT #1,buf1$
IF EOF(1) GOTO closeout
IF buf1$<>buf$ THEN PRINT "Error Reading ";nm$:e=e+1
GOTO dread
closeout:
CLOSE 1:KILL d$:buf$=lval$:nm$="Low Values":NEXT i
PRINT "There were ";e;"Errors Encountered During This Test"
IF ex=1 THEN RETURN
GOSUB waiter
GOTO mainmenu
errrtn:
IF ERR=61 THEN CLOSE 1:RESUME finwrite
IF ERR=52 THEN PRINT "The File is Not Open - Problem Has Occurred
With Diagnostic Program/Try a Backup"
IF ERR=53 THEN PRINT "Cannot Find File - Either Diagnostic
Program is Bad or a Problem with the Disk Media"
IF ERR=55 THEN PRINT "Attempt to Open a File Which is Already
Open - Diagnostic Program is Bad/Try a Backup"
```

```
IF ERR=57 THEN PRINT "Device I/O Error - Check Drive and
Interface"
IF ERR=68 THEN PRINT "Device Not Available - Problem with Power
or Interface"
IF ERR=70 THEN PRINT "Disk is Write-Protected"
IF ERR=74 THEN PRINT "Attempt to Read a Disk That is Not On the
Drive - Problem with Disk Media/Try Another Diskette"
PRINT:PRINT "Test has Ended With Fatal Error!":GOSUB waiter:CLOSE
1
ON ERROR GOTO 0
RESUME dmenu
'Serial Communications Diagnostic Module
sertest:
CLS:PRINT TAB(10) "Serial Communications Diagnostic":PRINT:PRINT
TAB(12) "1...Display Character Loopback Test":PRINT TAB(12)
"2...ASCII Values Loopback Test":PRINT "Remember to Attach Wire
Between Pins 5 and 9 of the Serial Communications Port"
PRINT:PRINT TAB(15) "Press Clear to End Test"
GOSUB kbd
IF ASC(a$)<>27 THEN continu
CLOSE  1:GOTO mainmenu
continu:
IF a$="1" THEN sw=0
IF a$="2" THEN sw=1
OPEN "com1: 300,n,8,1" AS 1 LEN=2000
CLS
PRINT CHR$(95);
process:
WHILE LOC(1)>0
a$=INPUT$(1,#1)
IF sw=1 THEN GOTO ascout
PRINT a$;CHR$(95);:GOTO wcheck
ascout:
PRINT ASC(a$);" ";CHR$(95);
wcheck:
WEND
GOSUB kbd
IF ASC(a$)=27 THEN CLOSE #1:GOTO sertest
PRINT #1,a$:GOTO process
'Exerciser Diagnostic Module
exercise:
ON BREAK GOSUB stopex
BREAK ON
CLS:ps=0:ex=1:PRINT TAB(10) "Exerciser Diagnostic":PRINT
INPUT "Test Monitor (Y or N) -";mn$:PRINT
INPUT "Test Printer (Y or N) - ";pt$
IF pt$="N" OR pt$="n" THEN GOTO ddck
PRINT:INPUT "Enter Printer Carriage Length (80 or 132)-";wp
IF wp<>132 THEN wp=80: WIDTH "lpt1:",wp
ddck:
```

```
PRINT:INPUT "Test Disk Drive(s) (Y or N) - ";dd$(0)
IF dd$(0)="N" OR dd$(0)="n" THEN GOTO exloop
PRINT:INPUT "How Many Drives to Test-";dd
FOR i=1 TO dd
PRINT:INPUT "Enter Drive ID (Name of Disk)-";DD$(i):NEXT i
procex:
IF mn$="N" OR mn$="n" THEN GOTO ptexck
a$="X":a=50:GOSUB dtest
ptexck:
IF pt$="N" OR pt$="n" THEN GOTO ddexck
a$="E":w=wp:x=10:GOSUB prex
ddexck:
IF dd$(0)="N" OR dd$(0)="n" THEN GOTO exloop
FOR i=1 TO dd
d$=dd$(1)+"Test":CLS:NEXT i

GOSUB dex
exloop:
ps=ps+1:GOTO procex
stopex:
CLS:CLOSE #1:PRINT TAB(10) "Exerciser Diagnostic":PRINT
PRINT TAB(10) "Total Passes through Exerciser-";ps
GOSUB waiter:ex=0:ON ERROR GOTO 0:BREAK OFF:GOTO mainmenu
'Mouse Diagnostic Module
mousetest:
CLS
PRINT TAB(10) "Mouse Diagnostic Module"
PRINT:PRINT TAB(5) "1...Range Test":PRINT TAB(5) "2...Button
Test"
PRINT:PRINT TAB(5)"Enter Test Number or Press Clear to End
Test":GOTO test
kbd:
a$=INKEY$:IF a$="" THEN GOTO kbd
RETURN
test:
GOSUB kbd
IF ASC(a$)=27 THEN mainmenu
IF a$="1" THEN GOTO range
IF a$="2" THEN GOTO btn
BEEP:GOTO test
range:
CLS
PRINT "Position the left edge of the Mouse on the left edge of
the Screen ":PRINT "and Click the button and hold it down."
