3141

COMMODORE™ CARE MANUAL: Diagnosing and Maintaining Your 64 or 128 System

CHRIS MORRISON AND
TERESA S. STOVER

COMMODORE™ CARE MANUAL: Diagnosing and Maintaining Your Commodore 64 or 128 System

COMMODORE™ CARE MANUAL: Diagnosing and Maintaining Your Commodore 64 or 128 System

CHRIS MORRISON AND
TERESA S. STOVER

TAB
TAB BOOKS Inc.
Blue Ridge Summit, PA

FIRST EDITION
FIRST PRINTING

Library of Congress Cataloging in Publication Data

Morrison, Chris.
Commodore care manual : diagnosing and maintaining your 64 or 128 system / by Chris Morrison and Teresa S. Stover.
p. cm.
Includes index.
ISBN 0-8306-3141-0 (pbk.)
1. Microcomputers—Maintenance and repair. 2. Commodore 64 (Computer) 3. Commodore 128 (Computer) I. Stover, Teresa S. II. Title.
TK887.M65 1988 88-4925
621.391'6—dc19 CIP

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

Reader Inquiry Branch
TAB BOOKS Inc.
Blue Ridge Summit, PA 17294-0214

Program List

Acknowledgments

The following individuals helped bring about the successful completion of this book, and for that we would like to express our hearty appreciation.

Craig A. Stover edited the manuscript, shot the photographs, provided moral support, and acted as our public relations representative through his incessant bragging about us.

Pat A. Mahony edited the manuscript, gave us encouragement and inspiration, and always made us feel like we were doing something worth doing.

Diana H. Price edited the manuscript, and offered us her unique blend of humor, sympathy, and food while having to work with the authors.

Ilva Klar created the manual line drawings and loaned us her Commodore 64 system for research and development of this book and the software.

We would especially like to thank Qubix Graphic Systems, Inc. for allowing Chris to create the electronic line drawings of the Qubix Designer Workstation.

Contents

Introduction

The computer is perhaps among the most revolutionary inventions in our history. Its presence influences our culture and each of our lives in many ways. Particular prominence belongs to the microcomputer, known as the "personal computer." You might use a personal computer such as the Commodore 64 or Commodore 128 to help manage a small business, to do schoolwork, to plan the home budget, to keep records, to write, or even to play games. Knowledge of computer software applications, however, is not necessarily accompanied by knowledge of the computer hardware and its functions.

Being a machine, the computer can malfunction and break down, just like any other device or appliance you might use. If you've had your computer for a while, and it's beginning to show signs of age and wear, you might have already experienced trips to a computer service center, with their disheartening repair bills. Although you are not a computer technician, and you do not know dc and ac electronics, you do feel you are capable of having more control over the care of your computer—saving yourself time and money.

This book helps 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 Commodore and to diagnose problems as they emerge.

This book provides you with three basic approaches to Commodore care: preventive maintenance, diagnostics when something goes wrong, and general repair once you discover the source of the problem.

These methods and procedures will help you better understand the internal workings of your system, enabling you to maintain a well-running system less likely to experience long periods of "downtime." You will save time from not having to deal with a broken computer, and save money from fewer trips to the computer service center. You might also gain a new understanding of BASIC programming. You will certainly gain more control from knowledge about your computer.

PREVENTIVE MAINTENANCE

To avoid breakdowns and their associated troubles, certain preventive measures are necessary. Following these procedures will help you maintain your computer in its best possible condition.

Chapter 1 outlines general practices that help keep your Commodore healthy. It also explains periodic preventive maintenance procedures that can stave off system problems. Troubleshooting techniques are explained to help you isolate problems.

Each succeeding chapter specializes in a particular Commodore subsystem: the system unit, the keyboard, the monitor, the printer, the cassette tape drive, the disk drive, and the serial communication interface. Each of these subsystem chapters provides a brief theory of operation, a list of possible problems with troubleshooting and repair techniques, and an explanation of the associated diagnostic program module.

REPAIR GUIDELINES

If you suspect a problem with some part of your computer system, refer to the appropriate subsystem chapter. Within that chapter, refer to the "Troubleshooting and Repair Guidelines" section. Then look up the problem that fits what your system is experiencing. Follow the troubleshooting and repair instructions in the order given. This becomes the computer's "treatment plan."

By following these instructions, you can often fix the problem yourself. If it's a more complex problem, you can take the system to a computer service center with a more educated in-depth explanation, a diagnosis of the problem, and a specific directive of what must be done. This saves the technician repair time, thereby saving you money.

DIAGNOSTICS

In addition to the "Troubleshooting and Repair Guidelines," the "System Diagnostic Program" can assist with the troubleshooting task. This BASIC program includes seven modules, each designed to perform a series of tests on a different component. You can buy the program on a diskette (refer to the last page of this book), or you can key it in yourself, because complete program listings for both the Commodore 64 and the Commodore 128 are provided in the appendices.

The System Diagnostic Program lets you gather information in order to determine whether the subsystem being tested has a problem, and to pinpoint where that problem lies. The "Troubleshooting and Repair Guidelines" indicate when a particular test should be performed. Instructions for running the module are provided in their own section, along with illustrations of expected test results. After executing these tests, you'll be able to make a diagnosis and take the appropriate action.

The program listings and detailed line-by-line explanations of this program "code" are also included in the corresponding chapters. This is particularly useful if you know how to program in BASIC, and would like to understand the workings and logic of the programs. This can also help you modify the program for your own purposes or for creating new modules for additional subsystems and peripherals.

In these line-by-line explanations, the code for the Commodore 64 is used as a model. The code for the C64 and C128 are similar, but not identical. Whenever there is a difference between the two, this is noted within the text. In addition, Appendix C summarizes the major programming differences between the two systems.

The program listings provided throughout this book use a particular convention. When a special symbol representing a special key or program output is part of an instruction, { } braces enclose the word describing the symbol. For example, consider the following instruction:

```
PRINT "{CLR/HOME}":RETURN
```

The words CLR/HOME are not actually in the instruction; rather, the CLR/HOME key is pressed as part of the instruction. The words are simply spelled out in these listings for your clarification.

Another example of this is found in the Color Test of the Monitor Diagnostic Module:

```
PRINT "{CTRL 1} Black" TAB(12)C$
```

Here, the words in braces, CTRL 1, are the keys that are pressed in order to change the color to black.

If you are not familiar with BASIC, you can easily skip the program explanations and still use the diagnostic modules successfully. The entire program is ready to use. Just create a backup of the program, then load it into your system.

EXERCISER

After you have your computer repaired, you will want to make sure that the problem has really been solved. The computer is a system of many components, and as such, when something goes wrong in one part of the computer, it could actually be caused by something in another part. The Exerciser Module of the System Diagnostic Program lets you make sure that the system as a whole is working properly. This is described in Chapter 8.

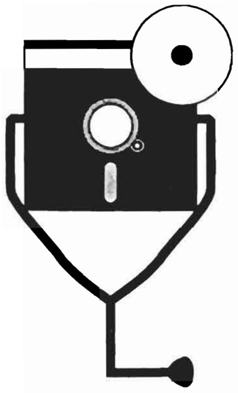
When you run the Exerciser Module after having a problem repaired, all system functions are exercised and run for a period of time. This is much like test-driving your car after repair. The program "burns-in" and tests the system to confirm that it works correctly and to ensure that there are no recurring problems.

CREATING YOUR OWN DIAGNOSTICS

If you know BASIC programming, this book provides you with the techniques necessary to develop new diagnostic routines for peripherals not covered in this book. Suppose you have another peripheral such as a digitizer tablet or a joystick, for which you want to develop a diagnostic program. The information in this book presents a systematic diagnostic approach that will help you learn how to predict the problems that can occur with that equipment.

Once you learn the types of problems that can occur, and you have the information needed to diagnose these problems, you can write a program or add a new routine appropriate to such a troubleshooting activity. Chapter 9 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. However, we cannot guarantee that your computer will behave as expected, nor can we be responsible for any malfunctions or damages that occur as a result of following the suggestions in this book. Nonetheless, the diagnostics and instructions should be sufficient to help you pinpoint problems when they arise.



Chapter 1 Getting Started

This chapter helps you to better understand your Commodore system, to use the System Diagnostic Program to troubleshoot problems, and to get these problems fixed through troubleshooting and repair procedures.

First, the Commodore system with its standard components and peripherals are described. Next, the System Diagnostic Program is discussed, along with a summary of the individual modules. Resources and tools necessary for troubleshooting and repairing Commodore malfunctions are listed. In addition, a section on general maintenance and troubleshooting techniques is included to provide a good foundation for computer care.

SYSTEM OVERVIEW

The typical Commodore computer system consists of the following components:

- ☐ system unit
- ☐ keyboard
- ☐ monitor
- ☐ printer
- ☐ cassette drive and/or disk drive(s)
- ☐ serial communication interface

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

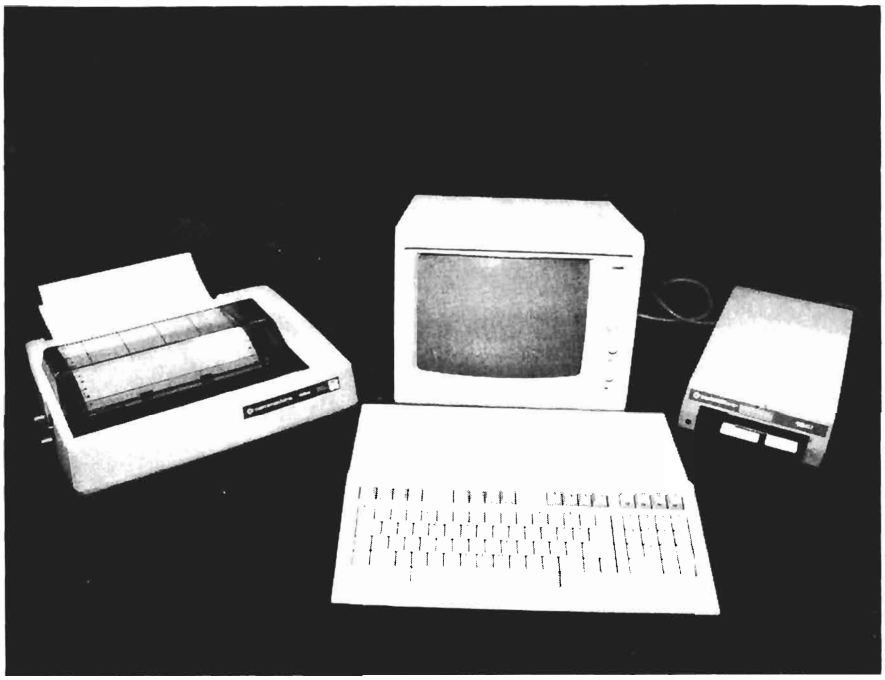


Fig. 1-1. Basic Commodore configuration.

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 System Unit

The system unit is the actual computer. On the Commodore systems, the system unit is housed in the same assembly as the keyboard. The system unit is the "brains" of the computer, and as such, is the single most important component of the computer.

The system unit houses the *central processing unit* (CPU), as well as the various interfaces for the computer's peripherals. The CPU consists of the *microprocessor* chip that runs the computer. It issues commands that cause calculations to be made and processes to be run. The CPU coordinates the various input and output devices. It transfers and processes data to and from the outside world.

The Commodore 64 system unit consists of the 6510 CPU chip with 64 kilobytes of memory. It also includes the disk drive controller, serial communication interface, serial printer interface, cassette interface, and monitor interface.

The system unit for the Commodore 128 includes the 6502/8502 CPU chips, a Z80 processor for running CP/M, 128 kilobytes of memory, a disk controller, serial communication interface, serial printer interface, cassette interface, and monitor interface. Just like the Commodore 64, the 128 system unit is housed in the same unit as the keyboard. Further information about the system unit is found in Chapter 2.

The Keyboard

The keyboard is the Commodore's primary input device from the outside world. Typed characters form commands, move the cursor, and enter data. The keyboard receives the code of the characters entered, sends them to the system unit for processing and interpretation, and displays the appropriate results on the monitor screen. Further information about the keyboard is provided in Chapter 2.

The Monitor

The monitor is the visual output for work done on the Commodore. It is also known as a *cathode ray tube* (CRT) or *video display*. The monitor is the output device that shows your input data from the keyboard, modem, joystick, or other input device. It also displays the result of system processing performed within the system unit. The Commodore usually uses a color monitor, and this can be either a standard television set or a computer monitor. Further information about the monitor is found in Chapter 3.

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 serial interface connection between the system unit and the printer causes data and processing results to be transferred from the system to the printer. You might have a dot-matrix or letter-quality printer. Learn more about the printer in Chapter 4.

The Cassette and Disk Drives

Both the cassette drive and disk drive are used to store and recall software programs and data by magnetically recording and recalling electronic bits. The cassette tape drive uses regular audio cassette tapes for storage, while the disk drive uses floppy diskettes as the storage media. Chapter 5 discusses the cassette drive in greater detail, and Chapter 6 does the same for the disk drive.

The Serial Communication Interface

The serial communication interface is the means for your Commodore to "talk" to the outside world. It can connect to, or *interface* with, a *modem*, which sends data to and receives data 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. For further information about the serial communication interface, refer to Chapter 7.

THE SYSTEM DIAGNOSTIC PROGRAM

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

System Diagnostic Program Summary

The block diagram in Fig. 1-2 outlines the structure of the System Diagnostic Program. As you can see, lines 5 –35 are set aside for explanation remarks as well as for the storage of any constants and variables that are used throughout 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 is the System Diagnostic Main Menu. The program code for the Main Menu Module, residing in lines 50–180, 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 Monitor."

The second module in the program is the Keyboard Diagnostic Module. This module runs a test which echoes and gives the numeric code, or *CHR$ code 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 200–740.

The third module is the Monitor Diagnostic Module. This module consists of six different tests residing in lines 800–1930:

- ☐ 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 Scroll Test provides a sliding character set to let you check that the monitor is scrolling correctly.
- ☐ The Color Test checks the 16 available colors with a color pallet test.
- ☐ The Sprite Test checks the special circuitry used to display and move sprites, which are used in many Commodore game applications.

The Printer Diagnostic Module is stored at lines 3500–4150. It is made up of five tests:

- ☐ 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.

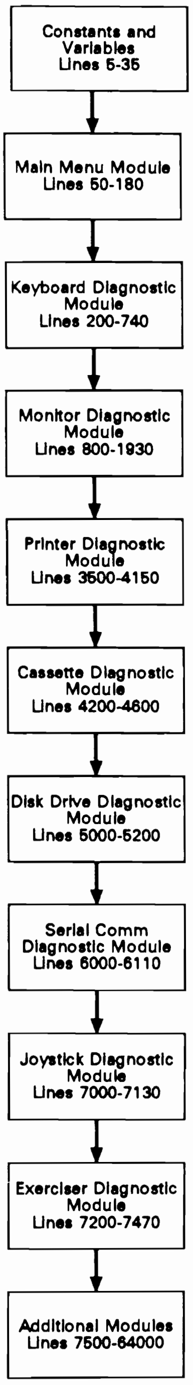

Constants and Variables Lines 5-35
Main Menu Module Lines 50-180
Keyboard Diagnostic Module Lines 200-740
Monitor Diagnostic Module Lines 800-1930
Printer Diagnostic Module Lines 3500-4150
Cassette Diagnostic Module Lines 4200-4600
Disk Drive Diagnostic Module Lines 5000-5200
Serial Comm Diagnostic Module Lines 6000-6110
Joystick Diagnostic Module Lines 7000-7130
Exerciser Diagnostic Module Lines 7200-7470
Additional Modules Lines 7500-64000

Fig. 1-2. System Diagnostic Program block diagram.

- ☐ 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 the feeding of a new continuous form page.

The Cassette Drive Diagnostic Module tests and exercises the two main functions of a cassette drive: reading and writing to the cassette. This module is found at lines 4200–4600.

Likewise, 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 both the 1541 and 1571 floppy disk drive models. Lines 5000–5200 contain this disk drive test.

Lines 6000–6110 contain the Serial Communication Diagnostic Module, which lets you test the serial communication 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 Joystick Diagnostic Module, located at lines 7000 through 7130, is the example used in Chapter 9 to demonstrate how you could go about developing and integrating your own diagnostic module for any additional peripherals you might have.

The final program module is the System Exerciser Module at lines 7200 to 7470. If you have just purchased a new Commodore, 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. This is particularly useful when the computer is first set up, or when it is moved, because that is when most problems take place. The Exerciser Module allows you to burn-in and check system functions automatically and unattended.

Five system utility subroutines are included at lines 2990 through 3300. These subroutines include the Clear Screen subroutine, the Get-a-Key subroutine, the Beep subroutine, the Key-to-End-Test subroutine, and the Key-to-Continue subroutine. These subroutines are explained later in this chapter.

Running the System Diagnostic Program

If you have purchased the diskette containing the System Diagnostic Program (refer to the back page of this book for more information), 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 BASIC on your system, and type in the appropriate version of the code as listed in the appendices. Appendix A provides the Commodore 64 program listing, and Appendix B provides the Commodore 128 version.

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

After keying the program in, be sure to save the program on your disk or tape, and make another backup copy of it.

NOTE: It is imperative that you key the program in *before* you have a system problem. If you wait until after you suspect a malfunction, you might not be able to enter the program for some reason attributed to this malfunction. Even if you are able to enter and run the program, your test results might be inaccurate and misleading.

From a Disk Drive. To run the diagnostic on your disk drive, follow these steps:

1. Boot up the system. BASIC is loaded automatically, and the OK prompt is displayed.
2. Place the diagnostic diskette into the disk drive.
3. Enter LOAD "diag",8. The System Diagnostic program is loaded into BASIC.
4. Enter RUN.

From a Cassette Drive. To run the diagnostic from a cassette tape drive, follow these steps:

1. Boot up the system. BASIC is loaded automatically, and the OK prompt is displayed.
2. Insert the cassette with the diagnostic program into the cassette drive.
3. Enter LOAD "diag". (The file name "diag" is simply the name we've given the System Diagnostic Program. Of course, you can name it whatever you like when you're entering the program yourself, or you can change the file name later.) The System Diagnostic Program is loaded into BASIC, and the OK prompt is displayed.
4. Enter RUN.

Once the program is loaded, the screen displays the System Diagnostic Main Menu. This menu allows you to choose from several diagnostic options, as shown in Fig. 1-3.

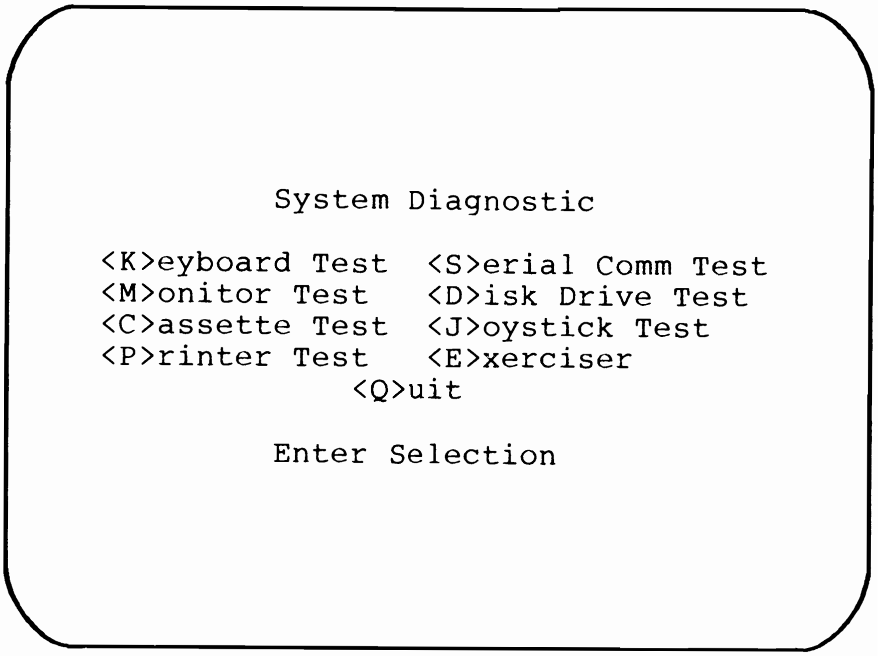
A selection is made by pressing the one-character command which is the first letter of the name of the desired test. For example, if you wish to run the keyboard test, enter K or k. The program is not character case-sensitive.

When you press a valid selection, the screen is cleared and you are either presented with a new menu, or the desired test begins 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.

When you press Q, the program "quits," or ceases operation, and returns you to the BASIC environment.

How the Main Menu Module Works

Now that you are familiar with the general structure of the Main Menu Module, you can begin to understand how the program works, line by line. A listing of the Main Menu Module is found in Fig. 1-4.



```
                  System Diagnostic

     <K>eyboard Test   <S>erial Comm Test
     <M>onitor Test    <D>isk Drive Test
     <C>assette Test   <J>oystick Test
     <P>rinter Test    <E>xerciser
                    <Q>uit

                  Enter Selection
```

Fig. 1-3. The System Diagnostic Main Menu.

Remember, you can skip this section if you are not familiar with BASIC programming, or if you are not interested in how the program works. Skipping this section and the others like it throughout the book will not in any way diminish your ability to use the System Diagnostic Program to troubleshoot and repair your Commodore.

Line 1 indicates the version of the code—Commodore 64 or Commodore 128:

```
1 REM "Commodore 64"
```

Line 5 indicates the title and copyright of the System Diagnostic Program:

```
5 REM "System Diagnostic Program Copyright 1988 TAB BOOKS Inc."
```

Line 10 uses the POKE statement to set the graphic character mode to be used in the test. Then an array is established for a keyboard map by using a DIMENSION statement to initialize variable K$ to 63:

```
10 POKE 53272,23:DIM K$ (63) :REM Set Character Set
```

C128 USERS: In the C128 code, the POKE and DIM K$ statements are modified to accommodate the larger keyboard.

Lines 20 through 24 contain DATA statements. This series of DATA statements

```
  1 REM "Commodore 64"
  5 REM "System Diagnostic Program Copyright 1988
    TAB BOOKS Inc."
 10 POKE 53272,23:DIM K$(63):REM Set Character Set
 20 DATA "INST/DEL","RETURN","CRSR RT","F7","F1","F3","F6",
    "CRSR UP","3","W","A"
 21 DATA "4","Z","S","E"," ","5","r","D","6","C","F",
    "T","X","7","Y","G","8","B"
 22 DATA "H","U","V","9","I","J","0","M","K","O","N",
    "+","P","L","-",".",":","@"
 23 DATA ",","£","*",";","CLR/HOME"," ","=","↑","/","1",
    "←"," ","2","SPACE"
 24 DATA " ","Q"
 25 FOR I=0 TO 62: READ K$(I):NEXT
 30 DIM R(8):DATA 1,2,4,8,16,32,64,128
 35 FOR I=0 TO 7:READ R(I):NEXT
 50 GOSUB 2990: PRINT TAB(12) "System Diagnostic":PRINT:
 60 PRINT TAB(3) "<K>eyboard Text  <S>erial Comm Test"
 70 PRINT TAB(3) "<M>onitor Test   <D>isk Drive Test":
 80 PRINT TAB(3) "<C>assette Test  <J>oystick Test":
 85 PRINT TAB(3) "<P>rinter Test   <E>xerciser":
 90 PRINT TAB(16) "<Q>uit":PRINT:PRINT TAB(12)
    "Enter Selection"
100 GOSUB 3000:REM Get Key Command
110 IF A$="K" OR A$="k" THEN 200
120 IF A$="M" OR A$="m" THEN 800
130 IF A$="P" OR A$="p" THEN 3500
135 IF A$="C" or A$="c" THEN 4200
140 IF A$="D" OR A$="d" THEN 5000
150 IF A$="S" OR A$="s" THEN 6000
160 IF A$="E" OR A$="e" THEN 7200
165 IF A$="J" OR A$="j" THEN 7000
170 IF A$="Q" OR A$="q" THEN END
180 GOSUB 3100:GOTO 80
```

Fig. 1-4. The System Diagnostic Main Menu Module listing.

serves to establish the keyboard map within the program. This map corresponds to the column/row layout of the keyboard so that the appropriate keycap value is displayed when a CTRL or COMMODORE (C=) key combination is pressed. This is necessary because a character might display that is not indicated on the actual keycap.

```
20 DATA "INST/DEL","RETURN","CRSR RT","F7","F1","F3","F6","CRSR
   UP","3","W","A"
```

```
21 DATA   "4","Z","S","E","    ","5","r","D","6","C","F","T","X","7",
   "Y","G","8","B"
22 DATA "H","U","V","9","I","J","0"," M","K","O","N"," + ","P","L",
   "-",".",":","@"
23 DATA ",","+","*",",","CLR/HOME","   ","=","    ","/","1","    ","   ",
   "2","SPACE"
24 DATA " ","Q"
```

For example, at position 0 of this keyboard map is the INSERT/DELETE key, position 1 is the RETURN key, and position 2 is the CURSOR RIGHT key.

C128 USERS: In the C128 code, line 24 is expanded to include the 8, 5, RUN/STOP, HELP, and TAB keys. There are also three additional lines of DATA instructions to accommodate the function keys available exclusively on the Commodore 128 keyboard. These include the LINE FEED, NO SCROLL, and ARROW keys.

The FOR-NEXT loop at line 25 causes the 62 values assigned in the keyboard map to be read into variable array R:

```
25 FOR I=0 TO 62: READ K$(I):NEXT
```

The (I) is the FOR-NEXT loop index, and this value determines that the characters fall into keyboard map K$, a variable that is used later in the Keyboard Diagnostic Module.

C128 USERS: The C128 code includes the 88 values assigned in the keyboard map to be read into variable array R.

Lines 30 and 35 initialize another array, designated as R, which is used later for the Sprite Test in the Monitor Diagnostic Module:

```
30 DIM R(8):DATA 1,2,4,8,16,32,64,128
35 FOR I=0 TO 7:READ R(I):NEXT
```

C128 USERS: The C128 code has two additional lines at this point. The first one loads an array with the CHR$ values of the eight function keys. Line 37 is a FOR-TO loop which reads these values into the array. Refer to Chapter 2, "How the Keyboard Module Works," for more information on how this array is used. Refer to Appendix B for the actual C128 code listing.

Lines 50 through 90 display the System Diagnostic Main Menu on the screen. Line 50 calls up the Clear Screen subroutine at Line 2990, which clears the screen and places the cursor at the top left corner. Then a series of PRINT commands display the title and the various menu options:

```
50  GOSUB 2990: PRINT TAB(12) "System Diagnostic":PRINT:
60  PRINT TAB(3) "<K>eyboard Test <S>erial Comm Test"
70  PRINT TAB(3) "<M>onitor Test <D>isk Drive Test":
80  PRINT TAB(3) "<C>assette Test <J>oystick Test":
85  PRINT TAB(3) "<P>rinter Test <E>xerciser":
90  PRINT TAB(16) "<Q>uit":PRINT:PRINT TAB(12) "Enter Selection"
```

Line 100 sends the program to line 3000 for the Get-a-Key subroutine:

```
100  GOSUB 3000:REM Get Key Command
```

When the user presses the key indicating the desired system module diagnostic test, the CHR$ code value of the key is placed into variable A$.

Lines 110 through 170 check to see which key was pressed for a particular menu choice. Each menu choice represents the diagnostic test for a different system module, and is chosen by pressing the indicated character. Pressing K or k initiates the Keyboard Diagnostic, just as pressing D or d initiates the Disk Drive Diagnostic, and so forth.

When you press the designated key to choose a particular module test, the program branches to the Get-a-Key subroutine, and the CHR$ code value of the key pressed is placed into variable A$. If A$ is equal to K or k, then line 110 causes the program to branch to line 200, which is the beginning of the Keyboard Diagnostic Module:

```
110  IF A$="K" OR A$="k" THEN 200
```

Likewise, if A$ is equal to M or m, then line 100 causes the program to branch to line 800, the beginning of the Monitor Diagnostic Module:

```
120  IF A$="M" OR A$="m" THEN 800
```

This is also the case for lines 130 through 165:

```
130  IF A$="P" OR A$="p" THEN 3500
135  IF A$="C" or A$="c" THEN 4200
140  IF A$="D" OR A$="d" THEN 5000
150  IF A$="S" OR A$="s" THEN 6000
160  IF A$="E" OR A$="e" THEN 7200
165  IF A$="J" OR A$="j" THEN 7000
```

Line 170 is similar, but slightly different. This line checks to see if you have pressed Q or q, indicating that you are finished with the program and wish to "quit" out of it:

```
170  IF A$="Q" OR A$="q" THEN END
```

In this case, rather than branching to another module, the program simply ends, and the system exits to the BASIC environment.

If the user does not press any of the valid keystrokes as presented in the menu, the program falls through to line 180. Line 180 goes to the Beep subroutine at line 3100:

```
180 GOSUB 3100:GOTO 80
```

After the beep, the GOTO statement branches the program back to Line 80 to await another user keystroke.

THE UTILITY SUBROUTINES

The System Diagnostic Program employs five subroutines (Fig. 1-5) used repeatedly throughout the code to execute program utility functions such as clearing the screen or emitting a beep. These subroutines are the following:

- ☐ the Clear Screen subroutine
- ☐ the Get-a-Key subroutine
- ☐ the Beep subroutine
- ☐ the Key-to-End-Test subroutine
- ☐ the Key-to-Continue subroutine

The Clear Screen Subroutine

The Clear Screen subroutine consists of just one line. The commands within line 2900 clear the screen and place the cursor at the top left corner of the screen:

```
2900 PRINT "{HOME/CLEAR}":RETURN:REM "Clear Screen Routine"
3000 GET A$:IF A$="" THEN 3000:REM "Get a Key Routine"
3005 K=PEEK(197):S=PEEK(653)
3010 RETURN
3099 REM "Beep Routine"
3100 FOR L=54272 TO 54296:POKE L,0:NEXT
3110 POKE 54296,15:POKE 54277,90:POKE 54278,1:
     POKE 54273,21:POKE 54272,31
3120 POKE 54276,33
3130 FOR T=1 TO 50:NEXT
3140 POKE 54276,16:RETURN
3200 PRINT TAB(7)"press any key to end test";
3210 GOSUB 3000
3220 RETURN
3300 PRINT TAB(7)"press any key to continue":RETURN
```

Fig. 1-5. The System Utility Subroutines.

```
2900 PRINT "{HOME/CLEAR}":RETURN:REM "Clear Screen Routine"
```

This is used to clear any messages or displays left on the screen before putting up a new display. It also places the cursor in an appropriate location so that the new display fits neatly within the limits of the screen. This subroutine is first encountered at line 50, when the screen must be cleared and the cursor placed at the HOME position before printing the System Diagnostic Main Menu on the screen.

The Get-a-Key Subroutine

The Get-a-Key subroutine begins at line 3000 and is used throughout the System Diagnostic Program. It is first encountered at line 100 in the Main Menu Module. The system branches to this subroutine to actually get the key that has been pressed on the keyboard. The value of this key is placed at the appropriate place in the program to perform the function required at that time.

```
3000 GET A$:IF A$="" THEN 3000:REM "Get a Key Routine"
3005 K=PEEK(197):S=PEEK(653)
3010 RETURN
```

The GET command in line 3000 retrieves the user's keystroke directly from the keyboard and places it into A$. The IF statement checks whether or not a key has actually been entered. If no key is entered, then variable A$ is equal to a *null*, meaning that there is nothing in the variable. The null is specified by the two double-quotes:

```
3000 GET A$:IF A$="" THEN 3000:REM "Get a Key Routine"
```

As long as A$ is equal to null, the instruction loops back on itself continuously. The loop is only broken when a key is pressed.

When a key is pressed, the program falls through to line 3005, which contains two "PEEK" statements. A PEEK reads a value out of memory:

```
3005 K=PEEK(197):S=PEEK(653)
```

When the Get-a-Key subroutine gets a key for the keyboard test, the value of the key is placed into memory location 197. Here, the program reads this value from memory to use it as an index into the column/row matrix of the keyboard map, which was built with the DATA statements starting at line 35. This index is used later in line 595, where the program gets the key value from the keyboard map.

The second PEEK statement on line 3005 sets variable S to memory location 653. This memory location contains the SHIFT/C=/CTRL condition of the key that has been pressed. If there is a zero in memory location 653, the SHIFT key has not been pressed. If location 653 contains a one, the SHIFT key has been pressed. If it contains a two, the C= key has been pressed. And if it contains a four, the

CTRL key has been pressed. (For further information about how these commands function in the program, refer to "How the Keyboard Diagnostic Module Works" in Chapter 2.)

The RETURN instruction sends the program back to where it was in the code before it branched to this subroutine to get a key:

```
3010  RETURN
```

The Beep Subroutine

Because the Commodore 64 does not have a specific command or ASCII character to create a beep, the Beep subroutine performs this function, using these statements:

```
3099  REM "Beep Routine"
3100  FOR L=54272 TO 54296:POKE L,0:NEXT
3110  POKE 54296,15:POKE 54277,90:POKE 54278,1:
      POKE 54273,21:POKE 54272,31
3120  POKE 54276,33
3130  FOR T=1 TO 50:NEXT
3140  POKE 54276,16:RETURN
```

The Commodore's built-in sound routines are used in this subroutine to produce a beep to indicate an input error. This first appears in the Main Menu Module at line 180, when the Beep subroutine is called up if the user presses a key that is not a valid command key.

The Beep subroutine begins at line 3099 with a REMARK statement:

```
3099  REM Beep Routine
```

To produce sounds, Commodore uses machine language commands expressed in numbers. A BASIC POKE statement lets you write values directly into the memory locations where these numeric machine language commands reside. Line 3110 begins with a FOR-NEXT loop including such a POKE statement:

```
3100  FOR L=54272 TO 54296:POKE L,0:NEXT
```

This line clears the *Sound Interface Device* (SID) registers to prepare for subsequent sound generation. A register is a special memory location that is an integral part of the chip's circuitry. To clear the registers, a value of zero is "POKEd" into the registers located at memory addresses 54272 through 54296.

C128 USERS: Line 3100 takes care of the entire Beep subroutine in the C128 code, reflecting the use of the C128 beep character, CHR$(7). The POKE and PEEK statements described below apply only to the C64 version of the subroutine.

Line 3110 consists of five POKE statements:

```
3110 POKE 54296,15:POKE 54277,90:POKE 54278,1:
     POKE 54273,21:POKE 54272,31
```

POKE 54296,15 sets the volume (at location 54296) at its highest setting (represented by the value 15). If you wanted to make the volume of the beep lower, you could enter a number smaller than 15.

POKE 54277,90 defines how fast the sound rises and falls from its peak level.

POKE 54278,1 defines the level to prolong a note. A value of one does not hold the sound very long, because a quick beeping effect is desired in this instance.

POKE 54273,21 and POKE 54272,31 specify which note on the musical scale the beep should be. If you were to refer to the Appendix of the Commodore User's Guide, you would find a table of musical notes, with the high and low oscillator frequencies. In this particular case, the note is an E.

Line 3120 is another POKE statement which sets a waveform. The value of 33 defines the *resonance* of the sound—how shrill or muted the tone is:

```
3120 POKE 54276,33
```

Line 3130 is a FOR-NEXT loop which is nothing more than a short delay loop. This stops the execution of the Beep subroutine, because other than this loop, there is actually no processing taking place in this line:

```
3130 FOR T=1 TO 50:NEXT
```

This sets the duration that the note is to be played as the beep. For example, if the TO value were set to 250 instead of 50, you would hear a quarter-note, which is one standard musical beat. The value of 50 provides a very short, crisp beep that seems to be the most appropriate for an alarm.

Line 3140 halts the beep:

```
3140 POKE 54276,16:RETURN
```

It then returns the program back to where it was in the code before it branched to this subroutine to sound a beep.

The Key-to-End-Test Subroutine

The Key-to-End-Test subroutine consists of three lines:

```
3200 PRINT TAB(7)"press any key to end test";
3210 GOSUB 3000
3220 RETURN
```

When the program branches here on a GOSUB instruction, the message - press any key to end test" is displayed on the screen. Line 3210 sends this subroutine to the Get-a-Key subroutine at line 3000. When a key is received in line 3000, the program returns to the place in the program from which it came.

The Key-to-Continue Subroutine

The Key-to-Continue subroutine is a one-line instruction at line 3300:

```
3300  PRINT TAB(7)"press any key to continue":RETURN
```

This subroutine is used several times throughout the System Diagnostic Program to allow the user to continue with the test at hand.

Program Modification and Adaptation

As you have seen, the System Diagnostic Program is written in Commodore 64/128 BASIC, and is structured to accommodate modification and additions.

You are not restricted, therefore, to simply testing the standard components of a typical Commodore system. Should you purchase a new peripheral device for your Commodore, such as a joystick, a different type of monitor, or a digitizer pad, you can write a new diagnostic module for that device and add it to this program. Chapter 9 covers the techniques, information, and references used for writing diagnostics for new devices.

TOOLS AND RESOURCES

Troubleshooting and repairing computers, or for that matter any electronic system, might seem to be a difficult 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 as to the nature of the malfunction. They repair the problem by making physical adjustments and replacing faulty components.

To perform such 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. Technicians use a good set of tools, and they have spare parts available to them. In addition, they have various information resources from which to draw.

Although you cannot expect to do as thorough a job as a computer technician without the same investment in training, tools, and spares, you can take cues by learning some of their basic principles and techniques. Then when your computer malfunctions, 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 documentation will provide the necessary specifics about your particular Commodore configuration. This information is vital when installing and setting up a new device, or if your computer malfunctions.

If you have inadvertently lost or thrown out important documentation, you can probably obtain a copy from your Commodore dealer, 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 obtain good-quality tools. This does not necessarily mean you should buy the most expensive ones 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. 1-6 illustrates these tools. Table 1-2 lists optional tools, and Fig. 1-7 provides their illustration. Table 1-3 lists cleaning supplies, and Fig. 1-8 illustrates them. The tables all list the functions, uses and approximate costs for the tools and cleaning supplies.

Spares

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

In your case, however, maintaining an inventory of spare parts for your Commodore would be costly and impractical. An alternative to this is to find someone (a friend, a colleague at work, an acquaintance in a user's group) who also owns a Commodore system. Work out a deal with this person enabling you to temporarily swap parts with each other when one of your systems malfunctions. You can use the other 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. A good practice might be to restrict swapping parts to major components, such as disk drives and monitors.

Find out where you can get the best prices for computer parts. There might be a good discount computer or electronics supply store in your area that stocks computer components at lower rates than a computer retailer. The want ads in

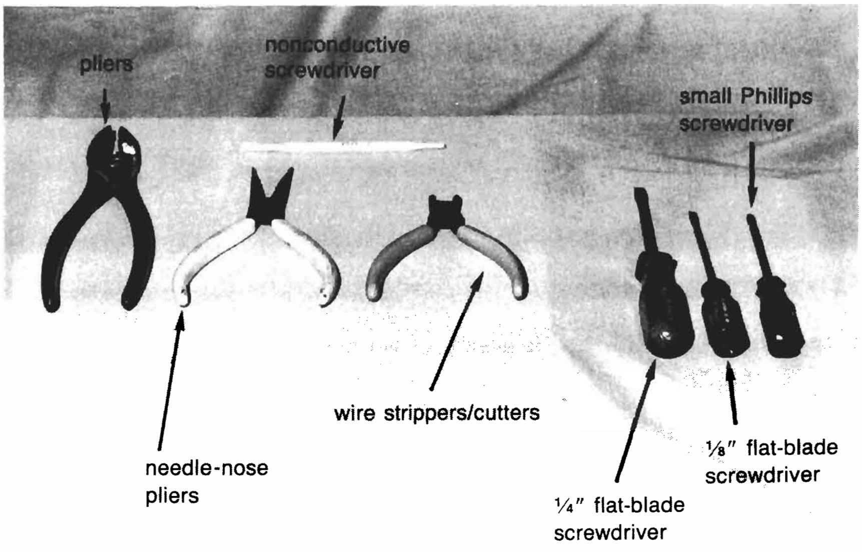

pliers
nonconductive screwdriver
small Phillips screwdriver
needle-nose pliers
wire strippers/cutters
⅛" flat-blade screwdriver
¼" flat-blade screwdriver

Fig. 1-6. Required tools.

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

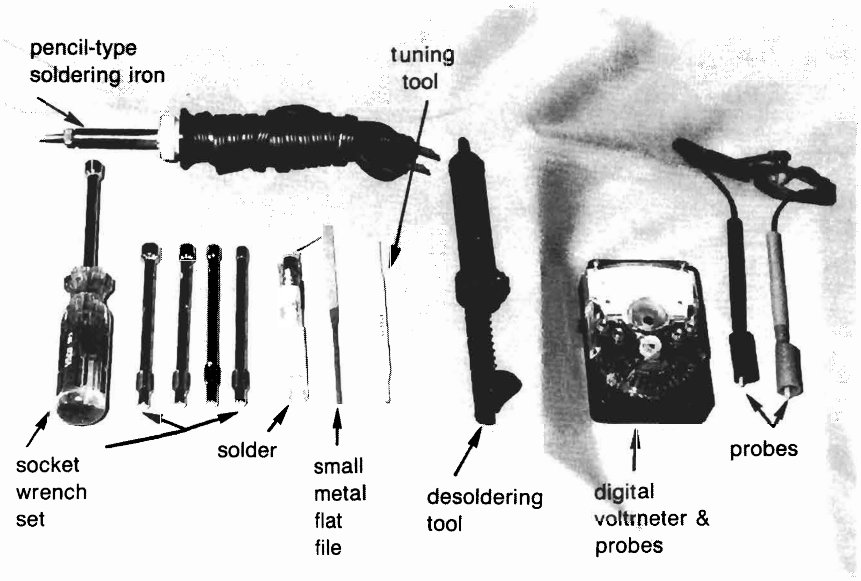


Fig. 1-7. Optional tools.

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

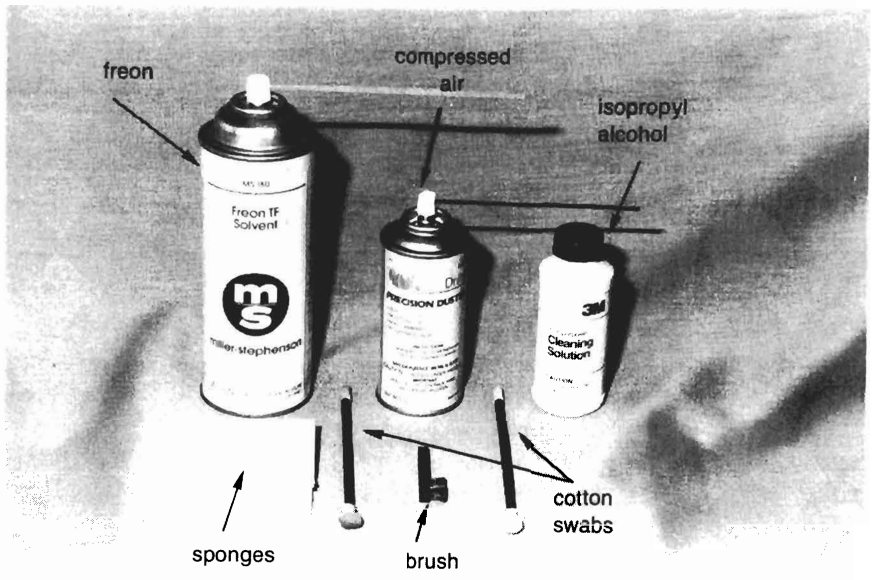


Fig. 1-8. Cleaning supplies.

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

the back pages of most computer magazines are good sources for used parts, as are computer flea markets and swap meets.

User Resources

Computer resources become more and more important, the more involved you become with your computer. It becomes less important to know *everything* if you have good resources that provide you with accurate information.

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

Keep in mind that a friend of yours with a Commodore system might know things you don't. Exchange information with any other Commodore users of your acquaintance. Everyone's computer experience and knowledge is different.

Commodore 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 Commodore computer magazines. They often provide good advice, information, and further resources.

TROUBLESHOOTING AND REPAIR GUIDELINES

This book is broken down in chapters according to major computer subsystems. If you're having trouble with your keyboard, turn to Chapter 2, "The System Unit/Keyboard." The 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 are not mechanically or technically "inclined," you should have little problem troubleshooting your Commodore, 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 could 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 grow. When in doubt, let the professionals do it. Don't put your computer in needless jeopardy.

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

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

MAINTENANCE TECHNIQUES

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

General System Care

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

Dust Covers. Buy dust covers to protect the monitor, system unit/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 the legs, of the integrated circuit (IC) chips on the system's printed circuit boards. These contacts have electric current flowing through them, so that they become ionized, and attract particles of dust. The dust attracts moisture out of the air. As this process 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. Get into the habit of covering your system when not in use.

Static Electricity. Static electricity is another computer foe. The IC chips throughout your system contain minute electronic circuits that can be easily damaged. These chips are particularly vulnerable to static electricity. If you live in a dry, cool climate, your computer stands an even greater risk of getting "zapped."

Most devices within the computer are protected from electricity up to 2–5 kilovolts. However, the electrostatic discharge from a human can be more than 20 kilovolts. This means that you could actually be the source of static electricity that can damage your system's components.

There are several ways to protect your computer from static electricity. You can spray the floor or carpet around the system with an anti-static solution, or you can lay down an anti-static mat under the computer. You can get anti-static spray as well as an anti-static 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 file cabinet, before you touch the computer.

This grounds you and keeps you from transferring static electricity to your computer.

If you ever handle a printed circuit board, do not touch the metal edge connectors. This will discharge the electricity in your body through the printed circuit board to the nearest IC chip.

Power Surge Protection. The electricity at the wall sockets is rated at 110/120 volts at 60 Hertz (cycles per second). There are moments when surges of electricity exceed 110/120 volts, and other moments when sags in voltage are lower than normal. You've probably experienced drops in electricity when your lights dim. This "brown-out" effect might cause your keyboard to temporarily lock up, and you might lose some data due to logic-related problems within the computer.

The most damaging 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 rectifies it to 5, 12, or 15 volts dc. These are the voltages with which the integrated circuits in the computer are designed to operate. When the computer's power supply receives a large power surge, it is not capable of suppressing it. This instantaneous spike is sent through to the chips, damaging or even destroying them.

Because boards and many chips are expensive to repair and replace, protect the computer from these indiscriminate surges by using a surge protector.

A surge protector isolates the voltage spikes in order to provide the system with a constant 110/120-volt supply. Obtain a surge protector at a computer or electronics supply store. Using a surge protector with your computer extends the life of its integrated circuits.

Food and Drink. Don't eat or drink near your computer. If you spill or drip something into your keyboard, you'll have an emergency cleaning job on your hands, or even the instant need for a new keyboard. Because the keyboard and system unit are included in the same housing, any liquid that spills in the keyboard could seep into the system unit, causing far more serious problems, such as shorting components out and causing permanent damage.

Cookie crumbs, sandwich particles, and dinner bits can all end up inside your keyboard and system unit, if you eat while working. This can destroy the keyboard contacts and render your keyboard useless. It can also damage components of the system unit.

Preventive Maintenance

You should perform the following preventive maintenance procedures for your computer at least every three months. These 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. 1-9), or the printer mechanism. This clears out any dust and dirt that gets into the workings in spite of the dust cover.

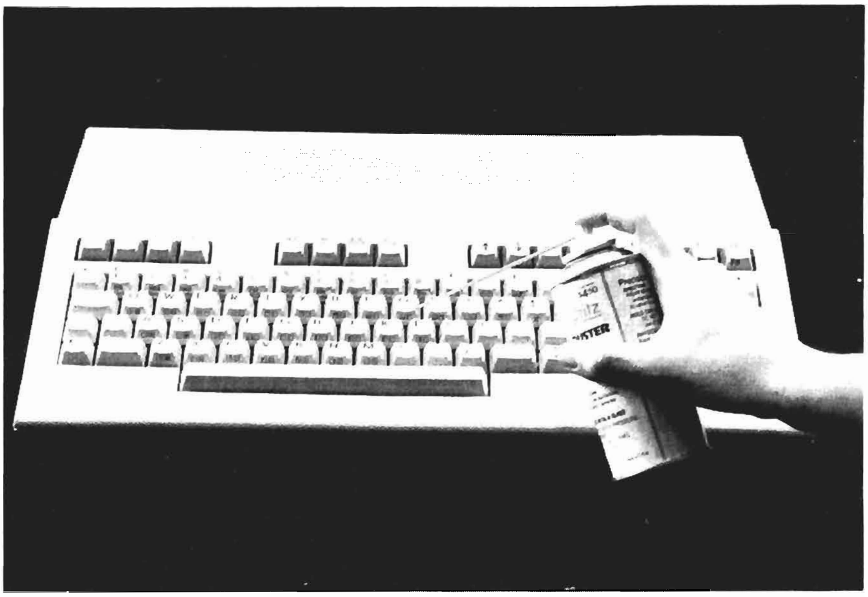


Fig. 1-9. Dusting the keyboard with compressed air.

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.

Disk drive head cleaning kits are available at most computer stores for approximately six dollars. Insert a special diskette permeated with a cleaning solution, enter a command that causes the diskette to rotate, and the read/write heads are cleaned in a flash.

If you use a cassette drive with your system, you can get cassette cleaner from a record or stereo store. You can also use cotton swabs lightly moistened with isopropyl alcohol to clean the read/write heads of the cassette drive.

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 lightly moistened with isopropyl alcohol to clean hard-to-reach areas.

Loose Cables. Several cables may be connected to the system unit at the rear and side. These cables connect interface circuits to the peripheral devices. The cable connectors are plugged into the system unit's receptacles. If the cable includes screws, they should be tightened down. These cables loosen when, for example, you move the printer or bump the modem. Suddenly a peripheral stops working because it's not getting the proper connection.

The solution to this is to use a small screwdriver and secure the cable into its receptacle. If the cable does not have screw mounts, then simply make sure it is correctly pressed into the receptacle.

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? Is the fuse burned out?
- ☐ Are all the connectors properly seated?
- ☐ Are all power switches turned to the ON position?

Each subsystem chapter that follows lists several more specific solutions to try when troubleshooting a problem.

When you've tried everything and the problem still is not fixed, the system unit itself is often involved. Refer to Chapter 2, "System Unit/Keyboard" for system unit troubleshooting procedures.

If those system unit measures do not effect a repair, it's time to take the computer in for servicing.

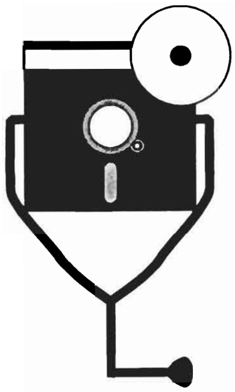
WHAT TO DO FIRST

To get started on the general care, preventive maintenance, troubleshooting, and repair of your Commodore, do the following:

- ☐ Read through this book to familiarize yourself with the concepts and organization of the material. 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, key 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 someone who also owns a Commodore system, and agree to loan and borrow components when needed for troubleshooting. Start establishing your computer user resources.
- ☐ Try out the diagnostics now, before you have a problem. Know how the results are supposed to appear under normal circumstances.
- ☐ Code any necessary modifications into the System Diagnostic Program. These modifications might have to do with accommodating another peripheral, or

a different kind of system, printer, or keyboard than what is allowed in this program. Test the modified program thoroughly, and when you're satisfied that the program is working as it should be, make a backup copy.

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



Chapter 2
The System Unit/ Keyboard

The system unit is the "brains," of the computer—the controller and coordinator for everything that takes place. It directs the activities of peripheral devices, issues commands, and processes input and output. Commands and data are input to the system unit for processing via the keyboard, which is the primary external link to the computer. The Commodore houses the system unit and keyboard within the same unit.

DESCRIPTION AND FUNCTION OF THE SYSTEM UNIT

The system unit consists of the following:

- ☐ central processing unit (CPU) and associated support chips
- ☐ random-access memory
- ☐ serial bus
- ☐ receptacles for optional peripheral devices such as the printer and modem
- ☐ operating system or basic input/output system (BIOS), which operates using read-only memory devices (ROMs).

Although there is no specific system unit module within the System Diagnostic Program, the modules testing each of the subsystems perform a sufficient job of exercising the system as a whole. This is because all subsystems are connected into and controlled by the system unit.

DESCRIPTION AND FUNCTION OF THE KEYBOARD

The keyboard is used to enter data, commands, and selections to work with your application software, whether the data are numbers in a spreadsheet, words in a word processor, or records in a database. The keyboard is your Commodore's primary input device.

The Commodore uses an *integral* keyboard, meaning it is housed together with the system unit. The CPU, memory, interfaces, and keyboard are all contained in one housing (Fig. 2-1).

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 CONTROL, INSERT, and DELETE keys. The C64 keyboard is a standard keyboard like this.

More "deluxe" system keyboards such as the C128 keyboard also include a ten-key pad, additional arrow cursor keys, programmable function keys, and other special function keys such as CAPS LOCK, NO SCROLL, and ALT (Fig. 2-2).

How the Keyboard Functions

Regardless of how basic or deluxe a keyboard is, however, they all work according to the same principle: pressing a key sends a code to the CPU in 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 unique "code" to be placed into a keyboard buffer. What this code consists of depends upon which key or key combination has been pressed.

The matrix of key and key-combination code is referred to as a *hard-wire mapping* of the keys. When you press a key, an electromechanical process takes place to generate a column and row value, indicating the keyboard location from where the key came.

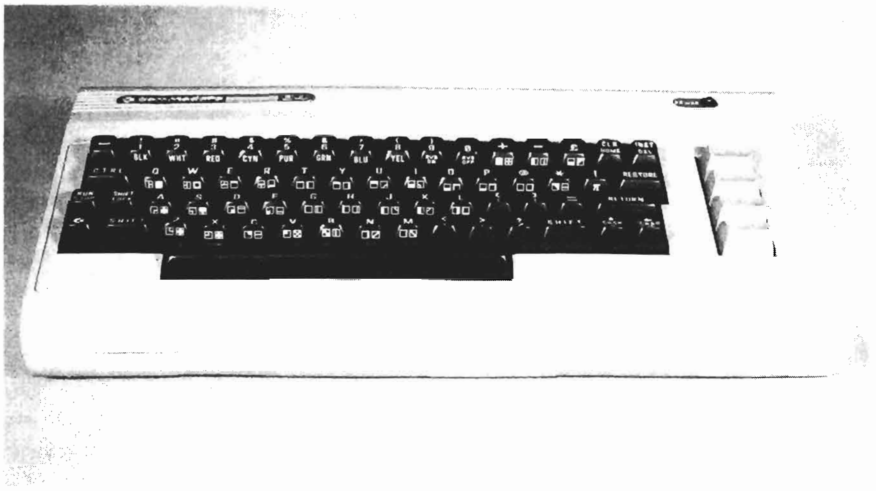


Fig. 2-1. Commodore 64 keyboard.

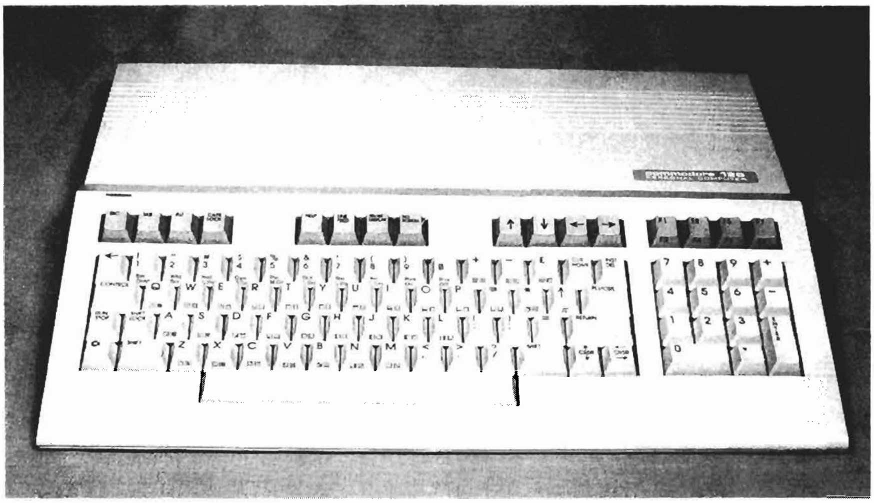


Fig. 2-2. Commodore 128 keyboard.

The physical action of pressing the key causes the CPU, which interfaces with the keyboard, to gather the column and row information, indicating where the key is located on the keyboard. The CPU also discerns whether the SHIFT, CTRL, or C= (Commodore logo) key is pressed.

This information is sent into a program residing within the *kernel* portion of the system's read-only memory (ROM), and it is here that the calculations are made about what the keystroke actually is and what it is expected to do.

The kernel handles input (keyboard) and output (monitor), or *I/O* functions for the Commodore. In this case, it takes the character and transfers it to the keyboard buffer. Further processing in turn transfers the character to the monitor's screen memory, thus displaying the character on the screen.

TROUBLESHOOTING AND REPAIR GUIDELINES

This section describes the problems that the system unit and keyboard could experience. It explains what causes these problems, and details the steps to fix them.

In subsequent chapters covering troubleshooting procedures for each of the subsystems, you'll learn that if nothing happens at all to begin with, the system unit is the suspect component to troubleshoot. In addition, if you try all the troubleshooting suggestions for a particular subsystem, but still cannot find the problem, again, the problem is likely to be within the system unit. Those sections direct you to this section to find and repair the system unit problem.

Problem: The Keyboard Doesn't Work At All

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

☑ Solutions

- ☐ The first thing to check is whether or not the system unit's red LED power light is illuminated. If it is not lit, the system unit is not getting power. Check your electrical outlet, the power strip if applicable, the wall switch if applicable, and the fuse box or circuit breaker.

  If these are all in order, the next thing to do is check the system's power supply. This is a fairly simple procedure, because the power supply is external to the system (Fig. 2-3). Remove the power supply and swap it on another Commodore system. If that system does not power up either, then buying and installing a new power supply will solve your problem. A Commodore power supply costs approximately $40.

- ☐ If the red LED light is illuminated, but the keyboard does not work, check the monitor for your standard screen prompts. These prompts indicate that you are in the BASIC environment by displaying the word READY and then a block cursor. If you do not see any of these, there is likely to be a problem with the system unit. Turn the computer off, Make sure the power supply cable is securely connected to the system, and turn the computer on again. If the screen display still does not look right, take the system unit in for servicing.

- ☐ If the red LED is illuminated and the system prompts are displayed correctly on the screen, but the keyboard still does not work, this indicates a keyboard problem rather than a system unit problem. Follow these steps:
  1. Turn the system off.
  2. Disconnect the power supply.
  3. Disconnect all attached peripheral devices (cassette drive, disk drive, printer, monitor, etc.).
  4. Remove the screws from the bottom of the system unit (Fig. 2-4).
  5. Open the system unit. You'll see a ribbon cable running from the keyboard onto the printed circuit board containing the CPU chip and many other

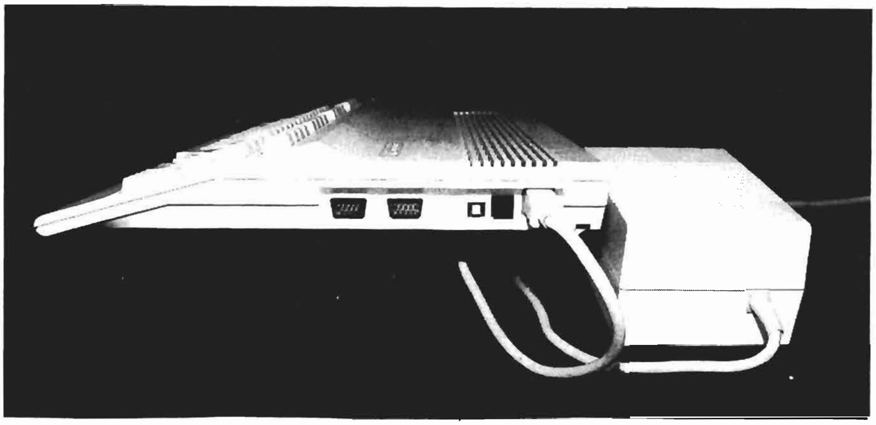


Fig. 2-3. System unit and power supply.

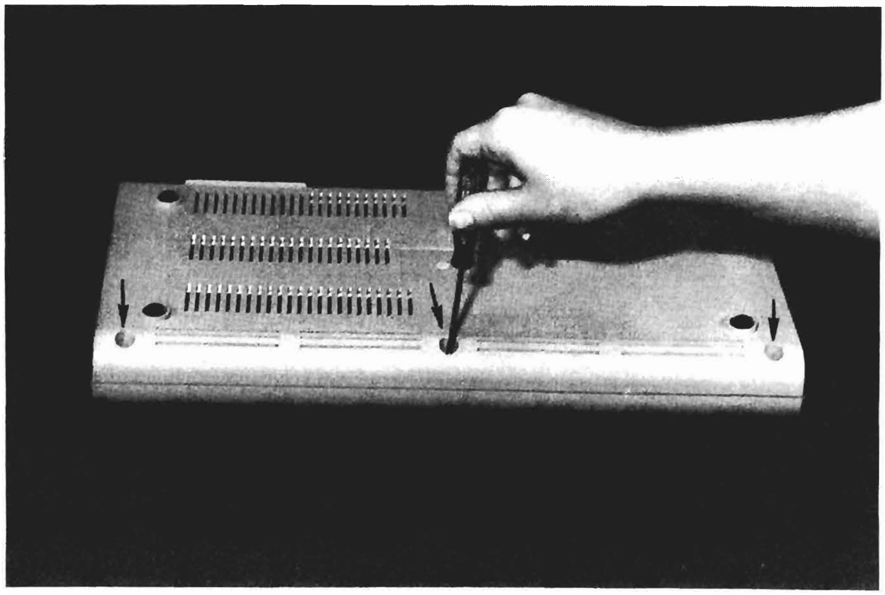


Fig. 2-4. Removing the system unit screws.

integrated circuits (Fig. 2-5). This board is referred to as the *motherboard* or *system board.*

6. Disconnect the ribbon cable from the board side and check the pins for *oxidation*. Over a period of time, oxidation can form on the cable pins and edge connector. Oxidation is caused by contact with air, moisture,

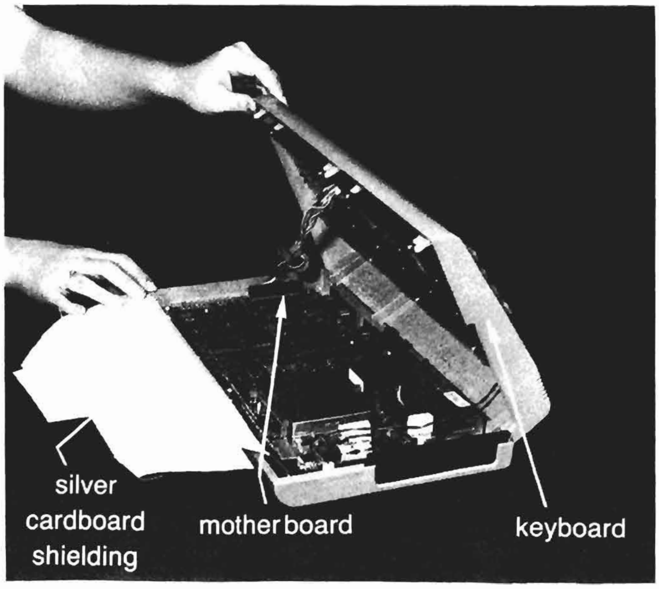


Fig. 2-5. Opening the system unit.

or a combination of the two in humidity. Oxidation tarnishes the metal, and can look like either a dull gray or black film.

7. If there is oxidation, remove it with a pencil eraser (Fig. 2-6). You can also use an emery cloth, very fine sandpaper (400-800 grit), or a lint-free cloth moistened with isopropyl alcohol.
8. Connect the cable to the edge connector again, close the system unit, reconnect the power supply and monitor, turn on the system unit, and check to see if the keyboard works now.

☐ Another possibility is that the integrated circuit chip which controls the keyboard circuitry has a problem. Take the system in for servicing, tell the technician that you did see the monitor prompts and have checked the keyboard connections. The technician can then check the keyboard IC, and if that's the problem, it can simply be replaced.

☐ If the keyboard circuitry is not the problem, replace the keyboard. Replacement keyboards are available through a Commodore dealer, service center, or magazine mail order. However, the keyboard should not need to be replaced unless it has been handled roughly or dropped.

Problem: Particular Keys Do Not Work

You might be typing along and notice what looks like a "typo." Perhaps there is an "a" that was left out of a word. You go back to the word and type the "a"

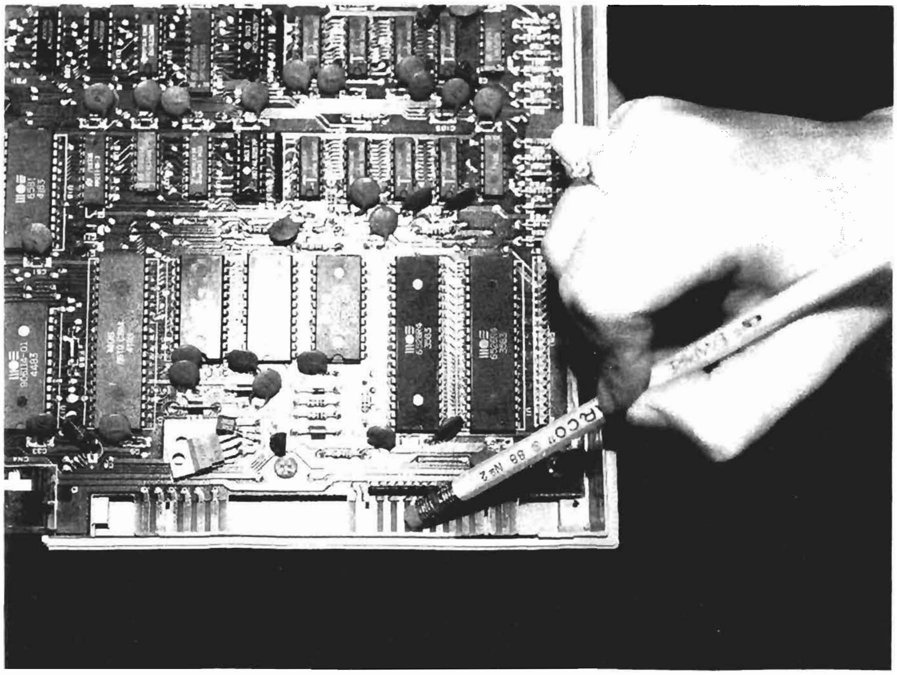


Fig. 2-6. Erasing oxidation.

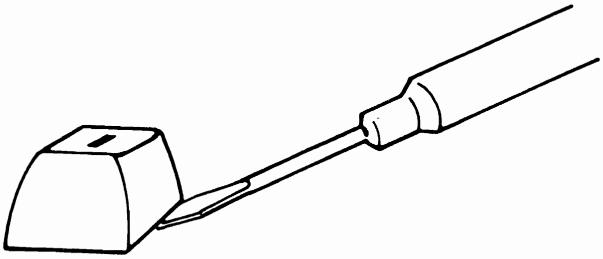


Fig. 2-7. Removing a key.

where it's supposed to be, but nothing happens; the "a" simply is not working, while all the other keys are working fine.

First run the Keyboard Diagnostic Module and confirm that the key does not work.

☑ Solution

☐ If the key does not work, use a flatblade screwdriver, or better yet, a key removal tool, to lift the key from the keyboard (Fig. 2-7). Spray contact cleaner on the key post (Fig. 2-8). Contact cleaner is available at electronics stores. Replace the keycap on the post and try 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

A sticking key stays down when you press it, causing its character to fill up the whole screen until you pry the key up again. There are three remedies for this problem.

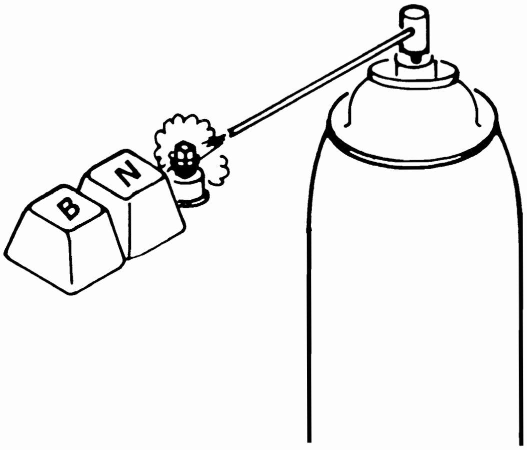


Fig. 2-8. Spraying contact cleaner on the key post.

☑ Solutions

- ☐ Lift the key off and spray contact cleaner on the key post as described in the previous solution. 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.
- ☐ If the key is not binding on the case, pop the key off and lightly file the edges of the key itself (Fig. 2-9). This should loosen the fit between the key and the key case so that the key can move more freely and no longer 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 is because the key springs soften as they get older.

☑ Solutions

- ☐ Call around to computer retailers to find keyboard springs for your keyboard. Lift off the keycap and then remove the spring from its post (Fig. 2-10). Replace the old spring with the new, and pop the keycap back on.
- ☐ If you can't find a replacement spring, use one from the keyboard that is rarely used. It could be one of the function keys, or perhaps one of the numbers on the 128 ten-key pad.

RUNNING THE KEYBOARD DIAGNOSTIC MODULE

The Keyboard Diagnostic Module consists of the Echo Key Test. This test displays, or echoes, any key pressed on the keyboard. If you press a special

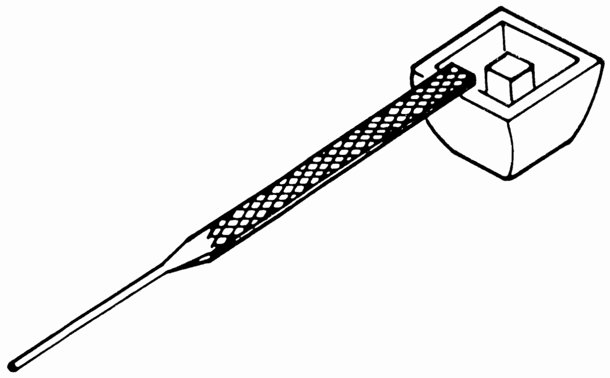


Fig. 2-9. Filing the edges of the key.

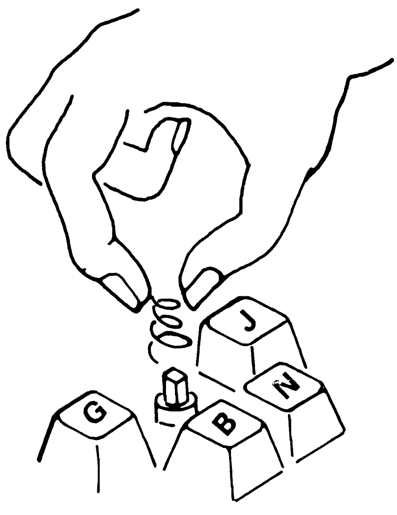


Fig. 2-10. Removing the key spring from the post.

character or key combination, the name of the character or combination is displayed.

There are two graphic character sets available on the Commodore. The first character set is the uppercase-only mode (set 1). The second character set is the upper/lowercase mode (set 2). To switch between the two modes, press the SHIFT and C= keys simultaneously (Fig. 2-11). Only one character set can be active at a time.

The graphic characters available with each of these character sets are shown on the front of the key caps. There are two characters: one on the left, and the other on the right (Fig. 2-12). If you're in the upper/lowercase mode, and you press the C= key with one of the keys, the graphic character shown on the left side of the key is displayed on the screen. If you're in the uppercase mode and you press the C= key with one of the keys, the graphic character on the right side of the key is shown.

The System Diagnostic Program automatically shifts into the default upper/lowercase mode when it's run. However, in the Keyboard Diagnostic, you can choose either character set. An informational message is displayed, indicating which character set is currently active. The menus and prompts also switch to the current character mode. In other words, if the upper/lowercase mode is active, all command prompts are displayed in upper/lowercase letters. If you press the SHIFT and the C= key while in the Keyboard Diagnostic, the menu shifts to all uppercase characters.

In either mode, when you press the C= key to test a graphic character, the resulting graphic character and its graphic CHR$ code are displayed.

The CHR$ code values are used to represent the standard display characters (such as A through Z), special control functions like CURSOR LEFT and CLEAR SCREEN, special graphic characters and symbols, and reverse video characters as a numeric code. The Commodore recognizes 256 such CHR$ code values. The CHR$ code

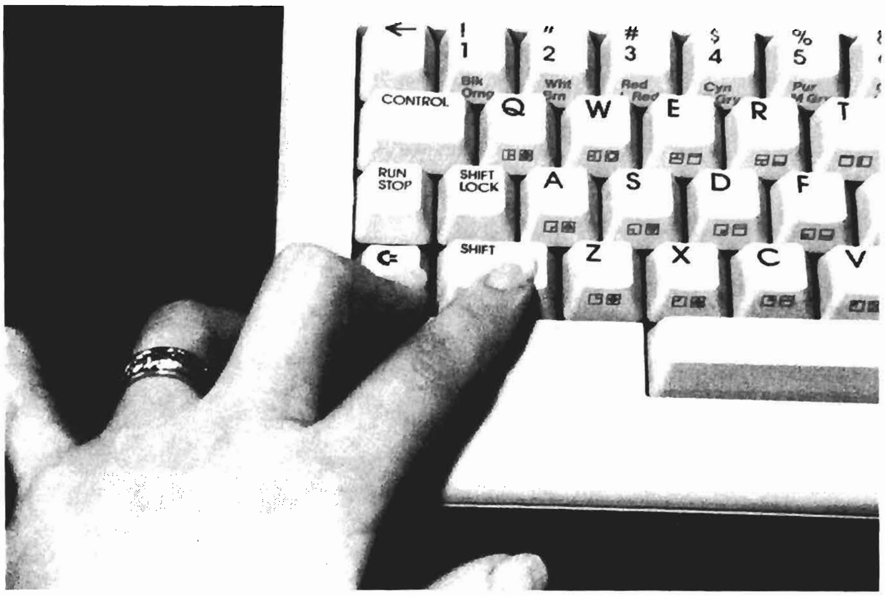


Fig. 2-11. Switching between the two character modes.

value is often used in BASIC programs to access a particular symbol or function that is not directly accessible from the keyboard. For example, on the Commodore 64, the BASIC command PRINT CHR$(83) displays a heart symbol.

The Commodore CHR$ code values are similar, but not identical, to the ASCII (American Standard Code for Information Interchange) code. Many personal computers map their keyboard characters and other special functions to ASCII values. The Commodore's CHR$ code is a variation of ASCII.

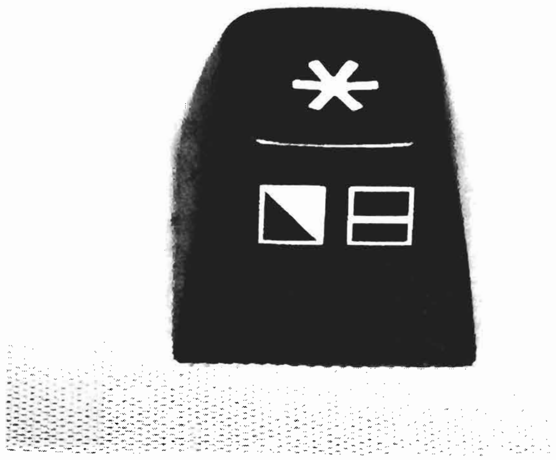


Fig. 2-12. Graphic characters.

When running the Keyboard Diagnostic Module, press a key to test whether it's working properly. This diagnostic should read and display, or echo, this key. The echo test lets you check whether or not a particular key, or even the entire keyboard, is working.

The Keyboard Diagnostic Module also displays the CHR$ value of the key. This lets you check whether the keys are sending the proper code to the computer to display the proper character or function. The Commodore User's Guide that is provided with your system includes a table that shows which key and/or function each CHR$ code value is meant to provide.

To run the Keyboard Diagnostic Module, follow the instructions provided in "Running the Diagnostic Program" in Chapter 1. Then press K or k to initiate the Keyboard Diagnostic Module.

Following the instructions given, press any key you wish to test. You can test any key on the keyboard: alphanumeric keys, cursor keys, function keys, special CTRL or C= keys. The key you press is displayed along with its CHR$ code value and any applicable key combination (Fig. 2-13).

If a character other than the one you pressed is displayed, or if no CHR$ code value is displayed, then you have accomplished the first troubleshooting step: there is something wrong with your keyboard. Refer to "Troubleshooting and Repair Guidelines" earlier in this chapter to get the keyboard working properly again.

When you finish testing, press the CLR/HOME key, and the program should return to the System Diagnostic Main Menu. If this does not happen, the CLR/HOME key is failing. In that case, press the RUN/STOP key to end the program, then troubleshoot the CLR/HOME key.

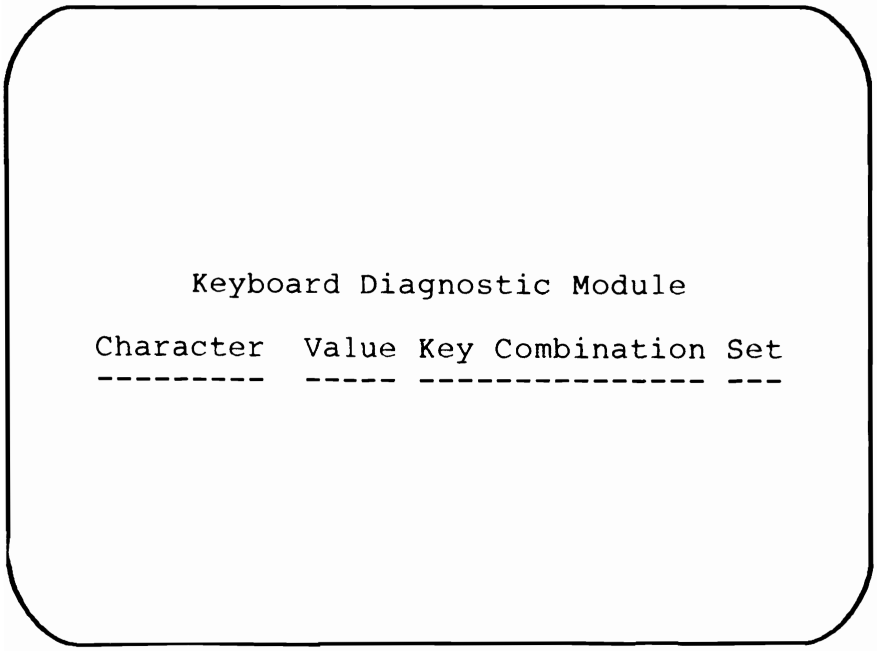

```
      Keyboard Diagnostic Module

Character  Value Key Combination Set
---------  ----- --------------- ---
```

Fig. 2-13. The Keyboard Diagnostic Module.

HOW THE KEYBOARD MODULE WORKS

When the user presses K or k from the main menu, the System Diagnostic Program branches to line 200, where the Keyboard Diagnostic Module begins. Figure 2-14 is a complete listing of the Keyboard Diagnostic Module.

Line 200 is a REMARK statement indicating the beginning of the Keyboard Diagnostic Module:

```
200 REM"Keyboard Diagnostic Module"
```

C128 USERS: The C128 code has an additional line, 205, which changes the values of the function keys so that they will actually display their keynames (F1, F2, F3, etc.) during this test, rather than displaying the code for their programmed functions. This replaces the function keys' current programming; however, their default designations are stored in system ROM. Turn the system off and on again to restore the function keys to their original programming after you have run this test.

Actual processing begins at line 210. The first command is a POKE statement, which places a value directly into a particular memory location. In this case, the program places a value of 23 into memory location 53272. This sets the character mode to set 2, the upper/lowercase mode.

```
210 POKE 53272,23:S1=1:GOSUB 630
```

The character set information indicating whether character set 1 or 2 is currently active is placed into variable S1. This instruction reflects the action of the POKE command, which caused the character set to be the upper/lowercase mode.

The next instruction is a GOSUB to line 630, which displays the headings on the monitor screen for the Keyboard Diagnostic Module.

C128 USERS: The 128 code uses a different POKE location and value in line 210.

Line 220 indicates that until the user enters a key to be placed in variable A$, variable KY$ remains null. Variable KY$ holds the display indications of any CTRL and C= key combinations, such as CTRL A, for later use in the test display.

```
220 A$="":KY$="":GOSUB 3000
```

The GOSUB instruction returns the program to the Get-a-Key subroutine at line 3000. As soon as the user presses a key to be tested, the Get-a-Key subroutine breaks its continuous loop, and the program continues on to line 230.

Line 230 sets variable N equal to the CHR$ value of A$, which is the variable holding the keystroke:

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

```
200 REM"Keyboard Diagnostic Module"
210 POKE 53272,23:S1=1:GOSUB 630
220 A$="":KY$="":GOSUB 3000
230 N=ASC(A$)
260 IF ASC(A$)=19 THEN GOTO 740
270 IF ASC(A$)=5 THEN KY$="ctrl 2:GOTO 600
280 IF ASC(A$)=17 THEN A$=" ":KY$="crsr down":GOTO 600
290 IF ASC(A$)=18 THEN KY$="rvs on":GOTO 600
300 IF ASC(A$)=20 THEN A$=" ":KY$="inst del":GOTO 600
310 IF ASC(A$)=28 THEN KY$="ctrl 3":GOTO 600
320 IF ASC(A$)=29 THEN A$=" ":KY$="cursor right":GOTO 600
330 IF ASC(A$)=30 THEN KY$="ctrl 6":GOTO 600
340 IF ASC(A$)=31 THEN KY$="ctrl 7":GOTO 600
350 IF ASC(A$)=141 THEN KY$="shift return":GOTO 600
360 IF ASC(A$)=142 THEN KY$="upper case":GOTO 600
370 IF ASC(A$)=144 THEN KY$="ctrl 1":GOTO 600
380 IF ASC(A$)=145 THEN KY$=" ":KY$="cursor up":GOTO 600
390 IF ASC(A$)=146 THEN KY$="rvs off":GOTO 600
400 IF ASC(A$)=147 THEN GOTO 740
410 IF ASC(A$)=148 THEN A$=" ":KY$="inst del":GOTO 600
420 IF ASC(A$)=156 THEN KY$="ctrl 5":GOTO 600
430 IF ASC(A$)=157 THEN A$=" ":KY$="cursor left":GOTO 600
440 IF ASC(A$)=158 THEN KY$="cntrl 8":GOTO 600
450 IF ASC(A$)=159 THEN KY$="ctrl 4":GOTO 600
460 IF ASC(A$)=133 THEN KY$="f1":GOTO 600
470 IF ASC(A$)=134 THEN KY$="f3":GOTO 600
480 IF ASC(A$)=135 THEN KY$="f5":GOTO 600
490 IF ASC(A$)=136 THEN KY$="f7":GOTO 600
500 IF ASC(A$)=137 THEN KY$="f2":GOTO 600
510 IF ASC(A$)=138 THEN KY$="f4":GOTO 600
520 IF ASC(A$)=139 THEN KY$="f6":GOTO 600
530 IF ASC(A$)=140 THEN KY$="f8":GOTO 600
560 IF S=0 OR S=1 THEN 600
570 IF S=2 THEN KY$="C= ":GOTO 595
580 IF S<>4 THEN GOTO 710
590 KY$="CTRL "
595 KY$=KY$+K$(K)
600 GOSUB 630
610 PRINT TAB(5) A$;TAB(13) N;TAB(23) KY$;TAB(35) S1
620 GOTO 220
630 GOSUB 2990
640 V=PEEK(53272)
650 IF V=21 THEN S1=1:GOTO 680
660 S1=2:PRINT TAB(7) "Keyboard Diagnostic Module":PRINT
```

```
670 PRINT TAB(2) "Character  Value Key Combination Set":GOTO 700
680 PRINT TAB(7) "keyboard diagnostic module":PRINT
690 PRINT TAB(2) "character  value key combination set"
700 PRINT TAB(2)"---------  ----- -------------- ---":RETURN
710 GOSUB 630
720 PRINT "An illegal shift code was encountered!"
730 PRINT "your keyboard needs service": GOTO 220
740 POKE 53272,23:PRINT "{BLUE}":GOTO 50
```

Fig. 2-14. The Keyboard Diagnostic Module listing.

This is done because some keystrokes cause some type of cursor movement. Ordinarily, the program prints the value held in A$ to display the graphic character. However, if it's a special character that causes a cursor movement, the program changes A$ to a blank so that the test display is not disrupted. Still, the CHR$ value of the key must be indicated, even though there is no graphic indication of the keystroke itself. So the program sets N to the CHR$ value of A$ before processing.

After this, line 260 checks to see if the CHR$ value of A$ is equal to 19. If it is, this indicates that the user has pressed the CLR/HOME key, which causes the program to go to line 740 to begin an exit subroutine:

```
260 IF ASC(A$)=19 THEN GOTO 740
```

The program performs some "housekeeping" routines before exiting back out to the System Diagnostic Main Menu, and these routines are found at line 740.

If the value of A$ is not equal to 19, the program falls through to line 270. Lines 270 through 530 check for exception keys, such as cursor movement keys, the RETURN key, color modes, and keys for which there is no actual graphic display character:

```
270 IF ASC(A$)=5 THEN KY$="ctrl 2:GOTO 600
280 IF ASC(A$)=17 THEN A$=" ":KY$="crsr down":GOTO 600
290 IF ASC(A$)=18 THEN KY$="rvs on":GOTO 600
300 IF ASC(A$)=20 THEN A$=" ":KY$="inst del":GOTO 600
310 IF ASC(A$)=28 THEN KY$="ctrl 3":GOTO 600
320 IF ASC(A$)=29 THEN A$=" ":KY$="cursor right":GOTO 600
330 IF ASC(A$)=30 THEN KY$="ctrl 6":GOTO 600
340 IF ASC(A$)=31 THEN KY$="ctrl 7":GOTO 600
350 IF ASC(A$)=141 THEN KY$="shift return":GOTO 600
360 IF ASC(A$)=142 THEN KY$="upper case":GOTO 600
370 IF ASC(A$)=144 THEN KY$="ctrl 1":GOTO 600
380 IF ASC(A$)=145 THEN KY$=" ":KY$="cursor up":GOTO 600
390 IF ASC(A$)=146 THEN KY$="rvs off":GOTO 600
400 IF ASC(A$)=147 THEN GOTO 740
410 IF ASC(A$)=148 THEN A$=" ":KY$="inst del":GOTO 600
420 IF ASC(A$)=156 THEN KY$="ctrl 5":GOTO 600
```

```
430 IF ASC(A$)=157 THEN A$=" ":KY$="cursor left":GOTO 600
440 IF ASC(A$)=158 THEN KY$="cntrl 8":GOTO 600
450 IF ASC(A$)=159 THEN KY$="ctrl 4":GOTO 600
460 IF ASC(A$)=133 THEN KY$="f1":GOTO 600
470 IF ASC(A$)=134 THEN KY$="f3":GOTO 600
480 IF ASC(A$)=135 THEN KY$="f5":GOTO 600
490 IF ASC(A$)=136 THEN KY$="f7":GOTO 600
500 IF ASC(A$)=137 THEN KY$="f2":GOTO 600
510 IF ASC(A$)=138 THEN KY$="f4":GOTO 600
520 IF ASC(A$)=139 THEN KY$="f6":GOTO 600
530 IF ASC(A$)=140 THEN KY$="f8":GOTO 600
```

This exception check is done for these particular keys so that these functions are not actually displayed. To prevent any disruption to the screen displays effected by these keys, variable A$ is set equal to null for each of these special cases.

The color commands are not "trapped" in this manner other than to display their key combinations. If the user activates the color green while in this test, for example, the display changes to green. This lets you test the key combinations used to obtain the different colors. As part of the "housekeeping" routine starting at line 740, the program resets the screen color back to the default of light blue before returning to the Main Menu.

Taking line 270 as an example for these exception cases, you see the following:

```
270 IF ASC(A$)=5 THEN KY$="ctrl 2":GOTO 600
```

The value 5 returned from a character variable (CHR$) means that the user pressed CTRL 2, which sets one of the colors. The variable KY$ stores the key combination. The program branches to line 600 for the actual test processing, because now that the program has received a user keystroke, the keystroke can be processed.

As you can see, there are quite a number of these exception checks. This ensures that all keys and all combinations can be tested. Some key combinations produce a color, as in the example at line 270. Other key combinations produce graphic characters. An example of this is pressing SHIFT 1 to have the ! symbol displayed and tested. Other examples include the graphic characters returned when the C= key is pressed alone or in combination with the SHIFT key to get one of the graphic characters displayed on the right or left of the keycap legend. This test lets the user know which key combination was pressed to display the graphic character.

C128 USERS: Three additional lines are added in the C128 code for this section: lines 272, 273, and 315. These lines provide exception checks for the TAB, LINE FEED, and ESC keys, respectively, which are not available on the C64.

If the user does not press any of these exception case keys, the program falls through to line 560.

Line 560 is an IF statement which sends the program to line 600 if variable

S contains a zero or a one, indicating a no-SHIFT or SHIFT condition, respectively. At this point, these character conditions are considered "normal"; they require no special handling.

```
560  IF S=0 OR S=1 THEN 600
```

However, if variable S were equal to two or four, a special character combination involving the C= or CTRL key would be indicated. In that case, the program would fall through to line 570.

Line 570 is another IF statement, indicating that if variable S is a two, then variable KY$ is set equal to C=, to represent the C= key in the display:

```
570  IF S=2 THEN KY$="C=  ":GOTO 595
```

The program then branches down to line 595.

If S is not equal to two, the program falls through to line 580:

```
580  IF S< >4 THEN GOTO 710
```

If variable S is not equal to four, the program falls through to line 710, which is an error routine signaling that an improper SHIFT condition, and possibly a keyboard problem, exists.

C128 USERS: Line 580 is modified in the C128 code to handle the ALT key available from the C128 keyboard, which offsets the possibility of such an improper SHIFT condition.

If variable S is equal to four, the program falls through to line 590:

```
590  KY$="CTRL "
```

Line 590 sets variable KY$ to CTRL for the display of the key combination. The program then falls through to line 595.

Line 595 goes to the keyboard map and retrieves the other character that was pressed in combination with the C= or CTRL key. This key is held in variable K$(K), the character value index that was retrieved from memory in line 550. The (K) holds the position of the major keycap from K$:

```
595  KY$=KY$+K$(K)
```

At this point, the program finally has all the information necessary to actually display the test results of the key(s) pressed, and can proceed to line 600.

Line 600 is a GOSUB to line 630. Line 630 starts the routine which prints the Keyboard Diagnostic Module headings, such as "Character," "Value," and so forth:

```
600  GOSUB 630
```

After the headings are displayed, the program returns to line 610. This is the line that actually prints the test results of the pressed key(s):

```
610  PRINT TAB(5) A$;TAB(13) N;TAB(23) KY$;TAB(35) S1
```

It tabs the display over five columns, then prints the contents of variable A$, which is the actual graphic character. Of course, if the pressed key(s) were an exception of some type, nothing would display in this field, because the exceptions were set to blanks to prevent the display from being adversely affected by the function.

Then the display tabs to position 13, and the contents of variable N is printed. At the start of this module, the CHR$ value of A$ was set to N. This was done for the exception keys which were converted to blank letters. If the actual CHR$(A$) were printed here, you would probably see just a blank, or a value of 32, which would not be useful information. So N holds the actual CHR$ code value of the key(s) that is being tested.

Next, the line tabs to position 23, and the key combination is printed, for example, CTRL 4. Finally, the line tabs to position 35, and either S1 or S2 is printed, to tell you which graphic character mode the system is currently employing, set 1 or set 2.

Line 620 sends the program back to line 220, the location where the program looks for another keystroke to test:

```
620  GOTO 220
```

This completes the process of analyzing the keystroke.

Line 630 begins the subroutine that displays the module's test title and headings. The first order of business for this subroutine is to clear the screen and place the cursor in the upper left corner HOME position.

```
630  GOSUB 2990
```

This is done with the Clear Screen subroutine at line 2990.

Line 640 is another PEEK instruction to read a value stored in memory. In this case, the program reads from memory location 53272, and places the value into variable V:

```
640  V=PEEK(53272)
```

Location 53272 holds the data indicating which graphic character set is currently active—the uppercase-only mode (set 1), or the upper/lowercase mode (set 2).

C128 USERS: Line 640 in the C128 code is modified for a different PEEK location of the current graphic character set.

Line 650 then has a related IF statement: if variable V contains 21, then the variable S1, which holds the graphic character set information, is set 1, the uppercase mode.

```
650  IF V=21 THEN S1=1:GOTO 680
```

If this statement is true, the program branches down to line 680.

C128 USERS: The C128 code is modified at line 650 to handle the different value of V for the character set mode.

If the statement in line 650 is not true, the program falls through to line 660, which assumes that if variable V is not a 21, then it must be a 23. The value 23 indicates that graphic character set 2 is active for the upper/lowercase mode.

```
660  S1=2:PRINT TAB(7) "Keyboard Diagnostic Module":PRINT
```

Line 660 first sets variable S1 to two for the upper/lowercase mode. Then it prints the title of the Keyboard Diagnostic Module followed by a blank line.

Line 670 goes on to print the test headings:

```
670  PRINT TAB(2) "Character Value Key Combination Set":
     GOTO 700
```

Then the program falls down to line 700.

If character set 1 had been in effect, the program would have branched from line 650 to line 680. Line 680 prints the same messages that lines 660 and 670 did, except in all uppercase letters to reflect the proper character mode:

```
680  PRINT TAB(7) "keyboard diagnostic module":PRINT
690  PRINT TAB(2) "character value key combination set"
```

This lets you change between the two character sets, and the headings reflect which set is active.

Line 700 then prints a row of dashes beneath the test headings. This row of dashes is valid for either graphic character set:

```
700  PRINT TAB(2)"————- ——- —————————- —-":
     RETURN
```

Line 700 then returns the program to the point in the code from which it came before it branched to this subroutine.

Recall that line 580 checks the key combination's SHIFT mode. If a value of zero, one, two, or four is not found there, the program branches down to line 710:

```
710  GOSUB 630
```

Line 710 is a GOSUB to line 630, which displays the diagnostic headings.

Lines 720 and 730 then print a message and return the program to line 220 to wait for another key to test:

```
720 PRINT "An illegal shift code was encountered!"
730 PRINT "your keyboard needs service": GOTO 220
```

This message probably indicates a problem with the keyboard. Refer to "Troubleshooting and Repair Guidelines" earlier in this chapter to help pinpoint the problem.

Line 740 begins the subroutine to exit the Keyboard Diagnostic Module and return to the System Diagnostic Main Menu. Line 740 is a POKE statement, poking a value of 23 into memory location 53272:

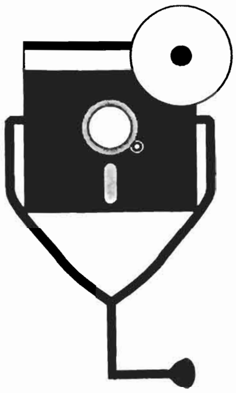
```
740 POKE 53272,23:PRINT "{BLUE}":GOTO 50
```

This resets the system to graphic set 2, for the upper/lowercase character mode. This ensures that when the program does return to the Main Menu, all the menu choices will be displayed in the proper type style.

That's the first part of the exit "housekeeping" routine. The second part prints CHR$(154) to return the screen color to the default of light blue, in case the user has been testing the different color commands as part of the keyboard test.

Finally, the GOTO statement returns the program to line 50, which is the beginning of the System Diagnostic Main Menu.

C128 USERS: Line 740 is modified in the C128 code for a different POKE location and value.



Chapter 3
The Monitor

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

DESCRIPTION AND FUNCTION OF THE MONITOR

The monitor is also known as a *cathode ray tube* (CRT) or *video display*. It is the screen on which you see the status of the program you're using.

All monitors used with the Commodore can display the two Commodore graphic character sets. Each set consists of 256 characters, and the sets are used one at a time. The television-set monitor accommodates 40 characters on each of 25 lines at a time (40 × 25 format), for 1000 on-screen character locations. All other monitors used with the Commodore accommodate 80 characters per line, with 25 lines per screen (80 × 25 format), for a total of 2000 on-screen character locations (Fig. 3-1).