gmouse:
a:MOUSE(0):IF a=0 THEN gmouse
PRINT:PRINT "Now place the Mouse on left edge of the
template.":PRINT "Roll the Mouse slowly to the right until
the right edge of the"
```

```
PRINT "Mouse is touching the right edge of the template then let
up on the button."
gmouse1:
b=MOUSE(0):IF b<=0 THEN gmouse1
x=MOUSE(1):y=MOUSE(2)
FOR i=1 TO 2
PSET (x,y),33:x=x+1:NEXT i
y=y+1:x=x-2
FOR i=1 TO 2
PSET (x,y),33:x=x+1:NEXT i
PRINT:PRINT "You should see a black box where the Mouse is
positioned on the screen."
PRINT "This should be near the right edge of the screen."
PRINT:GOSUB waiter:GOTO mousetest
btn:
CLS:PRINT TAB(10)"Button Test"
PRINT:PRINT "Click the Mouse Button once."
mlook:
a=MOUSE(0):IF a=0 THEN GOTO mlook
PRINT "Single Click is Okay"
PRINT:PRINT "Double-Click the Mouse Button."
dclick:
a=MOUSE(0):IF a<>2 THEN GOTO dclick
PRINT "Double Click is Okay":PRINT
PRINT "Now hold the Mouse Button down and drag Mouse a few
inches.":PRINT "Let up on the Button."
dragm:
a=MOUSE(0):IF a=0 THEN dragm
x=MOUSE(1):y=MOUSE(2)
dragm1:
b=MOUSE(0):IF b<0 THEN dragm1
x1=MOUSE(1):y1=MOUSE(2)
IF x=x1 AND y=y1 THEN PRINT "No Drag Occurred":GOTO waiter
PRINT "Drag Test is Okay.":GOSUB waiter:GOTO mousetest

waiter:
LOCATE 19,20:PRINT "Press Any Key to End Test";
GOSUB kbd
RETURN
kbd:
a$="":a$=INKEY$:IF LEN(a$)=0 THEN kbd
RETURN
```

Appendix C

CP/M BASIC System Diagnostic Program Listing

```
10 REM System Diagnostic Program
20 REM: Main Menu Module
30 GOSUB 2030:PRINT TAB(12) "System Diagnostic":PRINT:PRINT
   TAB(3) "<K>eyboard Test  <S>erial Comm Test":PRINT TAB(3)
   "<M>onitor Test   <D>isk Drive Test":PRINT TAB(3)  "<P>rinter
   Test   <E>xerciser":PRINT TAB(16)  "<Q>uit":PRINT
40 PRINT TAB(12) "Enter Selection:"
50 GOSUB 2040
60 IF A$="K" THEN 210
70 IF A$="k" THEN 210
80 IF A$="M" THEN 400
90 IF A$="m" THEN 400
100 IF A$="P" THEN 820
110 IF A$="p" THEN 820
120 IF A$="D" THEN 1450
130 IF A$="d" THEN 1450
140 IF A$="S" THEN 1730
150 IF A$="s" THEN 1730
160 IF A$="E" THEN 1790
170 IF A$="e" THEN 1790
180 IF A$="Q" THEN END
190 IF A$="q" THEN END
200 PRINT CHR$(7): GOTO 50
```

```
210 REM: Keyboard Diagnostic Module
220 GOSUB 2030:S=0:GOTO 250
230 GOSUB 2040
240 GOSUB 2030
250 PRINT TAB(7) "Keyboard Diagnostic Module":PRINT:PRINT TAB(13)
    "ASCII"
260 :PRINT TAB(3) "Character Value Key Combination":PRINT TAB(3)
    "--------- ----- ---------------"
270 IF S=1 THEN GOTO 290 ELSE S=1
280 GOSUB 2040
290 IF ASC(A$)=27 THEN 20
300 IF ASC(A$)<32 THEN GOTO 320
310 IF ASC(A$)<127 THEN GOTO 380
320 KY$="Ctrl"
330 IF ASC(A$)=28 THEN KY2$="\":GOTO 390
340 IF ASC(A$)=29 THEN KY2$="]":GOTO 390
350 IF ASC(A$)=30 THEN KY2$="6":GOTO 390
360 IF ASC(A$)=31 THEN KY2$="-":GOTO 390
370 N=ASC(A$)+96:KY2$=CHR$(N):GOTO 390
380 PRINT TAB(6) A$;:PRINT TAB(13) ASC(A$);:GOTO 230
390 PRINT TAB(6) A$;:PRINT TAB(13) ASC(A$);PRINT TAB(23) KY$+"
    "+KY2$:GOTO 230
400 GOSUB 2030:WIDTH 255
410 PRINT TAB(6) "Monitor Diagnostic":PRINT:PRINT TAB(5)
    "1...Display Test":PRINT TAB(5) "2...Alignment Test":PRINT
    TAB(5)     "3...Character Set Test":PRINT TAB(5) "4...Scroll
    Test":PRINT
420 PRINT TAB(5) "Press Esc to End Diagnostic"
430 GOSUB 2040:IF ASC(A$)=27 THEN 20
440 IF A$="1" THEN 500
450 IF A$="2" THEN 590
460 IF A$="3" THEN 630
470 IF A$="4" THEN 700
480 PRINT CHR$(7):GOTO 430
490 REM Character Echo Module
500 GOSUB 2030:PRINT "Enter Any Character to Fill
    Screen":PRINT:PRINT "Press Esc to End Test"
510 GOSUB 2040:IF LEN(A$)>1 THEN 510
520 IF A$=CHR$(27) THEN 400
530 FOR I=1 TO 23
540 FOR A=1 TO 80
550 IF A=79 THEN PRINT A$ ELSE PRINT A$;:NEXT A
560 NEXT I
570 IF EX=1 THEN GOTO 630
580 GOTO 510
590 GOSUB 2030:REM Alignment Test
600 FOR I=1 TO 1840
610 PRINT CHR$(1);:NEXT I
620 PRINT "Press Any Character to End Test";:GOSUB 2040:GOTO 400
630 GOSUB 2030:REM Character Set Test
```

```
640 FOR I=1 TO 255
650 IF I=7 OR I=9 OR I=10 OR I=11 OR I=12 OR I=13 OR I=26 OR I=28
    OR I=29 OR I=30 OR I=31 THEN GOTO 670
660 PRINT CHR$(I)+" ";
670 NEXT I
680 IF EX=1 THEN GOTO 720
690 PRINT:PRINT:PRINT "Press Any Character to End Test":GOSUB
    2040:GOTO 400
695 REM Scroll Test
700 GOSUB 2030:PRINT TAB(15) "Scroll Test"
710 INPUT "Enter Number of Repetitions-";A
720 N=31
730 FOR I=1 TO A
740 N=N+1:IF N>111 THEN N=32
750 FOR X=1 TO 80
760 PRINT CHR$(N);:N=N+1
770 IF N>111 THEN N=32
780 NEXT X
790 NEXT I
800 IF EX=1 THEN RETURN
810 PRINT:PRINT "Press Any Character to End Test";:GOSUB
    2040:GOTO 400
820 W=80: REM The Printer Diagnostic Module
830 GOSUB 2030:PRINT TAB(11) "Printer Diagnostic":PRINT:PRINT
    TAB(3) "1...Printer Setup":PRINT TAB(3)"2...Sliding Alpha
    Test":PRINT TAB(3)"3...Display Character Print Test":PRINT
    TAB(3)"4...Echo Character Print Test"
840 PRINT TAB(3)"5...Horizontal Tab Test":PRINT TAB(3)"6...Line
    Feed Test":PRINT:PRINT TAB(5)"Press Esc to End Diagnostic"
850 GOSUB 2040:IF ASC(A$)=27 THEN 20
860 IF A$="1" THEN 940
870 IF A$="2" THEN 1000
880 IF A$="3" THEN 1120
890 IF A$="4" THEN 1220
900 IF A$="5" THEN 1290
910 IF A$="6" THEN 1350
920 PRINT CHR$(7):GOTO 850
930 REM Printer Setup Module
940 GOSUB 2030:PRINT "Printer Setup":PRINT
950 INPUT "Enter the number of character positions, 80 or 132-";W
960 IF W<>80 OR W<>132 THEN W=80
970 WIDTH LPRINT W
980 GOTO 830
990 REM Sliding Alpha Test
1000 GOSUB 2030:PRINT "Sliding Alpha Test":PRINT:INPUT "Enter the
    number of repetitions-";X
1010 N=31
1020 FOR I=1 TO X
1030 L$="":N=N+1:IF N>111 THEN N=32
1040 FOR A=1 TO W
```

```
1050 L$=L$+CHR$(N):N=N+1
1060 IF N>111 THEN N=32
1070 NEXT A
1080 LPRINT L$;:NEXT I
1090 IF EX=1 THEN GOTO 1120
1100 GOTO 830
1110 REM Display Character Print Test
1120 GOSUB 2030:PRINT "Display Character Print Test":L$=""
1130 REM Enter pertinent Esc sequences for selection of graphics
     here
1140 FOR I=32 TO 128
1150 IF N=W THEN LPRINT L$:N=1:L$=""
1160 L$=L$+CHR$(I)+" ":N=N+2
1170 NEXT I
1180 IF LEN(L$)>0 THEN LPRINT L$
1190 IF EX=1 THEN GOTO 1240
1200 GOTO 830
1210 REM Echo Character Print Test
1220 GOSUB 2030:PRINT "Echo Character Print Test":PRINT:PRINT
     "Enter Character to Echo":PRINT:PRINT "Press Esc to End Test"