Types of Monitors

There are several types of monitors that can work with your Commodore system. These include the following:

☐ television set monitors (color or black-and-white)

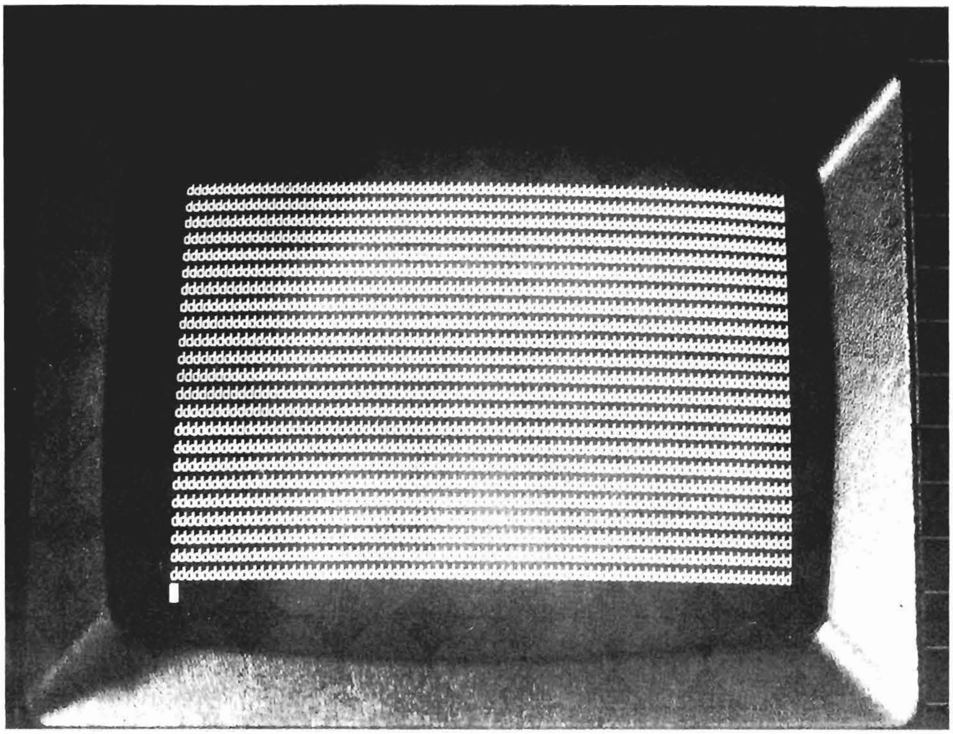


Fig. 3-1. The 80-column by 25-line screen format.

- ☐ monochrome monitors
- ☐ high-resolution monochrome graphics monitors
- ☐ high-resolution color graphics monitors

The monitor you obtain to work with your Commodore must be able to conform to the Commodore's RGB (red-green-blue) cable plug. Most computer monitors available can accommodate this type of cable.

The Television Set Monitor. The Commodore 64 and 128, like many other home computers, are designed to use a television set as the monitor. An external radio frequency (RF) converter is attached to the television via the television's VHF antenna leads (Fig. 3-2). If your television is cable-ready, you can use a coaxial cable connector instead of the antenna leads.

The Commodore cable attaches from the back of the Commodore to the RF converter. The RF converter includes a switch to let you choose between using the television as a computer monitor ("COMPUTER") and using the television to view broadcasts ("TV"). There is also a switch on the Commodore that allows selection of either Channel 3 or Channel 4 as the channel used for the computer mode.

The television-set monitor can display graphics and color, but only in the 40×25 text format (instead of the 80×25 text format on a standard computer monitor).

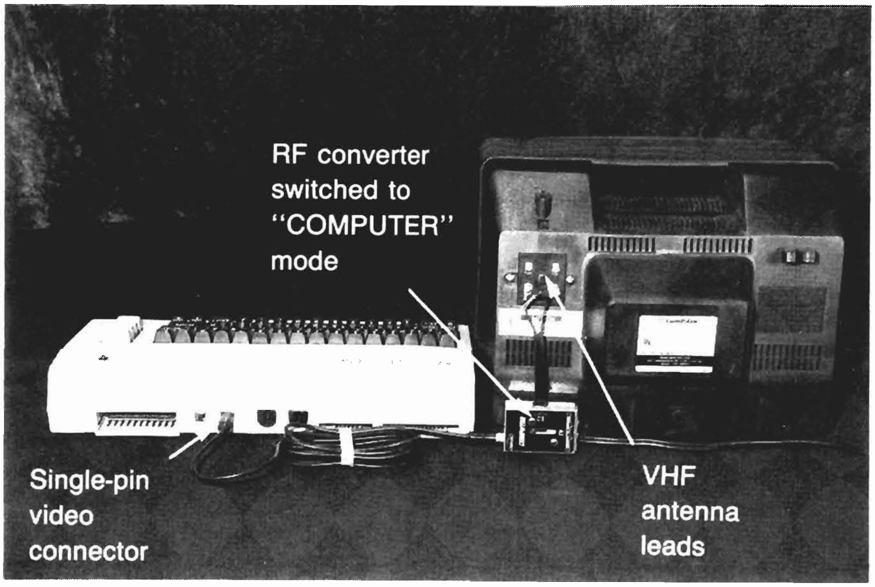

RF converter switched to "COMPUTER" mode
Single-pin video connector
VHF antenna leads

Fig. 3-2. RF converter attached to television.

The High-Resolution Graphics Monitor. The Commodore high-resolution graphics monitor illuminates every speck of light, or *pixel* picture element, in different combinations to form pictures. These pixels allow you to create and display computer drawings and graphics on the screen.

The greater the number of pixels, the better the screen resolution. The better the screen resolution, the sharper the displayed image. This is why the different types of high-resolution monitors are classified by the actual number of horizontal and vertical pixels they can accommodate on the screen at one time. The Commodore graphics monitors accommodate 320 × 200 pixels.

The Color Monitor. The color monitor is a high-resolution monitor that can display the pixels in different colors. A color monitor can be a vital asset when using your computer to do business graphics or design work.

If you are using a color television as the computer monitor, colors are already available to you. Another option is to buy a high-resolution color computer monitor that can be used with the special Commodore five-pin composite video port.

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 contact the back of the picture tube, which is coated with a material called phosphor. As the electrons strike the phosphor atoms, the atoms begin to glow. This reaction serves to form a small, clearly defined light spot (Fig. 3-3).

A number of these light spots are the dots that form the characters. On a mono-

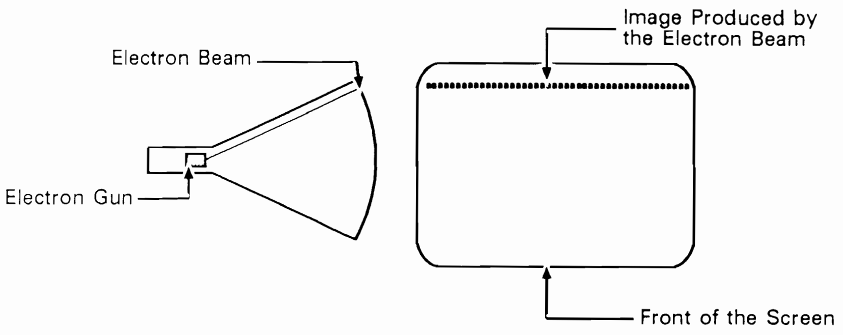


Fig. 3-3. Electron beam and display image.

chrome monitor, about seven of these light spots make up one character. On a high-resolution monitor, these light spots are the pixels which vary in density to form the characters, graphic elements, and drawings.

When a character on the keyboard is pressed to be displayed on the monitor, or when a particular application displays processing results, the program instructs the monitor to paint a certain pattern which forms the character representation. The specific pattern displayed is dependent upon the type of display matrix your monitor has.

When in the *text mode*, the monitor's display works on a character matrix. The display is divided into 25 horizontal rows. These rows are intersected by 40 to 80 vertical columns. The position at which a column and row intersect is one character location (Fig. 3-4).

The high-resolution, or color, display is a matrix as well, but a much more

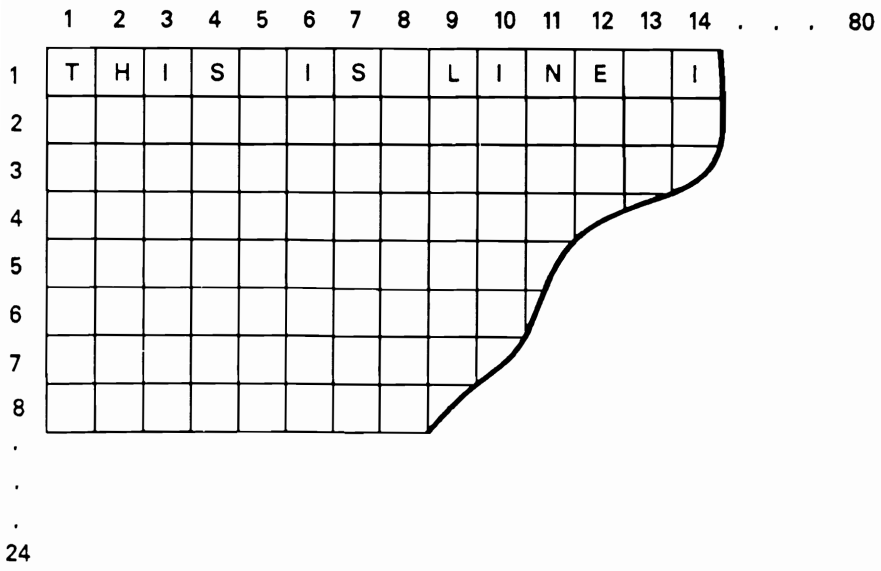


Fig. 3-4. Column/row matrix.

precise one, 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.

Character Generators. A *character generator* is a part of the system's read-only memory (ROM) which actually produces the series of pixel ON and OFF representations, or *bit map*. Character generators create the characters that are accessed either through the keyboard or through a program. The Commodore system employs two character generators, because there are two character sets available. These character generators are typically accessed whenever you are working within a text mode such as a word processing application.

The two character sets can only be used one at a time. You can move between the two character sets by pressing the SHIFT and C= key simultaneously. Character set 1 is the uppercase-only character mode. Character set 2 is the upper/lowercase character mode. Each of these character sets also have special graphic characters available. Refer to Chapter 2 for more information about accessing these graphic characters from the keyboard.

The Monitor Interface. The Commodore systems have two types of monitor interfaces available. The one you use depends on whether you're using a television or a computer monitor.

When using a television as your monitor, a round single-pin cable is used together with an RF converter to create the monitor interface. One end of this cable connects into the Commodore system unit, and the other end connects to the RF converter, which in turn connects to the VHF antenna terminals on the back of the television (see Fig. 3-2). The RF converter takes the computer's digital signals and converts them into the analog signals required by the television. In effect, the computer is functioning as a transmitting station to the television. When using a computer monitor instead of a television, the round five-pin RGB cable supplied with the monitor is the monitor interface.

Sprites. A very powerful feature of the Commodore systems is the ability to define special graphic shapes, called *sprites*. A sprite is a special display function in which a particular pattern or design can be described, and the image can be moved around the screen with very little programming. Up to eight of these sprites can be used at one time. These sprites are often used in games to represent quickly moving objects.

Special circuitry, referred to as *sprite registers*, is built into the Commodore system unit to control the sprites. A register is a specialized location within memory used for one particular function, distinct from the main system memory. Registers make it easy to move the sprites, as well as detect collisions, in which two or more sprites occupy one or more common pixel locations on the screen. Because special dedicated hardware handles these functions, the sprites move very quickly when viewed on the screen. This facilitates high-quality Commodore games, even when simply programmed with BASIC.

TROUBLESHOOTING AND REPAIR GUIDELINES

This section describes problems that can occur with the monitor, how to troubleshoot them, and how to 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, entitled "Running the Diagnostic Module."

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

- ☐ Check the brightness and contrast knobs, and make sure they are turned at least two-thirds of the way up. Someone else might have turned the controls down to prevent the screen from etching (see the next section on "Etching").
- ☐ 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. Test this by plugging another electrical device into the socket to see if it works. If it does not, an electrical supply problem is indicated. Check the fuse box or circuit breaker.
- ☐ Check the cabling to ensure that the monitor is properly attached to the display interface of the system unit.
- ☐ When the monitor is properly warmed up (it takes a minute or so), turn the brightness control knob up as far as it will go. A series of lines should be displayed across the screen (Fig. 3-5). If you don't see this, and the screen remains blank, then the monitor's power supply might be defective. Take the monitor in for servicing.
- ☐ If the monitor is making a crackling noise and the screen remains blank, this might indicate a problem with the display interface. If you run your hand next to the screen, the hairs on the back of your hand might become attracted to it.

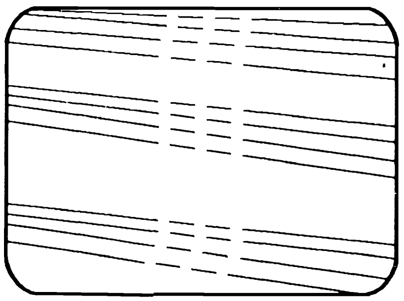


Fig. 3-5. High brightness lines.

First, try swapping a different monitor or television onto your system. If this monitor works, then your other monitor needs servicing. If the swapped monitor does not work either, a problem with the system unit itself is indicated. Refer to Chapter 2 for system unit troubleshooting procedures.

Problem: Static on the Television Screen

A problem with static is related to the previous problem of having nothing display on the screen. If you are working with a television monitor, and you have no picture but static, you could have a problem with the television set, the RF converter, or the computer system unit.

☑ Solutions

- ☐ First check the television. To do this, switch the RF converter to the television mode, or disconnect it altogether. Then change the television channel to a known working station. For example, if you know that you receive Channel 2 very well, turn it to Channel 2. If you still do not receive Channel 2 nor any other channel, there is a problem with the television's tuning or receiving mechanism, and it should be taken in for servicing.

- ☐ If there is a television picture, then you know that the television is working fine, and you should test the RF converter. The RF converter has two switch positions: COMPUTER and TV. Make sure that the converter is switched to the COMPUTER position. Check that the converter is properly connected to the correct antenna leads: VHF rather than UHF.

  Next, make sure the channel selector is set properly for the computer. The Commodore is set up so that it can operate from either Channel 3 or Channel 4. Select which channel the computer will use with the slide switch on the rear of the Commodore system unit/keyboard. The channel selected on the computer should be the channel the television is on.

  If the screen still shows just static, replace the RF converter. You can obtain one for approximately $5 at a stereo or electronics store.

- ☐ If you try all the previous solutions with the television set and the RF converter, but there is still static on the television screen, the computer system unit is probably having a problem. Refer to Chapter 2, "The System Unit/Keyboard" to troubleshoot system unit problems.

Problem: Etching

Etching, or *ghosting,* takes place on the screen when one particular format, such as the text editor menus or database screen, is displayed on the screen for long periods of time without changing. If the computer is left running with the same image displayed for many hours a day, or many days a week, the constant bombardment of the electrons on the same area of the picture tube phosphor can permanently burn, or etch, that image into the screen. When that happens, even when something else is displayed, you will still see a ghost image of the etched display. At the same time, the display you want to see will be fainter, more "washed out."

There isn't anything you can do to make an image that is etched into your display go away. 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.

Problem: Blank Character Location

Depending on the type of monitor you use, characters are displayed either in a character or a pixel matrix. A problem can occur in which one of these character locations in the matrix does not display anything; a character is entered to be placed at a certain location, but the character does not display.

The high-resolution 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. The problem might be that pixels are not displaying. Certain characters might be missing, which happens when the computer has lost a particular 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.

☑ Solution

- ☐ To troubleshoot this problem, run the Display Test of the Monitor Diagnostic Module. Press a character on the keyboard, and the entire screen should be filled with this character at every available character location. If you see that a particular character is missing, there is 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, and chips need to be replaced. Take the CPU board in for servicing.

Problem: Misaligned Display

Even if you are using a computer monitor instead of a television for your visual display, you can still have the same types of problems that can occur with a television. For example, the picture can become skewed, or it can appear compressed. It can also go out of alignment (Fig. 3-6).

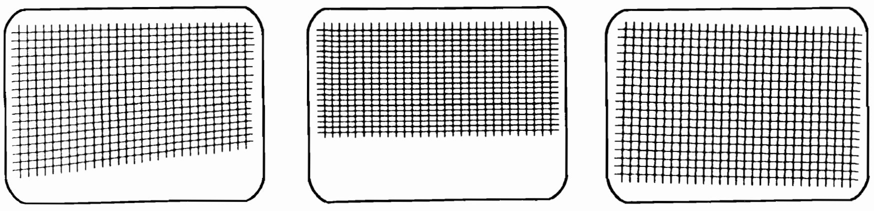


Fig. 3-6. Misaligned displays.

In many cases, it is more difficult to perceive such alignment problems when there is nothing but text on the screen. To check for proper alignment, try the following solutions.

☑ 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 the tuning wand (**do not** use a metal screwdriver!) to gently turn these potentiometers (''pots'') either clockwise or counterclockwise to bring the screen back into alignment. Note the direction you're turning the pot, and make only one-quarter of a turn at a time, so you can easily return the pot to its original position if necessary.
- ☐ If adjusting the 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, simply adjust the horizontal and vertical hold controls until the alignment grid looks straight.

Problem: A Particular Character Does Not Display

If you press a character key on the keyboard, but do not see it displayed on the monitor, there is probably a problem with the character generator ROM.

☑ Solution

- ☐ If you run the Display Test and do not see all the expected characters, or if the display is incorrect or garbled, a problem with the system unit itself is indicated. Refer to Chapter 2, ''The System Board/Keyboard'' to troubleshoot the system unit.

Problem: Colors Do Not Display Correctly

When using a color monitor (either the color television or a high-resolution color monitor), another problem you could encounter is that no colors display at all, or that the colors display the incorrect tones. Check the colors with the Color Test of the Monitor Diagnostic Module. The screen displays the names of the colors, then displays bars of color in the appropriate shades. If the color names and the actual colors do not match, there is a monitor color problem.

☑ Solutions

- ☐ First make sure that the monitor is actually a color television or a color monitor instead of a black-and-white television or monochrome monitor.
- ☐ If the monitor is displaying colors, but they're not the correct colors, there are two different adjustments you can make to rectify this problem.
  If you're using a television monitor, first check a normal television picture,

and see if that color is adjusted properly. If the picture doesn't look right, for instance, if faces look too red, or green, then you simply need to adjust the color and tint controls on the television set.

Once the television broadcast picture has displayed the correct colors, switch the television back to the computer mode and check the color. It's possible that the color of a television broadcast might look fine, but when working in the computer mode, the colors are wrong and/or fuzzy. When working with a computer monitor, you can simply adjust the color and tint controls of the monitor, just as you would with a television.

☐ After making the proper adjustments, if the colors are still not correct, you'll need to adjust a variable resistor inside the computer's system unit on the motherboard. To do this, first remove the cover to the system unit, following these steps:

1. Turn the system off and disconnect all peripheral cables.
2. Turn the system unit upside-down, resting the keyboard on foam or paper to protect it.
3. Remove the three phillips-head screws (Fig. 3-7).
4. While holding the case together, place the system unit keyboard-side up again.
5. Grasp the case on the bottom edge (by the SPACE BAR) and lift. The case is hinged and will fold back. Do this slowly and carefully to avoid jerking the keyboard cable from its connector.
6. Carefully bend back the silver cardboard cover which shields the motherboard (Fig. 3-8).

Now adjust the color trim. To do this, follow these steps:

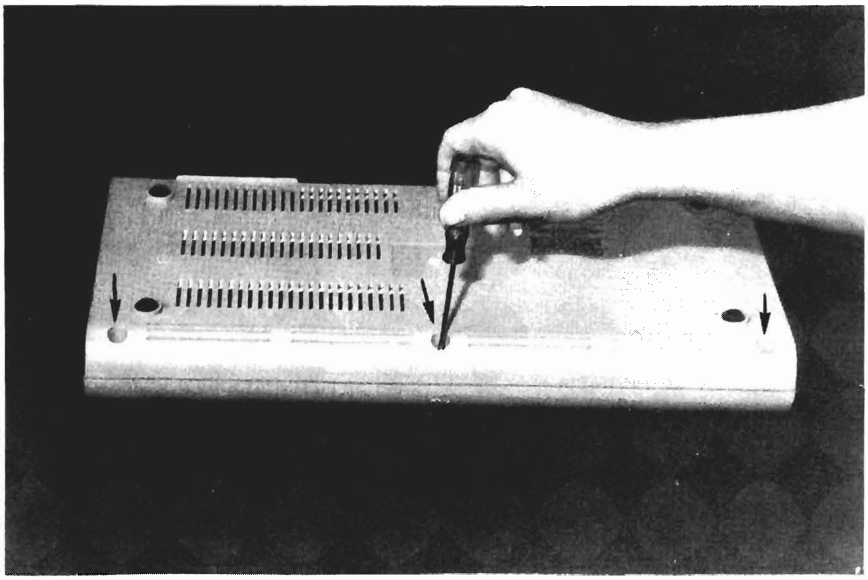


Fig. 3-7. Removing the system unit screws.

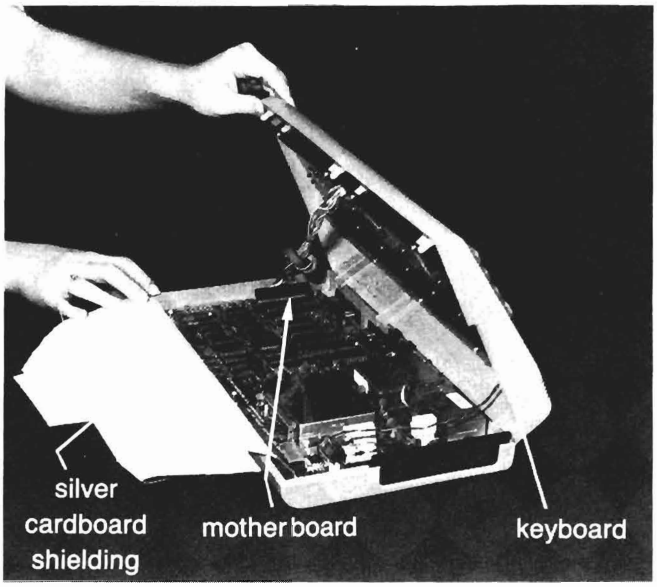


Fig. 3-8. Opening the system unit.

1. Reconnect the power supply, monitor, and disk or cassette drive.
2. Turn the system power on. While the system is open and powered-on, do not touch anything, and when making adjustments, use **only** a nonconductive (plastic) tuning wand. **Do not** use a metal screwdriver.
3. Load and run the System Diagnostic Program. Choose the Monitor Diagnostic, and then the Color Test within that diagnostic (refer to "Running the Monitor Diagnostic Module" later in this chapter for further instructions).
4. Prop up the upper part of the system unit case in order to expose the motherboard.
5. Locate the color trim variable resistor on the motherboard (Fig. 3-9).
6. Use your tuning wand (**do not** use a metal screwdriver) to adjust this resistor until you see satisfactory colors on the screen display.

Finally, close up the computer. To do this, follow these steps:

1. When the colors look correct, turn the system off again and disconnect the peripheral cables.
2. Replace the silver cardboard shield in its place over the motherboard.
3. Bring down the top cover of the system unit.
4. Holding the case together, turn the system unit upside-down. Again, protect the keyboard with a foam pad.
5. Replace the three phillips screws.
6. Turn the system unit keyboard-side up again, and connect all the peripheral and power supply cables.
7. Power up the system and run one or two passes of the Exerciser program (see Chapter 8, "The Exerciser") to ensure that the system is functioning properly.

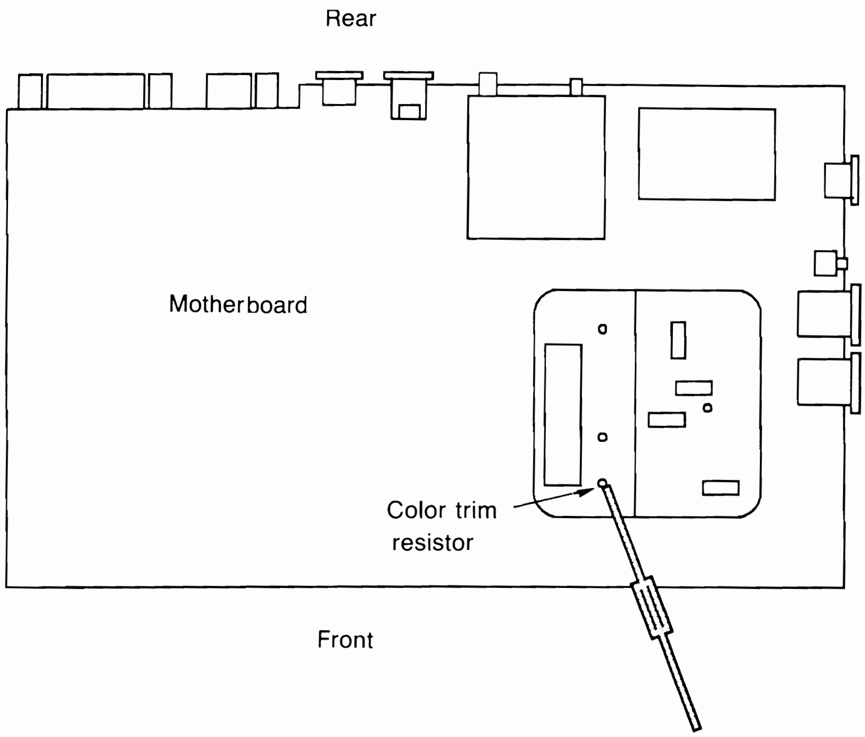


Fig. 3-9. Adjusting the color trim resistor.

Adjusting the color from inside the system unit does not affect the color and tint settings of the television or the monitor whatsoever.

Problem: Monitor Does Not Scroll Properly

When the screen is completely filled with text and you enter more characters on the next line, the entire screen moves itself up, pushing the top line off the screen, making 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 the scrolling, run the Scroll Test. Enter enough repetitions, about 30, 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 section of the system unit. Refer to Chapter 2 and troubleshoot the system unit.

Problem: Sprites Do Not Display Correctly

There are eight sprites available for use primarily as quickly moving objects in games. A problem that can occur is that one of these eight sprites might not be working properly. There is a sprite register for each one of the sprites in the display section of memory. A problem with the sprite memory register could cause a problem with the sprite display.

You might suspect a sprite problem when running a game, for example, if one of the objects do not move or look quite right. Confirm the problem by running the Sprite Test of the Monitor Diagnostic Module.

☑ Solution

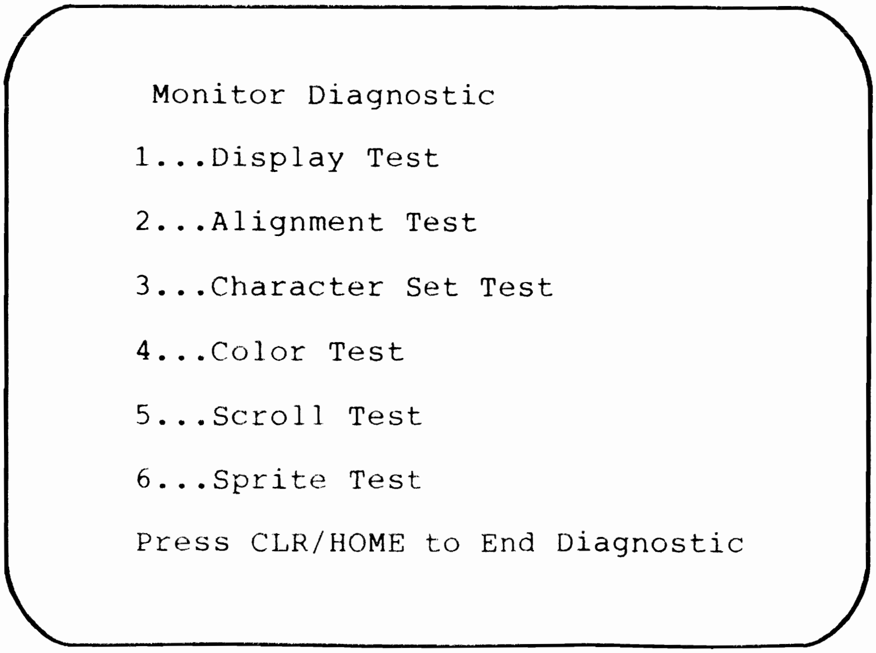
- ☐ If the sprite does not move, or if it is the wrong color (it should be white) or the wrong shape (it should be a square), there is a problem with the sprite circuitry in the system unit. If this is the case, refer to Chapter 2 and troubleshoot the system unit.

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. 3-10). To activate one of these tests, press the corresponding number. To end the Monitor Diagnostic Module, press the ESC key. The Diagnostic Main Menu will be displayed again.

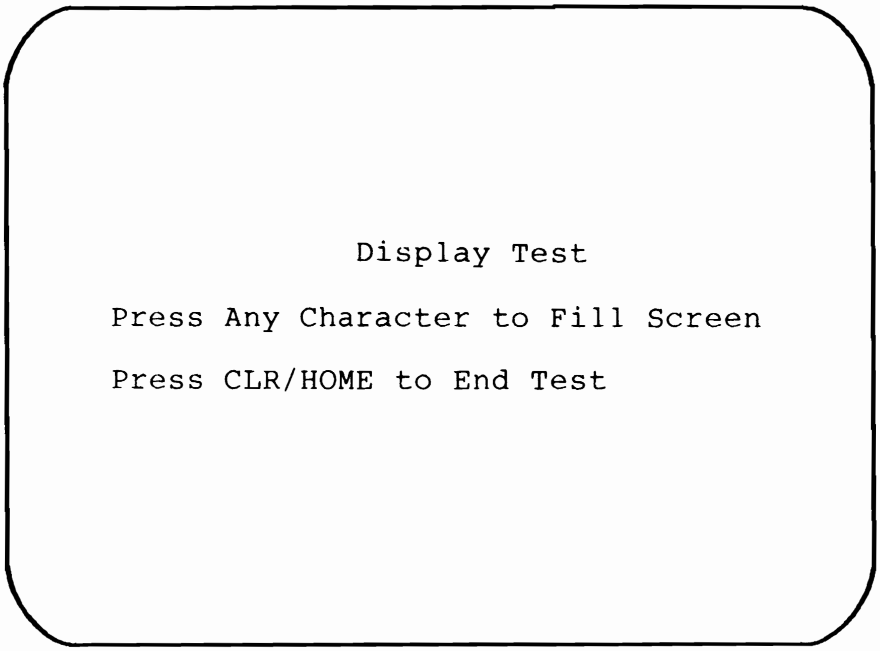
Display Test

To run the Display Test, press 1. The screen clears and then gives the test instruction (Fig. 3-11). Press a character on the keyboard. (Function keys and the



```
  Monitor Diagnostic

 1...Display Test

 2...Alignment Test

 3...Character Set Test

 4...Color Test

 5...Scroll Test

 6...Sprite Test

 Press CLR/HOME to End Diagnostic
```

Fig. 3-10. The Monitor Diagnostic Menu.



```
                 Display Test

     Press Any Character to Fill Screen

     Press CLR/HOME to End Test
```

Fig. 3-11. Display Test instructions.

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.

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 CLR/HOME to return to the Monitor Diagnostic Menu.

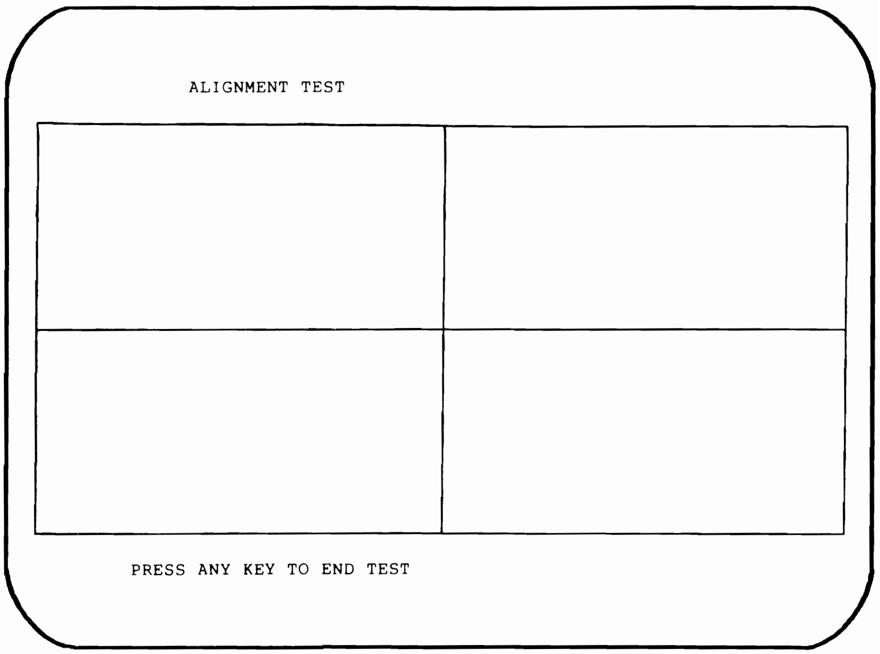
Both character sets, set 1 and set 2, can be used for this test. It's a good idea to run the Display Test twice, once for character set 1, and again for set 2.

The Display Test can also be used to validate a keyboard problem. Because the Keyboard Diagnostic relies on user input to display a character, a faulty key might not produce a particular character. If the Keyboard Diagnostic indicates a problem, check it again with the Display Test. The Display Test generates the characters by using a program, so if there is a character in this test that you did not see in the Keyboard Test, this confirms a keyboard problem. Refer to Chapter 2 to troubleshoot the keyboard.

Alignment Test

Pressing 2 from the Monitor Menu presents the Alignment Test, which displays the alignment box with horizontal and vertical lines resembling Fig. 3-12.

The alignment grid helps you look 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.



ALIGNMENT TEST

PRESS ANY KEY TO END TEST

Fig. 3-12. Alignment Test results.

Character Set Test

Press 3 to invoke the Character Set Test. This test checks whether any characters are missing.

You are prompted to enter the character set to be tested, set 1 or set 2. When you make the choice, the Character Set Test places all 256 display characters of that set (Fig. 3-13) on the screen. Pressing any key ends this test and displays the Monitor Diagnostic Menu again.

Color Test

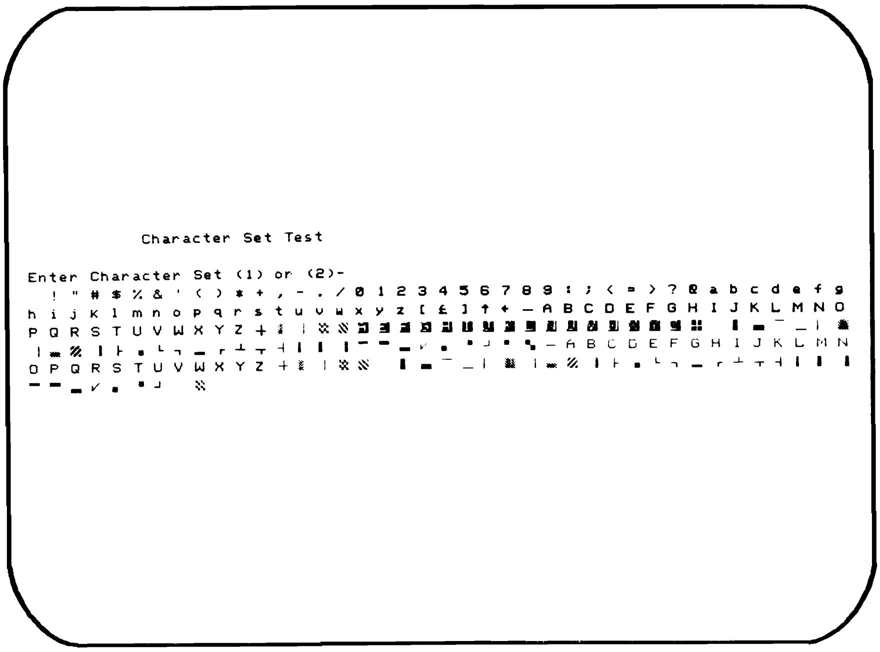
To test color on a color monitor, press 4 from the Monitor Diagnostic Menu. The screen displays the name of the color, then a bar of that color (see Fig. 3-14). All the colors available for your monitor are displayed in this manner.

This test is particularly useful not only to test the colors of the monitor, but also to adjust the colors if there is a problem with them. Leave this test displayed on the screen, and make the necessary adjustments, until all the colors look right. Refer to "Troubleshooting and Repair Guidelines" earlier in this chapter for the adjustment procedure.

When you press any key, the Color Test is cleared, and the Monitor Diagnostic Menu is displayed again.

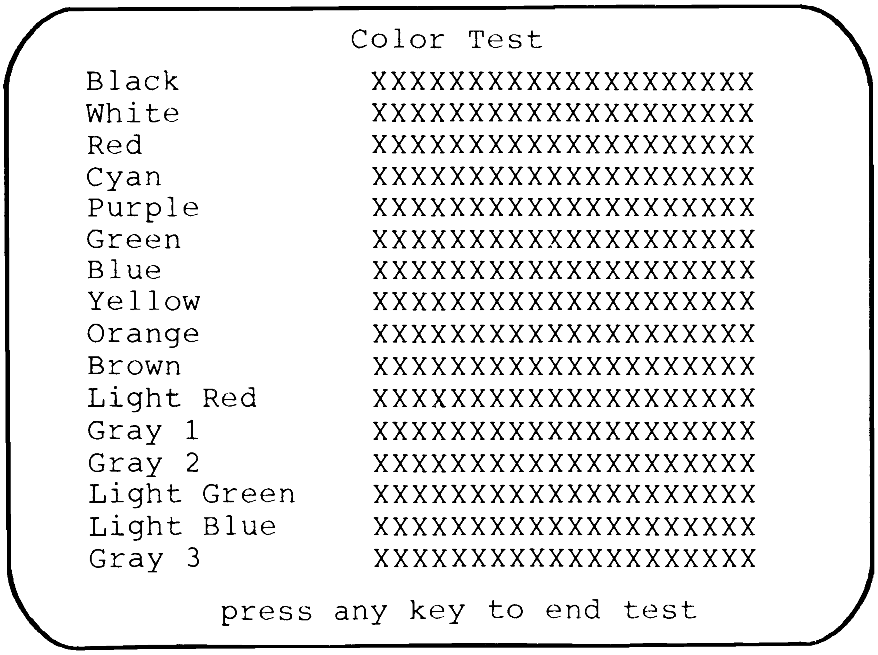
Scroll Test

To test scrolling and scroll rates, press 5 from the Monitor Diagnostic Menu. The Scroll Test prompts for the number of repetitions for the test. Enter a number



```
        Character Set Test

Enter Character Set (1) or (2)-
  ! " # $ % & ' ( ) * + , - . / 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 + ▒ │ ▒ ▧ █ █ █ █ █ █ █ █ █ █ █ █ █ █ █ █ ▚   ▌ ▬ ‾ _ │ ▓
│ ▃ ▨ ▎ ├ ▗ └ ┐ ▂ ┌ ┴ ┬ ┤ ▌ ▍ ▐ ▔ ▀ ▂ ✓ ▖   ▝ ┘ ▘ ▚ — 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. 3-13. Character Set Test results.



```
                 Color Test
   Black          XXXXXXXXXXXXXXXXXXXX
   White          XXXXXXXXXXXXXXXXXXXX
   Red            XXXXXXXXXXXXXXXXXXXX
   Cyan           XXXXXXXXXXXXXXXXXXXX
   Purple         XXXXXXXXXXXXXXXXXXXX
   Green          XXXXXXXXXXXXXXXXXXXX
   Blue           XXXXXXXXXXXXXXXXXXXX
   Yellow         XXXXXXXXXXXXXXXXXXXX
   Orange         XXXXXXXXXXXXXXXXXXXX
   Brown          XXXXXXXXXXXXXXXXXXXX
   Light Red      XXXXXXXXXXXXXXXXXXXX
   Gray 1         XXXXXXXXXXXXXXXXXXXX
   Gray 2         XXXXXXXXXXXXXXXXXXXX
   Light Green    XXXXXXXXXXXXXXXXXXXX
   Light Blue     XXXXXXXXXXXXXXXXXXXX
   Gray 3         XXXXXXXXXXXXXXXXXXXX

         press any key to end test
```

Fig. 3-14. Color Test results.

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 Fig. 3-15.

Sprite Test

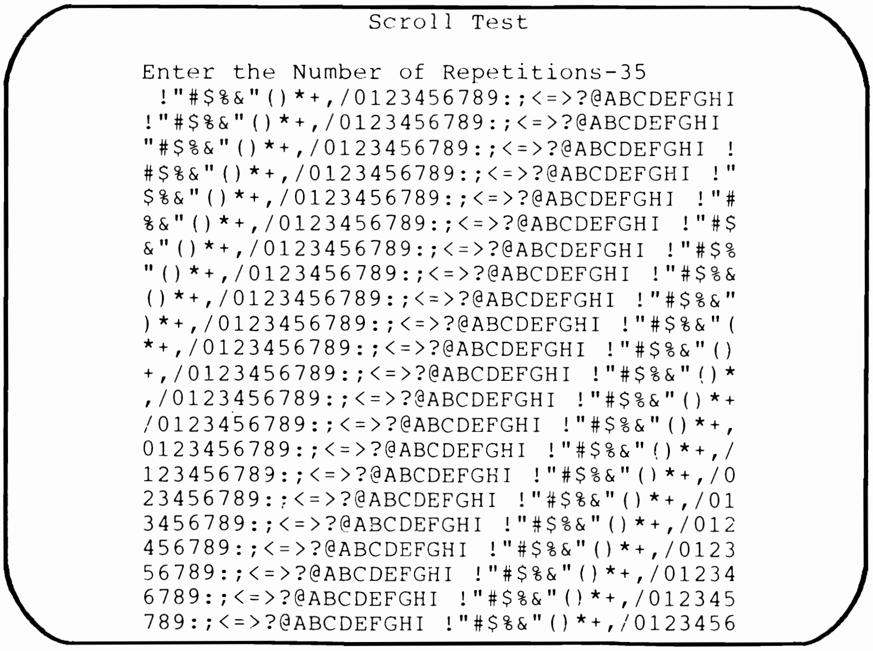
The Sprite Test is accessed by pressing 6 from the Monitor Diagnostic Menu. The first sprite appears at the left edge of the screen and moves to the right edge. Each of the eight sprites appear in succession, one at a time.

The sprites are represented by a white square (Fig. 3-16). If you do not see a sprite appear, if its movement is erratic, or if it is the wrong color or shape, there is a problem in the sprite-handling logic of your CPU. Refer to Chapter 2 and troubleshoot the system unit.

When monitor testing is complete, press CLR/HOME 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 800, which is the start of the Monitor Diagnostic Module (Fig. 3-17).



```
                    Scroll Test

Enter the Number of Repetitions-35
 !"#$%&"()*+,/0123456789:;<=>?@ABCDEFGHI
!"#$%&"()*+,/0123456789:;<=>?@ABCDEFGHI
"#$%&"()*+,/0123456789:;<=>?@ABCDEFGHI !
#$%&"()*+,/0123456789:;<=>?@ABCDEFGHI !"
$%&"()*+,/0123456789:;<=>?@ABCDEFGHI !"#
%&"()*+,/0123456789:;<=>?@ABCDEFGHI !"#$
&"()*+,/0123456789:;<=>?@ABCDEFGHI !"#$%
"()*+,/0123456789:;<=>?@ABCDEFGHI !"#$%&
()*+,/0123456789:;<=>?@ABCDEFGHI !"#$%&"
)*+,/0123456789:;<=>?@ABCDEFGHI !"#$%&"(
*+,/0123456789:;<=>?@ABCDEFGHI !"#$%&"()
+,/0123456789:;<=>?@ABCDEFGHI !"#$%&"()*
,/0123456789:;<=>?@ABCDEFGHI !"#$%&"()*+
/0123456789:;<=>?@ABCDEFGHI !"#$%&"()*+,
0123456789:;<=>?@ABCDEFGHI !"#$%&"()*+,/
123456789:;<=>?@ABCDEFGHI !"#$%&"()*+,/0
23456789:;<=>?@ABCDEFGHI !"#$%&"()*+,/01
3456789:;<=>?@ABCDEFGHI !"#$%&"()*+,/012
456789:;<=>?@ABCDEFGHI !"#$%&"()*+,/0123
56789:;<=>?@ABCDEFGHI !"#$%&"()*+,/01234
6789:;<=>?@ABCDEFGHI !"#$%&"()*+,/012345
789:;<=>?@ABCDEFGHI !"#$%&"()*+,/0123456
```

Fig. 3-15. Scroll Test results.

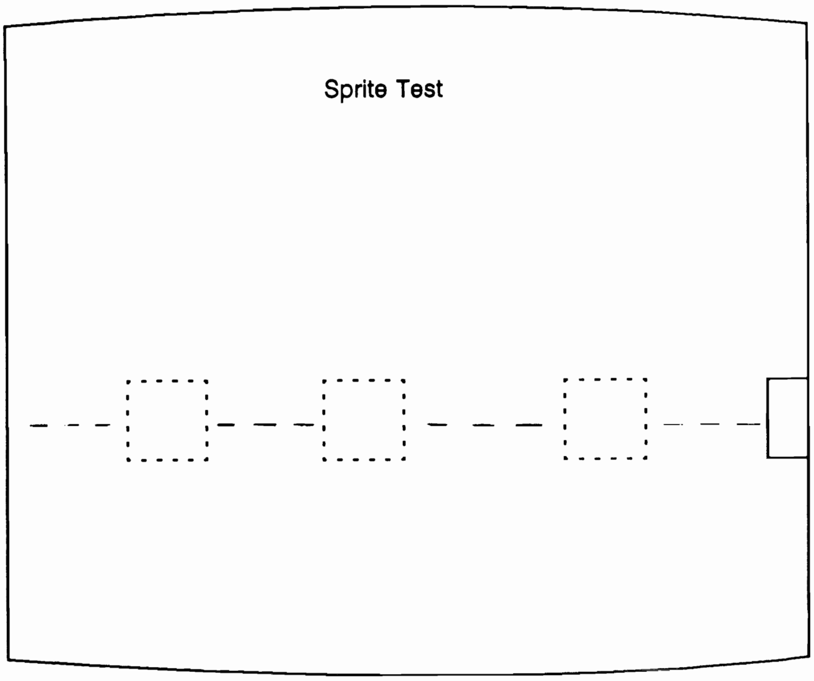


Fig. 3-16. Sprite Test results.

Line 800 goes first to the Clear Screen subroutine at line 2990. The REMARK statement indicates that this is the Monitor Diagnostic Module:

```
800  GOSUB 2990:REM "Monitor Diagnostic Module"
```

Six tests are included in the Monitor Diagnostic Module. Lines 810 through 880 print the test selection menu on the screen:

```
810  PRINT TAB(6) "Monitor Diagnostic":PRINT
820  PRINT TAB(5) "1 . . . Display Test":PRINT
830  PRINT TAB(5) "2 . . . Alignment Test":PRINT
840  PRINT TAB(5) "3 . . . Character Set Test":PRINT
850  PRINT TAB(5) "4 . . . Color Test":PRINT
860  PRINT TAB(5) "5 . . . Scroll Test":PRINT
870  PRINT TAB(5) "6 . . . Sprite Test":PRINT
```

Line 890 instructs the user to press the CLR/HOME key when they want to quit the Monitor Diagnostic and return to the System Diagnostic Main Menu:

```
890  PRINT TAB(5) "Press CLR/HOME to End Diagnostic"
```

Line 900 sends the program to the Get-a-Key subroutine at line 3000:

```
 800 GOSUB 2990:REM "Monitor Diagnostic Module"
 810 PRINT TAB(6) "Monitor Diagnostic":PRINT
 820 PRINT TAB(5) "1...Display Test":PRINT
 830 PRINT TAB(5) "2...Alignment Test":PRINT
 840 PRINT TAB(5) "3...Character Set Test":PRINT
 850 PRINT TAB(5) "4...Color Test":PRINT
 860 PRINT TAB(5) "5...Scroll Test":PRINT
 870 PRINT TAB(5) "6...Sprite Test":PRINT
 890 PRINT TAB(5) "Press CLR/HOME to End Diagnostic"
 900 GOSUB 3000
1000 IF ASC(A$)=19 THEN 50
1010 IF A$="1" THEN 1090
1020 IF A$="2" THEN 1180
1030 IF A$="3" THEN 1320
1040 IF A$="4" THEN 1440
1050 IF A$="5" THEN 1650
1070 IF A$="6" THEN 1770
1080 GOSUB 3100:GOTO 900
1090 GOSUB 2990:REM Display Character Set Test
1100 PRINT TAB(14) "Display Test":PRINT
1110 PRINT "Enter Any Character to Fill Screen":PRINT
1120 PRINT "Press CLR/HOME to End Test"
1130 GOSUB 3000
1132 PRINT "{CLR/HOME}"
1135 IF ASC(A$)=19 THEN 800
1140 FOR I=1 TO 999
1150 PRINT A$;:NEXT
1155 IF EX=1 THEN 1360
1160 GOTO 1130
1180 GOSUB 2990:REM Alignment Test
1185 POKE 53272,21
1190 PRINT TAB(13) "alignment test":
1200 PRINT CHR$(176);:FOR I=1 TO 18: PRINT CHR$(99);:NEXT I
1210 PRINT CHR$(178);:FOR I=1 TO 18: PRINT CHR$(99);:NEXT I
1220 PRINT CHR$(174):
1230 FOR I=1 TO 10: PRINT CHR$(98);TAB(19)CHR$(98);
     TAB(38)CHR$(98):NEXT I
1240 PRINT CHR$(171);:FOR I=1 TO 18 PRINT CHR$(99);:NEXT
1250 PRINT CHR$(123);:FOR I=1 TO 18 PRINT CHR$(99);:NEXT
1260 PRINT CHR$(179)
1270 FOR I=1 TO 10:PRINT CHR$(98);TAB(19) CHR$(98);TAB(38)
     CHR$(98):NEXT
1280 PRINT CHR$(173);:FOR I=1 TO 18:PRINT CHR$(99);:NEXT I
1290 PRINT CHR$(177);:FOR I=1 TO 18:PRINT CHR$(99);:NEXT
```

Fig. 3-17. The Monitor Diagnostic Module listing.