1230 L$="":GOSUB 2040:IF ASC(A$)=27 THEN GOTO 830
1240 FOR I=1 TO W
1250 L$=L$+A$:NEXT I
1260 LPRINT L$:IF EX=1 THEN GOTO 1290
1270 GOTO 1230
1280 REM Horizontal Tab Test
1290 GOSUB 2030:PRINT "Horizontal Tab Test"
1300 FOR I=1 TO W
1310 LPRINT TAB(I) "*":NEXT I
1320 IF EX=1 THEN GOTO 1350
1330 GOTO 830
1340 REM Line Feed Test
1350 GOSUB 2030:PRINT "Line Feed Test"
1360 LPRINT "This is the First Line.........................."
1370 FOR I=2 TO 5
1380 FOR A=1 TO I
1390 LPRINT CHR$(10);:NEXT A
1400 N$=STR$(I)
1410 LPRINT TAB(0)"There have been "+N$+" Line Feeds";:NEXT I
1420 LPRINT CHR$(12);:LPRINT "This is a new page............"
1430 IF EX=1 THEN RETURN
1440 GOTO 830
1450 REM: Disk Drive Diagnostic Module
1460 HVAL$="":LVAL$=""
1470 FOR I=1 TO 128:HVAL$=HVAL$+CHR$(170):NEXT I
1480 FOR I=1 TO 128:LVAL$=LVAL$+CHR$(85):NEXT I:IF EX=1 THEN GOTO
     1500
1490 GOSUB 2030:PRINT TAB(10)"Disk Drive Diagnostic":PRINT:INPUT
     "Which disk drive do you want to test-";D$:D$=D$+":Test"
1500 BUF$=HVAL$:NM$="High Values"
```

```
1510 ON ERROR GOTO 1670
1520 FOR I=1 TO 2
1530 OPEN "o",1,D$
1540 PRINT:PRINT "Writing ";NM$;" to Disk"
1550 PRINT #1,BUF$:GOTO 1550
1560 CLOSE #1
1570 OPEN "i",1,D$
1580 PRINT "Reading ";NM$;" from Disk":PRINT
1590 INPUT #1,BUF1$
1600 IF EOF(1) GOTO 1630
1610 IF BUF1$<>BUF$ THEN PRINT "Error Reading ";NM$:E=E+1
1620 GOTO 1590
1630 CLOSE #1:KILL D$:BUF$=LVAL$:NM$="Low Values":NEXT I
1640 PRINT "There Were ";E;"Errors Encountered During Test"
1650 IF EX=1 THEN RETURN
1660 GOSUB 2060:GOTO 20
1670 IF ERR=61 THEN CLOSE #1:RESUME 1560
1680 IF ERR=51 THEN PRINT "Internal Problem with MBASIC - Basic
     Interpreter is Bad"
1690 IF ERR=53 THEN PRINT "Cannot Find File - Problem with Disk
     Media"
1700 IF ERR=57 THEN PRINT "Disk I/O Error - Check Drive and
     Interface"
1710 IF ERR=62 THEN PRINT "Attempt to Read Past EOF - Problem
     with Disk Media"
1720 PRINT:PRINT "Test Has Ended With Fatal Error!":GOSUB
2060:RESUME 1490
1730 GOSUB 2030:PRINT TAB(5)"Serial communications code is too
     machine specific!"
1740 PRINT:PRINT "Refer to your individual owner's manual for
     specific information"