```
1300 PRINT CHR$(189):GOSUB 3200
1310 POKE 53272,23:GOTO 800
1320 GOSUB 2990:REM "Character Set Test"
1330 PRINT TAB(11) "Character Set Test"
1340 PRINT:PRINT "Enter Character Set (1) or (2)-"
1350 GOSUB 3000
1360 IF A$="1" THEN POKE 53272,21
1370 FOR I=32 TO 255
1380 IF I=128 OR I=130 OR I=131 OR I=132 OR I=141
     I=143 OR I=144 THEN 1410
1390 IF I=145 OR I=146 OR I=147 OR I=148 OR I=156
     OR I=157 OR I=158 THEN 1410
1395 IF I=159 THEN 1410
1400 PRINT CHR$(I);+" ";
1410 NEXT
1415 IF EX=1 THEN POKE 53272,23:GOTO 1440
1420 PRINT:GOSUB 3200
1430 POKE 53272,23:GOTO 800
1440 GOSUB 2990:REM "Color Test"
1450 PRINT TAB(15) "Color Test":PRINT
1460 "C$="XXXXXXXXXXXXXXXXXXXX"
1470 PRINT "{CTRL 1} Black";TAB(12) C$
1480 PRINT "{CTRL 2} White";TAB(12) C$
1490 PRINT "{CTRL 3} Red";TAB(12) C$
1500 PRINT "{CTRL 4} Cyan";TAB(12) C$
1510 PRINT "{CTRL 5} Purple";TAB(12) C$
1520 PRINT "{CTRL 6} Green";TAB(12) C$
1530 PRINT "{CTRL 7} Blue";TAB(12) C$
1540 PRINT "{CTRL 8} Yellow";TAB(12) C$
1550 PRINT "{C= 1} Orange";TAB(12) C$
1560 PRINT "{C= 2} Brown";TAB(12) C$
1570 PRINT "{C= 3} Lt. Red";TAB(12) C$
1580 PRINT "{C= 4} Gray 1";TAB(12) C$
1590 PRINT "{C= 5} Gray 2";TAB(12) C$
1600 PRINT "{C= 6} Lt. Green";TAB(12) C$
1610 PRINT "{C= 7} Lt. Blue";TAB(12) C$
1620 PRINT "{C= 8} Gray 3";TAB(12) C$
1625 IF EX=1 THEN PRINT "{C= 7}":GOTO 1680
1630 PRINT:GOSUB 3200
1640 PRINT "{C= 7}":GOTO 800
1650 GOSUB 2990:REM "Scroll Test"
1660 PRINT TAB(15) "Scroll Test":PRINT
1670 PRINT "Enter the Number of Repetitions-";A:PRINT
1680 N=31
1690 FOR I=1 TO A
```

Fig. 3-17. The Monitor Diagnostic Module listing. (Continued from page 64.)

```
1700 N=N+1:IF N>71 THEN N=32
1710 FOR X=1 TO 40
1720 PRINT CHR$(N);:N=N+1
1730 IF N>71 THEN N=32
1740 NEXT X
1750 NEXT I
1755 PRINT
1757 IF EX=1 THEN 1770
1760 GOSUB 3200
1765 GOTO 800
1770 GOSUB 2990:REM "Sprite Test"
1780 PRINT TAB(15)"Sprite Test"
1790 FOR S=2040 TO 2047:POKE S,13:NEXT
1792 FOR S=832 TO 894:POKE S,255:NEXT
1800 V=53248:POKE V+21,255
1810 FOR S=39 TO 46:POKE V+S,1:NEXT S
1815 R=0
1820 FOR I=0 TO 7
1830 POKE 53269,PEEK(53269)OR(2↑I)
1840 FOR X=24 TO 255
1850 POKE (V+I*2),X:POKE (V+I*2+1),152
1860 NEXT X
1870 POKE (V+16),R(I)
1880 FOR X=0 TO 65
1890 POKE (V+(I*2)),X:POKE (V+(I*2+1)),152
1895 NEXT X
1897 POKE V+16,PEEK(V+16)AND 254
1900 POKE 53269,PEEK(53269)AND(255-2↑I)
1910 NEXT I
1915 IF EX=1 THEN RETURN
1920 PRINT:GOSUB 3200
1930 GOTO 800
```

Fig. 3-17. The Monitor Diagnostic Module listing. (Continued from page 65.)

```
900 GOSUB 3000
```

When the user presses a key, its value is placed into variable A$. Line 100 checks whether the value of A$ is equal to 19, which is the CLR/HOME key. If it is, the program returns to line 50, the beginning of the System Diagnostic Main Menu:

```
1000 IF ASC(A$)=19 THEN 50
```

If the value of A$ is not equal to 19, lines 1010 through 1070 check to see whether the key pressed is a valid entry for this test, valid entries being numbers 1 through 6. If the user pressed a 1 for the Display Test, the program would branch to line 1090. If the user pressed 2 for the Alignment Test, the program would branch to line 1180, and so on:

```
1010  IF A$="1" THEN 1090
1020  IF A$="2" THEN 1180
1030  IF A$="3" THEN 1320
1040  IF A$="4" THEN 1440
1050  IF A$="5" THEN 1650
1070  IF A$="6" THEN 1770
```

If the user does not press a valid key, the program falls through to line 1080. Line 1080 sends the program to the Beep subroutine at line 3100 to signal the error. The program returns, then goes back to line 900 to let the user try again with another key:

```
1080  GOSUB 3100:GOTO 900
```

Line 1090 is the beginning of the Display Test. It goes to the Clear Screen subroutine at line 2990. The REMARK statement indicates that this is the Display Test:

```
1090  GOSUB 2990:REM Display Character Set Test
```

Lines 1100 through 1120 display the Display Test title and prompts on the screen:

```
1100  PRINT TAB(14) "Display Test":PRINT
1110  PRINT "Enter Any Character to Fill Screen":PRINT
1120  PRINT "Press CLR/HOME to End Test"
```

The program falls through to line 1130, which goes to the Get-a-Key subroutine at line 3000. When the user presses a key, the program returns to line 1132:

```
1130  GOSUB 3000
```

Line 1132 prints the CLR/HOME key, which clears the screen to prepare for the actual test:

```
1132  PRINT "{CLR/HOME}"
```

Line 1135 checks whether the ASCII value of A$ is 19, the CLR/HOME key. If it is, the program branches back to line 800 to display the Monitor Diagnostic Menu:

```
1135  IF ASC(A$)=19 THEN 800
```

If the key is not the CLR/HOME key, the program falls through to line 1140. This line begins a FOR-TO loop which runs from 1 through 999 for the number of characters needed to fill up the screen display:

```
1140  FOR I=1 TO 999
```

The program falls through to line 1150, which prints the value contained in A$, obtained from the Get-a-Key subroutine when the user pressed a key. It also includes the NEXT statement which loops the program back to line 1140 for 999 iterations:

```
1150 PRINT A$;:NEXT
```

Line 1155 checks whether the Exerciser Module is in effect. If EX is set to one, the program branches to the next test. If EX is equal to zero, the program falls through to the next line.

```
1155 IF EX=1 THEN 1360
```

Line 1160 branches back to line 1130 to wait for another keystroke. The program continues to display any key pressed until the user presses the CLR/HOME key.

```
1160 GOTO 1130
```

When the user presses 2 from the Monitor Diagnostic Menu, the program branches to the Alignment Test starting at line 1180.

Line 1180 uses the Clear Screen subroutine to prepare the screen for this test. The REMARK statement indicates the test name:

```
1180 GOSUB 2990:REM Alignment Test
```

Line 1185 sets the mode to the uppercase-only character mode. This accesses the characters necessary to draw the alignment grid for this test.

```
1185 POKE 53272,21
```

C128 USERS: Line 1185 in the C128 code reflects a different POKE location and value.

Line 1190 prints the test title on the screen:

```
1190 PRINT TAB(13) "alignment test":
```

Lines 1200 through 1300 paint the alignment grid on the screen. This is done with FOR-TO loops of the graphic characters which form bins, vertical bars, and horizontal bars.

```
1200 PRINT CHR$(176);:FOR I=1 TO 18: PRINT CHR$(99);:NEXT I
1210 PRINT CHR$(178);:FOR I=1 TO 18: PRINT CHR$(99);:NEXT I
1220 PRINT CHR$(174):
1230 FOR I=1 TO 10: PRINT CHR$(98);TAB(19)CHR$(98);
     TAB(38)CHR$(98):NEXT I
```

```
1240  PRINT CHR$(171);:FOR I=1 TO 18 PRINT CHR$(99);:NEXT
1250  PRINT CHR$(123);:FOR I=1 TO 18 PRINT CHR$(99);:NEXT
1260  PRINT CHR$(179)
1270  FOR I=1 TO 10:PRINT CHR$(98);TAB(19) CHR$(98);TAB(38)
      CHR$(98):NEXT
1280  PRINT CHR$(173);:FOR I=1 TO 18:PRINT CHR$(99);:NEXT I
1290  PRINT CHR$(177);:FOR I=1 TO 18:PRINT CHR$(99);:NEXT
1300  PRINT CHR$(189):GOSUB 3200
```

Once the alignment grid is painted, the program goes to the subroutine at 3200, which displays a prompt to enter any key to continue. When the user enters a key, the program returns to line 1310.

Line 1310 resets the character mode to the upper/lowercase mode with the POKE instruction. It then branches back to line 800, the beginning of the Monitor Diagnostic Module:

```
1310  POKE 53272,23:GOTO 800
```

C128 USERS: Line 1310 in the C128 code is modified to reflect a different POKE location and value.

When the user presses 3 from the Monitor Diagnostic Menu, the program branches to line 1320 to begin the Character Set Test. This test provides a display of the full graphic character set available through the computer's character generator, either Commodore character set 1 or character set 2.

At line 1320, the Clear Screen subroutine clears the screen. The REMARK statement indicates the name of the test:

```
1320  GOSUB 2990:REM "Character Set Test"
```

Line 1330 displays the test title on the screen:

```
1330  PRINT TAB(11) "Character Set Test"
```

Line 1340 prompts the user to indicate which character set is being used for the test:

```
1340  PRINT:PRINT "Enter Character Set (1) or (2)-"
```

Line 1350 then branches to the Get-a-Key subroutine at line 3000 to get the character pressed by the user in response to this prompt. The character is placed in variable A$, which is used in line 1360:

```
1350  GOSUB 3000
```

Line 1360 checks to see if the value of A$ is equal to one. If it is, a POKE reads into memory location 53272, set to 21, ensuring that the user is set up for the chosen character set:

```
1360  IF A$="1" THEN POKE 53272,21
```

C128 USERS: The C128 code at line 1360 is modified to reflect a different POKE location and value.

Line 1370 does a FOR-TO loop, indicating the range of characters that are to be displayed for this test:

```
1370  FOR I=32 TO 255
```

Lines 1380 through 1395 check for special cursor positioning characters which would alter the display. When the FOR-TO loop increments to one of these characters, lines 1380 through 1395 catch it and cause the program to ignore it by going to the end of the NEXT loop at line 1410. The program then increments to the next character string value:

```
1380  IF I=128 OR I=130 OR I=131 OR I=132 OR I=141
      I=143 OR I=144 THEN 1410
1390  IF I=145 OR I=146 OR I=147 OR I=148 OR I=156
      OR I=157 OR I=158 THEN 1410
1395  IF I=159 THEN 1410
```

If the next character is a valid, displayable character, it is placed in CHR$(I), and then line 1400 prints it on the screen. A blank space is placed after the character:

```
1400  PRINT CHR$(I);+" ";
```

Line 1410 is the NEXT statement which increments the character string value:

```
1410  NEXT
```

When all 255 characters have been printed, the program falls through to line 1415, which checks whether the EX exerciser flag is on. If it is set to one, the program sets the system to the upper/lowercase character mode, and then branches to line 1440:

```
1415  IF EX=1 THEN POKE 53272,23:GOTO 1440
```

C128 USERS: The C128 code at line 1415 is modified to reflect a different POKE location and value.

Line 1420 then branches to the subroutine at line 3200, which prints the prompt to enter any key to end test. Once the user enters any key, the program falls through to line 1430:

```
1420  PRINT:GOSUB 3200
```

Line 1430 POKEs memory location 53272 to 23 to make sure that it's in the upper/lowercase character mode. The program then returns to line 800, which is the start of the Monitor Diagnostic Menu:

```
1430  POKE 53272,23:GOTO 800
```

C128 USERS: The C128 code at line 1430 is modified to reflect a different POKE location and value.

When the user enters a 4 from the Monitor Diagnostic Menu, the program branches to line 1440 for the start of the Color Test.

NOTE: If you do not have a color monitor, you may delete this routine from the System Diagnostic Program, or simply ignore it.

Line 1440 clears the screen and includes the identifying REMARK statement:

```
1440 GOSUB 2990:REM "Color Test"
```

Line 1450 displays the test title at the top of the screen:

```
1450  PRINT TAB(15) "Color Test":PRINT
```

Line 1460 assigns a series of 20 Xs to variable C$. This is used for the color bar pattern:

```
1460  "C$="XXXXXXXXXXXXXXXXXXXX"
```

Lines 1470 through 1620 are PRINT statements that print the name of the color in that color, and then fills in the blanks in C$ with that color. This is done for all 16 colors:

```
1470  PRINT "{CTRL 1} Black";TAB(12) C$
1480  PRINT "{CTRL 2} White";TAB(12) C$
1490  PRINT "{CTRL 3} Red";TAB(12) C$
1500  PRINT "{CTRL 4} Cyan";TAB(12) C$
1510  PRINT "{CTRL 5} Purple";TAB(12) C$
1520  PRINT "{CTRL 6} Green";TAB(12) C$
1530  PRINT "{CTRL 7} Blue";TAB(12) C$
1540  PRINT "{CTRL 8} Yellow";TAB(12) C$
1550  PRINT "{C= 1} Orange";TAB(12) C$
1560  PRINT "{C= 2} Brown";TAB(12) C$
1570  PRINT "{C= 3} Lt. Red";TAB(12) C$
1580  PRINT "{C= 4} Gray 1";TAB(12) C$
1590  PRINT "{C= 5} Gray 2";TAB(12) C$
```

```
1600 PRINT "{C= 6} Lt. Green";TAB(12) C$
1610 PRINT "{C= 7} Lt. Blue";TAB(12) C$
1620 PRINT "{C= 8} Gray 3";TAB(12) C$
```

Line 1625 checks for the exerciser flag EX once again. If it is set to one, the color is reset to the default of light blue. Then the program branches to line 1680:

```
1625 IF EX=1 THEN PRINT "{C= 7}":GOTO 1680
```

At line 1630, a blank line is printed. The subroutine at line 3200 prompts the user to enter any key to end the test. This provides the user with ample time to make any necessary monitor color adjustments:

```
1630 PRINT:GOSUB 3200
```

Line 1640 resets the screen to its default of light blue, then the program branches back to line 800 to display the Monitor Diagnostic Menu:

```
1640 PRINT "{C= 7}":GOTO 800
```

When the user presses 5 from the Monitor Diagnostic Menu, the program branches to line 1650.

Line 1650 clears the screen and identifies the name of the test:

```
1650 GOSUB 2990:REM "Scroll Test"
```

Line 1660 displays the test title:

```
1660 PRINT TAB(15) "Scroll Test":PRINT
```

Line 1670 prompts the user to enter the number of repetitions for the scroll test. When entered, this value is placed into variable A:

```
1670 PRINT "Enter the Number of Repetitions-";A:PRINT
```

Line 1680 sets N to 31. N is the character string to be printed. The value 31 will be the character string value with which the line is to begin:

```
1680 N=31
```

Line 1690 is the beginning of a FOR-TO loop which indicates the number of lines to be printed as specified by the user in line 1670.

```
1690 FOR I=1 TO A
```

Line 1700 begins a series of instructions that serve to build and display a sliding character string, 40 characters in length. N starts with character string 31, as indicated in line 1680. Line 1700 increments N until it gets to character string 71 (40 characters later). When N is incremented to greater than 71, N is brought back to a value of 32:

```
1700  N=N+1:IF N>71 THEN N=32
```

Line 1710 is a FOR-TO loop which ensures that 40 characters are printed on each line:

```
1710  FOR X=1 TO 40
```

Line 1720 prints character string N, which will have a value between 32 and 71, depending on where it is in the loop. N is incremented once each time it passes through the loop:

```
1720  PRINT CHR$(N);:N=N+1
```

Line 1730 checks to see if N is greater than 71. If it is, N is set back to 32:

```
1730  IF N>71 THEN N=32
```

Lines 1740 and 1750 are the corresponding NEXT statements. The first one, NEXT X, is for the 40 characters filling a single line. The second one, NEXT I, is for the specified number of lines being printed to see the screen scrolling:

```
1740  NEXT X
1750  NEXT I
```

Line 1755 prints a blank line to prepare for the GOSUB instruction at line 1760:

```
1755  PRINT
```

Line 1757 checks the EX flag. If it is set to one, the program branches to line 1770:

```
1757  IF EX=1 THEN 1770
```

Line 1760 goes to the subroutine at line 3200, which prompts the user to press any key to end the test:

```
1760  GOSUB 3200
```

Line 1765 branches the program back to line 800 to display the Monitor Diagnostic Menu:

```
1765  GOTO 800
```

When the user presses 6 for the Sprite Test, the program branches to line

1770, which clears the screen and includes the identifying REMARK statement:

```
1770  GOSUB 2990:REM "Sprite Test"
```

Line 1780 prepares the screen display for the Sprite Test by printing the test name:

```
1780  PRINT TAB(15)"Sprite Test"
```

Line 1790 is a FOR-TO loop which sets all the sprites' pointers to 13 in order to activate them. Locations 2040 through 2047 are sprite firmware locations.

```
1790  FOR S=2040 TO 2047:POKE S,13:NEXT
```

Line 1792 is a sprite definition, indicating that the sprite is to appear as a white box, which is created with this FOR-TO instruction. POKE S,255 defines the shape of the sprite as a square. The cassette buffer is used to temporarily hold this sprite definition for the purpose of this test. Each of the eight sprites are defined in this manner, one at a time:

```
1792  FOR S=832 TO 894:POKE S,255:NEXT
```

Line 1800 sets V equal to 53248, which is another firmware location for the sprite registers. Memory location V+21 is POKEd to a value of 255. This informs the system that all eight sprites will be used:

```
1800  V=53248:POKE V+21,255
```

Line 1810 is another FOR-TO loop, which colors the sprites white. This is done with the instruction S=39 TO 46. The values 39 through 46 are the eight locations of the eight sprites. V, which was set in line 1800, is POKEd into the value of S, which is 29 through 46. The value 1 indicates that the sprites are to be colored white:

```
1810  FOR S=39 TO 46:POKE V+S,1:NEXT S
```

The program then falls through to line 1820, which moves the eight sprites, one at a time, across the screen with a FOR-TO loop:

```
1820  FOR I=0 TO 7
```

Line 1830 uses a POKE and PEEK logic statement in order to activate the individual sprites for individual testing. Because this is in a FOR-TO loop, it begins with zero, and increments up through seven, to test all eight sprites:

```
1830  POKE 53269,PEEK(53269)OR(2↑I)
```

Line 1840 is another FOR-TO loop which sets up the XY coordinates to place the location of the sprite. The sprite will move from left to right, which is represented by the values 24 (at the left edge of the screen) through 255 (almost to the right of the screen):

```
1840  FOR X=24 TO 255
```

Lines 1850 through 1890 consist of more POKE statements and FOR-TO statements which cause the sprite to continue on to the right edge of the screen. Line 1870 uses the array R, which was initialized in line 35:

```
1850  POKE (V+I*2),X:POKE (V+I*2+1),152
1860  NEXT X
1870  POKE (V+16),R(I)
1880  FOR X=0 TO 65
1890  POKE (V+(I*2)),X:POKE (V+(I*2+1)),152z
1895  NEXT X
```

C128 USERS: Lines 1790 through 1880 of the C128 code are completely modified to reflect the system's method of programming sprites. This particularly involves the special sprite-related commands, SPRITE and MOVSPR. Refer to Appendix B for the actual code.

Line 1897 resets the sprite X pointer back to the left edge of the screen:

```
1897  POKE V+16,PEEK(V+16)AND 254
```

Line 1900 turns off the sprite currently being tested so that the next sprite can be tested in the same manner:

```
1900   POKE 53269,PEEK(53269)AND(255-2   I)
```

Line 1910 is a NEXT statement which lets the program move on to the next sprite to be tested. This loop continues until all eight sprites are tested from left to right:

```
1910  NEXT I
```

Line 1915 checks for the EX flag. If it is set to one, the program returns to the Exerciser module:

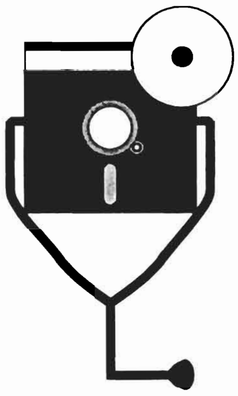
```
1915  IF EX=1 THEN RETURN
```

Line 1920 prints a blank line, and then branches to the subroutine which prompts the user to press any key to end the Sprite Test:

```
1920  PRINT:GOSUB 3200
```

Line 1930 returns the program back to the Monitor Diagnostic Menu:

```
1930  GOTO 800
```



Chapter 4 The Printer

Although visual computer output is available from the monitor, this output is often needed in a physical *hardcopy* form as well. Printed output allows more detailed scrutiny of the information, and allows presentation to others. The printer is the means with which you can make a paper copy of computer processing results.

DESCRIPTION AND FUNCTION OF THE PRINTER

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

Types of Printers

There are a myriad of printers available on the market today which work with the Commodore. A printer is a standard peripheral, but you need to make sure that the printer has a serial bus interface connector. Aside from that, Commodore printer choices consist of the dot-matrix printer and the letter-quality printer.

The Dot-Matrix Printer. If a character on the monitor screen were magnified, 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 or more dots. The series of dots used to build characters is called a *dot matrix.*

The dot-matrix printer uses the same method as the monitor does 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 are called *bidirectional* printers, and they lay characters down on a page much faster, since no time is wasted in sending the print head back to the "beginning" of a new line.

Some dot matrix printers offer several *qualities* of print. These different qualities might include a 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 and more dots to make a darker, more solid-looking character with better resolution. The tradeoff 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 in terms of the output that can be created.

There are also dot-matrix printers that can print in different colors. The major difference is that the printer uses a multicolored ribbon.

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 people who use computers to

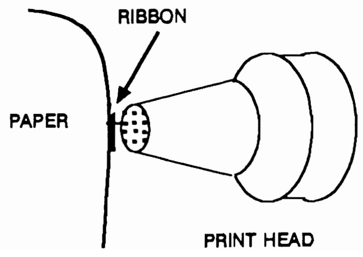


Fig. 4-1. Dot-matrix print head.

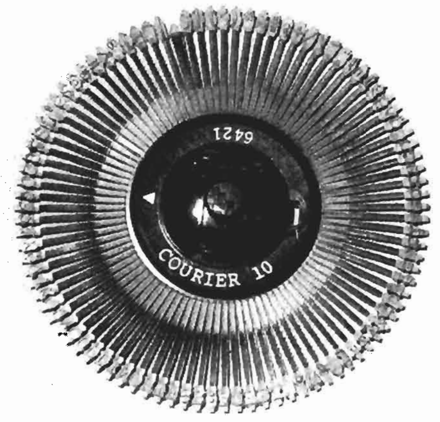


Fig. 4-2. Daisy wheel.

do mass mailings as part of their business. Letters printed on a letter-quality printer look as though they were individually typed on a typewriter, not mass-produced with a computer. In fact, any document that you want 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. 4-2).

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 principle similar to the typewriter, except that the motor-driven printer is much faster.

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. 4-3). They work in the same general manner as the daisy wheel.

Fig. 4-3. Character cup.

Printer Interfaces

The printer is physically connected, or *interfaced* to the computer through a receptacle on the back of the system unit which accepts the printer's cable.

There are two major ways a printer can interface with a computer: *parallel* and *serial*. The difference between these two interfaces is the manner in which the data is transferred. A parallel interface pushes seven or eight bits of data in one transfer. The serial interface sends data one bit at a time, in a series (Fig. 4-4). Commodore systems only support serial printers. Parallel printers are not compatible with these systems.

TROUBLESHOOTING AND REPAIR GUIDELINES

There are a number of problems the printer might experience, whether it is a dot-matrix or letter-quality printer. The following sections describe these potential problems, indicate how to diagnose them, and explain the steps necessary to repair them.

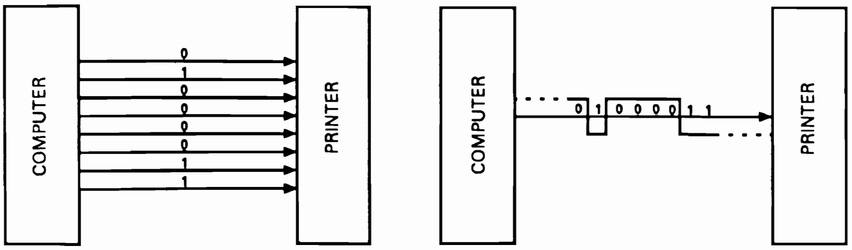
Problem: The Printer Does Not Work

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

☑ Solutions

☐ First check that power is applied to the printer. The printer has a power indicator light on the top or front of the chassis. If the printer's 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. Check whether this plug is controlled by a light switch.

☐ Make sure the printer is in the proper mode. Most printers have an ONLINE mode which indicates that it is ready to receive data from the computer. They might also have a LOCAL mode which allows you to physically manipulate the printer: roll the paper, prepare the printer for a different form, or adjust



Parallel Transfer of Data Serial Transfer of Data

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

the platen. 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 printer. 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 the application program is addressing the printer correctly. The printer might not be working because the correct printer driver has not been installed for this application.

The driver is activated by a printer installation program provided with your application software. This special utility program lets you tell the program what make of printer is being used 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. Even off-brand printers always have an equivalent to one of these choices. If you do not run the application's printer installation program, the driver might not be installed properly, and 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 7, "The Serial Communication Interface."

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 can occur when printing on a multiple copy form. While the top copy prints well, subsequent copies print poorly or not at all.

☑ Solutions

☐ Run the Sliding Alpha or Display Print Character Test of the Printer Diagnostic 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 platen to accommodate paper of heavier thicknesses, or multipart forms. Use the forms adjustment to move the head closer to the platen. 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 a lint-free cloth or cotton swab moistened with isopropyl alcohol.
- ☐ If the print quality is still unacceptable after changing the ribbon, adjusting the print head, and cleaning the element, refer to the printer documentation. Call the manufacturer for more information.

Problem: A Horizontal Tab Is Not Working

If the printer seems to be 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 the tabs do not print in this diagonal, check whether something is obstructing the *platen* (the cylinder around which the paper rolls) 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 does not solve the problem, refer to the printer documentation, or contact the manufacturer.

Problem: Paper Does Not Feed Properly

The printer might not feed the paper correctly. Certain commands issued from the application program to the printer can cause the paper to feed several lines, like carriage returns on a typewriter. When working with special forms like checks and invoices, the 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 enter a form feed command at a certain defined point, thus causing the printer to roll up to the next form.

Problems can occur when the specified vertical spacing does 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 of the Printer Diagnostic Module to check the carriage control. This test prints lines with variable line spacing and then feeds the paper up to the next page.
- ☐ If the Line Feed Test does not take place as expected, check that the paper has been fed into the printer correctly.
- ☐ If you are 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.
- ☐ Check the platen adjustment. There is a lever that unlocks the platen when

rolling the paper in. In order for the paper to automatically feed with the forms tractor, the platen must be set at the locked position.

☐ Check for a scrap of paper or other foreign object that might be lodged in the platen or in the paper feed area.

☐ If you suspect that the printer is inserting extra spaces between lines, run the Sliding Alpha Test of the Printer Diagnostic. The output should appear single-spaced. If there are blank line spaces between the printed lines, faulty option switch setting might be causing the extra line feeds. Check your printer documentation for more information about the switch settings.

☐ If none of the above suggestions repair the line spacing problem, refer to your printer documentation for the manufacturer recommendations.

Problem: Certain Characters Do Not Print

Sometimes a certain character does not print. The character exists 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 there is a black box or blank space where the character should be. This means different things, depending on whether you have a dot-matrix printer or a letter-quality printer.

☑ Solutions

☐ If you're working with a dot-matrix printer, run the Display Character Print or Echo Character Test of the Printer Diagnostic. If you see that black box or blank space where a character is supposed to be, then the printer is probably not set up to accommodate graphics.

As discussed earlier, there are several types of dot-matrix printers. One type accommodates graphic characters as well as the standard alphanumeric characters. Another type operates only in the character mode. If yours is a character printer, you cannot print graphic characters.

It might be confusing to see the graphic characters on your monitor display, and yet not be able to print them. This is because different makes of printers support different character sets. Some character sets are more limited than others. Therefore, be sure to send only valid characters to the printer.

The Commodore provides two graphic character sets. You might be attempting to print a character that is not in the currently active character set. The character set printed is the one that is active when the printer is powered on.

☐ In the case of a letter-quality printer, the daisy wheels are limited in size and can usually only accommodate about 100 characters, upper- and lowercase. Because of this, it is likely that the daisy wheel you're using does not include any special graphic characters. (However, you can get special graphic daisy wheels consisting of such special characters as part of their character sets.)

☐ If you're having trouble with regular alphanumeric characters, rather than

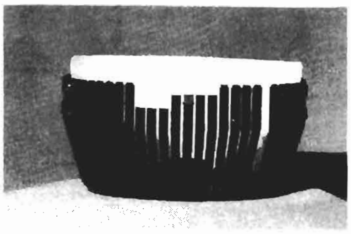


Fig. 4-5. Broken character cup.

graphic ones, on your letter-quality printer, the problem is of an entirely different nature.

For example, if the a's do not appear wherever they're supposed to, run the Sliding Alpha or Echo Character Test of the Printer Diagnostic to see if both the uppercase A and the lowercase a are missing. If this is the case, 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. 4-5). Remove the daisy wheel and examine it to confirm that it is indeed damaged. Buy a new daisy wheel to replace the broken one.

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. Only part of the a, part of the r, and part of the m, might be displayed, and the segment of the character that's missing is in the same location of each character, such as the upper left or lower right (Fig. 4-6).

☑ 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 solenoids that drives the dot matrix pins in the head has probably malfunctioned or worn out, and that would explain the partially formed characters. Take the printer in for servicing to have this solenoid motor repaired or replaced.

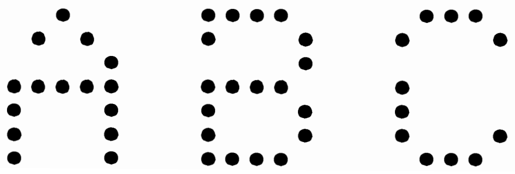


Fig. 4-6. Incompletely printed characters.

Problem: Installing a New Printer

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

For example, suppose you have been outputting your software processing results to a particular dot-matrix printer with no problem whatsoever. One day, you decide to upgrade to a letter-quality printer. You take the old printer out and plug the new printer into your computer, and think you're finished. Upon running an application and attempting to print the results, however, you discover that either the printer does not print at all, or that there are unexpected results. This happens because, as far as the computer system is concerned, the new printer is a completely foreign device. It is not addressable. As such, the computer and new printer do not know how to effectively communicate with one another.

☑ Solutions

- ☐ 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 application programs include 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.

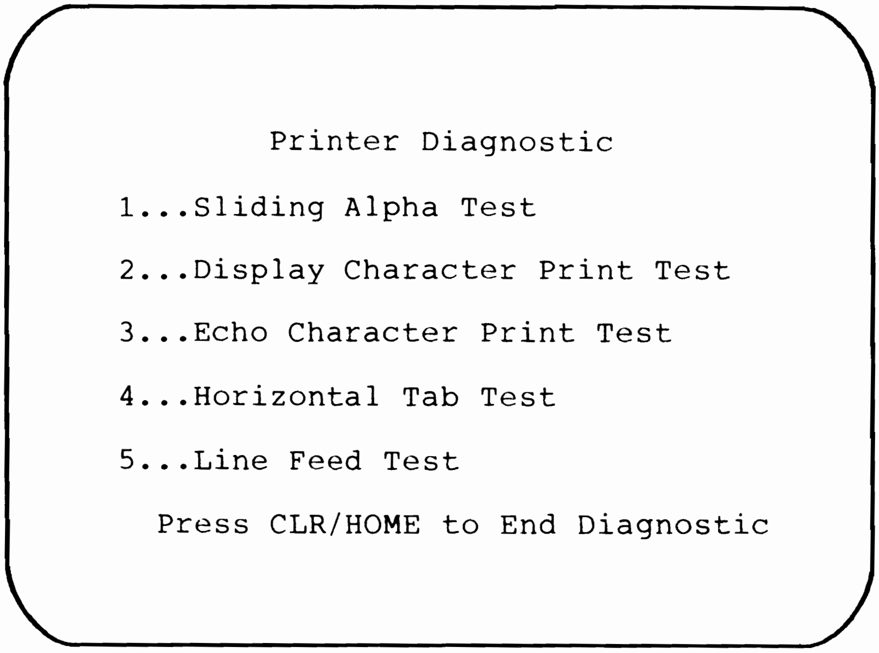
  The device driver lets you specify which printer is being used, and thus can let the program communicate correctly with the printer. Refer to the user manual of the application software for more information on configuring the device driver.

- ☐ When connecting a printer to the Commodore, ensure that any software programs that use the printer are set up to the proper interface. This information should be provided with your various user documentation; documentation for the computer system, the printer, and the software are all vital for configuration.

- ☐ You might encounter other problems when setting up a new printer. Again, read the printer documentation thoroughly before installing. Other documentation that might help includes Chapter 6, "Input/Output Guide," of the *Commodore 64 Programmers Reference Guide*. The user guide for any software programs with which you're going to use the printer can also be a good resource. Consult the printer manufacturer for further information.

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 or p to initiate the Printer Diagnostic Module. The screen clears, and the Printer Diagnostic Menu



```
              Printer Diagnostic

     1...Sliding Alpha Test

     2...Display Character Print Test

     3...Echo Character Print Test

     4...Horizontal Tab Test

     5...Line Feed Test

       Press CLR/HOME to End Diagnostic
```

Fig. 4-7. The Printer Diagnostic Menu.

is displayed as shown in Fig. 4-7. To activate one of the printer tests, press the corresponding number, 1 through 5.

Sliding Alpha Test

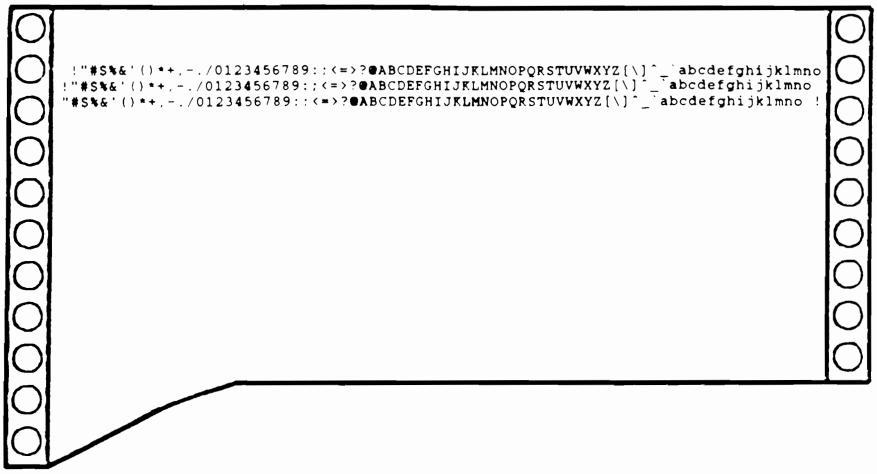
The printer Sliding Alpha Test is similar to the sliding alpha routine used in the Scroll Test of the Monitor Diagnostic Module. On the printer, this test serves two functions: to fully exercise the printer, and to get a quick look at the full alphanumeric display character set available for your printer.

When you press 1, 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 types a line full of as many repetitions 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 the number of lines you specified (Fig. 4-8).

If nothing happens when you try to run this test, or if the results do not look similar to the illustration, refer to "Troubleshooting and Repair Guidelines" earlier in this chapter.

Display Character Print Test

The Display Character Print Test displays all characters available within the



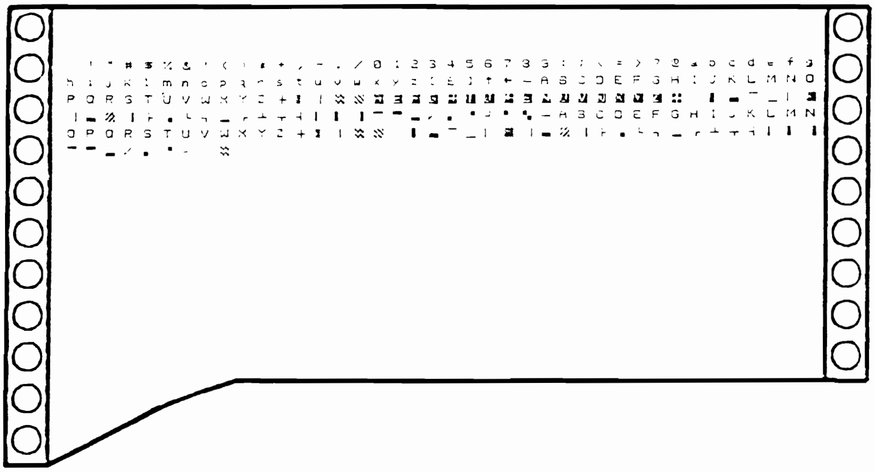
```
 !"#S%&'()*+,-./0123456789::<=>?●ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno
!"#S%&'()*+,-./0123456789:;<=>?●ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno
"#S%&'()*+,-./0123456789::<=>?●ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !
```

Fig. 4-8. Sliding Alpha Test results.

printer's character set, including graphic characters, and is used to check whether all characters are printing satisfactorily.

This test checks both character sets. You must have a dot-matrix printer in order to print the graphic characters.

To run the Display Character Print Test, simply press 2. All characters in the character set are printed (Fig. 4-9). 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 O
```

Fig. 4-9. Display Character Print Test results.

Echo Character Print Test

The Echo Character Print Test lets you enter a character and have it immediately printed, or echoed, on the printer. The entered character is echoed across all 80 columns of the printer. This test lets you ensure that a certain character is indeed printing, and lets you check its appearance. Make sure it's a complete character, and that it's not broken nor faded. This test also lets you check whether all columns along the printer carriage are accepting characters.

To run the Echo Character Print Test, press 3. You are prompted to enter the character to be echoed. 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 (Fig. 4-10). 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

The Horizontal Tab Test checks all combinations of tabbing across the printer carriage. To run the Horizontal Tab Test, press 4. 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. In this way, every horizontal tab position is checked. The printed result is a diagonal line of asterisks running from the upper left corner to the bottom right corner of the paper (Fig. 4-11).

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

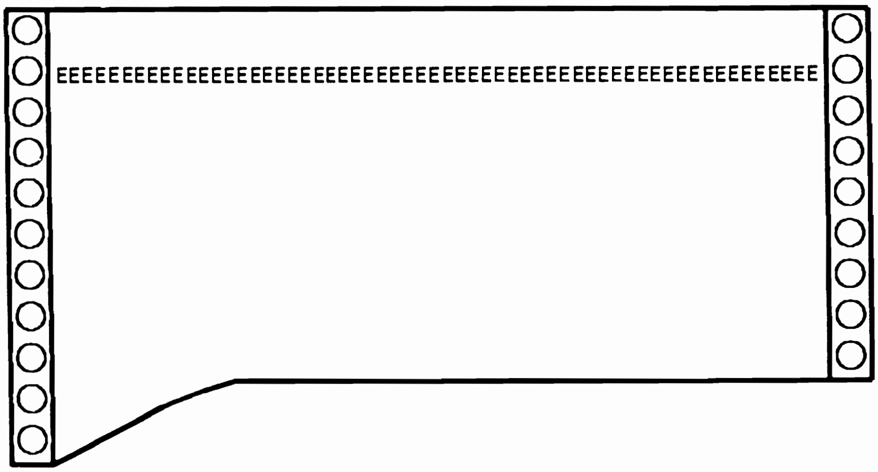

EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE

Fig. 4-10. Echo Character Print Test results.

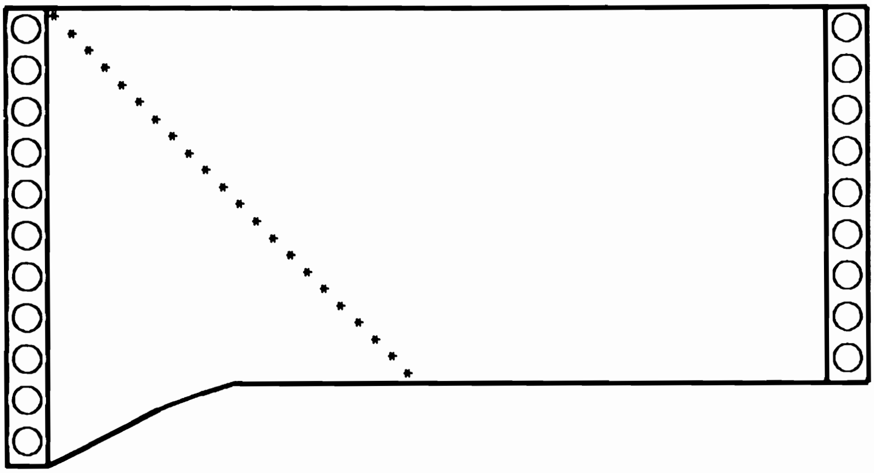


Fig. 4-11. Horizontal Tab Test results.

Line Feed Test

Line feed problems are manifested by 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 5 from the Printer Diagnostic Menu. Like the Horizontal Tab Test, this test needs no additional parameters entered, and runs as soon as it is selected.

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 line, gives five line feeds, and types a fifth line. Finally a form feed command is issued, and the printer rolls up to the next page and prints a line.

Line spacing and form feeding are often controlled by switch settings. If the results are unsatisfactory, refer to "Troubleshooting and Repair Guidelines" in the previous section for more information.

HOW THE PRINTER MODULE WORKS

Selecting P or p from the System Diagnostic Main Menu causes the program to branch to line 3500, which is the beginning of the Printer Diagnostic Module (Fig. 4-12).

```
3500  REM "Printer Diagnostic Module"
```

Line 3510 begins with a GOSUB instruction to line 2900 to clear the screen and position the cursor:

```
3510  GOSUB 2900:PRINT TAB(11) "Printer Diagnostic":PRINT
3515  PRINT TAB(3)"1 . . . Sliding Alpha Test":PRINT
```

```
3500 REM "Printer Diagnostic Module"
3510 GOSUB 2900:PRINT TAB(11) "Printer Diagnostic":PRINT
3515 PRINT TAB(3)"1...Sliding Alpha Test":PRINT
3517 PRINT TAB(3)"2...Display Character Print Test":PRINT
3518 PRINT TAB(3)"3...Echo Character Print Test":PRINT
3519 PRINT TAB(3)"4...Horizontal Tab Test":PRINT
3520 PRINT TAB(3)"5...Line Feed Test":PRINT
3530 PRINT TAB(5)"Press CLR/HOME to End Diagnostic
3535 OPEN 1,4,7:REM "Open as Upper/Lower case"
3540 GOSUB 3000:IF ASC(A$)<>19 THEN 3560
3550 CLOSE 1:GOTO 50
3560 IF A$="1" THEN 3630
3570 IF A$="2" THEN 3750
3580 IF A$="3" THEN 3920
3590 IF A$="4" THEN 4010
3600 IF A$="5" THEN 4070
3610 GOSUB 3100:GOTO 3540
3630 REM "Sliding Alpha Test"
3640 GOSUB 2990:PRINT TAB(11) "Sliding Alpha Test":PRINT
3645 INPUT "Enter the number of repetitions-";X
3650 N=31
3660 FOR I=1 TO X
3670 L$="":N=N+1:IF N>111 THEN N=32
3680 FOR A=1 TO 80
3690 L$=L$+CHR$(N):N=N+1
3700 IF N>111 THEN N=32
3710 NEXT A
3720 PRINT#1,L$;:NEXT I
3730 IF EX=1 THEN GOTO 3760
3735 CLOSE 1
3740 GOTO 3500
3750 REM "Display Character Print Test"
3760 GOSUB 2990:PRINT TAB(6)"Display Character Print
     Test":L$="":N=1
3770 CLOSE 1
3780 OPEN 1,4:REM "Open as Upper Case Only"
3790 FOR I=32 TO 255
3800 IF N=80 THEN PRINT#1,L$;:N=1:L$="":GOTO 3820
3810 L$=L$+CHR$(I)+" ":N=N+1
3820 NEXT I
3830 CLOSE 1
3840 OPEN 1,4,7:REM "Open as Upper/Lower Case"
3850 FOR I=32 TO 255
3860 IF N=80 THEN PRINT#1,L$;:N=1:L$="":GOTO 3880
```

Fig. 4-12. The Printer Diagnostic Module listing.

```
3870 L$=L$+CHR$(I)+" ":N=N+1
3880 NEXT I
3890 IF L$<>"" THEN PRINT#1,L$
3900 IF EX=1 THEN GOTO 3950
3905 CLOSE 1
3910 GOTO 3500
3915 REM "Echo Character Print Test"
3920 GOSUB 2990:PRINT TAB(7)"Echo Character Print Test":
     PRINT
3925 PRINT TAB(5)"Enter the Character to Echo":PRINT
3935 PRINT "Press the CLR/HOME Key to End the Test"
3940 GOSUB 3000:IF ASC(A$)=19 THEN CLOSE 1:GOTO 3500
3950 L$=""
3970 FOR I=1 TO 80
3980 L$=L$+A$:NEXT I
3990 PRINT#1,L$
4000 GOTO 3940
4010 REM "Horizontal Tab Test"
4020 GOSUB 2990:PRINT TAB(10)"Horizontal Tab Test"
4030 FOR I=1 TO 80
4040 PRINT #1,TAB(I)"*":NEXT I
4050 IF EX=1 THEN GOTO 4080
4060 CLOSE 1:GOTO 3500
4070 REM "Line Feed Test"
4080 GOSUB 2990:PRINT TAB(13)"Line Feed Test"
4100 FOR I=2 TO 10
4110 FOR A=1 TO I
4115 PRINT#1,"":NEXT A
4120 N$=STR$(I-1)
4130 PRINT#1,"There have been ";N$;" line feeds";:NEXT I
4140 IF EX=1 THEN RETURN
4150 CLOSE 1:GOTO 3500
```

Fig. 4-12. The Printer Diagnostic Module listing. (Continued from page 89.)

```
3517   PRINT TAB(3)″2 . . . Display Character Print Test″:PRINT
3518   PRINT TAB(3)″3 . . . Echo Character Print Test″:PRINT
3519   PRINT TAB(3)″4 . . . Horizontal Tab Test″:PRINT
3520   PRINT TAB(3)″5 . . . Line Feed Test″:PRINT
3530   PRINT TAB(5)″Press CLR/HOME to End Diagnostic
```

The balance of line 3510, together with lines 3515 through 3530, displays the Printer Diagnostic Menu.