1750 PRINT:PRINT "to implement on your system"
1760 GOSUB 2060:GOTO 20
1770 REM Exerciser Module
1780 WIDTH 255
1790 GOSUB 2030:PS=0:EX=1:PRINT TAB(10)"Exerciser
     Diagnostic":PRINT
1800 INPUT "Test Monitor (Y or N)-";MN$:PRINT
1810 INPUT "Test Printer (Y or N)-";PT$
1820 IF PT$="N" OR PT$="n" THEN GOTO 1840
1830 PRINT:INPUT "Enter the printer carriage length (80 or 132)-
     ";WP:WIDTH LPRINT WP
1840 PRINT:INPUT "Test Disk Drive(s) (Y or N)-";DD$(0)
1850 IF DD$(0)="N" OR DD$(0)="n" THEN GOTO 1890
1860 PRINT:INPUT "How Many Drives to Test-";DD
1870 FOR I=1 TO DD
1880 PRINT:INPUT "Enter Drive ID-";DD$(I):NEXT I
1890 REM Here's where we have to set a break trap
1900 IF MN$="N" OR MN$="n" THEN GOTO 1920
1910 A$="X":A=50:GOSUB 530
```

```
1920 IF PT$="N" OR PT$="n" THEN GOTO 1940
1930 A$="E":W=WP:X=10:GOSUB 1010
1940 IF DD$(0)="N" OR DD$(0)="n" THEN GOTO 1990
1950 FOR I=1 TO DD
1960 D$=DD$(I)+":Test":GOSUB 2030
1970 GOSUB 1470
1980 NEXT I
1990 PS=PS+1:GOTO 1900
2000 GOSUB 2030:CLOSE #1:PRINT TAB(10)"Exerciser
     Diagnostic":PRINT
2010 PRINT "Total Passes through Exerciser-";PS
2020 GOSUB 2060:EX=0:ON ERROR GOTO 0:GOTO 20
2030 PRINT CHR$(26);:RETURN
2040 A$="":A$=INKEY$:IF LEN(A$)=0 THEN 2040
2050 RETURN
2060 PRINT:PRINT:PRINT TAB(10) "Press Any Character to End
     Test":GOSUB 2040:GOTO 20
```

Appendix D

Microsoft BASIC Conversions

This book provides the System Diagnostic Program in three versions of Microsoft BASIC: MS-DOS BASICA or GWBASIC, Macintosh Microsoft BASIC, and CP/M MBASIC. All three versions are very similar to one another due to their all having been written by Microsoft Corporation. Most differences tend to be machine-dependent. Table D-1 charts the significant differences among the three versions.

In addition to the instructions listed in Table D-1, there are several notable differences between the various versions of BASIC. Most notable is the fact that in the Apple version, line numbers are not required:

CP/M
MS-DOS

```
010 IF A=1 THEN GOTO 50
 •
 •
 •
050 PRINT "The End"
```

Macintosh

```
IF A=1 THEN GOTO MESS1
 •
 •
 •
MESS1:
PRINT "The End"
```

Table D-1. Microsoft BASIC Conversions.

FUNCTION	*IBM BASICA*	*MACINTOSH MICROSOFT BASIC*	*CP/M MBASIC*
clearing the screen	CLS	CLS	PRINT CHR$(26)
positioning cursor on the screen	LOCATE 1,1	LOCATE 1,1	MULTIPLE PRINT/ PRINT TAB STATEMENTS
function key processing	ON KEY(N)	not implemented	not implemented
opening files	OPEN "File" AS #1 for input	OPEN "I",1,"File"	OPEN "x"1,"File"
accessing serial communication	OPEN "COM1:"300,n,8,1" AS 1 LEN=2000	OPEN "COM1:	not available
determining special function keys	Check length of key input. Length>1 is a "special" key. The second byte contains the value: A$=INKEY$ IF LEN(A$)>1 THEN...	look for values outside 32-128	look for values outside (< >) 32-128
character attribute	COLOR N N=attribute	CALL TEXTMODE(N) N=attribute	not available

Another difference is that Macintosh BASIC operates in several different windows. Editing a program is done in the "LIST" window. Commands such as RUN are given in the "COMMAND" window. Visual output generated by "PRINT" statements occur in the "OUTPUT" window.

Still another difference is that IBM BASICA and the clones' GWBASIC support a 25th status line, while the Macintosh and CP/M systems do not.

There are a number of other differences. Refer to your BASIC programming manual for more details.

Index

Index

Edited by Marianne Krcma

Other Bestsellers From TAB

☐ **80386—A PROGRAMMING AND DESIGN HANDBOOK—Penn Brumm and Don Brumm**

The basis of IBM's much-anticipated OS/2 operating system and their new Personal System/2 computers, the 80386 microprocessor promises new standards in microcomputer power, speed, and versatility. Now, with the cooperation of 80386 designers from Intel Corporation, Penn and Don Brumm have provided the first complete sourcebook on this advanced processor, including an overview of its capabilities and in-depth information for programmers and designers. 448 pp., 150 illus.

Paper $19.95 **Hard $29.95**
Book No. 2937

☐ **CLIPPER™: dBASE® COMPILER APPLICATIONS—Gary Beam**

Whether you are a novice Clipper user in need of hands-on guidance in mastering the compiler's many special functions and features or an experienced program developer looking for new techniques to enhance your programming efficiency, this book is a must. And, to save you the time and effort of typing the programs and routines included by Beam, there is a set of two ready-to-run disks available that include both application program code and interactive utilities. 190 pp., 37 illus.

Paper $16.95 **Book No. 2917**

☐ **POWER PROGRAMMING WITH ADA® FOR THE IBM PC®—John Winters, Ph.D.**

This excellent new guide puts Ada programming within easy understanding. Whether you'd simply like to find out how Ada works or you need a fundamental knowledge of Ada to compete more effectively in the Defense Department-related marketplace, John Winters leads you easily and effectively through the principles of Ada programming from step one to actual program writing. He even includes an extensive glossary filled with sample code that's an ideal programming reference. 220 pp., 153 illus.

Paper $16.95 **Hard $24.95**
Book No. 2902

☐ **WORKING WITH FOCUS®: AN INTRODUCTION TO DATABASE MANAGEMENT—Clifford A. Schaffer**

Every aspect of this powerful, business-oriented software system is covered . . . from building and entering a database, the parts of FOCUS, and the format of data fields to entering data, FOCUS reports, the text editor, the Dialogue Manager, and user-n language. With the help of this comprehensive reference, you'll be able to develop applications more than twice as fast under FOCUS than general-purpose languages. 256 pp., 91 illus.

Paper $22.95 **Book No. 2810**

☐ **INTRODUCTION TO TELECOMMUNICATIONS SYSTEMS—P. H. Smale**

This sourcebook covers the whole range of telecommunication principles, beginning with a basic discussion of wave theory and continuing with the principles behind radio, television, telephone, and digital networks. Covers all the latest technology in telecommunications: local area networks (LANs), cellular phones, cable television, direct broadcasting by satellite, high-definition television, optic fiber systems, and mobile radio systems. 160 pp., 181 illus.

Paper $14.95 **Book No. 2924**

☐ **WORKING WITH ORACLE®—Jack L. Hursch, Ph.D., and Carolyn J. Hursch, Ph.D.**

This easy-to-understand guide shows you how you can use the ORACLE database management system to open the lines of communication between all the computers in your company—micros, minis, and mainframes. The ideal supplement to the ORACLE user's manual, this book addresses all the standard operations and features of ORACLE version 5.0. with instructions on ORACLE's query language, SQL. 240 pp., 75 illus.

Paper $19.95 **Book No. 2916**

☐ **SYSTEMS DESIGN UNDER CICS COMMAND AND VSAM—Alex Varsegi**

Here is a comprehensive summary of CICS functions, design considerations, and related software products to acquaint you with the concept of on-line data processing and its established role in current computer design. You'll cover system design, screen painting techniques using SDF to create input/output maps, the use of CECI, VSAM (virtual storage access method), all the CICS commands and how they relate to system design. 272 pp., 204 illus., 6″ × 9″.

Hard $28.95 **Book No. 2843**

☐ **Smart Apples: 31 ARTIFICIAL INTELLIGENCE EXPERIMENTS WITH THE APPLE II®, II + ®, *//e*®, *//c*®, and IIGS®—Delton T. Horn**

This unique book will help you enter the world of AI using only an Apple computer. The treasury of programs can turn your computer into an intelligent competitor, a witty conversationalist, an artist, poet, musician, or writer. Horn includes an introduction to intelligence research and covers the milestones in AI development. You'll cover such intriguing topics as the Turning Test, game applications, computer-generated stories, and more. 200 pp., 47 illus.

Paper $12.95 **Hard $18.95**
Book No. 2775

Other Bestsellers From TAB

☐ **PRODOS® INSIDE AND OUT—Dennis Doms and Tom Weishaar**

This introduction to programming with BASIC. SYSTEM gives practical tips and how-to advice on everything from booting your system to assembly language programming with ProDOS. You'll even cover such hard-to-find topics as subdirectories and their use, programming examples for the F)ield and B)yte options of text files, the use of BSAVE and BLOAD parameters to save disk space, and more! 270 pp., 113 illus.

Paper $16.95 **Hard $24.95**
Book No. 2745

☐ **THE ILLUSTRATED HANDBOOK OF DESKTOP PUBLISHING AND TYPESETTING—Michael L. Kleper**

Now, one of the nation's top authorities in desktop publishing has written *the definitive sourcebook* on the subject! Far more than a cursory overview of what desktop publishing can accomplish, this is an in-depth look at the entire scope of the innovative technology—from the use of personal computers and commercially available software to prepare professional-quality newsletters or reports to computerized input workstations for commercial phototypesetting. 784 pp., more than 800 illus., Extra Large Format 8 1/2″ × 11″.

Paper $29.95 **Hard $49.95**
Book No. 2700

☐ **ANALYSIS WITH REFLEX™—Roberts**

This exceptional, straightforward guide shows you a totally new way to look at your data using Reflex on any MS-DOS system with color and/or graphics capabilities, including the IBM®PC. You'll use it for reviewing production rates, analyzing questionnaire results, producing reports and presentations, keeping an employee database, tracking sales performance by individual and product, and much more! 224 pp., 150 illus., 7″ × 10″.

Paper $16.95 **Book No. 2712**

☐ **THE MICRO TO MAINFRAME CONNECTION—Brumm**

Highlighting the data handling capabilities offered when microcomputer versatility is combined with mainframe performance power, Brumm supplies planning checklists, details on computer linking techniques and software packages, LANs (local area networks), and public network systems. It's a complete guide to state-of-the-art options available for taming your ever-increasing flow of paperwork. 224 pp., 54 illus.

Paper $15.95 **Hard $22.95**
Book No. 2637

*Prices subject to change without notice.

Look for these and other TAB books at your local bookstore.

TAB BOOKS Inc.
P.O. Box 40
Blue Ridge Summit, PA 17294

Send for FREE TAB catalog describing over 1200 current titles in print.

OR ORDER TOLL-FREE TODAY: **1-800-233-1128**
IN PENNSYLVANIA AND ALASKA, CALL: **717-794-2191**

PC Care Manual: Diagnosing and Maintaining Your MS-DOS, CP/M or Macintosh System

If you are intrigued with the possibilities of the programs included in *PC Care Manual: Diagnosing and Maintaining Your MS-DOS, CP/M or Macintosh System* (TAB Book No. 2991), you should definitely consider having the ready-to-run disk containing the software applications. This software is guaranteed free of manufacturer's defects. (If you have any problems, return the disk within 30 days, and we'll send you a new one.) Not only will you save the time and effort of typing the programs, the disk eliminates the possibility of errors that can prevent the programs from functioning. Interested?

Available on 5¼" disk for MS/DOS and CP/M systems, and 3½" disk for the Macintosh at $24.95 for each disk plus $1.00 shipping and handling.

I'm interested. Send me:

_____ Macdisk for the Macintosh (6243S)
_____ disk for CP/M systems (6430S)
_____ disk for MS/DOS systems (6650S)
_____ TAB BOOKS catalog

Check/Money Order enclosed for $_____
plus $1.00 shipping and handling for each tape or disk ordered.

_____ VISA _____ MasterCard

Account No. _____ Expires _____

Name _____

Address _____

City _____ State _____ Zip _____

Signature _____

Mail To: **TAB BOOKS Inc.**
Blue Ridge Summit, PA 17294-0840

OR CALL TOLL-FREE TODAY: **1-800-233-1128**
IN PENNSYLVANIA AND ALASKA, CALL: **717-794-2191**

(Pa. add 6% sales tax. Orders outside U.S. must be prepaid with international money orders in U.S. dollars.)
Prices subject to change without notice.

TAB 2991

So that TAB BOOKS Inc. can better fill your reading needs . . .

please take a moment to complete and return this card. We appreciate your comments and suggestions.

1. I am interested in books on the following subjects:

- ☐ automotive
- ☐ aviation
- ☐ business
- ☐ computer, hobby
- ☐ computer, professional
- ☐ engineering (specify) _____
- ☐ other (specify) _____
- ☐ other (specify) _____
- ☐ electronics, hobby
- ☐ electronics, professional
- ☐ finance
- ☐ how to, do-it-yourself

2. I own/use a computer:

- ☐ IBM _____
- ☐ Apple _____
- ☐ Commodore _____
- ☐ Other (specify) _____
- ☐ Macintosh _____
- ☐ ATARI _____
- ☐ Amiga _____

3. This card came from TAB book (specify title and/or number):

_____

4. I purchase books:

- ☐ from general bookstores
- ☐ from technical bookstores
- ☐ from college bookstores
- ☐ other (specify) _____
- ☐ through the mail
- ☐ by telephone
- ☐ by electronic mail

Comments _____

_____

Name _____

Company _____

Address _____

City _____

State _____ Zip _____

See page 207 for a Special Companion Diskette Offer.

I'm interested. Send me:

- _____ Macdisk for Macintosh systems (6243S)
- _____ disk for MS DOS systems (6650S)
- _____ disk for CP/M systems (6430S)
- _____ TAB BOOKS catalog

Charge my credit card for _____ ($24.95 plus $1.00 shipping and handling for each disk ordered).

_____ VISA _____ MasterCard

Account No. _____ Expires _____

Name _____

Address _____

City _____ State _____ Zip _____

Signature _____

OR CALL TOLL FREE TODAY: 1-800-233-1128
IN PENNSYLVANIA AND ALASKA, CALL 717-794-2191

(Pa. add 6% sales tax. Orders outside U.S. must be prepaid with international money orders in U.S. dollars.)
Prices subject to change without notice.

TAB 2991

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 9 BLUE RIDGE SUMMIT, PA 17214

POSTAGE WILL BE PAID BY ADDRESSEE

TAB BOOKS Inc.
P.O. Box 40
Blue Ridge Summit, PA 17214-9988

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 9 BLUE RIDGE SUMMIT, PA 17214

POSTAGE WILL BE PAID BY ADDRESSEE

TAB BOOKS Inc.
P.O. Box 40
Blue Ridge Summit, PA 17214-9988