Line 3535 opens the printer in the upper/lowercase character set mode:

```
3535   OPEN 1,4,7:REM ″Open as Upper/Lower case″
```

Line 3540 accesses the keyboard input subroutine, and waits for the user to press a key indicating the printer test choice:

```
3540   GOSUB 3000:IF ASC(A$)< >19 THEN 3560
```

If the user presses the CLR/HOME key (character string value 19), the program goes to line 3550, which closes the printer and returns to the System Diagnostic Main Menu:

```
3550   CLOSE 1:GOTO 50
```

When the user presses a key other than CLR/HOME, the program branches to lines 3560 through 3600:

```
3560   IF A$="1" THEN 3630
3570   IF A$="2" THEN 3750
3580   IF A$="3" THEN 3920
3590   IF A$="4" THEN 4010
3600   IF A$="5" THEN 4070
```

These particular instructions check whether values 1, 2, 3, 4, or 5 reside in variable A$. For example, if the user presses 1, the program branches to line 3630, the beginning of the Sliding Alpha Test.

If the user presses something other than 1 through 5, the program falls through to line 3610, sounding the beep which alerts the user that an invalid function has been entered:

```
3610   GOSUB 3100:GOTO 3540
```

The program returns to line 3540, at which point the user can try again with a valid key.

When the user presses 1, the program branches to line 3630 for the Sliding Alpha Test:

```
3630   REM "Sliding Alpha Test"
```

Line 3640 clears the screen and displays the Sliding Alpha Test title. The INPUT statement at line 3645 prompts for the number of repetitions to be printed. This number is placed into variable X:

```
3640   GOSUB 2990:PRINT TAB(11) "Sliding Alpha Test":PRINT
3645   INPUT "Enter the number of repetitions- ";X
```

Variable N holds the value of the characters to be included in the line which successively builds the sliding alpha pattern. In line 3650, N is set to 31. This is one less than 32, the character string value for a blank space:

```
3650   N=31
```

Line 3660 is the beginning of a FOR-NEXT loop:

```
3660   FOR I=1 TO X
```

This determines the number of lines to be printed, so it is FOR one to X; X being the number of repetitions specified at line 3645.

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

```
3670   L$="":N=N+1:IF N>111 THEN N=32
```

N starts out as 31, and then becomes 32, the character code for the blank space character. If N is greater than 111, N is reset back to 32. The characters with 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. It also causes the string to slide by one character for each line that is printed.

Another FOR-NEXT loop begins at Line 3680:

```
3680   FOR A=1 TO 80
3690   L$=L$+CHR$(N):N=N+1
```

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

Line 3700 begins a test similar to the test done in line 3670:

```
3700   IF N>111 THEN N=32
3710   NEXT A
```

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

Once all the characters have built an 80-character line, which is the printer carriage width, the program proceeds to line 3720:

```
3720   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. The NEXT I means that one of the repetitions has been carried out. This routine prints the lines for the specified number of repetitions.

When the number of repetitions entered into the INPUT statement for X is satisfied, the program goes to line 3730, which checks whether variable EX is equal to one. If it is, then the Exerciser Module is in control, and the program branches to line 3760. If EX is equal to zero, then the program proceeds to lines 3735 and 3740. The printer is closed, and the program branches back to display the Printer Diagnostic Menu:

```
3730  IF EX=1 THEN GOTO 3760
3735  CLOSE 1
3740  GOTO 3500
```

When 2 is pressed, the program goes to the Display Character Print Test starting at line 3750:

```
3750  REM Display Character Print Test
3760  GOSUB 2990:PRINT TAB(6)"Display Character Print
      Test":L$="":N=1
```

Line 3760 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 3770 closes the printer so it can be opened to print in the uppercase graphics character mode:

```
3770  CLOSE 1
```

Line 3780 reopens the printer in this mode:

```
3780  OPEN 1,4:REM "Open as Uppercase Only"
```

Lines 3790 through 3820 form a FOR-NEXT loop which prints the complete uppercase graphic character set:

```
3790  FOR I=32 TO 255
3800  IF N=80 THEN PRINT#1,L$;:N=1:L$="":GOTO 3820
3810  L$=L$+CHR$(I)+" ":N=N+1
3820  NEXT I
```

Lines 3830 and 3840 close and reopen the printer in the upper/lowercase mode:

```
3830  CLOSE 1
3840  OPEN 1,4,7:REM "Open as Upper/Lowercase"
```

Line 3850 prints the upper/lowercase graphic character set, cycling through values 32 to 255. These are the values for the standard CHR$ display codes (blank space, A through Z, 1 through 0, etc.).

```
3850  FOR I=32 TO 255
```

Line 3860 checks to see if variable N is equal to the carriage width of 80 characters:

```
3860  IF N=80 THEN PRINT#1,L$;:N=1:L$="":GOTO 3880
```

If N=80, a full line has been built, and L$ is output to the printer, device #1.

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 and the program skips to line 3880, the NEXT I statement.

If N is not equal to 80 at line 3860, the program falls through to 3870, which concatenates the next character onto the L$ and adds one to N:

```
3870   L$=L$+CHR$(I)+" ":N=N+1
```

This loop continues until all character positions have been printed:

```
3880   NEXT I
```

Because a full line is not built for the part of the character set, L$ must be dumped after the FOR-NEXT loop. Line 3890 accomplishes this:

```
3890   IF L$<>"" THEN PRINT#1,L$
```

Line 3900 checks whether the EX flag is set to one. If it is, the Exerciser Module is operating, and the program branches to line 3950:

```
3900   IF EX=1 THEN GOTO 3950
```

Line 3905 closes the printer:

```
3905   CLOSE 1
```

Line 3910 returns the program to display the Printer Diagnostic Menu:

```
3910   GOTO 3500
```

When the user enters 3 from the Printer Diagnostic Menu, the program branches to line 3915 for the Echo Character Print Test:

```
3915   REM "Echo Character Print Test"
```

At 3920, the screen is cleared, and the test name is printed:

```
3920   GOSUB 2990:PRINT TAB(7)"Echo Character Print Test":
       PRINT
```

Line 3925 prompts the user to enter the character to be echoed on the printer:

```
3925   PRINT TAB(5)"Enter the Character to Echo":PRINT
```

Line 3935 prompts the user to press the ESC key to end the test:

```
3935   PRINT "Press the CLR/HOME Key to End the Test"
```

Line 3940 goes to the Get-a-Key subroutine at line 3000. As before, if the value of A$ is equal to character string value 19 (the CLR/HOME key), the program closes the printer and then branches back to the Printer Diagnostic Menu:

```
3940   GOSUB 3000:IF ASC(A$)=19 THEN CLOSE 1:GOTO 3500
```

Line 3950 sets L$ to a null value:

```
3950   L$=""
```

Once the user enters a valid key to be echoed, the program falls to 3970, which is a FOR-NEXT loop:

```
3970   FOR I=1 TO 80
```

As before, 80, the carriage length, is used as the parameter for how far the character is to be printed.

Line 3980 takes the entered character and joins it with L$ for the length of the carriage:

```
3980   L$=L$+A$:NEXT I
```

Once all positions in the carriage width are filled, the program proceeds to line 3990.

Line 3990 outputs the contents of L$ to the printer:

```
3990   PRINT#1,L$
```

Line 4000 returns the program to line 3940, at which point the user can either enter another character to be echoed, or press CLR/HOME to end this test and return to the Printer Diagnostic Menu:

```
4000   GOTO 3940
```

Pressing 4 from the Printer Diagnostic Menu branches the program to the Horizontal Tab Test starting at line 4010:

```
4010   REM Horizontal Tab Test
4020   GOSUB 2990:PRINT TAB(10)"Horizontal Tab Test"
```

The screen is cleared and the test title is displayed.

Line 4030 begins another FOR-NEXT loop for the length of 80, the printer carriage length:

```
4030   FOR I=1 TO 80
```

Line 4040 prints a series of asterisks using the TAB statement. This causes horizontal tabbing:

```
4040   PRINT #1,TAB(I)"*":NEXT I
```

The TAB statement uses variable I of the FOR-NEXT loop as the tab value. It prints an asterisk, then loops back to repeat for the next value of variable I. This causes the asterisks to be printed in a stairstepped row.

After printing this diagonal row of asterisks, the program proceeds to line 4050, which checks whether variable EX is equal to one. If EX is equal to one, the Exerciser Module continues at line 4080. If EX is not equal to one, the program falls through to line 4060, which closes the printer and returns the program to display the Printer Diagnostic Menu:

```
4050   IF EX=1 THEN GOTO 4080
4060   CLOSE 1:GOTO 3500
```

The final test is activated by pressing 5. The program branches to line 4070 for the Line Feed Test:

```
4070   REM "Line Feed Test"
4080   GOSUB 2990:PRINT TAB(13)"Line Feed Test"
```

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

Line 4100 begins one of two FOR-NEXT loops:

```
4100   FOR I=2 TO 10
```

This causes vertical line spacing with lines 2 through 10.

Line 4110 begins the second FOR-NEXT loop that has the printer producing blank lines with simple carriage returns:

```
4110   FOR A=1 TO I
```

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

Line 4115 prints the contents of the line to the printer, then goes on to the next cycle of the FOR-NEXT loop:

```
4115   PRINT#1,"":NEXT A
```

The program falls through to line 4120, which sets N$ equal to the string value of I. The (I-1) part of the statement makes sure that the first line feed actually takes place on the first line:

```
4120   N$=STR$(I-1)
```

This creates orderly output for line 4130, which is the message indicating the number of lines that have been tabbed:

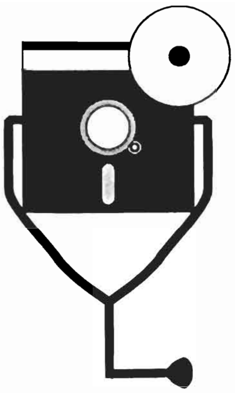
```
4130   PRINT#1,"There have been ";N$;" line feeds";:NEXT I
```

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

After this is completed, line 4140 checks to see if EX is equal to one. If it is not, the program proceeds to Line 4150:

```
4140   IF EX=1 THEN RETURN
4150   CLOSE 1:GOTO 3500
```

This causes the program to close the printer and then branch back to line 3500, returning the program to display the Printer Diagnostic Menu.



Chapter 5
The Cassette Drive

The previous chapters have primarily covered computer input and output, the keyboard being the major input device, and the monitor and printer being output devices. However, when you wish to do anything even slightly sophisticated with your computer, in addition to the facilities of input and output, you need one more function: mass storage.

The system unit of the Commodore 64 can store 64 kilobytes (abbreviated 64K) of information at one time. The Commodore 128 system unit memory can store 128 kilobytes of information at one time. This is fine until you want to turn the computer off. The Commodore systems have what is called a *volatile* memory. This means that in order for the memory chips to retain their information, power must be continuously applied to their circuitry. Once the computer is powered off, everything that was stored in that 64 or 128K is erased.

To prevent the loss of valuable programs and data, mass storage devices such as the cassette tape drive and the floppy diskette drive are used. When you command the computer to save a file, the program currently in system memory is written onto the magnetic media, whether it is a cassette tape or a diskette. The file is written in *digital* form, as a series of ON and OFF bits.

Likewise, when you command the computer to restore a saved file, the system retrieves the information stored on the media (tape or diskette). It places it into system memory just as it was last saved. This facility also allows you to buy prerecorded programs and load them into your system.

While Chapter 6 covers the disk drive, this chapter deals solely with the cassette drive.

DESCRIPTION AND FUNCTION OF THE CASSETTE DRIVE

The cassette tape drive used with the Commodore is very similar to the cassette machine in your home stereo or in your car. It has the same mechanics, and works on the same principles. It loads the same size and type of cassette tape. It has record/playback head(s), and several controls: RECORD, PLAY, FAST FORWARD, REWIND, STOP, and EJECT. It might also have a tape counter, which is particularly useful when storing several files on one tape.

The difference between the computer cassette drive and your stereo's cassette player is that instead of recording and playing back music through a speaker, the cassette drive records and "plays back" digital information through a computer.

This difference indicates the two storage functions the cassette drive provides: recording, or *writing* information from the computer onto the tape; and playing back, or *reading* information from the tape to the computer.

How the Cassette Drive Functions

The computer and the cassette drive operate in two entirely different languages. The computer understands digital information consisting of ones and zeros, or ON and OFF. The cassette drive, on the other hand, understands *analog* information consisting of waveforms, or signal noise.

Fortunately, the two devices can translate their respective languages to accommodate one another. This is accomplished with the cassette drive interface cable. When the cassette drive cable is plugged into the appropriate connector on the back of the Commodore, the translator is set up (Fig. 5-1).

Then, when there is data or a program you would like to save to tape, you can type the appropriate command and press the RECORD button on the cassette drive. The computer's digital information goes through the cassette interface and is converted into an analog noise signal (Fig. 5-2). This "sound" is then input to the cassette drive and recorded directly onto the tape media.

When there is previously stored data or a prerecorded program you would like to retrieve, it works in the reverse. Load the cassette holding that information, give the appropriate retrieve command from the computer, and press the PLAY button on the cassette drive. The analog signal on the cassette tape moves from the cassette drive and through the cassette interface. This time, the analog signal is converted into a corresponding digital signal. This signal is then input to the computer and the information is placed into system memory. You can now access the information for display on the monitor and further processing as desired.

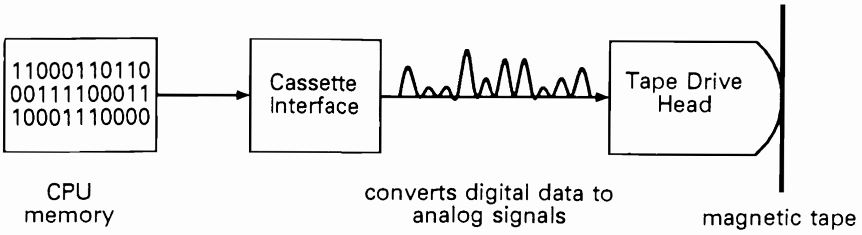


Fig. 5-1. The cassette tape drive interface.

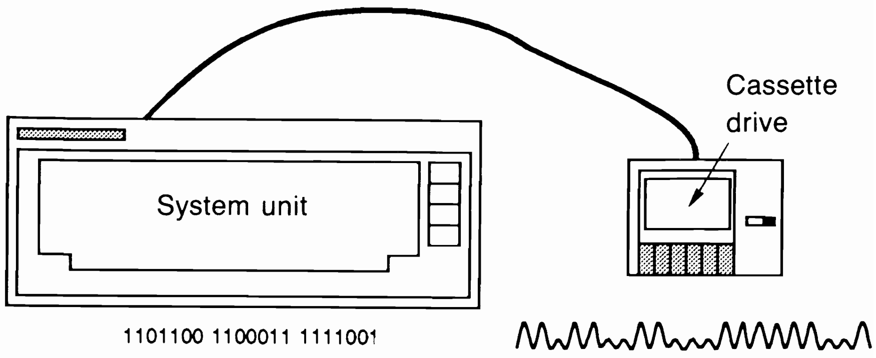


Fig. 5-2. Digital to analog signal conversion.

The Cassette Tape Media

The material in the cassette which stores this information is a *magnetic media*. It is a type of plastic, called "mylar," that has been coated with iron oxide. Iron oxide is a particular material that can hold a magnetic charge. When recording, the record head of the cassette drive mechanism causes a specific magnetic pattern in the form of an analog signal to be recorded onto this tape media (Fig. 5-3).

The tape media is wound from one spool to a second spool, and this is all encased within a cassette. As the spools are turned with the PLAY, FAST FORWARD, or REWIND controls on the cassette drive, the tape is fed from one spool to the other in the indicated direction. This facilitates a linear, or *sequential*, access to the recorded data (as opposed to a *random* access, which is the case with a disk drive).

Various lengths of cassette tapes are available: 30-minute, 60-minute, 90-minute, and 120-minute. When choosing the length, consider whether you'll be using a new tape for each new program or set of information, in which case you might choose one of the tapes that are shorter in length. Also consider that the 120-minute tape is made of a thinner mylar, and as such, it might be more fragile than the others, and therefore might not be as reliable.

There are also varying degrees of cassette tape quality available from many different manufacturers. Choose the quality in accordance with how you intend to use the tape. For example, in the case of recording audio, lower-quality tapes might be acceptable for recording a conversation or a meeting, but a higher-quality tape is more desirable for recording high-fidelity music. Likewise, use low-quality

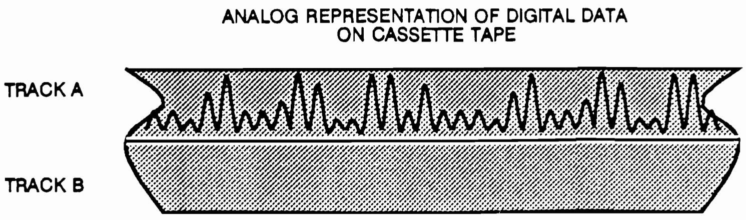


Fig. 5-3. Analog signals recorded on cassette tape.

tapes to store the less important data, and high-quality tapes to store important data and programs. And remember to make backups of everything.

NOTE: Whenever you write a program or data to tape, always make one or two backups, just in case something goes wrong.

The Cassette Tape Drive

Your cassette drive might be the Commodore VIC Datasette, or it might be a standard cassette tape machine. The Commodore Datasette cassette drive is made for the specific purpose of reading and writing data. Because of this, the Datasette has no volume control, tone control, or speaker. This simplifies the unit, because there is no need for a speaker. The computer is not interested in audible sound, but only the signal. Volume and tone settings are necessary, but the VIC Datasette sets these functions internally, to prevent faulty read or write operations.

Nonetheless, any cassette tape machine can work successfully as the Commodore's cassette drive, as long as it includes the RECORD and PLAY functions. Be careful to set the volume and tone close to the higher settings, however. Ensure that the settings are the same when you record material and when you play it back. If the settings are changed, the data might not read back correctly.

Whichever type of cassette drive you use, it will include the RECORD, PLAY, FAST FORWARD, REWIND, STOP, and EJECT functions (Fig. 5-4). It will also have the head mechanism, which is the device that magnetically records the data onto the tape or reads the recorded information from tape. Finally, the cassette drive will include the motor, which rotates the cassette reels to advance the tape.

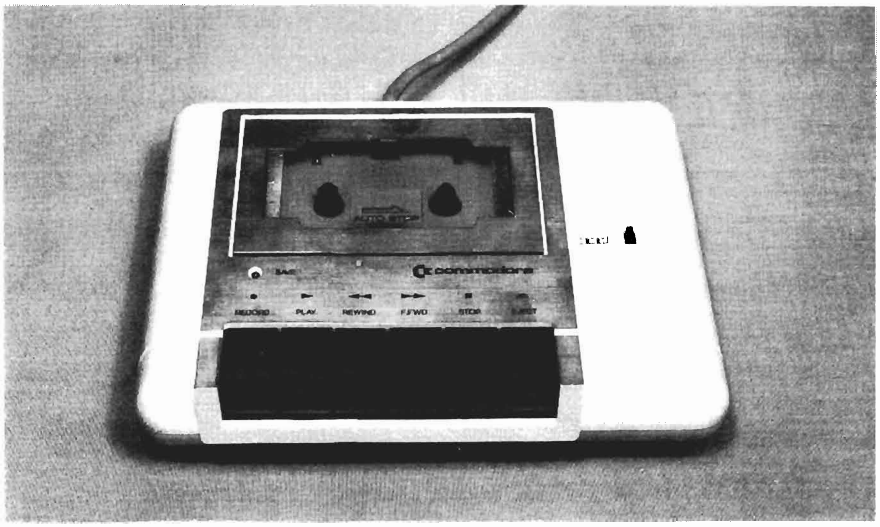


Fig. 5-4. Commodore VIC Datasette.

TROUBLESHOOTING AND REPAIR GUIDELINES

There are several problems possible with the cassette drive. The following sections identify these problems, indicate how to diagnose them, and describe specific repair procedures.

Problem: The Cassette Drive Does Not Work

If nothing happens when you turn the cassette drive power on, insert a tape, and press one of the buttons, there is a problem with the cassette, with power, or with the system unit.

☑ Solutions

- ☐ First check the cassette tape itself. If you pressed PLAY or FAST FORWARD and nothing happened, it could mean that the tape is wound all the way through. Press REWIND. If it begins rewinding, then this certainly was the case. If it does not rewind, and if the red LED does not light, a power problem is indicated.
- ☐ Check the power source. The VIC Datasette power is derived from the Commodore system unit itself. Whenever the Commodore system unit is powered on, the tape drive should power on along with it. The LED on the front of the Commodore indicates that the system unit has power. If the LED does not light up, check the power receptacle to which the Commodore is connected. Check to see whether that receptacle is working, and whether it is controlled by a switch.

  A standard cassette tape drive is plugged directly into its own power receptacle. In this case, check the power receptacle to which the cassette drive is connected.
- ☐ If the Commodore system is on and functioning, and the power receptacle is working fine, then check the cabling. As noted, the VIC Datasette drive cable is a combination power cable and interface cable. A standard cassette drive has its own power cable connected into the AC wall socket, and a separate interface cable. In both cases, the interface cable runs from the back edge of the cassette drive into an edge connector on the back of the Commodore system unit (Fig. 5-5). Check that the cable is properly attached to the edge connector.

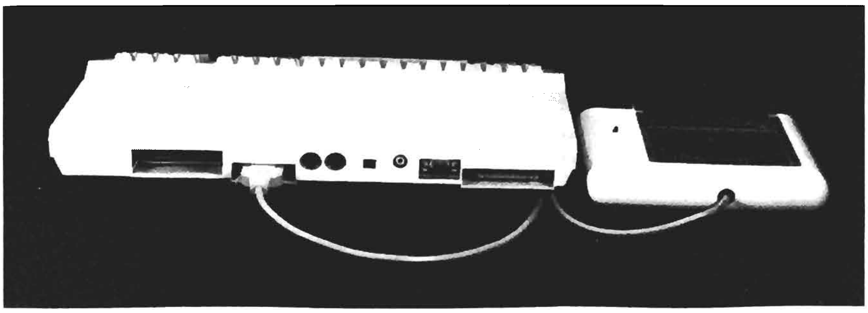


Fig. 5-5. Cassette drive interface connection.

☐ If the cable looks secure on both ends, note that the cassette drive must be connected to the system unit prior to powering on the system. If you tried to connect the drive while the system unit is powered on, the system might not have recognized that the drive was there. To check this, turn the system off, turn it on, and try to load the tape again.

☐ If the tape still does not load, try swapping a different tape drive on your system. If the swapped tape drive works on your system, there is a problem with your cassette drive, and it should be replaced (it probably would cost as much to repair a defective tape drive as it would to buy a new one).

☐ If the swapped tape drive does not work on your system, try using your tape drive on a different system. If your tape drive works on a different system, there is a problem with your system unit. Refer to the "Troubleshooting and Repair Guidelines" in Chapter 2, and follow the instructions for troubleshooting the system unit.

Problem: Drive Is not Reading or Writing Correctly

When attempting to read data, if the LOAD message remains on the screen for an unusually long time, perhaps five minutes, for example, this indicates that the system could not complete the load process. If the LOAD message is normal, but garbled results or no results are displayed when you try to run the program, there is either a problem with the cassette tape, with the cassette drive, or with the system unit.

To test whether there is a write problem, you could write a program to tape, then rewind it and give the VERIFY command. VERIFY is a Commodore function which reads the tape and checks whether the material on the tape matches the material currently in system memory. If this test fails, there is a write problem, which again could originate from the cassette tape, cassette drive, or system unit. The majority of the remedies for both read and write problems are the same, and are as follows.

☑ Solutions

☐ First run the Cassette Tape Diagnostic, particularly the Read/Write Test described in the next section, "Running the Cassette Drive Diagnostic." Use a new, blank cassette tape.

If the cassette drive passes the test, this indicates that the tape you tried to read to or write from is defective. Use its backup. If the test fails, there might be a problem in the read/write circuitry of the system unit. There are other tests you should try, however, before delving into the computer itself.

☐ Load a cassette that you know is good—one that contains a program loaded many times in the past without fail. If it does not load at first, try it again several more times. Sometimes the tape will eventually read, but this is often an indication of a problem with the cassette tape media. Immediately make a backup, if you don't have one already.

- ☐ If the test still fails, try loading yet another tape that has recently been successfully loaded. If this tape loads, this means the other tape was defective. Discard it and use its backup.

- ☐ If the second tape does not load either, try loading the tape on another system. If the tape still does not load, the tape is defective. If it does load on the other system, create a backup copy of the tape and try it again on your system.

  If the backup works on your system, the tape was defective. However, if the backup still does not work on your system, there might be a more serious problem with the cassette drive or the system unit.

- ☐ If you are using a standard cassette drive rather than the Commodore VIC Datasette, check the levels of the volume and tone controls. If either of these is set too low, the data signal will not be interpreted properly, causing an unsuccessful read or write.

  A good rule of thumb is to maintain both volume and tone controls at least three-quarters of the way up. The Read/Write Test of the Cassette Diagnostic Module can also help you determine the best levels.

- ☐ If you're not having trouble reading from a cassette tape, but are having trouble writing, check the cassette tape itself—it might be write-protected. Cassette tapes usually have a write-protect tab at the top, and if it has been broken off, nothing can be saved on this tape.

  You can easily make the tape write-enabled again by placing a small piece of cellophane tape over the write-protect gap (Fig. 5-6).

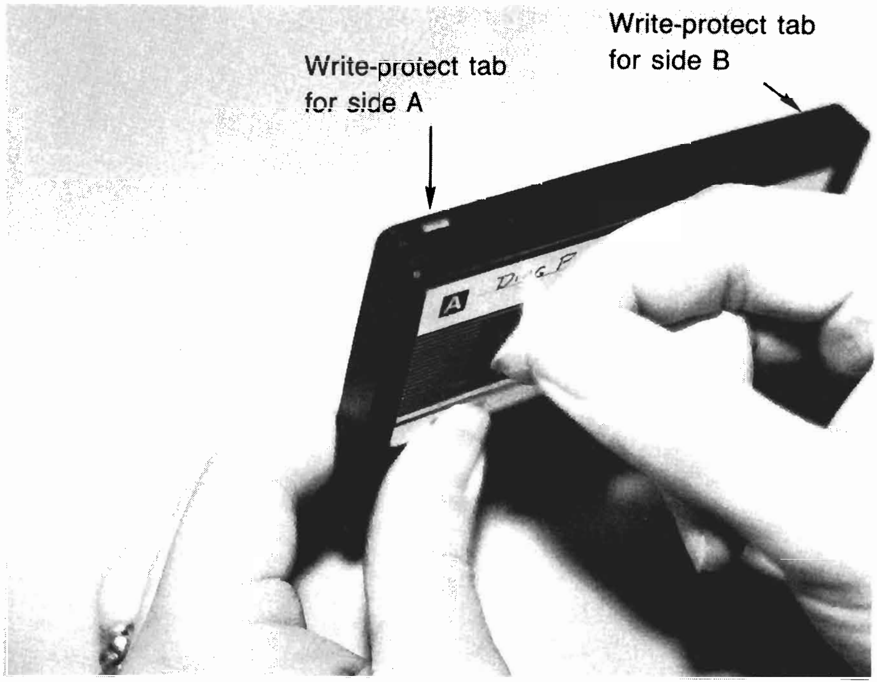


Fig. 5-6. Write-enabling a write-protected cassette tape.

☐ If the system is attempting to read and write information, but the results are not accurate, the problem might be that the read/write head in the cassette drive needs cleaning.

The cassette's mylar tape is coated with iron oxide that comes into direct contact with the cassette drive's read/write head. This head magnetizes the tape when writing (recording), or reads the magnetic pattern in the tape when reading (playing).

Because of this direct contact, the iron oxide coating can rub off on the head. Over a period of time, this residue on the head can begin to inhibit its functions of reading and writing data.

To clean the iron oxide residue from the cassette drive head, moisten a cotton swab or lint-free cloth with isopropyl alcohol. Rub the swab or cloth over the head (Fig. 5-7). Now attempt to load the tape again.

☐ If there is still trouble reading or writing, try cleaning the cassette drive interface edge connector. This edge connector can oxidize, and the oxidation can prevent the necessary electrical connection.

To clean off the oxidation, use the pink eraser from the end of a pencil, and rub it over the edge connector (refer to Fig. 2-6 for a photograph of this procedure). Black or gray residue will be removed by the eraser.

Reconnect the cable and attempt to read or write the problem cassette tape again.

☐ If problems persist, swap a different cassette drive on your system and load a tape. If you are able to read/write a tape on the swapped tape drive, there is a problem with your original cassette drive, and it should be replaced.

☐ If the swapped tape drive does not work on your system, swap your cassette drive on a different system and load a tape. If you are able to read/write a tape on the different system, a problem with your system unit is indicated. Refer to the "Troubleshooting and Repair Guidelines" in Chapter 2, "The System Unit/Keyboard," and follow further instructions for troubleshooting the system unit.

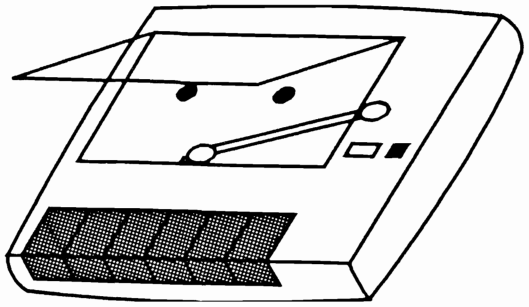


Fig. 5-7. Cleaning the tape head.

RUNNING THE CASSETTE DRIVE DIAGNOSTIC MODULE

To run the Cassette Drive Diagnostic Module, follow the instructions provided in "Running the Diagnostic Program" in Chapter 1. Then press C to initiate the Cassette Drive Diagnostic Module. The Cassette Drive Diagnostic Menu is displayed (Fig. 5-8).

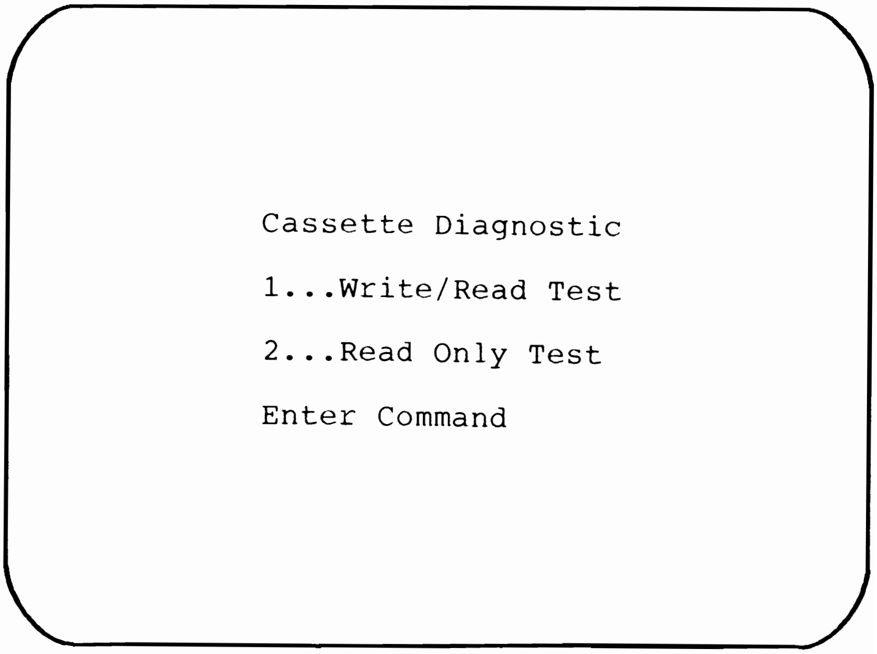
The Cassette Drive Diagnostic Module consists of two tests: the Write/Read Test and the Read-Only Test. Use a blank cassette tape to run this test. The test results might prove more accurate when using a new tape for testing. If you don't have a blank cassette, at least use one containing data that is no longer needed. This test will overwrite any existing data on the tape.

The Write/Read Test writes two files of 100 records each on the tape. It then reads what it wrote, to ensure a match and therefore verify the integrity of both the write and read circuitry. You should run this test and keep the tape as a test tape for future use.

The Read-Only Test reads the two files of 100 records created in the Write/Read Test. This is particularly useful when you suspect that the system is experiencing problems with the read circuitry. Run this test on the test tape made previously with the Write/Read Test to verify whether or not there is a problem.

To select the Write/Read Test, choose 1 from the Cassette Drive Diagnostic Menu. You are prompted to insert a blank cassette tape into the cassette drive. Make sure this tape is fully rewound to the beginning. Then, as prompted, select the RECORD button, and press any key to continue.

The Write/Read Test then writes two files out to the tape. The first one has



```
Cassette Diagnostic

1...Write/Read Test

2...Read Only Test

Enter Command
```

Fig. 5-8. The Cassette Drive Diagnostic Menu.

an alternating pattern of 10101010. The second one has the opposite pattern of 01010101. The program writes 100 records of these patterns to the tape.

Once this is done, you are prompted to press the cassette drive STOP button, rewind the tape to the beginning, press the PLAY button, and press any key to continue. The program reads both files containing the 100 records of alternating bit patterns. As it reads, it compares each record to the value that was written. If there is a discrepancy, an error message is displayed, indicating the number of errors encountered during the test.

To select the Read-Only Test, enter 2 from the Cassette Drive Diagnostic Menu. You are prompted to load the cassette tape to be read into the cassette drive. Be sure to have the tape rewound to the beginning. Press PLAY, and then press any key to continue.

HOW THE CASSETTE DRIVE MODULE WORKS

When the user presses C or c from the System Diagnostic Main Menu, the program branches to line 4200. A REMARK statement at line 4200 indicates the beginning of the Cassette Drive Diagnostic Module (Fig. 5-9):

```
4200   REM "Cassette Diagnostic Module"
```

Lines 4210 through 4250 clear the screen and display the Cassette Drive Diagnostic Menu:

```
4210   GOSUB 2900
4220   PRINT TAB(10) "Cassette Diagnostic":PRINT
4230   PRINT TAB(10) "1 . . . Write/Read Test":PRINT
4240   PRINT TAB(10) "2 . . . Read Only Test":PRINT
4250   PRINT TAB(10) "Enter Command"
```

Line 4260 sends the program to the Get-a-Key subroutine at line 3000. A valid character here is either 1 or 2, corresponding to the two tests within this diagnostic:

```
4260   GOSUB 3000
```

Lines 4270 and 4280 check for these values and branch to the appropriate places in the code:

```
4270   IF A$="1" THEN 4310
4280   IF A$="2" THEN 4420
```

If a key other than 1 or 2 was pressed, the program falls through to line 4290, which sends for the Beep subroutine at line 3100. The program then proceeds on to line 4300, which returns the program back to line 4260, letting the user enter another key:

```
4200 REM "Cassette Diagnostic Module"
4210 GOSUB 2900
4220 PRINT TAB(10) "Cassette Diagnostic":PRINT
4230 PRINT TAB(10) "1...Write/Read Test":PRINT
4240 PRINT TAB(10) "2...Read Only Test":PRINT
4250 PRINT TAB(10) "Enter Command"
4260 GOSUB 3000
4270 IF A$="1" THEN 4310
4280 IF A$="2" THEN 4420
4290 GOSUB 3100
4300 GOTO 4260
4310 GOSUB 2990:PRINT TAB(13)"Write/Read Test":PRINT
4320 PRINT "Place a blank tape in the cassette drive"
4330 PRINT "and press RECORD.":PRINT
4340 GOSUB 3300:GOSUB 3000:PRINT
4350 OPEN 1,1,1, "Test-Tape 1"
4360 FOR I=1 TO 100
4370 PRINT#1,CHR$(170):NEXT
4380 CLOSE 1:OPEN 1,1,1,"Test-Tape 2"
4390 FOR I=1 TO 100
4400 PRINT#1 CHR$(85):NEXT
4405 CLOSE 1
4410 PRINT:PRINT "Rewind the tape and press PLAY"
     PRINT:GOSUB 3200
4415 GOTO 4460
4420 GOSUB 2990
4430 PRINT TAB(13) "Read Only Test":PRINT
4440 PRINT "Place the test tape in the cassette drive"
4450 PRINT "and press PLAY"
4455 PRINT:GOSUB 3300:GOSUB 3000:PRINT
4460 OPEN 1,1,0,"Test-Tape 1"
4470 FOR I=1 TO 100
4480 INPUT#1,A$
4490 IF A$<>CHR$(170) THEN E=E+1
4495 NEXT
4500 CLOSE 1
4510 OPEN 1,1,0,"Test-Tape 2"
4520 FOR I=1 TO 100
4530 INPUT#1,A$
4540 IF A$<>CHR$(85) THEN E1=E1+1
4550 NEXT I
4560 CLOSE 1:PRINT
4570 PRINT TAB(3) "There were ";E;" High Value errors"
     :PRINT
```

Fig. 5-9. The Cassette Drive Diagnostic Module listing.

```
4580 PRINT TAB(3) "There were ";E1;" Low Value errors"
     :PRINT
4590 GOSUB 3200
4600 GOTO 50
```

Fig. 5-9. The Cassette Drive Diagnostic Module listing. (Continued from page 108.)

```
4290   GOSUB 3100
4300   GOTO 4260
```

When the user presses 1, the program branches to line 4310 for the Write/Read Test. Here, the screen is cleared with the Clear Screen subroutine, and the title is displayed at the top of the screen:

```
4310   GOSUB 2990:PRINT TAB(13)"Write/Read Test":PRINT
```

Lines 4320 through 4340 display the necessary user prompts:

```
4320   PRINT "Place a blank tape in the cassette drive"
4330   PRINT "and press RECORD.":PRINT
4340   GOSUB 3300:GOSUB 3000:PRINT
```

The program then falls through to line 4350, which opens the tape device and assigns the name of "Test-Tape 1" to the first file that is written with this test. "Opening" the tape device means that it is logically connected to the system:

```
4350   OPEN 1,1,1, "Test-Tape 1"
```

Line 4360 is the beginning of a FOR-TO loop, in which I is equal to 1 through 100. This serves to write 100 records onto Test-Tape 1:

```
4360   FOR I=1 TO 100
```

Line 4370 actually prints the record which will be written 100 times. The PRINT #1 statement instructs the program to direct the output to file #1, which is the cassette drive opened in line 4350. Its output is the character string 170, which is a pattern with every other bit on: 10101010.

```
4370   PRINT#1,CHR$(170):NEXT
```

At line 4380, the cassette drive is closed with the CLOSE 1 statement. It is opened again immediately, and given a new file name of "Test-Tape 2."

```
4380   CLOSE 1:OPEN 1,1,1,"Test-Tape 2"
```

Line 4390 begins another FOR-TO loop that goes from one to 100. This time, it's output is character string 85, which is a pattern with every other bit off: 01010101.

```
4390   FOR I=1 TO 100
4400   PRINT#1 CHR$(85):NEXT
```

Once the FOR-TO loop is complete, line 4405 closes the cassette drive:

```
4405   CLOSE 1
```

Line 4410 prompts the user to rewind the tape, press PLAY, and press any key to continue:

```
4410   PRINT:PRINT "Rewind the tape and press PLAY"
       PRINT:GOSUB 3200
```

Line 4415 sends the program to line 4460, the beginning of the Read Test:

```
4415   GOTO 4460
```

When the user selects 2 from the Cassette Drive Diagnostic Menu, the Read-Only Test is activated, beginning at line 4420. This test is also the second part of the Write/Read Test. Line 4420 itself clears the screen with the Clear Screen subroutine:

```
4420   GOSUB 2990
```

Line 4430 displays the title of the test, and the following lines print the necessary user prompts:

```
4430   PRINT TAB(13) "Read Only Test":PRINT
4440   PRINT "Place the test tape in the cassette drive"
4450   PRINT "and press PLAY"
4455   PRINT:GOSUB 3300:GOSUB 3000:PRINT
```

Line 4460 opens the cassette drive device in a read mode, rather than a write mode as in the first test. This is indicated by the 0 instead of a 2 in the second operand. It is looking to read a file called "Test-Tape 1," which was created by the Write/Read Test. This is also the point where the Write/Read Test branches after it has completed writing the test files:

```
4460   OPEN 1,1,0,"Test-Tape 1"
```

Line 4470 is a FOR-TO loop for 100 records. Instead of writing information to tape as before, it reads information from the tape, placing it into variable A$:

```
4470   FOR I=1 TO 100
4480   INPUT#1,A$
```

Line 4490 checks variable A$ for accuracy. If it is not equal to 170, which was the value written to Test-Tape 1, error counter E is set equal to one. From there, it is incremented with each error encountered through each pass of the FOR-TO loop.

After A$ is compared, the program falls through to the NEXT statement at line 4495, which sends the program back through another loop. This continues until 100 passes have been completed:

```
4490   IF A$< >CHR$(170) THEN E=E+1
4495   NEXT
```

4500 closes the cassette drive device:

```
4500   CLOSE 1
```

Line 4510 immediately opens the device again. This time the program looks for the Test-Tape 2 file created in the Write/Read Test:

```
4510   OPEN 1,1,0,"Test-Tape 2"
```

Line 4520 begins a FOR-TO loop, and again it reads from one to 100, inputting its data into variable A$. This time it compares the data with character string value 85. If A$ is not equal to 85, the error counter is incremented as before:

```
4520   FOR I=1 TO 100
4530   INPUT#1,A$
4540   IF A$< >CHR$(85) THEN E1=E1+1
```

Line 4550 sends the program back through the FOR-TO loop:

```
4550   NEXT I
```

Once the 100 passes through the loop are completed, line 4560 closes the cassette drive:

```
4560   CLOSE 1:PRINT
```

Line 4570 displays the number of high value errors encountered during the test, if any; while line 4580 displays the number of low value errors encountered:

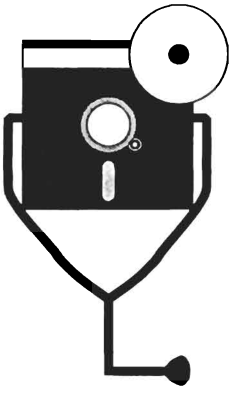
```
4570   PRINT TAB(3) "There were ";E;" High Value errors"
       :PRINT
4580   PRINT TAB(3) "There were ";E1;" Low Value errors"
       :PRINT
```

Line 4590 goes to line 3200, which prompts the user to press any key to end the test:

```
4590    GOSUB 3200
```

Finally, the program falls through to line 4600, which sends it back to line 50 to display the System Diagnostic Main Menu:

```
4600    GOTO 50
```



Chapter 6 The Disk Drive

Computer data is broken down into a series of ON and OFF bits: ones and zeros. Eight of these ON and OFF bits constitute a *byte*. Most computer dimensions are in terms of bytes, particularly the amount of data that can be stored in system memory, or the amount of data that can be stored on a diskette.

Every computer system unit is rated for a certain capacity of data it can store at one time within system memory. The capacity of the Commodore 64 is 64 kilobytes of data (approximately 64,000 bytes), while that of the Commodore 128 is 128 kilobytes of data.

This capacity is not large enough to hold software programs and all your data. Two or three application programs could easily use up this capacity, leaving no room for additional software or data with which to use the programs. The disk drive provides the additional storage necessary to supplement system memory.

DESCRIPTION AND FUNCTION OF THE DISK DRIVE

The disk drive provides a means for high-speed storage for programs and data, and as such, it is one of the most useful peripheral devices you can use with the Commodore (Fig. 6-1). With the disk drive, you can quickly store and access programs such as spreadsheets, databases, and word processors. The disk drive can also 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.

Throughout this chapter, the terms "disk," "diskette," "disk drive," and "drive" are used. *Disk* is the generic term for the floppy disk media on which the data is written. *Diskette* also refers to the floppy diskette media. *Disk drive* and *drive* are both generic terms for the floppy disk drive.

Types of Disk Drives

The Commodore systems can accommodate two different types of disk drives. The 1541 disk drive is a low-capacity drive which can store 175 kilobytes of data on a diskette. The 1571 disk drive is a high-capacity drive which can store twice as much: 350 kilobytes.

Both the 1541 and the 1571 are *external* disk drives; they reside in their own housing external to the system unit, and use their own power supply (Fig. 6-2).

The first computers with disk drives were the mainframe computers used in large corporations. The storage media used were 14-inch metal platters (Fig. 6-3). 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. 6-4). They were called "floppy" because, unlike their metal predecessors, the recording media consisted of a smaller platter made of thin, flexible plastic and encased in a protective cardboard covering. The technology produced by the microcomputer revolution miniaturized everything about the computer, including the disk drive.

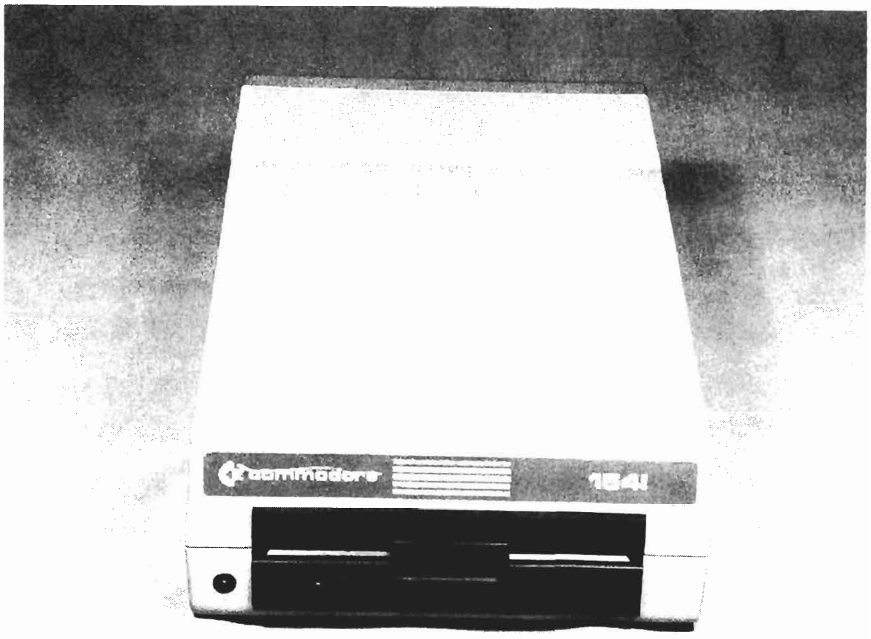


Fig. 6-1. The Commodore disk drive.

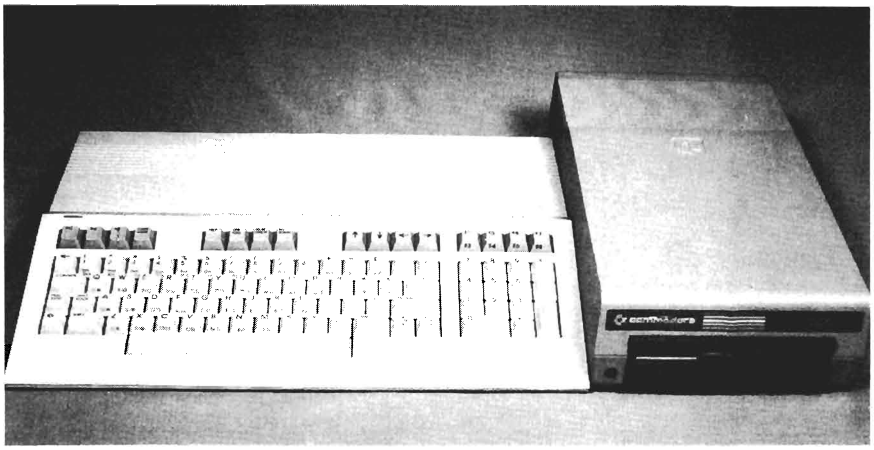


Fig. 6-2. The Commodore disk drive is external to the system unit.

The microcomputer's components continued to shrink in various ways. Among these components were the disk drive and diskette. Soon the 5¼-inch floppy diskette with its 5¼-inch disk drive became the standard (Fig. 6-5). These diskettes were smaller and easier to use than the eight-inch versions. The Commodore systems use these 5¼-inch floppy diskettes.

How a Disk Drive Functions

The disk drive attaches to the computer system unit by means of the disk interface built into the Commodore system's motherboard. The serial bus connector for this interface is located at the rear of the system unit and is where the disk drive cable is connected.

Up to four disk drives can be *daisy-chained* together (Fig. 6-6). This means that the first disk drive cable connects from the system unit to the first disk drive; the second connects between the first and second disk drive; the third connects

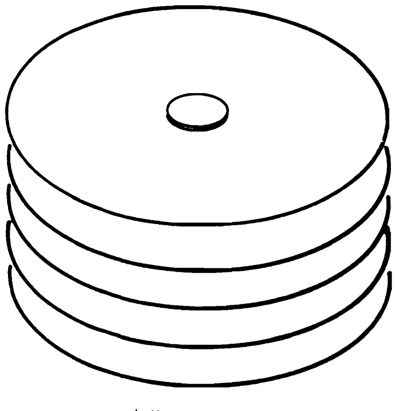


Fig. 6-3. The 14-inch mass storage platters.

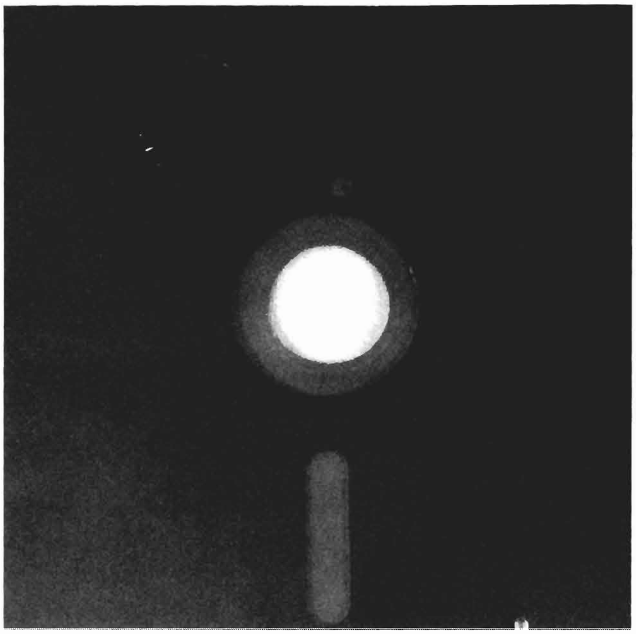


Fig. 6-4. The eight-inch floppy diskette.

between the second and third, and so on. Having multiple disk drives provides greater ease of use and versatility than just having one.

The disk drive includes a motor which spins the diskette at 300 to 400 revolutions per minute (rpm). The disk drive also consists of a head-arm assembly which locates, reads, and writes data to and from the disk according to the computer's internal program instructions (Fig. 6-7).

Fig. 6-5. A 5¼-inch floppy diskette.

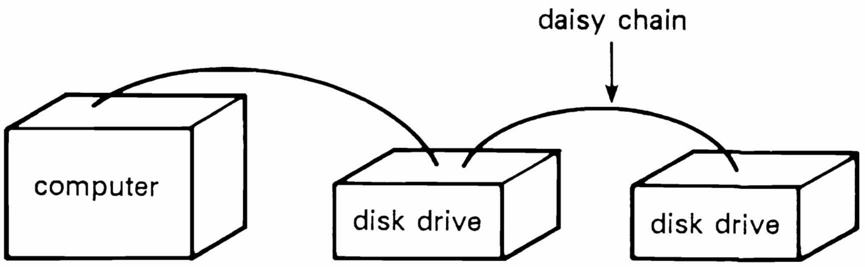


Fig. 6-6. Disk drive daisy chain.

The diskette is a magnetic recording medium made with iron oxide, similar to the material used on cassette tapes for a stereo cassette player. The magnetic material 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. 6-8).

The floppy diskette has a means for protecting the programs and data stored there from being overwritten or erased. The 5¼-inch diskette includes a small cutout on the side called the *write-protect notch.* As long as the write-protect notch is exposed, the disk drive is able to write information onto the diskette as well as read from it (Fig. 6-9). New boxes of diskettes always include a package of adhesive foil tapes. When one of these tapes is placed over a diskette's write-protect notch, the disk drive's ability to write information onto the diskette is "turned off." The tape blocks the write logic sensor in the disk drive, and no data can be written onto this diskette.

When you insert your diskette into the disk drive and close the drive latch, a switch inside the drive tells the system that a diskette has been inserted. The drive motor then spins the diskette.

There is a small hole located near the center of the diskette. This hole is used

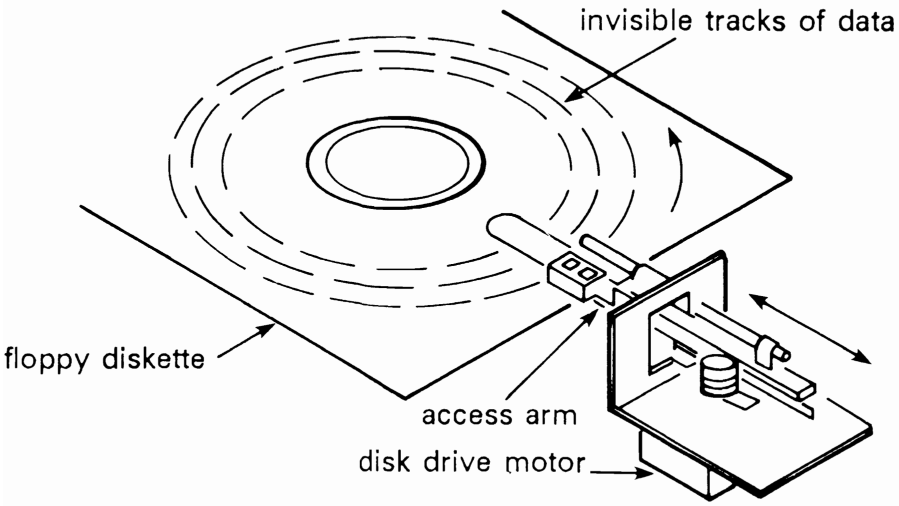


Fig. 6-7. Disk drive head-arm assembly.

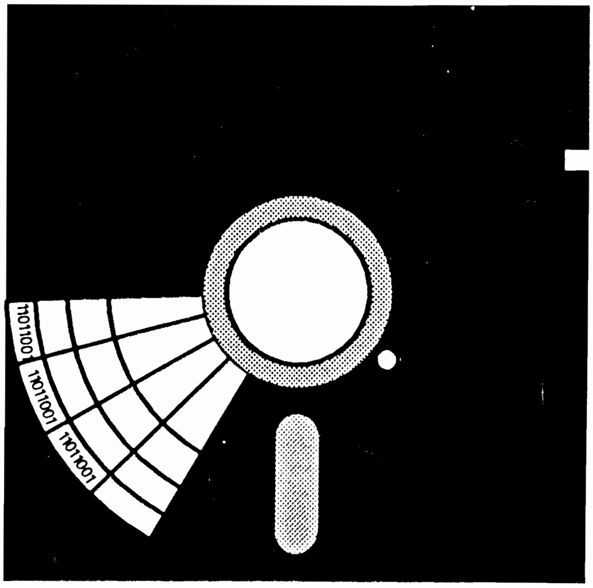


Fig. 6-8. Data written on the diskette.

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. It 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 (a *micron* is one millionth of a *meter*)

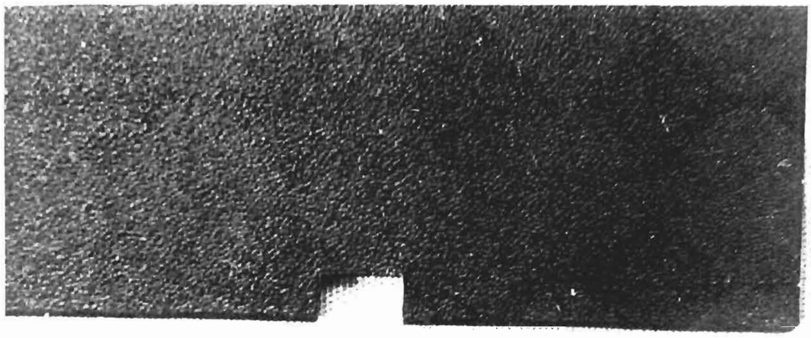


Fig. 6-9. Write-protect notch.

above the diskette on a cushion of air. Software and hardware logic 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 series of concentric circles. Each circle is a track. There are 35 tracks on a 175-kilobyte 1541 disk drive, and 70 tracks on a 350-kilobyte 1571 disk drive. When the diskette is formatted, each track is divided into sectors (Fig. 6-10).

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 sector. One sector on either of the Commodore disk drives contains 256 characters.

When a read command is issued by the operating system or the application program, the disk drive positions 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

Disk drive problems can be caused by the diskette, the disk drive, or the system unit/motherboard. To offset diskette problems, always maintain current backups of your software programs and data.

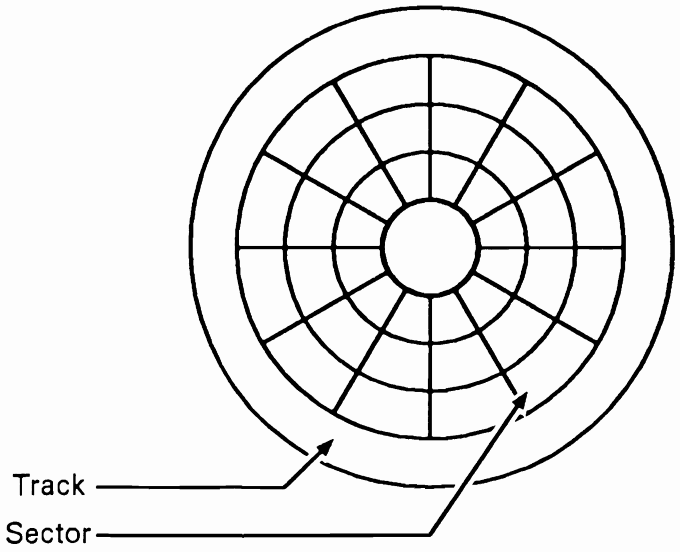


Fig. 6-10. Diskette tracks and sectors.

If your troubleshooting efforts point to a problem in the disk drive, take it in for servicing. The disk drive can require many precision adjustments that should be performed by computer repair professionals. If a disk drive problem turns out to actually be a problem with the motherboard, refer to Chapter 2 for system unit troubleshooting procedures.

NOTE: While running the System Diagnostic Program, if the disk drive encounters an error in writing or reading data, the error message is not automatically displayed. A drive error is indicated by the drive's red LED flashing on and off. If you see this condition, press RUN/STOP, then enter GOTO 5180. This returns the system to the location in the Disk Drive Diagnostic Module that displays the error code, the message, and the track and sector which contains the error.

Problem: The Disk Drive Does Not Work

If you insert a diskette into a disk drive but cannot read from or write to it, try the following solutions.

Solutions

- ☐ Check how the diskette is inserted. If it's inserted upside-down, the drive cannot read from or write to it. The diskette label should be face up.
- ☐ Make sure the diskette is formatted correctly for your system. For example, a diskette formatted for a 1571 cannot be used on a 1541 disk drive. (However, you can use a diskette formatted for a 1541 on a 1571 disk drive, although it would not be the most efficient use of the drive, because the diskette is only formatted for half of its potential storage capacity.)
- ☐ 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 executing a read or write operation. If the system boots up, the cursor is displayed on the screen, and the fan is running, 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 has its own power supply. First make sure that power is being supplied to the drive. Check that the drive is properly plugged in. If it is, try using a different electrical outlet. See whether the outlet being used for the drive is controlled by a light switch.
- ☐ If the power checks out but the drive's indicator still does not light, there is most likely a problem with the drive's power supply. Check the power supply's fuse located on the rear of the disk drive (Fig. 6-11). Pull the fuse out by inserting the end of a flatblade screwdriver into the fuse slot and turning it counterclockwise until the fuse holder is disengaged. Examine the fuse. If it is blown, replace it with another of the same rating. If the fuse is still good, take the drive in for servicing.

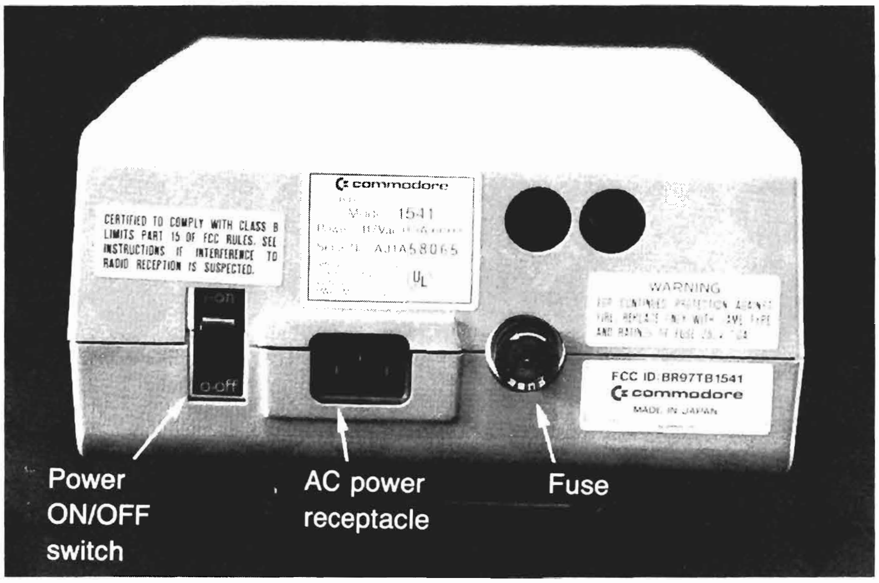


Fig. 6-11. Disk drive fuse location.

☐ If the operating system does not boot up, check that the disk drive interface cable is properly attached between the drive and the CPU board in the system unit.

☐ If your Commodore configuration includes more than one disk drive, there might be a problem with the daisy-chain arrangement that could be preventing the disk drive from getting power. Each disk drive must have its own unique address which is set internally in the drive itself. To set the drive's address, follow these steps:

1. Turn off the disk drive.
2. Disconnect the cables from the drive.
3. Turn the drive upside-down.
4. Remove the four phillips screws from the bottom of the drive (Fig. 6-12).
5. Hold the case together and turn it rightside-up again.
6. Remove the top of the disk drive housing.
7. Remove the two screws from the side of the metal housing shield (Fig. 6-13).
8. Remove the metal housing shield.
9. Locate the address jumper block on the printed circuit board (Fig. 6-14).
10. Refer to Table 6-1 to determine which jumper should be cut to give the disk drive its proper address for its place along the daisy chain.
11. Use a small pair of diagonal wire cutters to cut the jumper(s) as needed.
12. Replace the metal housing.
13. Replace the upper case of the disk drive and reinstall the phillips screws.
14. Reconnect the disk drive cable.
15. Test the disk drive again to see if this solved the problem. If not, take the drive in for servicing.

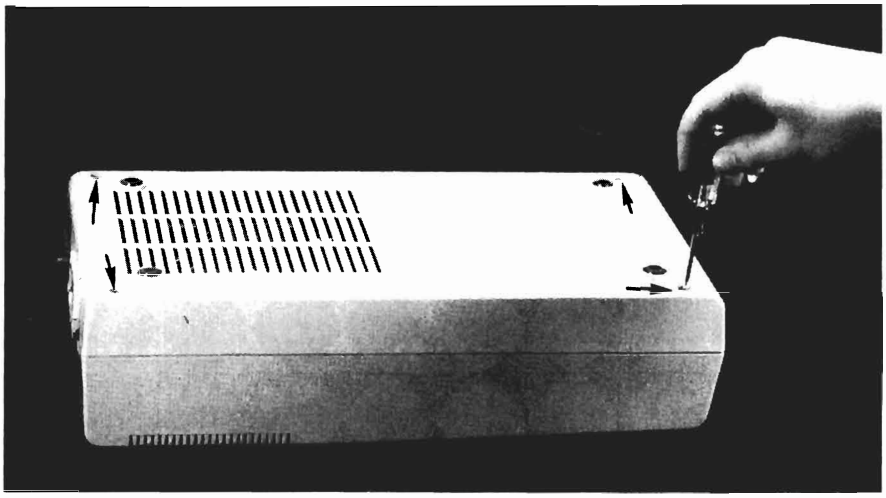


Fig. 6-12. Removing the drive screws.

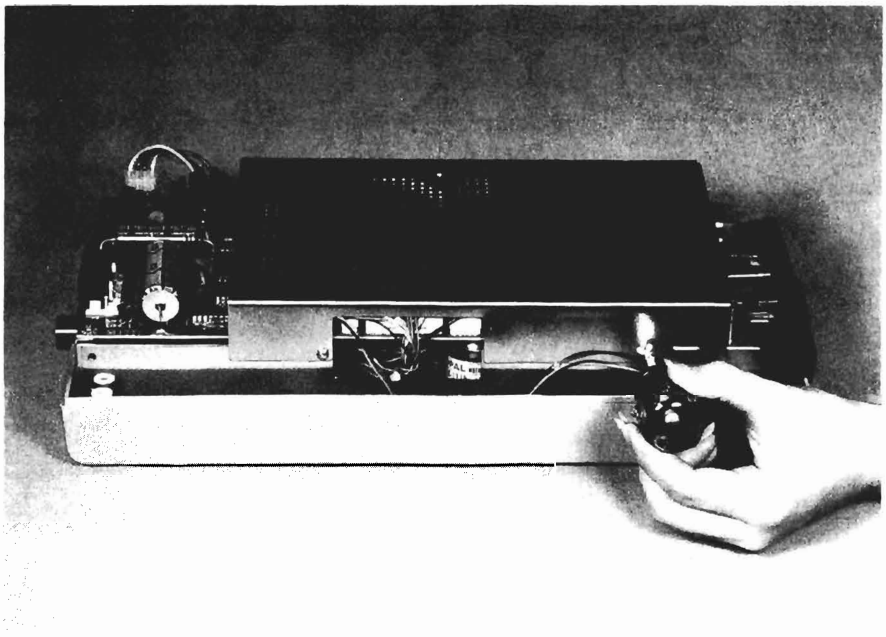


Fig. 6-13. Removing the metal housing.

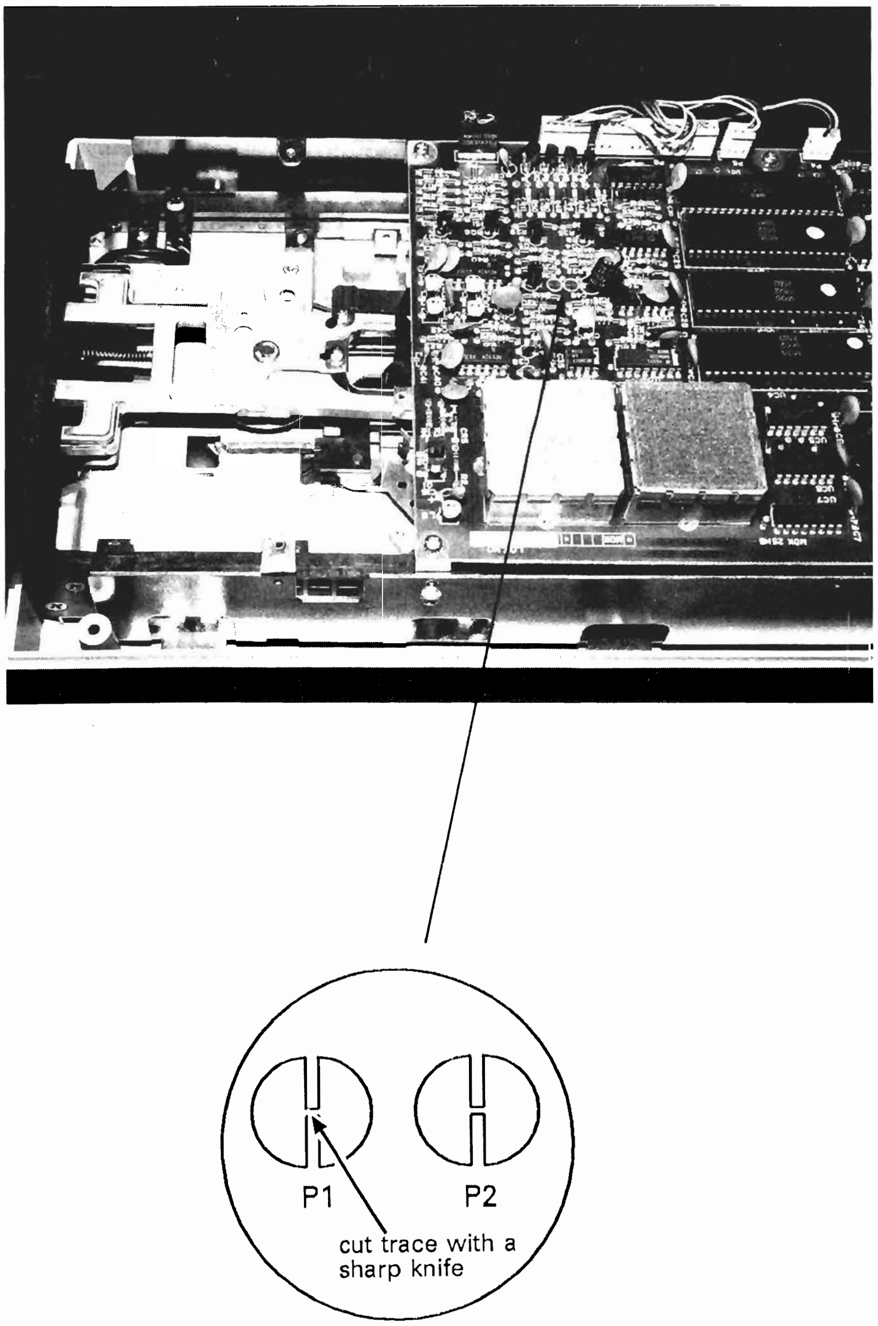


Fig. 6-14. Address jumper block.

Table 6-1. Disk Addresses.

Device Address	P1	P2
8	Intact	Intact
9	Cut	Intact
10	Intact	Cut
11	Cut	Cut

Problem: Cannot Read or Write to the Diskette

A common disk drive problem is that the system cannot read data from or write data to a diskette. In this case, the first thing to check is the diskette itself.

☑ Solutions

- ☐ If the system is not writing, and you are using a brand new diskette, first make sure that it is the correct type of diskette for your drive. There are hard-sectored and soft-sectored, single-density and double-density, single-sided and double-sided diskettes.

- ☐ If the diskette is the correct type for the drive, the problem might be that the diskette is not formatted. There are many diskette manufacturers, and the same make of 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 all new diskettes in order for them to work with your computer.

  The format program places operating system information and a disk directory program onto the new diskette. The format program also tests the diskette for bad sectors. If any bad sectors are found, the format program blocks them out. The diskette can still be used, and valuable data is prevented from being written onto bad areas.

- ☐ If the system is able to read from a diskette, but cannot write to it, or if you've run the Disk Drive Diagnostic Module and the 26 Write Protect On error message was returned, look at the diskette and see whether it is write-protected.

- ☐ If you've been using a particular diskette, and now find that the computer can no longer 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 continuous use. This is one reason why keeping a current backup copy of each diskette is so important. If a diskette wears out, and the message 25 Write Error is displayed, simply use its backup diskette.

- ☐ If you still cannot read from or write to the diskette, try a different diskette, one that you have recently used and know is not defective. If the system has no problem with this diskette, the original diskette you were trying to use was defective. Use its backup.

☐ If you cannot read from or write to the second diskette, there might be a problem with the drive. To check this, try the diskette on another system. If it does not read or write on that system, this indicates a bad diskette. However, if it does read and write on the other system, your disk drive is probably faulty.

☐ To begin troubleshooting the disk drive, run the Disk Drive Diagnostic Module on 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 two or three error messages are returned, a bad sector on the disk is probably the cause. When a new diskette is formatted, all bad sectors are blocked out. Sometimes a sector can go bad after formatting. In this case, the disk can be salvaged with reformatting. Reformatting lets you start over with a "fresh" disk, and will block out the bad sectors. Remember, however, that when a diskette is reformatted, all data on it is erased. You cannot reformat without losing the data on the diskette.

If the diagnostic test returns a lengthy series of read errors, either the disk drive or the drive controller read circuit on the motherboard is probably 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 on the motherboard is fine. The problem lies in the disk drive, which should be taken in for servicing.

Problem: Cannot Load a Program or Read a Data File

If attempts to boot up the system only caused a read error message to be returned, there could be a problem with the disk drive's read circuitry.

☑ Solutions

☐ If the system includes a second disk drive, place the program diskette in that drive, and try reading the program from there. If the system still cannot read from the other drive, this indicates a bad program diskette. Load the backup diskette.

If the disk drive does load, load the Disk Drive Diagnostic Module. Run the diagnostic on the drive that would not load the program. If the test runs successfully, and the successful test completion message is returned from the diagnostic, then nothing was wrong with the disk drive. It was probably just a problem with the diskette. Try a different diskette, and it should work.

If the diagnostic test fails, this indicates that the disk drive is defective, and it should be taken in for servicing.

☐ If the system cannot boot up, read data from or write data to any of the disk drives, then the problem almost certainly resides on the disk drive interface circuit within the system unit, and not on the disk drive itself. Refer to Chapter 2 to troubleshoot the motherboard.

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, a result of the disk drive being shaken or dropped. A head crash can also be caused by contamination on the disk or in the drive (Fig. 6-15).

If the head crashes, there is a grinding noise, and the disk is scratched. This can be catastrophic to the 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 a warped diskette is placed 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, the disk drive will go out of adjustment and it will have to be taken in for servicing.

If the system cannot read from, write to, or format the disk, this indicates a head crash. Another indication is a scratchy hissing sound heard when a diskette is inserted into the drive.

☑ Solution

- ☐ When the head crashes, the drive must be taken in for servicing. 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. Because diskettes are magnetic media, 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 motor's magnetic field could garble the data.

Although the 5¼-inch diskette is enclosed in 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, the emulsion could be altered, and data lost.

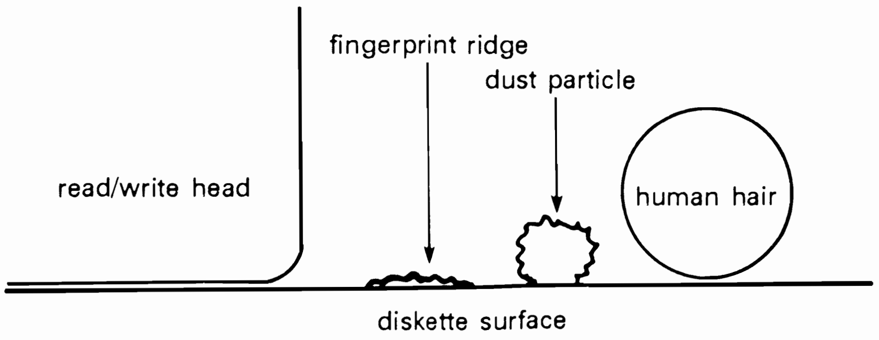


Fig. 6-15. Diskette contamination.

Problem: 20 Read Error

If the 20 Read Error message is displayed, this indicates that the disk drive cannot locate the header portion of a sector that the system is trying to access. This means that it's trying to access a nonexistent sector, or that the sector header is damaged in some way. 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. This could indicate either a defective diskette or a defective disk drive.

☑ Solutions

- ☐ Try to read a known, good diskette, perhaps the test diskette, in the same drive. If the drive can read this diskette, it means the first diskette is defective. If the drive still cannot read the diskette, it means the disk drive is probably defective.
- ☐ Try to read the diskette in a different drive, if available. If the diskette can be read in the different drive, it means the first drive is defective. If the diskette cannot be read in the different drive, it means the diskette media is defective.

Problem: 21 Read Error

If the 21 Read Error message is displayed, the drive cannot find a *sync marker* for the track that the system was trying to access. The sync marker is a starting point for the diskette. It consists of data placed on the diskette by the formatting process. This error indicates either that the diskette is not formatted, or that the disk drive does not contain a diskette. The drive door might have been inadvertently closed without the diskette being inserted.

☑ Solutions

- ☐ Check that there is indeed a diskette in the drive.
- ☐ Insert another diskette that you know is formatted, perhaps the test diskette. Run the Disk Drive Diagnostic again. If the diagnostic runs without errors, the original diskette probably was not formatted. If the diagnostic returns the same error message, the disk drive might be out of alignment. Take it in for servicing to have it realigned.

Problem: 23 Read Error

If the Disk Drive Diagnostic returns the message 23 Read Error, this indicates a *checksum* error, which occurred when reading data into internal memory. A checksum is a method by which the computer internally tests the information to ensure that it has been read properly. A checksum is actually data that accompanies the data being read. The checksum is used for comparison purposes; it compares the data being read in from the diskette to the data that is being transferred to system memory. This should be the same data. If it is not, then a checksum error is returned.

☑ Solution

- ☐ If this error message occurs frequently, there is probably a problem with grounding. Make sure that the diskette drive is properly grounded with a grounded plug.

Problem: 24 Read Error

If the Disk Drive Diagnostic returns the 24 Read Error message, a hardware error has occurred while the drive was reading data or header information from the diskette into system memory.

☑ Solution

- ☐ This indicates a problem with electrical grounding. Make sure that the diskette drive is properly grounded with a grounded plug.

Problem: 25 Read Error

If the Disk Drive Diagnostic returns the 25 Write Error message, there is a data mismatch. When information from system memory is written to a disk, it goes through a process called *verification*. In this process, the data being written is immediately read back and compared to the data in system memory. If the two sets of data do not match, this error message is returned.

☑ Solution

- ☐ If this error occurs frequently on the same diskette, there is probably a disk media problem. Using a backup of the diskette or a new diskette will solve the problem.

Problem: 26 Write Protect On

If the 26 Write Protect On error message is returned while running the Disk Drive Diagnostic, this indicates that the system is trying to write information onto a diskette that is write-protected.

☑ Solution

- ☐ Remove the diskette from the drive, then remove the adhesive write-protect tape from the edge of the diskette. Insert the diskette back into the drive, and the system will now be able to write information onto the diskette. Remember, however, that you probably originally placed the write-protect tape on the diskette in order to protect data already written on it. Make sure the system does not overwrite valuable data, now that you have write-enabled the diskette.

Problem: 27 Read Error

The 27 Read Error message indicates that a checksum error (see the earlier discussion of the 23 Read Error) occurred while the drive was reading the header of a particular sector.

☑ Solution

- ☐ Check the electrical grounding of the disk drive. Make sure that the diskette drive is properly grounded with a grounded plug.

Problem: 28 Write Error

The 28 Write Error message indicates that after the drive had written data to the diskette, it was not able to locate the sync mark for the next sector. This can indicate a mechanical problem in the drive motor or with the head positioning mechanism. This could also indicate an improperly formatted diskette.

☑ Solutions

- ☐ Try writing to another diskette that you know is formatted properly. Run the Disk Drive Diagnostic.
- ☐ If the same error message reoccurs, the disk drive might be defective, and it should be taken in for servicing.

Problem: 62 File Not Found

If the 62 File Not Found message is returned, this means that the drive could not locate the file being requested.

☑ Solutions

- ☐ Run the Disk Drive Diagnostic on the diskette. If this message is returned again, there is a problem with the diskette media itself.
- ☐ Run the diagnostic with a different diskette, preferably a new one. If the diagnostic does not detect the error, this means the original diskette was defective. If the error message is returned again, this means the disk drive is probably defective, and it should be taken in for servicing.

Problem: 66 Illegal Track and Sector

If the 66 Illegal Track and Sector message is returned, this indicates that the drive is trying to access a nonexistent track or sector on the diskette. This could happen particularly if you indicate that the drive is a 1571 drive when it is actually a 1541 drive. Because the 1571 has a higher capacity than the 1541, the test evidently attempted to access a track and sector beyond the limits of the drive.

☑ Solutions

- ☐ Run the Disk Drive Diagnostic Test again and make sure that you specify the proper drive and diskette type (1541 or 1571).
- ☐ If the error message is still returned, there might be a problem with the disk drive. Swap the drive on another system unit. If the drive fails on another system, it is defective and should be taken in for servicing.

- ☐ If the drive works on the other system, there is probably a problem with the system unit. Swap a different disk drive onto the system. If the new drive does not work either, refer to Chapter 2 to troubleshoot the system unit.

Problem: 74 Drive Not Ready

If the 74 Drive Not Ready error message is returned, it means that the drive contains no diskette, or that the drive door is open. It could also mean that the diskette is not formatted.

☑ Solutions

- ☐ Check to see if the drive contains a diskette.
- ☐ If it does, make sure the drive door is closed.
- ☐ If the error message persists, check whether the diskette is formatted or not. The easiest way to do this is to try another diskette that you know is formatted.

PREVENTIVE MAINTENANCE

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 head mechanism.

Dust can be a big enemy to the disk drive motors. Dirt can collect on the head mechanism and cause the head to crash. Covering the system with a dust cover protects disk drives from dust and dirt.

When moving the 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.

Obtain a disk drive head cleaning kit from a computer store. Insert the special diskette, which is permeated with a cleaning solution, into the drive. Enter a read command to cause the diskette to rotate. The read/write heads are then cleaned as they pass over the cleaning diskette.

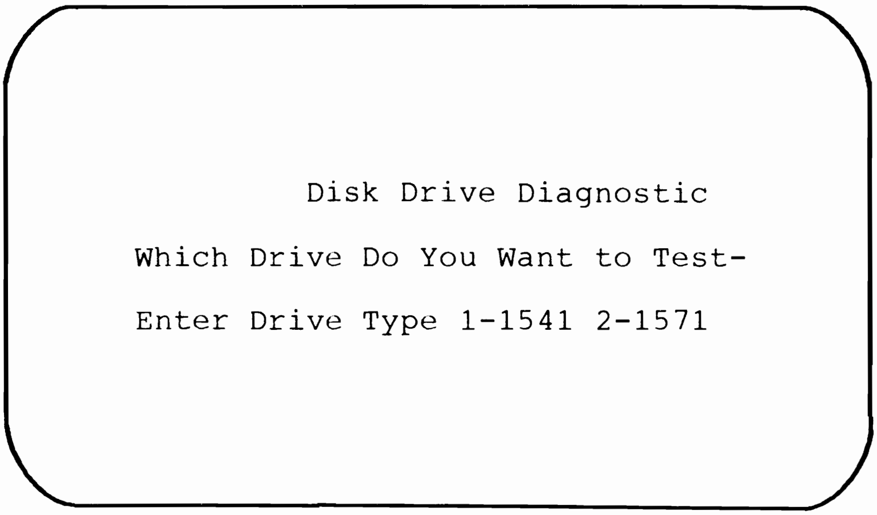
RUNNING THE DISK DRIVE DIAGNOSTIC MODULE

To run the Disk Drive Diagnostic Module, follow the instructions provided in "Running the Diagnostic Program" in Chapter One. Then press D to initiate the Disk Drive Diagnostic Module (Fig. 6-16).

This diagnostic checks the disk drive and 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 either the 1541 or 1571 disk drive, and you are prompted to enter which one the system uses.

When running the test, it is best to use a freshly formatted, empty diskette. If necessary, a diskette containing data may also be used, although this would not constitute as thorough a test. 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.



```
              Disk Drive Diagnostic

   Which Drive Do You Want to Test-

   Enter Drive Type 1-1541 2-1571
```

Fig. 6-16. The Disk Drive Diagnostic Module.

Because the information already residing on the disk is not tested, it cannot be damaged or wiped out. However, as always, make sure you have complete current backups of all programs and files, especially those residing on any diskette being tested.

In response to the prompt, indicate the number of the drive to be tested: 8 to 15. The diagnostic creates a file called TEST, and makes four distinct passes through the disk.

The first pass writes a series of 254-byte records that contain a 254-character string of CHR$ value 170. CHR$ value 170 is a character that has every other bit 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 CHR$ 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 254-character records of CHR$ 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 CHR$ value 170. When the diskette is full, the file is again closed.

CHR$ 170 has every odd bit turned off, and every even bit turned on. CHR$ 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.

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 CHR$ 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 the user presses D or d from the System Diagnostic Main Menu, the program branches to line 5000. A REMARK statement in the program marks the beginning of the Disk Drive Diagnostic Module (Fig. 6-17).

```
5000 REM: "Disk Drive Diagnostic Module"
```

Lines 5010 and 5020 initialize two variables, HV$ for high values, and LV$ for low values. HV$ builds a string of the high CHR$ value 170. The loop continues eight times, essentially building an eight-character record:

```
5010 FOR I=1 TO 8:HV$=HV$+CHR$(170):NEXT
5020 LV$="":FOR I=1 TO 8:LV$=LV$+CHR$(85):NEXT
```

Line 5020 does the same thing, building a string of the low CHR$ value 85 eight times.

Line 5030 clears the screen and displays the diagnostic title. Line 5035 then displays a prompt for the user to enter the disk drive to be tested. The user would enter a value such as 8 to designate the first drive, 9 to designate the second drive, or any drive designation up through 15. This drive designation number is placed in variable D:

```
5030 GOSUB 2990:PRINT TAB(10)"Disk Drive Diagnostic":PRINT
5035 INPUT "Which Drive Do You Want To Test-";D:PRINT
```

Line 5036 checks that a valid drive number selection has been made. If an invalid drive number has been entered, the program displays an error message and loops back to line 5035 for the user to try again:

```
5036 IF D<8 OR D>15 THEN PRINT "Invalid Disk Number":
     GOTO 5035
```

Line 5037 prompts the user to specify the drive type. A value of one indicates the 1541 drive, and a two indicates the 1571 drive. If a value other than one or two is entered, line 5038 causes the program to default to the 1541 drive:

```
5037 INPUT "Enter Drive Type 1-1541 2-1571";T
5038 IF T< >2 THEN T=1
```

Line 5040 sets variable BF$ equal to HV$. The name variable NM$ is initialized to the character literal "High Values":

```
5040 BF$=HV$:NM$="High Values"
```

```
5000 REM: "Disk Drive Diagnostic Module"
5010 FOR I=1 TO 8:HV$=HV$+CHR$(170):NEXT
5020 LV$="":FOR I=1 TO 8:LV$=LV$+CHR$(85):NEXT
5030 GOSUB 2990:PRINT TAB(10)"Disk Drive Diagnostic":PRINT
5035 INPUT "Which Drive Do You Want To Test-";D:PRINT
5036 IF D<8 OR D>15 THEN PRINT "Invalid Disk Number":
     GOTO 5035
5037 INPUT "Enter Drive Type 1-1541 2-1571";T
5038 IF T<>2 THEN T=1
5040 BF$=HV$:NM$="High Values"
5050 FOR I=1 TO 2
5060 OPEN 1,D,2,"TEST,s,w"
5070 PRINT:PRINT"Writing ";NM$;" to Disk"
5072 IF T=2 THEN Z=33600
5074 IF T=1 THEN Z=16800
5075 FOR A=1 TO Z
5077 B1$=""
5080 PRINT#1,BF$
5090 NEXT A
5095 CLOSE 1
5100 OPEN 1,D,2, "TEST,s,r"
5110 PRINT "Reading ";NM$;" from Disk":PRINT
5115 FOR A=1 TO Z
5120 INPUT#1,B$
5140 IF B$<>BF$ THEN PRINT "Error Reading ";NM$:E=E+1
5150 NEXT
5160 CLOSE 1
5165 OPEN 15,8,15,"s0:TEST":CLOSE 15
5170 BF$=LV$:NM$="Low Values":NEXT I
5180 PRINT:PRINT "There were ";E;" errors encountered"
5182 OPEN 15,8,15:INPUT#15,DS,DS$,T,S:PRINT:PRINT DS;
     ",",DS$+",";T;",";S:PRINT
5183 CLOSE 15
5185 IF EX=1 THEN RETURN
5190 GOSUB 3200
5200 GOTO 50
```

Fig. 6-17. The Disk Drive Diagnostic Module listing.

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

Line 5050 is the beginning of a FOR-TO loop. The test runs through two passes: one for the high values of CHR value 170, and one for the low values of CHR value 85.

```
5050 FOR I=1 TO 2
```

The FOR-TO loop allows the same code to be used in both cases, simply inserting the different variables as appropriate.

The program proceeds to line 5060. The disk file is opened using the disk number that was stored in variable D in line 5035. The file is named "TEST," and is opened in the output mode as a sequential file:

```
5060 OPEN 1,D,2,"TEST,s,w"
```

Line 5070 prints a blank line and then displays an informational message indicating that the contents of NM$, which was initialized in line 5040 to "High Values," is being written to disk:

```
5070 PRINT:PRINT"Writing ";NM$;" to Disk"
```

Line 5072 determines the number of records to write, based on the value of T. If T equals two, then variable Z is set to 33600, which fills up all available free sectors on a 1571 diskette. Otherwise, Z is set to 16800, the number of eight-byte records needed to fill up the sectors available on a 1541 diskette:

```
5072 IF T=2 THEN Z=33600
5074 IF T=1 THEN Z=16800
```

Line 5075 causes 33600 or 16800 records to be written to the dsk. This completely fills up the diskette:

```
5075 FOR A=1 TO Z
5077 B1$=""
```

Line 5080 is where the actual output to disk takes place. The PRINT #1 statement from variable BF$ writes the high values out to the disk:

```
5080 PRINT#1,BF$
```

Line 5090 is the NEXT statement which serves to loop the program back to the Write Test code:

```
5090 NEXT A
```

Line 5095 closes the file after all records have been written:

```
5095 CLOSE 1
```

Line 5100 reopens TEST, which is the name designated for the file created in the Write Test:

```
5100  OPEN 1,D,2, "TEST,s,r"
```

Line 5110 displays the message indicating that the high values of NM$ are being read from the disk:

```
5110  PRINT "Reading ";NM$;" from Disk":PRINT
```

Line 5115 begins a FOR-TO loop which causes the program to read the 33600 or 16800 records, depending on the drive type:

```
5115  FOR A=1 TO Z
```

Line 5120 inputs the data being read from the disk into variable B$:

```
5120  INPUT#1,B$
```

When all the data have been read, the program branches to line 5160 to end the test.

If the end-of-file has not yet been reached, the program falls through to line 5140, which compares variable B1$ to variable B$. B1$ contains the high values read from the disk in line 5120, while B$ contains the high values written into the disk in line 5080:

```
5140  IF B$< >BF$ THEN PRINT "Error Reading ";NM$:E=E+1
```

In other words, this statement compares the data that was written in the Write Test to the data that was read in the Read Only Test. They should be exactly the same. If there is a discrepancy between the two, a message is displayed, indicating a high-value error reading.

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.

The program proceeds to line 5150, a NEXT statement to increment the FOR-TO loop begun in line 5115:

```
5150  NEXT
```

There was an end-of-file trap at line 5130. When the end of the file is reached, the program branches here to line 5160 to close the test file:

```
5160  CLOSE 1
```

The CLOSE 1 statement closes the disk drive.

Line 5165 erases the high-values TEST file from the disk drive:

```
5165  OPEN 15,8,15,"s0:TEST":CLOSE 15
```

Line 5170 sets BF$ to CHR$(85) and reads the disk for the low values:

```
5170  BF$=LV$:NM$="Low Values":NEXT I
```

Once the FOR-NEXT loop is completed, the program proceeds to line 5180.

Line 5180 displays a summary of the number of read errors encountered during the test, such as "There were 2 errors encountered," or better yet, "There were 0 errors encountered:"

```
5180  PRINT:PRINT "There were ";E;" errors encountered"
```

If read errors are returned, there could be a problem with either the diskette or the disk drive itself. Refer to "Troubleshooting and Repair Guidelines" earlier in this chapter for further action.

Line 5182 displays the error code, error message, faulty track and sector for any error encountered in the course of this test:

```
5182  OPEN 15,8,15:INPUT#15,DS,DS$,T,S:PRINT:PRINT DS;
      ",",DS$+",";T;",";S:PRINT
5183  CLOSE 15
```

If a read or write error takes place during this test, the error message does not display automatically. Such an error is indicated by the disk drive light flashing on and off. If this happens, press the RUN/STOP key on the keyboard, then enter GOTO 5182. The System Diagnostic Program resumes at this point, and line 5182 displays the error message and information.

Line 5185 checks for the EX flag. If it equal to one, this indicates that the Exerciser Module is in effect, in which case the program returns to the Exerciser Module portion of the code. If it is equal to zero, the program falls through to line 5190:

```
5185  IF EX=1 THEN RETURN
```

Once the test is complete, it prompts the user to enter any character to continue and then branches back to line 50, the beginning of the System Diagnostic Main Menu:

```
5190  GOSUB 3200
5200  GOTO 50
```

Chapter 7 The Serial Communication Interface

With the proper equipment, the Commodore can "talk" to many other devices: a modem, a printer, a mouse—even another computer. Data and information can be sent to and received from these peripherals. Communicating among different devices requires the services of the serial communication interface.

DESCRIPTION AND FUNCTION OF THE SERIAL COMMUNICATION INTERFACE

The serial communications port is a 25-pin edge connector located on the back of the Commodore system. An additional interface, such as the VIC-MODEM or Commodore RS-232 cartridge, must be plugged into this connector in order to use the port as a modem or printer interface.

The interface reaches the outside world by means of a cable to the other device with which communication is to take place. It is through this cable that the appropriate signals and data are passed to and from the other device.

Types of Serial Communication Devices

The Commodore serial communication interface is used to communicate with devices such as the modem, printer, and another computer.

Interfacing with a Modem. The modem is a device widely used for serial communication. It allows transmission and reception of computer data 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.

The modem converts this digital signal into sound waves, or *analog* signals for transmission over telephone lines. This part of the conversion is referred to as *modulation.*

The modem at the receiving end then converts the analog signal back into a digital signal, and this part of the conversion is referred to as *demodulation.* The information once again takes the form of ones and zeros, so the computer connected to the receiving modem can successfully interpret it (Fig. 7-1).

Modems can only operate in pairs. For two computers to communicate with one another, each computer must be connected to a modem that has matching configurations. This would include communication speed, data size, start bits, and other parameters that are discussed later in this chapter.

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 the networks' centers of information. By coupling the modem to a telephone, then dialing a specific telephone number (which is often a local or toll-free number), the Commodore can be linked into a network's mainframe computer. You can then use the network service to research a wide range of information, as well as to communicate with other users. Online services can also provide banking, shopping, and airline reservation services through the use of the Commodore's serial communication interface and the modem (Fig. 7-2).

The Commodore systems have two different types of modems available. One is the RS-232 modem, and the other is the VIC-MODEM. The VIC-MODEM connects into one of the game ports on the right side of the system unit. An RS-232 modem uses the RS-232 cable, and connects to the serial port at the rear of the system unit.

Interfacing with a Printer. The *serial bus interface* is most commonly used for the printer on a Commodore system. A serial bus is a daisy-chained, high-

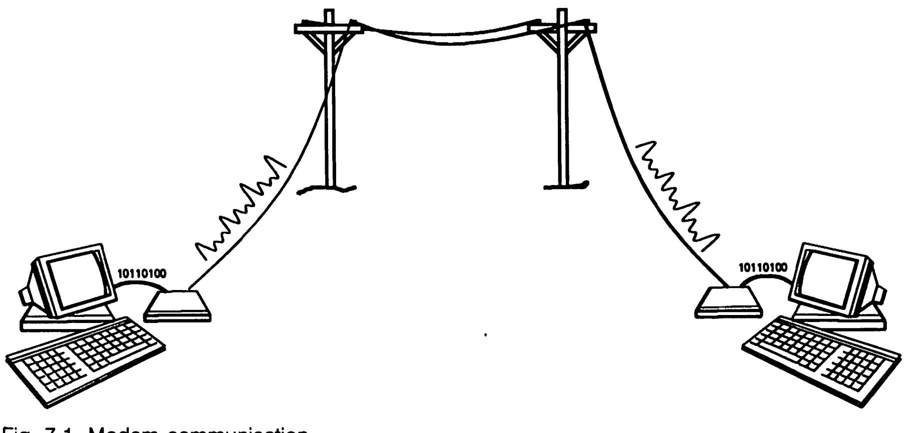


Fig. 7-1. Modem communication.

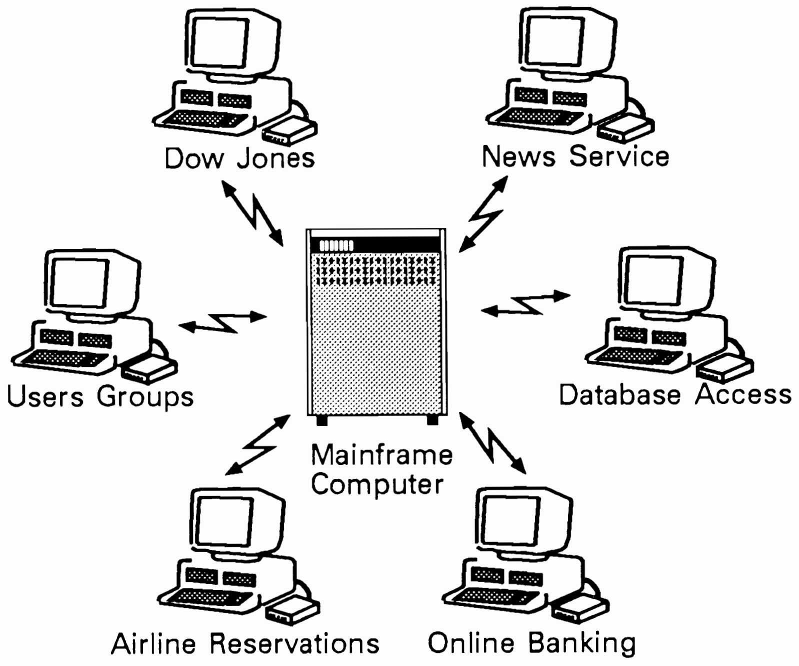


Fig. 7-2. Online services.

speed serial interface. It is located on the back of the Commodore system unit, and allows daisy-chaining of up to eight devices. It is used to interface disk drives and Commodore-supplied printers.

However, if you have a serial printer using the serial communication interface, it would be cabled to the serial communication port instead of the serial bus port. The printer is typically attached to the computer with an interface cable, or *hard wire*.

Just like with the modem, either type of printer must be configured with the same data speed and size parameters that are set within the Commodore. This is because the printer must know what to expect from the computer, and how to handle it.

In some cases, 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.

Interfacing Directly with Another Computer. If your Commodore resides in an environment in which there are computers in relatively close proximity (less than 50 feet), these computers can be hard-wired together 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 phone lines are not needed in a hard-wire connection because of the close proximity of the computers.

When computers are hard-wired together, data can be sent 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 diminish and are lost due to the resistance of the wire through which the signals are traveling.

How the Serial Communication Interface Functions

Serial communication is a function of timing and speed. Several factors contribute to keeping the communicating devices synchronized with one another so that they can interpret 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 the serial communication interface, communication software that works from the serial communication interface is needed. An example of this kind of software is a modem communication program. If you subscribe to an online network service such as Dow Jones, The Source, or CompuServe, their user guides explain the communication software packages that work properly with their devices.

Communication Parameters. Because data is sent one bit at a time in serial communication, certain parameters must be specified when first setting up communication software. These parameters pertain to data transmission speed, the size of the data characters being sent, whether start and stop bits are 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 the computer is communicating. Online service user documentation details the particular parameters needed.

The printer, modem, or other device with which the Commodore is going to communicate must have the communication interface variables set identically on their end in order for the communication link to be established and maintained. These parameters are set either through the software or with switches on the communication interface board. Check the device documentation for specific details.

When you are running communication software for the first time, access the setup mode to set the communication parameters. Once these parameters are set, they can usually be stored on the program disk so that they are automatically set each time the program is used.

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 are 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 usually the number of bits that can be transmitted in a second. The differences in baud rates are analogous to the different ways a car horn might be sounded. Three short beeps can be heard much faster than three long beeps.

In the same way, 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 of which both devices are aware, the system knows how long each ON bit lasts, and how long each OFF bit lasts. Most communication interfaces for personal computers have a baud rate of 300 or 1200 bits per second.

Higher-speed communication, in the range of 4800 and 9600 bits per second, is available, but is used primarily for communicating with devices that are hard-wired to a system through an interface cable. A hard-wire setup does not require an intermediary such as a modem in order to establish communication. A printer, or even a computer that is hard-wired to another computer, is able to operate at the faster speeds. There are modems available that can run at such high speeds, but they are prohibitively expensive, especially for the small business or home user.

Data Size. Another parameter, *data size,* refers to the size of the data being sent: how many bits make up a standard character or word. A standard character of data consists 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 that it knows what to expect (Fig. 7-3).

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.

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 or each (Fig. 7-4).

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*).

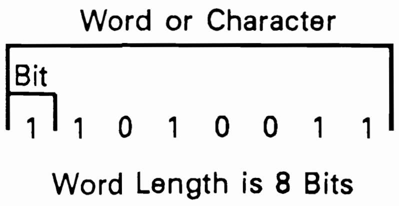


Fig. 7-3. Bits, characters, and words.

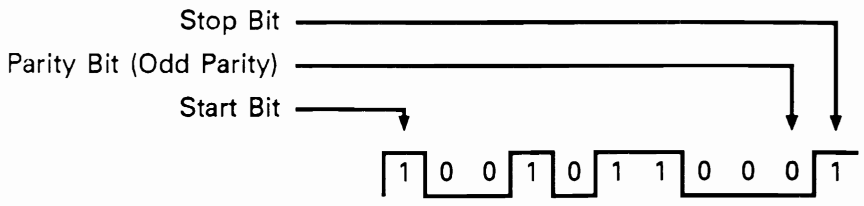


Fig. 7-4. The stop bit, start bit, and parity bit.

For example, if the system is set up with odd parity from the modem to the computer, and a character or word is received with an even number of bits, parity is not correct. This might mean that one of the character bits has dropped off during transmission. A failure of parity indicates an error in transmission, and the communication software either prompts you to try the data transmission again, or it restarts it automatically for you.

Directionality. Depending on the device with which the Commodore is communicating, the data might be sent in one direction only, or data might be sent in both directions—to and from both devices. Printers are *unidirectional* devices, meaning they only receive data; they cannot transmit data back to the computer. When two computers are linked up through modems, on the other hand, the data is said to be *bidirectional,* because data can be sent and received to and from both systems.

Whether the data is unidirectional or bidirectional depends on the type of device to which the data is being sent, as well as the software being used for the communication transfer. Some parameters in the support software might need to be set. There might also be switches to set on the serial communication interface on the motherboard. This is 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.

To solve this problem, an emulation program in the communication software can disguise the Commodore and make it "pretend" that it is another computer of a different design. The DEC VT100 is a common emulation mode, available with communication software, in which the Commodore 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 can indicate whether or not this protocol is 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 Commodore has a 2000-byte communication buffer. In the process of receiving a file that may be 16,000 bytes long, the communication buffer is quickly filled up. When this happens, the Commodore sends a "suspend" command to the other system, temporarily stopping communication transfer. This suspension is the XOFF command. When the buffer is emptied through computer processing, such as displaying characters on the screen, the Commodore 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, 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. 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 ensure understanding.

If one person talks much faster than the other is able to comprehend what is being said, there will be a breakdown in communication: words will be missed, and the original thought will be misunderstood.

The *manner* in which protocols are used is important as well. If two people meet, and 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 will be confusion and lack of understanding.

The modem handshaking protocol operates under the same principle. Many signals are used to establish modem communication. The following is a listing of the handshaking protocol used to establish and maintain communication.

1. Run the program.
   The TR (Terminal Ready) signal is issued.
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. You 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 "single file." The functional signals ensure that both machines are synchronized with one another, and that they 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 the 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 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 is located on the back of the system unit. It derives its power from the system unit's power supply. Because of this, the problem of the communication board simply not working never occurs, unless of course the system unit as a whole is not working.

There are two possible serial communication problems. 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

The inability to establish communication might manifest itself in not being able to dial up, not being able to make the connection after dialing, or not being able to detect the carrier after connecting.

☐ Solutions

☐ Check that the cable is properly connected and seated to the system unit serial communication interface port.

☐ Make sure that the proper interface cable is connected to the serial communication interface. The serial communication interface is usually a *male connector,* meaning it is a pins-type connector.

- ☐ Run the Serial Communication Diagnostic Module (described in the following section of this chapter). If the test fails, and the character entered is not echoed on the screen, try the test using another serial cable.

- ☐ Swap the VIC-MODEM or RS-232 cartridge on another Commodore system. If it works, the problem is with the serial communication port on your Commodore. Refer to Chapter 2 for instructions on cleaning the edge connector. If it still does not work on your system, take the system unit in for repair.

  If the VIC-MODEM or RS-232 cartridge does not work on another system, the problem lies in the modem/cartridge. Either take it in for repair, or replace it.

- ☐ 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? Was the appropriate character or word size specified? 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 Commodore, run the diagnostic program on it to see whether its communication interface is working properly.

- ☐ 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 pins. These cables might expect data from different pins, look for other signals, or have signals tied together. If this is the case, cable that has the correct pin configuration.

- ☐ Most modems provide additional assistance with troubleshooting, usually through a row of red indicator lights across the front status panel (Fig. 7-5). If your modem includes such an indicator panel, check that certain signals are functioning. For instance, when the modem is properly cabled into the Commodore, and the communication software is being run, the lights over TR and DSR should be illuminated. These lights indicate that the computer and the modem are ready and that the interface signals are turned on and ready

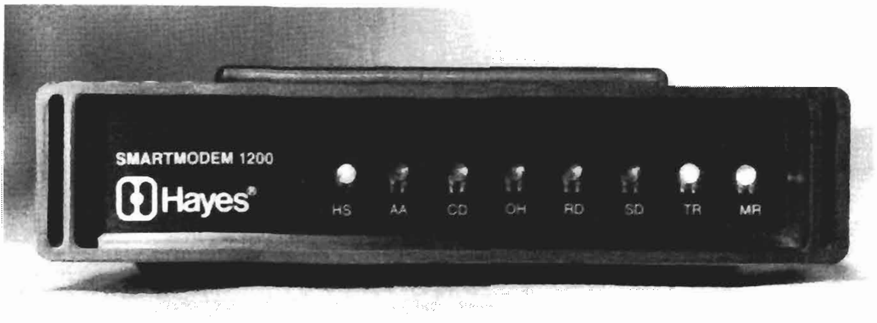


Fig. 7-5. Modem status panel.

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 ("receive data") or TD ("transmit data") are illuminated. When a key is pressed, these lights indicate that the data for the key code are indeed being sent.

☐ The modem might also have a test mode (check the modem documentation to see if this is the case). This test mode loops a data pattern back through the modem. It takes a pattern of characters from the computer, sends the pattern through the modem back to itself, and then returns the pattern to the computer. This *internal loopback test* checks the modem to make sure that it is sending and receiving as it should. Use this test mode if communication still cannot be established. It helps troubleshoot one suspected modem problem at a time.

☐ A *carrier* is a tone that the modem expects before it can send data. It's like saying "Hello" when the phone rings and you pick up the receiver. After saying "Hello," communication can be established, and information from the caller can be transmitted.

This carrier can be heard when using a *smart modem* that dials the phone automatically. You would first hear the telephone ring through the modem speaker, and then a high-pitched tone would sound. This audible signal is the carrier, and it tells the system that the modem is ready to accept data, either from your system or from the other system. The CD ("carrier detect") light is illuminated. This indicates that the other system has answered and been given the carrier.

If the carrier signal is not received, it could mean 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 through the lines. This usually has to do with one of the parameter settings, such as the word size or parity. Return to the communication software parameter setup mode, and check all the settings.

☑ Solution

☐ If you get a mixture of garbled characters and good characters—for example, the word "login" is sent, but "loxxn" is received—you probably have a poor telephone connection. Hang up and redial.

Problem: Printer Does Not Work

If the 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 a switch must be set in order to use the serial communication line. Refer to the printer documentation.

☐ The printer also needs to know the usual communication parameters, such as the baud rate being run, 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.

RUNNING THE SERIAL COMMUNICATION INTERFACE DIAGNOSTIC MODULE

The Serial Communication Diagnostic Module initializes the serial communication interface for testing. 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.

NOTE: The Serial Communication Diagnostic Module works only with an RS-232 modem, not the VIC-MODEM.

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 connect to the peripheral with which you want to communicate. In the connector at this end, attach a wire between pins 2 and 3 (Fig. 7-6).

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 is pressed at the Commodore keyboard is sent out to the device, immediately received in through these pins and sent back to the computer, and the character is echoed back and displayed on the screen (Fig. 7-7).

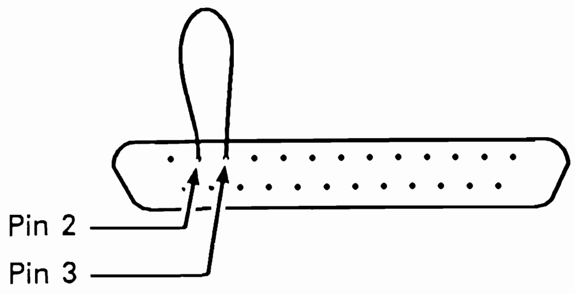


Fig. 7-6. Wire bridge across pins 2 and 3.

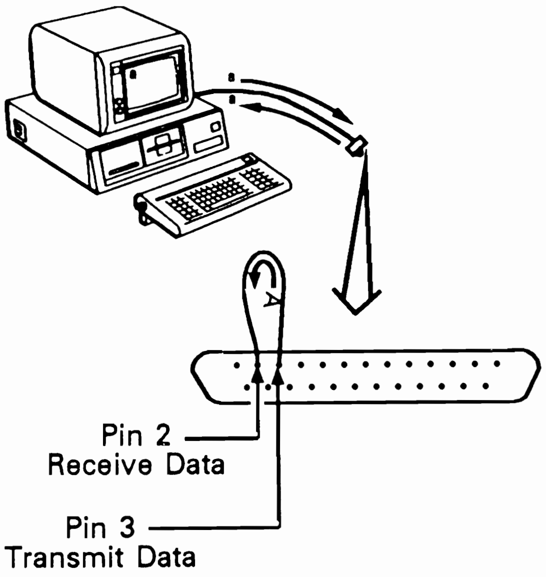


Fig. 7-7. Loopback diagram.

There are plugs available that already have pins 2 and 3 wired together. Such a plug is called a "25-pin serial loopback connector," and can be purchased at a computer or electronic supply store. The plug fits directly onto the serial port, and accomplishes the same loopback test (Fig. 7-8).

The instructions to the Serial Communication Diagnostic Module appear on the screen (Fig. 7-9). The Serial Communication Diagnostic consists of a single test: the Display Character Loopback Test. You enter a character on the keyboard, and it is sent through the communication loop created by the loopback wire. The loopback causes the character to be immediately sent back and received as output displayed on the monitor screen. This checks the functions of the serial communication interface by both transmitting and receiving data.

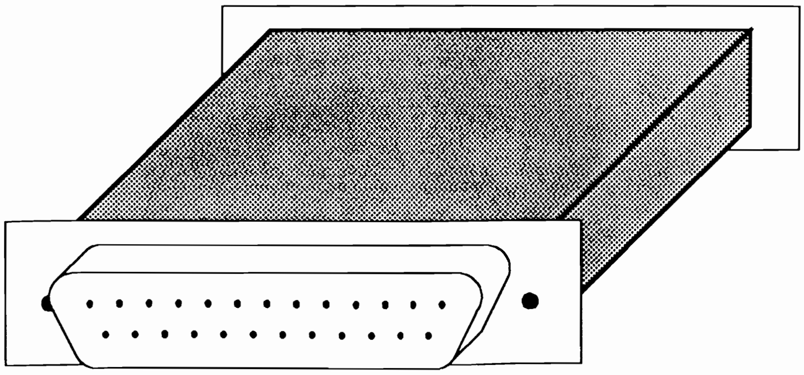
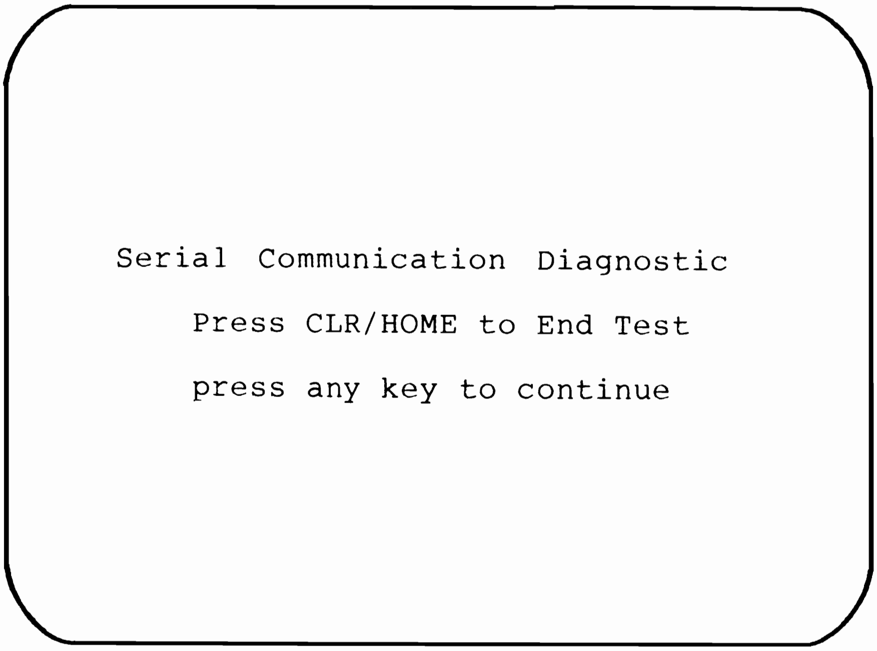


Fig. 7-8. Loopback connector.



```
Serial Communication Diagnostic

    Press CLR/HOME to End Test

    press any key to continue
```

Fig. 7-9. Serial Communication Diagnostic instructions.

HOW THE MODULE WORKS

Pressing S from the System Diagnostic Main Menu causes the program to branch to line 6000, where the Serial Communication Diagnostic Module begins (Fig. 7-10).

Line 6000 begins by clearing the screen and positioning the cursor at the HOME position, and is followed by the REMARK statement indicating that this is the Serial Communication Diagnostic Module:

```
6000 GOSUB 2990:REM "Serial Comm Module"
```

Line 6010 prints the title of the diagnostic on the screen:

```
6010 PRINT TAB(3) "Serial Communication Diagnostic":
     PRINT
```

Line 6020 prints the user instruction to press the CLR/HOME key to end the test:

```
6020 PRINT TAB(7)"Press CLR/HOME to End Test":PRINT
```

Line 6025 then instructs the user to enter any key to continue, once the previous messages have been read:

```
6025 GOSUB 3300
```

```
6000 GOSUB 2990:REM "Serial Comm Module"
6010 PRINT TAB(3) "Serial Communications Diagnostic":
     PRINT
6020 PRINT TAB(7)"Press CLR/HOME to End Test":PRINT
6025 GOSUB 3300
6030 GOSUB 3000:GOSUB 2990
6040 OPEN 2,2,3,CHR$(6)+CHR$(0)
6050 GET #2,A$
6060 IF A$="" THEN 6070
6065 PRINT A$:
6070 GET A$:IF A$="{CLR/HOME}" THEN 6100
6075 IF A$="" THEN 6050
6080 PRINT#2,A$;:GOTO 6050
6100 CLOSE 2:GOTO 50
```

Fig. 7-10. The Serial Communication Diagnostic Module listing.

When the user presses a key, the program goes to line 6030, which clears the screen again:

```
6030  GOSUB 3000:GOSUB 2990
```

Line 6040 opens the serial communication interface, which is designated as device #2. There are two character strings: 6 and 0. These character strings establish parameters for the serial communication line being opened. The first character string, 6, indicates that the test is to be run at 300 baud, which is a sufficient speed for this test. The second character string, 0, specifies that no parity checking is to take place.

```
6040  OPEN 2,2,3,CHR$(6)+CHR$(0)
```

The GET instruction in line 6050 opens the receive side of the serial communication interface, and places the character residing in the receive buffer into variable A$. This prepares the program for testing.

```
6050  GET #2,A$
```

Line 6060 checks whether there is a value in A$. If there is none, and A$ is a null, the program goes to line 6070 to get a value:

```
6060  IF A$="" THEN 6070
```

Line 6065 prints the value contained in A$ on the screen:

```
6065  PRINT A$:
```

The GET statement in line 6070 causes the program to go to the keyboard and look for the user's keystroke. This keystroke is placed into variable A$. If this keystroke is the CLR/HOME key, the program branches to 6100, where the serial communications interface is closed and the test ended:

```
6070  GET A$:IF A$="{CLR/HOME}" THEN 6100
```

If A$ does not contain the CLR/HOME key, the program continues with testing the keystroke, falling through to line 6075. If A$ contains nothing, the program branches from line 6075 to line 6050:

```
6075  IF A$="" THEN 6050
```

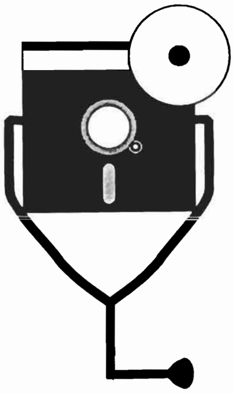
If A$ contains a keystroke other than CLR/HOME, line 6080 outputs it to the serial communication interface, which has been defined as device #2:

```
6080  PRINT#2,A$;:GOTO 6050
```

Once A$ is output, the program returns to line 6050 to accept additional values for testing.

Line 6100 closes the serial communication interface which was designated as device #2. Then the program returns to line 50, the beginning of the System Diagnostic Main Menu:

```
6100  CLOSE 2:GOTO 50
```



Chapter 8
Post-Repair Test and Burn-In

Chapters 2 through 8 have each covered individual components of the Commodore system, examining and treating the problems that can occur within a particular subsystem. This chapter looks at the Commodore system as a whole, and provides a diagnostic module that exercises the entire system to check for problems.

If something does go wrong with the computer and you repair it or take it in for professional servicing, the first thing you want to do when it is returned is check it out. This chapter describes and provides methods for testing the repaired system to make sure everything works as it should.

When picking up the computer from servicing, treat it like an automobile repair. If your car had been in for servicing with a mechanic, you wouldn't just pay the bill and drive off. You would want to test drive it, or have the mechanic show the work that had been done. If the car had a brake problem, you would check to see that the brakes worked properly. If oil was leaking, you would see that the leak was stopped. Only when satisfied that the problem was fixed would you pay the bill and drive away.

Do the same when picking up the computer from servicing. Have the service technician describe what was done, explain why it was done, and demonstrate that the problem has been fixed. However, keep in mind that even while checking the component at the repair shop, not all system functions can be checked. You'll probably just see a few tests for the specific problem, demonstrating that the particular component is working correctly.

For example, if the system unit had a problem, you probably would have just taken the system unit in for servicing, leaving the printer, disk drive, and modem at home. But the computer is a system, consisting of a number of devices and peripherals which all work together. Because of this, sometimes when there is a problem in one device, it can manifest itself in another device. So even when the technician demonstrates that the system unit is working fine, the complete status is unknown until the unit is assembled at home with the entire system in its usual configuration.

The diagnostic module covered in this chapter is a program that thoroughly exercises the Commodore when it's new, when its configuration changes, or when it returns from servicing. This Exerciser Module checks various components of the Commodore system and leaves it running in a continuous mode so that it can ''burn in'' for several hours, even overnight.

Burn-in thoroughly tests and exercises the system, and should be done for 12 to 24 hours. If there is a problem with the computer, 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.

FUNCTION OF THE EXERCISER

The System Exerciser Module calls on diagnostic tests that are already a part of individual component diagnostic modules elsewhere in the System Diagnostic Program. Specifically, it branches to routines in the Monitor Diagnostic Module, Printer Diagnostic Module, and Disk Drive Diagnostic Module.

When the System Exerciser Module is accessed, a menu is displayed. This menu prompts you to specify the devices to be tested, the monitor, printer, and/or disk drives. You can specify any combination of devices to be included or excluded from the Exerciser. These component parameters may be set up in whatever combination is convenient and appropriate for the system.

Once the parameters are set up, the System Exerciser Module immediately begins to perform the continuous test on the specified devices until you press the RUN/STOP key. This means you could run the Exerciser for hours, even overnight if desired.

If the monitor is to be tested, the Scroll Test, the Character Set Test, and the Display Test are run. For more information on these tests, refer to Chapter 3.

If the printer is to be tested, 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 the disk drive is to be tested, the Read/Write Test is run, with records filling the disk and then being read back. The disk drives include some of the most delicate components of the entire system, and should be tested after repair. For more information on the Read/Write Test, refer to Chapter 6.

The System Exerciser begins with the monitor diagnostics, then goes to the printer diagnostics, and finally the disk drives. When complete, the Exerciser cycles back through to the monitor to start the process again. This cycle continues until the RUN/STOP key is pressed to interrupt the Exerciser. Upon interruption, a message is displayed which indicates the number of cycles the Exerciser has run through.

Because the Exerciser tests the three major components of the Commodore system, the monitor, the disk drive(s), and the printer, it can indicate whether there is a problem. If it's an obvious problem, it will probably surface immediately. If there is a marginal or intermittent problem elsewhere 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, the Exerciser is also testing an extensive section of the system unit's machine instruction set.

RUNNING THE SYSTEM EXERCISER

When ready to run the Exerciser on the Commodore, make sure that the components to be tested are ready for hours of testing. If the printer is part of the test, make sure it is loaded with a sufficient quantity of continuous-form paper. Check that there is enough ribbon to last throughout the duration of the printer test. Blank, formatted diskettes should be in each of the drives being tested. Although you might want to dim the monitor display, 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 "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 a prompt asks if you wish to test the monitor, yes or no. Enter y or n, either upper- or lowercase, and press RETURN. A second prompt asks if the printer is to be tested. Again, enter y or n.

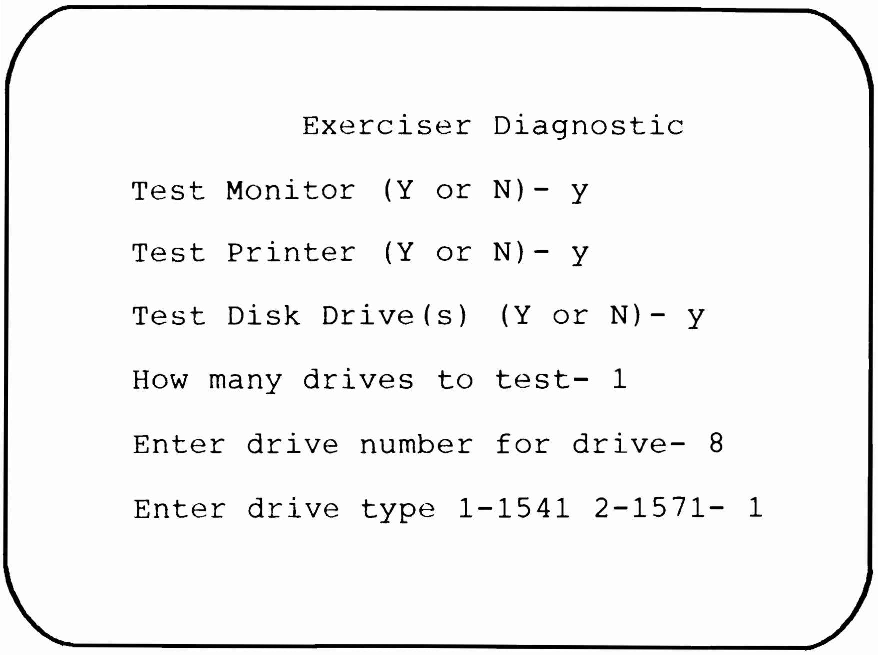
A final prompt asks if the disk drive(s) are to be tested. Enter either y or n. If the answer is yes, you are then prompted to enter the number of drives to be exercised. All the disk drives on the Commodore can be tested through the Exerciser. Enter 1, 2, 3 or 4. Then you are asked to enter the drive IDs for each drive being tested. This ID is the disk drive number, such as 8 for the first drive, 9 for the second drive, or 10 for the third drive (Fig. 8-1).

Any combination of Exerciser parameters can be selected. Once these parameters are all entered, the program begins exercising the system. If the monitor is included in the Exerciser, the screen is filled with the results of the Display Test. The screen then clears and runs through the Character Set Test. Finally the Scroll Test is run for 50 repetitions.

If the printer is included in the Exerciser, it begins upon completion of the monitor testing. Ten repetitions of the Sliding Alpha Test are printed. It then runs through the Display Character Print Test and then prints a line of Es for the Echo Character Print Test. The Horizontal Tab Test and finally the Line Space Test are run, for a thorough printer test.

If the disk drive(s) is included in the Exerciser, it begins upon completion of the printer testing. The Write/Read Test is run on each disk drive specified for testing.

When this is completed, the program loops back and tests the monitor again,



```
              Exerciser Diagnostic

     Test Monitor (Y or N)- y

     Test Printer (Y or N)- y

     Test Disk Drive(s) (Y or N)- y

     How many drives to test- 1

     Enter drive number for drive- 8

     Enter drive type 1-1541 2-1571- 1
```

Fig. 8-1. Exerciser Module prompts.

then the printer, and so forth. The Exerciser runs in this continuous loop, running the tests in a repeated cycle. To halt the Exerciser, press the RUN/STOP key. This ends the System Exerciser Module, and also halts the entire System Diagnostic Program.

To return to the Exerciser, enter GOTO 7430. The system returns to the system Diagnostic Program at line 7430 in the code. This causes the message indicating the number of passes the program has completed to be printed. You are then 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.

If the Exerciser is interrupted while it's running the disk drive test, you might see a file called "TEST" show up in the diskette's directory. Delete this file from the diskette. Normally, the diagnostic removes this file after it completes its run.

If one of the tests fail, the Exerciser halts. So if the printer runs out of paper, or if something happens to the disk drive, the program stops at the point at which the problem took place. An error message describing the problem is displayed, so that you can take the appropriate action.

HOW THE SYSTEM EXERCISER MODULE WORKS

When the user chooses E from the System Diagnostic Main Menu, the program branches to line 7200 for the beginning of the System Exerciser Module (Fig. 8-2).

Line 7200 begins by assigning variable EX equal to one. EX is the Exerciser flag that is used in several modules throughout the diagnostic program. EX is ordinarily equal to zero, allowing the monitor, printer, and disk drive diagnostic

```
7200 EX=1:GOSUB 2990:REM "Exerciser Module"
7210 PRINT TAB(10)"Exerciser Diagnostic":PRINT
7220 INPUT "Test Monitor (Y or N)-";MN$:PRINT
7230 INPUT "Test Printer (Y or N)-";PT$:PRINT
7240 INPUT "Test Disk Drive(s) (Y or N)-";DD$
7250 IF DD$="N" OR DD$="n" THEN 7310
7260 PRINT:INPUT "How many drives to test-";DD
7265 DN="":DIM DN(DD):DIM DT(DD)
7270 FOR I=1 TO DD
7280 PRINT:INPUT "Enter drive number for drive-";DN(I)
7290 PRINT:INPUT "Enter drive type 1-1541 2-1571-";DT(I)
7300 NEXT
7310 IF MN$="N" OR MN$="n" THEN 7330
7320 A$="X":A=50:GOSUB 1140
7340 IF PT$="N" OR PT$="n" THEN 7380
7350 OPEN 1,4,7
7360 A$="E":X=10:GOSUB 3650
7370 CLOSE 1
7380 IF DD$="N" OR DD$="n" THEN 7310
7390 FOR I=1 TO DD
7395 GOSUB 2990
7400 D=DN(I):T=DT(I):GOSUB 5040
7410 NEXT
7420 PS=PS+1:GOTO 7310
7430 GOSUB 2990:CLOSE 1
7440 PRINT TAB(10)"Exerciser Diagnostic":PRINT
7450 PRINT TAB(3)"Total Passes through Exerciser-";PS
7460 GOSUB 3200
7470 EX=0:GOTO 50
```

Fig. 8-2. The System Exerciser Module listing.

modules to function in its usual standalone mode with user input. However, when EX is set to 1, the program skips the various parts of the program which require user intervention, such as entering a particular value.

```
7200 EX=1:GOSUB 2990:REM "Exerciser Module"
```

The GOSUB statement sends the program to the Clear Screen subroutine at line 2990. Finally, the REMARK statement indicates that this is the beginning of the Exerciser module.

Line 7210 prints the title of the module on the screen:

```
7210 PRINT TAB(10)"Exerciser Diagnostic":PRINT
```

Line 7220 prompts the user to indicate whether or not the monitor is to be tested. The answer is placed into variable MN$:

```
7220  INPUT "Test Monitor (Y or N)-";MN$:PRINT
```

Line 7230 prompts the user to indicate whether or not the printer is to be tested. The answer is placed into variable PT$:

```
7230  INPUT "Test Printer (Y or N)-";PT$:PRINT
```

Line 7240 prompts the user to indicate whether or not the disk drive(s) are to be tested. The answer is placed into variable DD$:

```
7240  INPUT "Test Disk Drive(s) (Y or N)-";DD$
```

Because the user's system can have more than one drive, and different types of drives, further prompts are required regarding the disk drive test.

Line 7250 checks to see if the user has answered "no" to the Test Disk Drive(s) prompt. If so, the program branches to line 7310, skipping the code asking for further information about the disk drive(s) to be tested:

```
7250  IF DD$="N" OR DD$="n" THEN 7310
```

If the user answered "yes" to the Test Disk Drive(s) prompt, the program falls through to line 7260, which prompts the user for the number of drives to be tested. The answer is placed into numeric variable DD:

```
7260  PRINT:INPUT "How many drives to test-";DD
```

Line 7265 sets dimension DN equal to null, in order to clear it. Disk number variable DN is dimensioned by the disk quantity variable DD. Also, disk type variable DT is dimensioned by the disk quantity variable DD. The variable DT will indicate whether the disk type in question is a 1541 or 1571 disk drive:

```
7265  DN="":DIM DN(DD):DIM DT(DD)
```

Line 7270 is a FOR-TO loop, looping through the number of drives being tested:

```
7270  FOR I=1 TO DD
```

Line 7280 prompts the user for the drive number, and places the answer into array DN(I). If the first drive is drive #8, for example, the first element of the array becomes an 8:

```
7280  PRINT:INPUT "Enter drive number for drive-";DN(I)
```

Line 7290 is a similar instruction. The user is prompted to enter the drive type: 1 signifies a 1541 drive, and 2 signifies a 1571 drive. The answer is placed into array DT(I):

```
7290 PRINT:INPUT "Enter drive type 1-1541 2-1571-";DT(I)
```

Line 7300 is the NEXT statement that corresponds with the FOR-TO loop that began in line 7270. This causes the code to loop as many times as there are drives to test:

```
7300 NEXT
```

Line 7310 begins the actual system exercising. Here the program examines which options the user has selected, then goes about executing them. This line begins with an IF statement for the Monitor Diagnostic. If the user has chosen not to exercise the monitor, the program branches to line 7330:

```
7310 IF MN$="N" OR MN$="n" THEN 7330
```

If the user did choose to exercise the monitor, the program falls through to line 7320, which sets variable A$ equal to X, which is used as the display character in the monitor scroll test, and sets variable A equal to 50 for the number of repetitions to scroll. Finally, GOSUB 1140 sends the program back to the Monitor Diagnostic Module in order to perform the Scroll Test with these preset parameters:

```
7320 A$="X":A=50:GOSUB 1140
```

In the Monitor Diagnostic Module code, there are various checks to see whether or not EX is equal to one. If it is, the Exerciser is active, and the program jumps past any lines that require user input. The Exerciser runs through the various tests of the Monitor Diagnostic through to the Sprite Test. At that point, at line 1915, there is a statement indicating that if the Exerciser is active, it should RETURN. So when the program runs to that line, it returns back to line 7340.

Line 7340 then does the same check for variable PT$, the printer flag. If the user has chosen not to test the printer, the program branches to line 7380:

```
7340 IF PT$="N" OR PT$="n" THEN 7380
```

If the user did choose to test the printer, the program falls through to line 7350, which opens the printer as upper/lowercase device #1:

```
7350 OPEN 1,4,7
```

Line 7360 assigns variable A$ equal to E. This E is used as the character to be echoed in the Echo Character Print Test in the Printer Diagnostic. It also sets X equal to ten for the number of lines to print in the Sliding Alpha test. Finally, there is a GOSUB to line 3650, treating the entire Printer Diagnostic Module like a large subroutine:

```
7360 A$="E":X=10:GOSUB 3650
```

Just as in exercising the monitor, there are checks for whether EX equals one throughout the Printer Diagnostic code. When the Exerciser gets to line 4140, it returns here to the Exerciser Module.

Line 7370 closes device #1, the printer:

```
7370 CLOSE 1
```

Line 7380 checks if the user opted to exercise the disk drive(s). If not, the program branches to line 7310:

```
7380 IF DD$="N" OR DD$="n" THEN 7310
```

If the user did choose to exercise the disk drive(s), the program falls through to line 7390, which is a FOR-TO loop. Variable DD holds the user input of the number of drives to be tested, so this loops for the number of drives:

```
7390 FOR I=1 TO DD
```

Line 7395 goes to the Clear Screen subroutine at line 2990, which clears the screen of leftover messages and prompts from the previous test:

```
7395 GOSUB 2990
```

Line 7400 sets variable D, the disk number, equal to DN(I). DN is the array in which the disk number is stored. I is an index. In the first pass through the first disk, the program goes through DN(I). The second part of the statement sets variable T, the disk type, equal to DT(I). Then the program goes back to line 5040 into the Disk Drive Diagnostic Module to run the write and read tests.

```
7400 D=DN(I):T=DT(I):GOSUB 5040
```

When the Exerciser completes its run through the Disk Drive Diagnostic, it returns to line 7410, the NEXT statement corresponding to the FOR-TO loop at line 7390:

```
7410 NEXT
```

When all the specified disk drives have been tested, the program falls to line 7420. Line 7420 sets variable PS equal to PS+1. When the user halts the program, this variable indicates the number of passes the Exerciser has run, which is especially useful if the Exerciser has been run for several hours or even overnight.

The program then branches to line 7310, which returns the program to the process loop at the beginning of the Exerciser module, with the prompts for which devices to test:

```
7420  PS=PS+1:GOTO 7310
```

The Exerciser is an endless loop which the user can halt by pressing the RUN/STOP key on the keyboard. When this is done, to continue with the rest of the program, the user must enter the statement GOTO 7430.

Line 7430 is the last segment of the Exerciser Module. It branches to the Clear Screen subroutine at line 2990, then closes device #1, in case the printer or disk drive was left open at the time the program was halted:

```
7430  GOSUB 2990:CLOSE 1
```

Line 7440 prints the Exerciser Diagnostic title on the screen again:

```
7440  PRINT TAB(10)"Exerciser Diagnostic":PRINT
```

Line 7450 prints the total number of test passes on the screen, using variable PS from line 7420, which has been counting the passes:

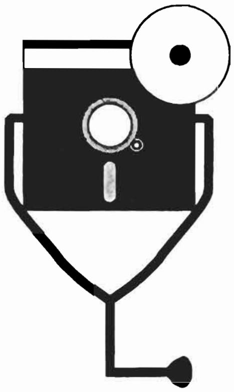
```
7450  PRINT TAB(3)"Total Passes through Exerciser-";PS
```

Line 7460 goes to the subroutine at line 3200, which prints the user prompt to enter any character to end the test:

```
7460  GOSUB 3200
```

Line 7470 returns EX equal to zero again, to set the System Diagnostic Program back to its usual mode of user-directed operation. The program then returns to line 50, which is the beginning of the System Diagnostic Main Menu:

```
7470  EX=0:GOTO 50
```



Chapter 9 Writing Diagnostics for Other Peripherals

When there is a problem somewhere in the system, you want to be able to isolate and identify its source. The System Diagnostic Program presented in this book provides the means for doing this with the system unit, keyboard, monitor, printer, disk drives, and serial communication interface. You are able to perform the necessary repairs on the indicated subsystem, based on this information.

However, if your system utilizes other peripherals not covered by this program, it might be desirable to be able to write new diagnostic modules for them. In this way, the System Diagnostic Program can grow with the computer system, and new components will not be left out of the diagnostic loop that benefits the rest of the system.

Using the joystick as an example, this chapter explains how to identify the basic functions of a new device, and describes the types of problems you might have with it. This chapter also tells how to find the information necessary for developing the diagnostic that tests the device functions. Advice on flowcharting and coding is provided, along with integration and testing procedures. This chapter 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 but would like to be, obtain one of the many excellent books on learning BASIC. There are also BASIC classes available through computer stores, community colleges, and adult education programs.

TYPES OF MICROCOMPUTER PERIPHERALS

There are many interesting peripheral devices available for the Commodore systems. These peripherals include digitizer tablets, which let you draw and draft with the computer. Voice recognition devices can recognize voice patterns and even respond to verbal commands. Voice synthesizers can speak to you.

Other popular peripheral devices include the bar code reader, light pen, mouse—even the video cassette recorder. The Commodore game paddles and joysticks allow you to play various interactive games.

In addition to these peripherals, more and more devices are constantly being invented, marketed, and made readily available to the Commodore computer user.

DEVELOPING A NEW DIAGNOSTIC MODULE

Begin considering the development of a new diagnostic module for the new peripheral as soon as you have acquired some familiarity with it. When this new device becomes an integral part of the permanent system, and not just a new "toy," its own diagnostic should be developed.

Learn the Functions of the Device

The first thing to do when developing a new diagnostic module for a new peripheral is to learn everything possible about how the device functions. Find out specifically what the device is intended for and what it is designed to do. Why did you buy it? What does it do? Experiment with the device. Explore its possibilities. Try to do real work (or play) with it.

Read the Device Manual

Read the device documentation to learn about capabilities of which you may not have been aware. 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 that will prove useful in developing and programming the diagnostic module. Never throw away any documentation that comes with a device.

The operation section of the manual 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.

In addition to installation instructions, the technical section of the manual describes various options and their settings. It also provides suggestions on setting up your existing hardware and software in order for the new device to work properly and efficiently.

The manual usually also includes a troubleshooting guide which outlines problems that can occur and suggests remedial procedures to follow in response to these problems. Examine the troubleshooting guide as a first step to developing a new diagnostic module. It will make you familiar with typical problems for which the diagnostic should test. The diagnostic must be developed before trouble strikes—don't wait until there is a problem before looking at the troubleshooting guide.

The troubleshooting guide provides information about the particular device, but does not take into account the software running with the device, the hardware configuration of the system, or the type of interface. In other words, the troubleshooting guide does not know your particular system, and therefore, cannot specifically address your exact troubleshooting requirements. It would be impossible for any troubleshooting guide to consider all these items, simply because there are infinite possibilities and combinations.

Because of this, you can benefit from developing a customized troubleshooting diagnostic for the device as it works with your particular system. The diagnostic can then be integrated into the System Diagnostic Program as a new module.

Decide What to Test For

Once you know the functions of the device, take the inverse of these functions and test for them. In other words, if a device is supposed to perform a particular function, you should test whether it does *not* perform this function.

Further research, either in the system technical manual, the device documentation, or the BASIC programming manual, is necessary to find out how to write programs that check for particular functions.

Create a Program Flowchart

Once you know the functions and problems for which to test, you can develop the program flowchart. This flowchart becomes the program design and the "road map" to writing the actual diagnostic code.

The flowchart outlines the various test routines that will exist within the module, as well as any GOTO statements and FOR-TO loops. The flowchart can also detail any new program functions and commands you have learned through research on the particular device. Be as detailed as possible in the flowchart. This makes subsequent coding much easier.

Flowcharting is the method by which the process or "flow" of a program can be charted graphically. Arrows connect process and decision boxes, representing 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.

Create the flowchart in two stages. In the first stage, express the fundamental ideas of the program in plain English. Figure 9-1 is a flowchart for a simple segment of a program asking users to enter their name, and then greeting them by name.

In the second stage, write a more detailed flowchart, translating the English statements into BASIC commands. Now follow the arrows and add the line numbers to the code to write the program as follows:

```
10 PRINT {CLR/HOME}
20 INPUT "Enter your name-";A$
30 PRINT "Hello ";A$
40 END
```

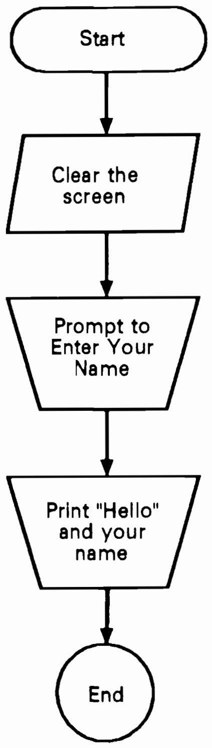


Fig. 9-1. A simple program flowchart.

Flowcharts can be detailed or brief according to personal preference. If you're a beginning or novice programmer, however, the more detailed the flowchart is, the easier and more trouble-free the coding will tend to be. The objective is to establish the program's flow and logic to the point where the program can easily be written.

Write the Diagnostic Module

When the program flowchart is diagrammed, you're ready to begin actual coding. When developing a new test, first make sure that all aspects of the system are functioning properly. Do not develop a test on a malfunctioning system, or with a peripheral that is experiencing a problem. If a test were developed on a malfunctioning system, as soon as the problem were fixed, the results of the test might well become null and void, and the test code itself could become obsolete.

When exploring new BASIC instructions for the new device, follow examples presented in the BASIC manual, just to try them out. Educate yourself in this manner, and get a good feel for what these instructions do before "hacking" the System Diagnostic Program.

After creating a flowchart of your ideas and converting them to BASIC code, write a stand alone version of the new diagnostic module and test it by itself before integrating it with the System Diagnostic Program. Refer to the code of existing modules in Appendix A or Appendix B (depending on your system) to familiarize yourself with the proper techniques.

Include the standard utility subroutines used throughout the System Diagnostic Program to simplify and streamline the new code.

Integrate the New Module

When you insert the new module into the System Diagnostic Program, modify lines 50 through 180 of the program where the Main Menu Module resides. Add the module's title and identifier to the PRINT statement that lists the menu options on the screen. Also, insert the appropriate IF statement for the new module so that the program branches to the proper line number where the new code begins.

Once this is done, add the code to the program. Have the line numbers of the new code begin around 8000, or wherever the last diagnostic module stopped.

Test the New Diagnostic Module

Finally, test the diagnostic as an integrated module of the System Diagnostic Program. Try out all the new tests with as many possibilities as you can imagine. Try your best to find "bugs" and "break" the program. If and when you do find bugs, fix the code where the problem occurred.

Also test the other modules of the System Diagnostic Program, to find out if the new program is interacting with the original code in any adverse ways. This testing process ensures accurate results when using the diagnostic in an actual troubleshooting situation.

DEVELOPING THE JOYSTICK 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, a concrete example is provided in this section. This example uses the joystick as the new peripheral for which a new diagnostic module will be developed.

Development of the Joystick Diagnostic Module is described, step-by-step. This includes researching the functions of the joystick, writing the new code for the Joystick Diagnostic, and testing the new module as integrated with the entire System Diagnostic Program.

Between the general principles outlined in the previous section and the concrete examples described in this section, you should gain a good understanding of how a diagnostic module can be developed for any peripheral added to your Commodore system.

Functions of the Joystick

The joystick is a pointing and "firing" device (Fig. 9-2). It is used to position the cursor on the screen, typically in games. It is also used to "fire" a graphic object, like a missile, at another graphic object, as part of the game. With the joystick, you can "fly" a jet airplane, move a graphic robot, or control a laser cannon.

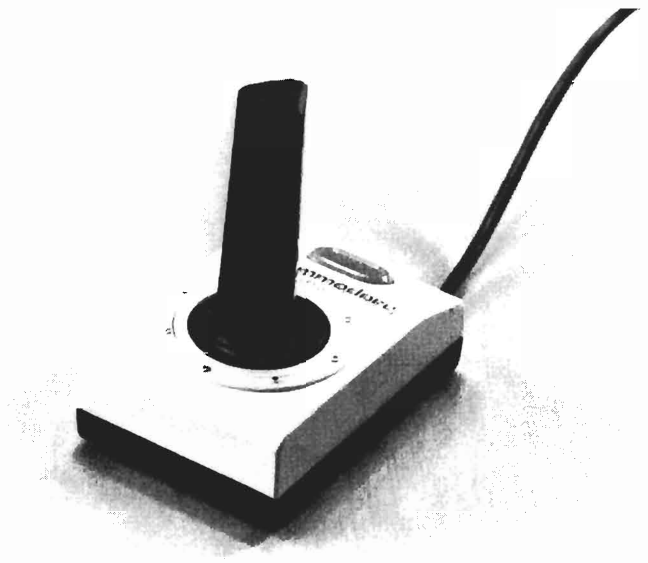


Fig. 9-2. The Commodore joystick.

The joystick is a mechanical device with five internal switches. Four of these switches control the four directional positions at which the joystick can point: top, bottom, right, and left. The fifth switch controls the fire button at the top of the joystick lever.

Two of the directional switches can be activated at one time. For example, if the joystick is pointed diagonally toward the top right, both the top and right joystick switches are activated. This means that there are eight different directional positional positions that can be used, in addition to the fire button control.

Tracking a Graphic Object. The appropriate object on the screen moves in accordance with the movement of the joystick. When the joystick is moved to the right, the object on the screen tracks to the right. When the joystick is moved to the left, the object tracks to the left. When the joystick is moved diagonally, the object likewise tracks diagonally. The joystick causes the particular object on the screen to follow its movements.

Pressing the FIRE Button. In addition to the pointing lever, the joystick includes the FIRE button. The graphic object can be positioned at a certain location, and then when the fire button is pressed, the computer is informed to take a certain action as programmed. By clicking the button, the computer knows that something is to be done with the item on the screen where the cursor is pointing. Typically, this would be shooting something in a game.

Information about the Joystick

There are several sources that can provide the information necessary to write programs for the joystick, particularly a diagnostic program for the joystick.

The Commodore BASIC manual has a great deal of valuable information for programming in Commodore BASIC. The joystick user's manual provides more specific information. The amount of information included in the joystick manual depends on the manufacturer. The manual will often include information on installation and maintenance, uses and applications for the joystick, and perhaps sample programs and a troubleshooting guide.

One of the best sources of information on programming the joystick, is the *Commodore Programmer's Reference Guide*. In the I/O chapter, there is information about programming the joystick. In addition, one of the appendices includes documentation on the pinout configuration of the game port to which the joystick is connected.

Find out all the possible positions in which a joystick can be placed. Find out how the joystick button is activated. Find out new uses for the joystick of which you might not have been aware. Any sample programs in the documentation will provide a good idea of the proper approach and special commands for developing a program for your own purposes.

Another very beneficial means for researching the joystick is an old defective joystick. If there is a defective joystick available, or if yours goes bad and needs to be replaced, experiment with the old one. It's already defective, so nothing can be hurt by taking it apart and exploring the works. This will help you understand the component better, which will later aid troubleshooting endeavors.

TROUBLESHOOTING AND REPAIR GUIDELINES

The joystick user's manual will usually provide information on maintenance, troubleshooting, and possibly repair. This information, along with the following possible problems, help you decide what kinds of diagnostic tests are needed to troubleshoot these particular malfunctions.

Problem: Joystick Does Not Work At All

If the joystick does not work at all, there could be a problem with the interface connection, with the joystick itself, or with the system unit. To isolate where the problem lies, try the following solutions in the order given.

☑ Solutions

- ☐ Make sure the joystick is firmly seated in its game paddle port (Fig 9-3). Unplug it, and carefully plug it back in and see if it works.
- ☐ If it still does not work, unplug the joystick cable from its game paddle port, and plug it into the second port. If the joystick works in the other game paddle port, this indicates a problem in the original port in the system unit. Refer to Chapter 2 for further instructions on troubleshooting the system unit.
- ☐ If the joystick does not work in either of the two game paddle ports, try operating the joystick, along with the diagnostic program, on another system. If the joystick works on the other system, this indicates a problem with your system unit. Refer to Chapter 2 for system unit troubleshooting procedures.

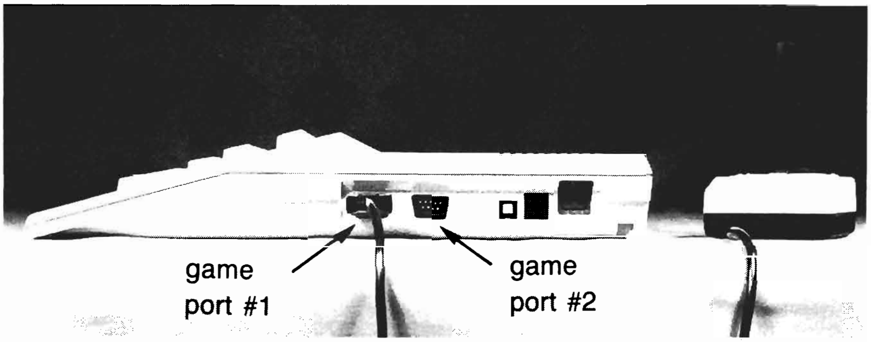


Fig. 9-3. The joystick connected to game port #1.

- ☐ If the joystick does not work on the other system, this indicates a faulty joystick. The easiest thing to do in this case is to simply replace it.

Problem: Joystick Does Not Work In a Particular Direction

The joystick has eight directional positions to which it can point: north, south, east, west, northeast, northwest, southeast, and southwest. If the joystick does not respond to one or more of these positions, this indicates a problem with the internal direction switch.

✓ Solutions

- ☐ Check the cable connection and make sure the joystick connection is properly plugged in to the game paddle port.
- ☐ Following instructions found in the joystick user guide, carefully disassemble the joystick. Spray contact cleaner onto the suspect switch. This should "unstick" the switch.
- ☐ Swap another joystick onto your system, if available. If the other joystick works on your system, then your joystick is defective, and should be replaced. If the other joystick does not work on your system, this indicates a problem with the system unit. Refer to Chapter 2 for further solutions.

Problem: Fire Button Does Not Work

The switch that controls the fire button can also get stuck.

✓ Solutions

- ☐ Check the cable connection and make sure that the joystick connection is properly plugged in to the game paddle port.
- ☐ Following instructions found in the joystick user guide, carefully disassemble the joystick. Spray contact cleaner onto the suspect switch. This should "unstick" the switch.

- ☐ Swap another joystick onto your system, if available. If the other joystick works on your system, then your joystick is defective, and should be replaced. If the other joystick does not work on your system, this indicates a problem with your system unit. Refer to Chapter 2 for further solutions.

Joystick Tests

Once you have a fair understanding of the joystick, determine the kinds of tests that would be most helpful in such a diagnostic module.

The Game Port Option. Because the Commodore includes two game paddle ports, the diagnostic user instructions should include a prompt for which port is to be tested. This will allow separate testing of both ports.

The Direction Test. The joystick is a series of switches which determine the direction of movement or firing action of the object it is controlling. Because of this, one particularly useful test would be an on-screen display tracking the direction in which the joystick is pointing. This would let you diagnose whether a particular switch or combination of switches is malfunctioning.

In such a direction test, when the joystick is in its neutral position, the screen is clear. If the joystick is moved away from you, the word NORTH would appear on the screen. Likewise, if the joystick is moved to your left, the word WEST would appear on the screen. And if you were to move the joystick diagonally to the lower right, the word SOUTHEAST would be displayed.

In order to develop such a directional test, determine how information is sent from the joystick to the system unit, and how this information is accessed. This is a relatively simple process. The joystick is connected to one of two game ports located on the right side of the system unit (Fig. 9-3). One of the chips on the system unit's motherboard is an input port, accepting and processing input from these game ports.

When one of the joystick switches is activated, a particular memory location is set in this input port chip. On the Commodore 64, this is memory location 56320 for the first port, and memory location 56321 for the second port. This is a one-byte location which contains a numeric value. The following table indicates the numeric value which corresponds to each joystick position:

neutral	0
up	1
down	2
left	4
up left diagonal	5
down left diagonal	6
right	8
up right diagonal	9
down right diagonal	10

The Joystick Diagnostic Module must include a tight loop that continuously examines the appropriate game port memory location. As long as the value is zero and the joystick is in the neutral position, the loop goes around continuously. As soon as the user moves the joystick, the value changes, and the program can go to a table that holds some comparison statements. If the value were one, the program would display the word "NORTH" on the screen. If the value changed to two, the program would display the word "SOUTH." This would let the user make sure that the joystick's "sense of direction" is working properly.

The Fire Test. A second test necessary for joystick diagnostics involves the fire button. When the fire button is pressed, a value of 15 is set into the game port memory location. This can serve two purposes: to test the fire button, and to quit from the Joystick Diagnostic and return to the System Diagnostic Main Menu.

The Joystick Module Flowchart

Once you have determined the functions to be tested as part of the Joystick 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 drawn in the form of a program flowchart.

The program flowchart indicates the commands to be used when the code for this program is actually being written. To help write the flowchart, and later the actual code with all the attendant details, have the BASIC programming manual nearby as a ready reference. The *Commodore Programmer's Reference* is also an excellent resource. These books will help you research any specialized instructions used to access commands for the joystick.

Once you are familiar with the special commands, instructions, formats, and other idiosyncrasies about programming the joystick on the Commodore, the module can be diagrammed in the flowchart. This flowchart reflects your research about the joystick functions, and details the particulars of the two tests you've decided to develop for this module. Figure 9-4 shows a possible flowchart for the Joystick Diagnostic Module.

Writing the Joystick Diagnostic Module

Once the flowchart is developed and diagrammed, the actual diagnostic code can be written. Again, refer to the BASIC programming reference books. Also refer to Appendix A if you have a Commodore 64, or Appendix B if you have a Commodore 128, and see examples of how the other diagnostic modules have performed certain activities. Go ahead and copy or refer to the simple routines, such as the Clear Screen and Get-a-Key subroutines.

The Joystick Diagnostic Module Listing. Based on the flowchart, the program code might look similar (although it could have very many variations) to the listing as shown in Fig. 9-5.

How the Joystick Diagnostic Module Works. Just as was done with the other modules within the System Diagnostic Program, the first things to be done in this module are to clear the screen, position the cursor, provide an identifying

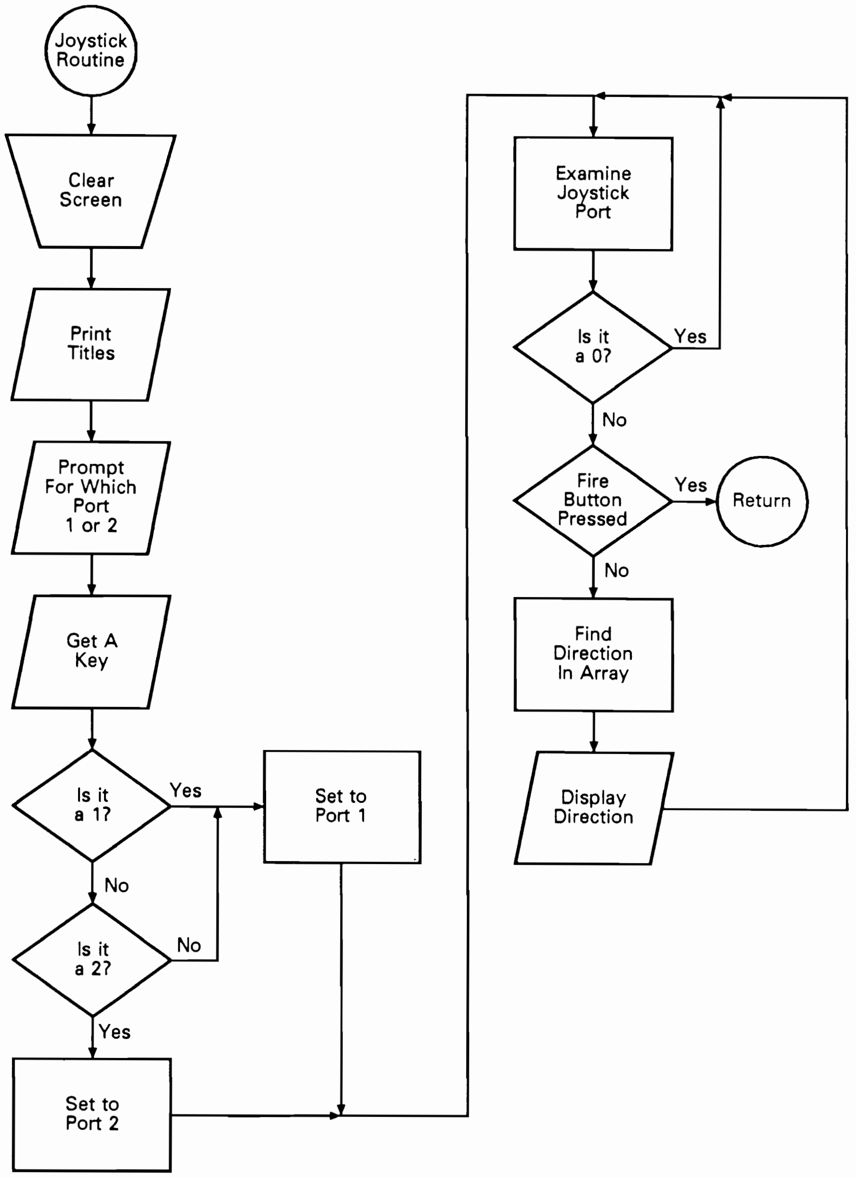


Fig. 9-4. The Joystick Diagnostic Module program flowchart.

REMARK statement, and print the module's title on the screen. This begins at line 7000, which clears the screen, positions the cursor, and identifies the module:

```
7000 D$="":DIM D$(10):GOSUB 2990:
     REM "Joystick Module"
```

Line 7002 begins a FOR-NEXT loop which initializes an array:

```
7000 D$="":DIM D$(10):GOSUB 2990:
     REM "Joystick Module"
7002 FOR I=1 TO 10
7004 READ D$(I):NEXT
7006 DATA "NORTH","SOUTH","","WEST"
7007 DATA "NORTHWEST","SOUTHWEST"
7008 DATA "","EAST","NORTHEAST","SOUTHEAST"
7010 PRINT TAB(10) "Joystick Diagnostic":PRINT
7020 PRINT TAB(7) "Which Port to Test 1 or 2-";P
7030 IF P<>2 THEN P=1
7040 PRINT:PRINT "Press Fire Button to End Test"
7050 IF P=1 THEN J=56320
7055 IF P=2 THEN J=56321
7060 VL=PEEK(J)
7070 FB=VL AND 16
7080 JD=15-(VL AND 15)
7090 IF FB=16 THEN 7110:REM Fire button was pressed
7100 GOTO 50
7110 IF JD=0 THEN 7060
7120 PRINT D$(JD)
7130 GOTO 7060
```

Fig. 9-5. The Joystick Diagnostic Module listing.

```
7002 FOR I=1 TO 10
```

Line 7004 reads variable D$, which is indexed by I. I is set from one to ten, meaning there are ten pieces of data loaded into the array. The data for the array follow in lines 7006, 7007, and 7008:

```
7004 READ D$(I):NEXT
7006 DATA "NORTH","SOUTH","","WEST"
7007 DATA "NORTHWEST","SOUTHWEST"
7008 DATA "","EAST","NORTHEAST","SOUTHEAST"
```

These data specify the position indicators that are to be printed on the screen depending on the direction in which the joystick is moved. These data are placed in the array according to the possible values that can be returned when the program PEEKs into the memory location variable where the joystick input information resides.

C128 USERS: Lines 7002 through 7009 are modified in the C128 code to reflect the different switch positions of the directions in the array.

Line 7010 prints the name of the Joystick Diagnostic at the top of the screen:

```
7010 PRINT TAB(10) "Joystick Diagnostic":PRINT
```

Line 7020 prompts the user to select the game port to be tested, port 1 or port 2. The value entered is placed into variable P:

```
7020 PRINT TAB(7) "Which Port to Test 1 or 2-";P
```

Line 7030 fixes the program so that if the value entered is anything but a two, P is automatically set to one. This ensures that only the values one or two are ever placed into variable P:

```
7030 IF P< >2 THEN P=1
```

Line 7040 prints a message that instructs the user to end the test by pressing the FIRE button on the joystick:

```
7040 PRINT:PRINT "Press Fire Button to End Test"
```

Line 7050 checks to see if P is equal to one. If it is, then variable J is set equal to 56320, which is the memory location that holds the input information for game port #1. If P is equal to two, then J is set to 56321, which is the memory location that holds the input information for game port #2:

```
7050 IF P=1 THEN J=56320
7055 IF P=2 THEN J=56321
```

Line 7060 sets variable VL equal to the contents of location J, which is set up either for game port #1 or game port #2:

```
7060 VL=PEEK(J)
```

C128 USERS: Lines 7050 through 7060 are modified to reflect the use of the C128 joystick JOY command.

Line 7070 sets variable FB equal to variable VL, which is derived from the PEEK statement, after going through the logical AND of 16 in the array. This instruction filters out some unnecessary data that it picked up from the PEEK operation. All that is necessary for the purposes of this program is that the input value equal a number between zero and 16:

```
7070 FB=VL AND 16
```

Line 7080 sets variable JD equal to 15 minus VL after it went through the logical AND statement. This again filters out extra data, and will also help check when the FIRE button is depressed on the joystick:

```
7080  JD=15-(VL AND 15)
```

Line 7090 checks to see if variable FB is equal to 16:

```
7090  IF FB=16 THEN 7110:REM Fire button was pressed
```

If it is, the program branches to line 7110. If it is not equal to 16, the program falls through to line 7100, which loops back to line 50, the beginning of the System Diagnostic Main Menu:

```
7100  GOTO 50
```

If FB is not equal to 16, the FIRE button was not pressed, and the program branches to line 7110. Line 7110 checks whether variable JD (joystick direction) is equal to zero. If it is, the joystick is in the neutral position, so the program returns to line 7060:

```
7110  IF JD=0 THEN 7060
```

If JD is not equal to zero, the program falls through to line 7120, which prints the contents of variable D$. D$ is the array set up at the beginning of the module, indexed by variable JD. Therefore, if JD contains a value of one, for example, then the word NORTH is printed on the screen, indicating that the user is pointing the joystick forward.

```
7120  PRINT D$(JD)
```

The program then falls through to line 7130, which immediately sends the program back to line 7060 for the loop looking for another value again. In this manner, the user can continue to check the various joystick directions.

```
7130  GOTO 7060
```

When the user wants to end the diagnostic, the FIRE button can be tested at the same time. If the FIRE button does not cause the program to exit to the System Diagnostic Main Menu, there is probably a problem with the button. Press the RUN/STOP key to end execution of the entire program, then refer to the "Troubleshooting and Repair Guidelines" earlier in this chapter to continue with diagnostic and repair efforts.

If the FIRE button is working correctly, the program branches all the way back to line 50 for the beginning of the System Diagnostic Main Menu, at which point the user can choose another diagnostic module or quit the program.

Integrating the Module with the System Diagnostic Program

Once the code is written for the Joystick Diagnostic Module, you must integrate it with the rest of the System Diagnostic Program.

First, modify the System Diagnostic Main Menu routine, which displays and accesses all the diagnostic module options. Look at lines 50 through 90 of the program where the Main Menu Module resides. Add the Joystick Diagnostic identifier to the PRINT statement that lists the menu options on the screen. Also, insert the appropriate IF statement for the Joystick Diagnostic identifier in lines 110-170, so that the program branches to the beginning of the Joystick Diagnostic Module when the appropriate key, perhaps a J, is pressed.

Once this is done, add the code to the program. A good place to append the new code is starting at line 7000, as was done in this chapter's example. This begins the new module where the last diagnostic module stopped.

Testing the Joystick Diagnostic Module

The last thing left to do is to test the Joystick Diagnostic as it now resides as an integrated module of the System Diagnostic Program. Try out the tests in all the different directions. Try to "break" the program, and if you do, find and fix the code where the problem occurred.

Once you have completed these research, flowcharting, coding, integration, and testing procedures, the Joystick Diagnostic Module is ready for real work whenever troubleshooting and diagnostics are needed.

Appendix A
Commodore 64 System Diagnostic Program Listing

```
1 REM "Commodore 64"
  5 REM "System Diagnostic Program Copyright 1988
    TAB BOOKS Inc."
 10 POKE 53272,23:DIM K$(63):REM Set Character Set
 20 DATA "INST/DEL","RETURN","CRSR RT","F7","F1","F3","F6",
    "CRSR UP","3","W","A"
 21 DATA "4","Z","S","E"," ","5","r","D","6","C","F",
    "T","X","7","Y","G","8","B"
 22 DATA "H","U","V","9","I","J","0","M","K","O","N",
    "+","P","L","-",".",":","@"
 23 DATA ",","£","*",";","CLR/HOME"," ","=","↑","/","1",
    "←"," ","2","SPACE"
 24 DATA " ","Q"
 25 FOR I=0 TO 62: READ K$(I):NEXT
 30 DIM R(8):DATA 1,2,4,8,16,32,64,128
 35 FOR I=0 TO 7:READ R(I):NEXT
 50 GOSUB 2990: PRINT TAB(12) "System Diagnostic":PRINT:
 60 PRINT TAB(3) "<K>eyboard Text  <S>erial Comm Test"
 70 PRINT TAB(3) "<M>onitor Test   <D>isk Drive Test":
 80 PRINT TAB(3) "<C>assette Test  <J>oystick Test":
 85 PRINT TAB(3) "<P>rinter Test   <E>xerciser":
 90 PRINT TAB(16) "<Q>uit":PRINT:PRINT TAB(12)
    "Enter Selection"
```

```
100 GOSUB 3000:REM Get Key Command
110 IF A$="K" OR A$="k" THEN 200
120 IF A$="M" OR A$="m" THEN 800
130 IF A$="P" OR A$="p" THEN 3500
135 IF A$="C" or A$="c" THEN 4200
140 IF A$="D" OR A$="d" THEN 5000
150 IF A$="S" OR A$="s" THEN 6000
160 IF A$="E" OR A$="e" THEN 7200
165 IF A$="J" OR A$="j" THEN 7000
170 IF A$="Q" OR A$="q" THEN END
180 GOSUB 3100:GOTO 80
200 REM"Keyboard Diagnostic Module"
210 POKE 53272,23:S1=1:GOSUB 630
220 A$="":KY$="":GOSUB 3000
230 N=ASC(A$)
260 IF ASC(A$)=19 THEN GOTO 740
270 IF ASC(A$)=5 THEN KY$="ctrl 2:GOTO 600
280 IF ASC(A$)=17 THEN A$=" ":KY$="crsr down":GOTO 600
290 IF ASC(A$)=18 THEN KY$="rvs on":GOTO 600
300 IF ASC(A$)=20 THEN A$=" ":KY$="inst del":GOTO 600
310 IF ASC(A$)=28 THEN KY$="ctrl 3":GOTO 600
320 IF ASC(A$)=29 THEN A$=" ":KY$="cursor right":GOTO 600
330 IF ASC(A$)=30 THEN KY$="ctrl 6":GOTO 600
340 IF ASC(A$)=31 THEN KY$="ctrl 7":GOTO 600
350 IF ASC(A$)=141 THEN KY$="shift return":GOTO 600
360 IF ASC(A$)=142 THEN KY$="upper case":GOTO 600
370 IF ASC(A$)=144 THEN KY$="ctrl 1":GOTO 600
380 IF ASC(A$)=145 THEN KY$=" ":KY$="cursor up":GOTO 600
390 IF ASC(A$)=146 THEN KY$="rvs off":GOTO 600
400 IF ASC(A$)=147 THEN GOTO 740
410 IF ASC(A$)=148 THEN A$=" ":KY$="inst del":GOTO 600
420 IF ASC(A$)=156 THEN KY$="ctrl 5":GOTO 600
430 IF ASC(A$)=157 THEN A$=" ":KY$="cursor left":GOTO 600
440 IF ASC(A$)=158 THEN KY$="cntrl 8":GOTO 600
450 IF ASC(A$)=159 THEN KY$="ctrl 4":GOTO 600
460 IF ASC(A$)=133 THEN KY$="f1":GOTO 600
470 IF ASC(A$)=134 THEN KY$="f3":GOTO 600
480 IF ASC(A$)=135 THEN KY$="f5":GOTO 600
490 IF ASC(A$)=136 THEN KY$="f7":GOTO 600
500 IF ASC(A$)=137 THEN KY$="f2":GOTO 600
510 IF ASC(A$)=138 THEN KY$="f4":GOTO 600
520 IF ASC(A$)=139 THEN KY$="f6":GOTO 600
530 IF ASC(A$)=140 THEN KY$="f8":GOTO 600
560 IF S=0 OR S=1 THEN 600
570 IF S=2 THEN KY$="C= ":GOTO 595
580 IF S<>4 THEN GOTO 710
590 KY$="CTRL "
```

```
 595 KY$=KY$+K$(K)
 600 GOSUB 630
 610 PRINT TAB(5) A$;TAB(13) N;TAB(23) KY$;TAB(35) S1
 620 GOTO 220
 630 GOSUB 2990
 640 V=PEEK(53272)
 650 IF V=21 THEN S1=1:GOTO 680
 660 S1=2:PRINT TAB(7) "Keyboard Diagnostic Module":PRINT
 670 PRINT TAB(2) "Character  Value Key Combination Set":GOTO 700
 680 PRINT TAB(7) "keyboard diagnostic module":PRINT
 690 PRINT TAB(2) "character  value key combination set"
 700 PRINT TAB(2)"---------  ----- -------------- ---":RETURN
 710 GOSUB 630
 720 PRINT "An illegal shift code was encountered!"
 730 PRINT "your keyboard needs service": GOTO 220
 740 POKE 53272,23:PRINT "{BLUE}":GOTO 50

 800 GOSUB 2990:REM "Monitor Diagnostic Module"
 810 PRINT TAB(6) "Monitor Diagnostic":PRINT
 820 PRINT TAB(5) "1...Display Test":PRINT
 830 PRINT TAB(5) "2...Alignment Test":PRINT
 840 PRINT TAB(5) "3...Character Set Test":PRINT
 850 PRINT TAB(5) "4...Color Test":PRINT
 860 PRINT TAB(5) "5...Scroll Test":PRINT
 870 PRINT TAB(5) "6...Sprite Test":PRINT
 890 PRINT TAB(5) "Press CLR/HOME to End Diagnostic"
 900 GOSUB 3000
1000 IF ASC(A$)=19 THEN 50
1010 IF A$="1" THEN 1090
1020 IF A$="2" THEN 1180
1030 IF A$="3" THEN 1320
1040 IF A$="4" THEN 1440
1050 IF A$="5" THEN 1650
1070 IF A$="6" THEN 1770
1080 GOSUB 3100:GOTO 900
1090 GOSUB 2990:REM Display Character Set Test
1100 PRINT TAB(14) "Display Test":PRINT
1110 PRINT "Enter Any Character to Fill Screen":PRINT
1120 PRINT "Press CLR/HOME to End Test"
1130 GOSUB 3000
1132 PRINT "{CLR/HOME}"
1135 IF ASC(A$)=19 THEN 800
1140 FOR I=1 TO 999
1150 PRINT A$;:NEXT
1155 IF EX=1 THEN 1360
1160 GOTO 1130
1180 GOSUB 2990:REM Alignment Test
```

```
1185 POKE 53272,21
1190 PRINT TAB(13) "alignment test":
1200 PRINT CHR$(176);:FOR I=1 TO 18: PRINT CHR$(99);:NEXT I
1210 PRINT CHR$(178);:FOR I=1 TO 18: PRINT CHR$(99);:NEXT I
1220 PRINT CHR$(174):
1230 FOR I=1 TO 10: PRINT CHR$(98);TAB(19)CHR$(98);
     TAB(38)CHR$(98):NEXT I
1240 PRINT CHR$(171);:FOR I=1 TO 18 PRINT CHR$(99);:NEXT
1250 PRINT CHR$(123);:FOR I=1 TO 18 PRINT CHR$(99);:NEXT
1260 PRINT CHR$(179)
1270 FOR I=1 TO 10:PRINT CHR$(98);TAB(19) CHR$(98);TAB(38)
     CHR$(98):NEXT
1280 PRINT CHR$(173);:FOR I=1 TO 18:PRINT CHR$(99);:NEXT I
1290 PRINT CHR$(177);:FOR I=1 TO 18:PRINT CHR$(99);:NEXT
1300 PRINT CHR$(189):GOSUB 3200
1310 POKE 53272,23:GOTO 800
1320 GOSUB 2990:REM "Character Set Test"
1330 PRINT TAB(11) "Character Set Test"
1340 PRINT:PRINT "Enter Character Set (1) or (2)-"
1350 GOSUB 3000
1360 IF A$="1" THEN POKE 53272,21
1370 FOR I=32 TO 255
1380 IF I=128 OR I=130 OR I=131 OR I=132 OR I=141
     I=143 OR I=144 THEN 1410
1390 IF I=145 OR I=146 OR I=147 OR I=148 OR I=156
     OR I=157 OR I=158 THEN 1410
1395 IF I=159 THEN 1410
1400 PRINT CHR$(I);+" ";
1410 NEXT
1415 IF EX=1 THEN POKE 53272,23:GOTO 1440
1420 PRINT:GOSUB 3200
1430 POKE 53272,23:GOTO 800
1440 GOSUB 2990:REM "Color Test"
1450 PRINT TAB(15) "Color Test":PRINT
1460 "C$="XXXXXXXXXXXXXXXXXXXX"
1470 PRINT "{CTRL 1} Black";TAB(12) C$
1480 PRINT "{CTRL 2} White";TAB(12) C$
1490 PRINT "{CTRL 3} Red";TAB(12) C$
1500 PRINT "{CTRL 4} Cyan";TAB(12) C$
1510 PRINT "{CTRL 5} Purple";TAB(12) C$
1520 PRINT "{CTRL 6} Green";TAB(12) C$
1530 PRINT "{CTRL 7} Blue";TAB(12) C$
1540 PRINT "{CTRL 8} Yellow";TAB(12) C$
1550 PRINT "{C= 1} Orange";TAB(12) C$
1560 PRINT "{C= 2} Brown";TAB(12) C$
1570 PRINT "{C= 3} Lt. Red";TAB(12) C$
1580 PRINT "{C= 4} Gray 1";TAB(12) C$
```

```
1590 PRINT "{C= 5} Gray 2";TAB(12) C$
1600 PRINT "{C= 6} Lt. Green";TAB(12) C$
1610 PRINT "{C= 7} Lt. Blue";TAB(12) C$
1620 PRINT "{C= 8} Gray 3";TAB(12) C$
1625 IF EX=1 THEN PRINT "{C= 7}":GOTO 1680
1630 PRINT:GOSUB 3200
1640 PRINT "{C= 7}":GOTO 800
1650 GOSUB 2990:REM "Scroll Test"
1660 PRINT TAB(15) "Scroll Test":PRINT
1670 PRINT "Enter the Number of Repetitions-";A:PRINT
1680 N=31
1690 FOR I=1 TO A
1700 N=N+1:IF N>71 THEN N=32
1710 FOR X=1 TO 40
1720 PRINT CHR$(N);:N=N+1
1730 IF N>71 THEN N=32
1740 NEXT X
1750 NEXT I
1755 PRINT
1757 IF EX=1 THEN 1770
1760 GOSUB 3200
1765 GOTO 800
1770 GOSUB 2990:REM "Sprite Test"
1780 PRINT TAB(15)"Sprite Test"
1790 FOR S=2040 TO 2047:POKE S,13:NEXT
1792 FOR S=832 TO 894:POKE S,255:NEXT
1800 V=53248:POKE V+21,255
1810 FOR S=39 TO 46:POKE V+S,1:NEXT S
1815 R=0
1820 FOR I=0 TO 7
1830 POKE 53269,PEEK(53269)OR(2↑I)
1840 FOR X=24 TO 255
1850 POKE (V+I*2),X:POKE (V+I*2+1),152
1860 NEXT X
1870 POKE (V+16),R(I)
1880 FOR X=0 TO 65
1890 POKE (V+(I*2)),X:POKE (V+(I*2+1)),152
1895 NEXT X
1897 POKE V+16,PEEK(V+16)AND 254
1900 POKE 53269,PEEK(53269)AND(255-2↑I)
1910 NEXT I
1915 IF EX=1 THEN RETURN
1920 PRINT:GOSUB 3200
1930 GOTO 800

2900 PRINT "{HOME/CLEAR}":RETURN:REM "Clear Screen Routine"
3000 GET A$:IF A$="" THEN 3000:REM "Get a Key Routine"
```

```
3005 K=PEEK(197):S=PEEK(653)
3010 RETURN
3099 REM "Beep Routine"
3100 FOR L=54272 TO 54296:POKE L,0:NEXT
3110 POKE 54296,15:POKE 54277,90:POKE 54278,1:
     POKE 54273,21:POKE 54272,31
3120 POKE 54276,33
3130 FOR T=1 TO 50:NEXT
3140 POKE 54276,16:RETURN
3200 PRINT TAB(7)"press any key to end test";
3210 GOSUB 3000
3220 RETURN
3300 PRINT TAB(7)"press any key to continue":RETURN

3500 REM "Printer Diagnostic Module"
3510 GOSUB 2900:PRINT TAB(11) "Printer Diagnostic":PRINT
3515 PRINT TAB(3)"1...Sliding Alpha Test":PRINT
3517 PRINT TAB(3)"2...Display Character Print Test":PRINT
3518 PRINT TAB(3)"3...Echo Character Print Test":PRINT
3519 PRINT TAB(3)"4...Horizontal Tab Test":PRINT
3520 PRINT TAB(3)"5...Line Feed Test":PRINT
3530 PRINT TAB(5)"Press CLR/HOME to End Diagnostic
3535 OPEN 1,4,7:REM "Open as Upper/Lower case"
3540 GOSUB 3000:IF ASC(A$)<>19 THEN 3560
3550 CLOSE 1:GOTO 50
3560 IF A$="1" THEN 3630
3570 IF A$="2" THEN 3750
3580 IF A$="3" THEN 3920
3590 IF A$="4" THEN 4010
3600 IF A$="5" THEN 4070
3610 GOSUB 3100:GOTO 3540
3630 REM "Sliding Alpha Test"
3640 GOSUB 2990:PRINT TAB(11) "Sliding Alpha Test":PRINT
3645 INPUT "Enter the number of repetitions-";X
3650 N=31
3660 FOR I=1 TO X
3670 L$="":N=N+1:IF N>111 THEN N=32
3680 FOR A=1 TO 80
3690 L$=L$+CHR$(N):N=N+1
3700 IF N>111 THEN N=32
3710 NEXT A
3720 PRINT#1,L$;:NEXT I
3730 IF EX=1 THEN GOTO 3760
3735 CLOSE 1
3740 GOTO 3500
3750 REM Display Character Print Test
3760 GOSUB 2990:PRINT TAB(6)"Display Character Print
```

```
     Test":L$="":N=1
3770 CLOSE 1
3780 OPEN 1,4:REM "Open as Upper Case Only
3790 FOR I=32 TO 255
3800 IF N=80 THEN PRINT#1,L$;:N=1:L$="":GOTO 3820
3810 L$=L$+CHR$(I)+" ":N=N+1
3820 NEXT I
3830 CLOSE 1
3840 OPEN 1,4,7:REM "Open as Upper/Lower Case"
3850 FOR I=32 TO 255
3860 IF N=80 THEN PRINT#1,L$;:N=1:L$="":GOTO 3880
3870 L$=L$+CHR$(I)+" ":N=N+1
3880 NEXT I
3890 IF L$<>"" THEN PRINT#1,L$
3900 IF EX=1 THEN GOTO 3950
3905 CLOSE 1
3910 GOTO 3500
3915 REM "Echo Character Print Test"
3920 GOSUB 2990:PRINT TAB(7)"Echo Character Print Test":
     PRINT
3925 PRINT TAB(5)"Enter the Character to Echo":PRINT
3935 PRINT "Press the CLR/HOME Key to End the Test"
3940 GOSUB 3000:IF ASC(A$)=19 THEN CLOSE 1:GOTO 3500
3950 L$=""
3970 FOR I=1 TO 80
3980 L$=L$+A$:NEXT I
3990 PRINT#1,L$
4000 GOTO 3940
4010 REM Horizontal Tab Test
4020 GOSUB 2990:PRINT TAB(10)"Horizontal Tab Test"
4030 FOR I=1 TO 80
4040 PRINT #1,TAB(I)"*":NEXT I
4050 IF EX=1 THEN GOTO 4080
4060 CLOSE 1:GOTO 3500
4070 REM "Line Feed Test"
4080 GOSUB 2990:PRINT TAB(13)"Line Feed Test"
4100 FOR I=2 TO 10
4110 FOR A=1 TO I
4115 PRINT#1,"":NEXT A
4120 N$=STR$(I-1)
4130 PRINT#1,"There have been ";N$;" line feeds";:NEXT I
4140 IF EX=1 THEN RETURN
4150 CLOSE 1:GOTO 3500

4200 REM "Cassette Diagnostic Module"
4210 GOSUB 2900
4220 PRINT TAB(10) "Cassette Diagnostic":PRINT
```

```
4230 PRINT TAB(10) "1...Write/Read Test":PRINT
4240 PRINT TAB(10) "2...Read Only Test":PRINT
4250 PRINT TAB(10) "Enter Command"
4260 GOSUB 3000
4270 IF A$="1" THEN 4310
4280 IF A$="2" THEN 4420
4290 GOSUB 3100
4300 GOTO 4260
4310 GOSUB 2990:PRINT TAB(13)"Write/Read Test":PRINT
4320 PRINT "Place a blank tape in the cassette drive"
4330 PRINT "and press RECORD.":PRINT
4340 GOSUB 3300:GOSUB 3000:PRINT
4350 OPEN 1,1,1, "Test-Tape 1"
4360 FOR I=1 TO 100
4370 PRINT#1,CHR$(170):NEXT
4380 CLOSE 1:OPEN 1,1,1,"Test-Tape 2"
4390 FOR I=1 TO 100
4400 PRINT#1 CHR$(85):NEXT
4405 CLOSE 1
4410 PRINT:PRINT "Rewind the tape and press PLAY"
     PRINT:GOSUB 3200
4415 GOTO 4460
4420 GOSUB 2990
4430 PRINT TAB(13) "Read Only Test":PRINT
4440 PRINT "Place the test tape in the cassette drive"
4450 PRINT "and press PLAY"
4455 PRINT:GOSUB 3300:GOSUB 3000:PRINT
4460 OPEN 1,1,0,"Test-Tape 1"
4470 FOR I=1 TO 100
4480 INPUT#1,A$
4490 IF A$<>CHR$(170) THEN E=E+1
4495 NEXT
4500 CLOSE 1
4510 OPEN 1,1,0,"Test-Tape 2"
4520 FOR I=1 TO 100
4530 INPUT#1,A$
4540 IF A$<>CHR$(85) THEN E1=E1+1
4550 NEXT I
4560 CLOSE 1:PRINT
4570 PRINT TAB(3) "There were ";E;" High Value errors"
     :PRINT
4580 PRINT TAB(3) "There were ";E1;" Low Value errors"
     :PRINT
4590 GOSUB 3200
4600 GOTO 50
```

```
5000 REM: "Disk Drive Diagnostic Module"
5010 FOR I=1 TO 8:HV$=HV$+CHR$(170):NEXT
5020 LV$="":FOR I=1 TO 8:LV$=LV$+CHR$(85):NEXT
5030 GOSUB 2990:PRINT TAB(10)"Disk Drive Diagnostic":PRINT
5035 INPUT "Which Drive Do You Want To Test-";D:PRINT
5036 IF D<8 OR D>15 THEN PRINT "Invalid Disk Number":
     GOTO 5035
5037 INPUT "Enter Drive Type 1-1541 2-1571";T
5038 IF T<>2 THEN T=1
5040 BF$=HV$:NM$="High Values"
5050 FOR I=1 TO 2
5060 OPEN 1,D,2,"TEST,s,w"
5070 PRINT:PRINT"Writing ";NM$;" to Disk"
5072 IF T=2 THEN Z=33600
5074 IF T=1 THEN Z=16800
5075 FOR A=1 TO Z
5077 B1$=""
5080 PRINT#1,BF$
5090 NEXT A
5095 CLOSE 1
5100 OPEN 1,D,2, "TEST,s,r"
5110 PRINT "Reading ";NM$;" from Disk":PRINT
5115 FOR A=1 TO Z
5120 INPUT#1,B$
5140 IF B$<>BF$ THEN PRINT "Error Reading ";NM$:E=E+1
5150 NEXT
5160 CLOSE 1
5165 OPEN 15,8,15,"s0:TEST":CLOSE 15
5170 BF$=LV$:NM$="Low Values":NEXT I
5180 PRINT:PRINT "There were ";E;" errors encountered"
5182 OPEN 15,8,15:INPUT#15,DS,DS$,T,S:PRINT:PRINT DS;
     ",",DS$+",";T;",";S:PRINT
5183 CLOSE 15
5185 IF EX=1 THEN RETURN
5190 GOSUB 3200
5200 GOTO 50

6000 GOSUB 2990:REM "Serial Comm Module"
6010 PRINT TAB(3) "Serial Communications Diagnostic":
     PRINT
6020 PRINT TAB(7)"Press CLR/HOME to End Test":PRINT
6025 GOSUB 3300
6030 GOSUB 3000:GOSUB 2990
6040 OPEN 2,2,3,CHR$(6)+CHR$(0)
6050 GET #2,A$
6060 IF A$="" THEN 6070
6065 PRINT A$:
```

```
6070 GET A$:IF A$="{CLR/HOME}" THEN 6100
6075 IF A$="" THEN 6050
6080 PRINT#2,A$;:GOTO 6050
6100 CLOSE 2:GOTO 50

7000 D$="":DIM D$(10):GOSUB 2990:
     REM "Joystick Module"
7002 FOR I=1 TO 10
7004 READ D$(I):NEXT
7006 DATA "NORTH","SOUTH","","WEST"
7007 DATA "NORTHWEST","SOUTHWEST"
7008 DATA "","EAST","NORTHEAST","SOUTHEAST"
7010 PRINT TAB(10) "Joystick Diagnostic":PRINT
7020 PRINT TAB(7) "Which Port to Test 1 or 2-";P
7030 IF P<>2 THEN P=1
7040 PRINT:PRINT "Press Fire Button to End Test"
7050 IF P=1 THEN J=56320
7055 IF P=2 THEN J=56321
7060 VL=PEEK(J)
7070 FB=VL AND 16
7080 JD=15-(VL AND 15)
7090 IF FB=16 THEN 7110:REM Fire button was pressed
7100 GOTO 50
7110 IF JD=0 THEN 7060
7120 PRINT D$(JD)
7130 GOTO 7060

7200 EX=1:GOSUB 2990:REM "Exerciser Module"
7210 PRINT TAB(10)"Exerciser Diagnostic":PRINT
7220 INPUT "Test Monitor (Y or N)-";MN$:PRINT
7230 INPUT "Test Printer (Y or N)-";PT$:PRINT
7240 INPUT "Test Disk Drive(s) (Y or N)-";DD$
7250 IF DD$="N" OR DD$="n" THEN 7310
7260 PRINT:INPUT "How many drives to test-";DD
7265 DN="":DIM DN(DD):DIM DT(DD)
7270 FOR I=1 TO DD
7280 PRINT:INPUT "Enter drive number for drive-";DN(I)
7290 PRINT:INPUT "Enter drive type 1-1541 2-1571-";DT(I)
7300 NEXT
7310 IF MN$="N" OR MN$="n" THEN 7330
7320 A$="X":A=50:GOSUB 1140
7340 IF PT$="N" OR PT$="n" THEN 7380
7350 OPEN 1,4,7
7360 A$="E":X=10:GOSUB 3650
7370 CLOSE 1
7380 IF DD$="N" OR DD$="n" THEN 7310
7390 FOR I=1 TO DD
```

```
7395 GOSUB 2990
7400 D=DN(I):T=DT(I):GOSUB 5040
7410 NEXT
7420 PS=PS+1:GOTO 7310
7430 GOSUB 2990:CLOSE 1
7440 PRINT TAB(10)"Exerciser Diagnostic":PRINT
7450 PRINT TAB(3)"Total Passes through Exerciser-";PS
7460 GOSUB 3200
7470 EX=0:GOTO 50
```

Appendix B

Commodore 128 System Diagnostic Program Listing

```
1 REM "Commodore 128"
5 REM "System Diagnostic Program Copyright 1988
  TAB BOOKS Inc."
10 POKE 2604,22:DIM K$(88):REM Set Character Set
20 DATA "inst/del","return","cursor right","f7","f1",
   "f3","f6","cursor up","3"
21 DATA "w","a","4","z","s","e"," ","5","r","D","6",
   "c","f","t","x","7","y","g"
22 DATA "8","b","h","u","v","9","i","j","0","n","k",
   "o","n","+","p","l","-","."
23 DATA ":","@",",","£","*",";","clr/home"," ","=",
   "↑","/","1","←"," ","2"
24 DATA "space"," ","Q","run stop","help","8","5",
   "tab"
25 DATA "2","4","7","1","esc","+","-","line feed"
26 "data " ","6","9","3"," ","0",".","up arrow",
   "down arrow","left arrow"," "
27 DATA "right arrow","no scroll"
29 FOR I=0 TO 88: READ K$(I):NEXT
30 DIM R(8):DATA 1,2,4,8,16,32,64,128
35 FOR I=0 TO 7:READ R(I):NEXT
36 DIM F(8):DATA 133,137,134,138,135,139,136,140
```

```
 37 FOR A=1 TO 8:READ F(A):NEXT
 40 REM "System Diagnostic Main Menu"
 50 GOSUB 2990:PRINT TAB(12) "System Diagnostic":PRINT:
 60 PRINT TAB(3) "<K>eyboard Text  <S>erial Comm Test"
 70 PRINT TAB(3) "<M>onitor Test   <D>isk Drive Test":
 80 PRINT TAB(3) "<C>assette Test  <J>oystick Test":
 85 PRINT TAB(3) "<P>rinter Test   <E>xerciser":
 90 PRINT TAB(16) "<Q>uit":PRINT:PRINT TAB(12)
    "Enter Selection"
100 GOSUB 3000:REM Get a Key
110 IF A$="K" OR A$="k" THEN 200
120 IF A$="M" OR A$="m" THEN 800
130 IF A$="P" OR A$="p" THEN 3500
135 IF A$="C" or A$="c" THEN 4200
140 IF A$="D" OR A$="d" THEN 5000
150 IF A$="S" OR A$="s" THEN 6000
160 IF A$="E" OR A$="e" THEN 7200
165 IF A$="J" OR A$="j" THEN 7000
170 IF A$="Q" OR A$="q" THEN END
180 GOSUB 3100:GOTO 80
200 REM"Keyboard Diagnostic Module"
205 FOR I=1 TO 8:KEY I,CHR$(F(I)):NEXT
210 POKE 2604,22:S1=1:GOSUB 630
220 A$="":KY$="":GOSUB 3000
230 N=ASC(A$)
260 IF ASC(A$)=19 THEN GOTO 740
270 IF ASC(A$)=5 THEN KY$="ctrl 2:GOTO 600
272 IF ASC(A$)=9 THEN A$=::KY$="tab":GOTO 600
273 IF ASC(A$)=(10) THEN A$="":KY$="line feed":
    GOTO 600
274 IF ASC(A$)=13 THEN KY$="return":GOTO 600
280 IF ASC(A$)=17 THEN A$=" ":KY$="cursor down":
    GOTO 600
290 IF ASC(A$)=18 THEN KY$="rvs on":GOTO 600
300 IF ASC(A$)=20 THEN A$=" ":KY$="inst del":GOTO 600
310 IF ASC(A$)=28 THEN KY$="ctrl 3":GOTO 600
315 IF ASC(A$)=27 THEN A$="":KY$="esc":GOTO 600
320 IF ASC(A$)=29 THEN A$=" ":KY$="cursor right":
    GOTO 600
330 IF ASC(A$)=30 THEN KY$="ctrl 6":GOTO 600
340 IF ASC(A$)=31 THEN KY$="ctrl 7":GOTO 600
350 IF ASC(A$)=141 THEN KY$="shift return":GOTO 600
360 IF ASC(A$)=142 THEN KY$="upper case":GOTO 600
370 IF ASC(A$)=144 THEN KY$="ctrl 1":GOTO 600
380 IF ASC(A$)=145 THEN KY$=" ":KY$="cursor up":GOTO 600
390 IF ASC(A$)=146 THEN KY$="rvs off":GOTO 600
400 IF ASC(A$)=147 THEN GOTO 740
```

```
410 IF ASC(A$)=148 THEN A$=" ":KY$="inst del":GOTO 600
420 IF ASC(A$)=156 THEN KY$="ctrl 5":GOTO 600
430 IF ASC(A$)=157 THEN A$=" ":KY$="cursor left":GOTO 600
440 IF ASC(A$)=158 THEN KY$="cntrl 8":GOTO 600
450 IF ASC(A$)=159 THEN KY$="ctrl 4":GOTO 600
460 IF ASC(A$)=133 THEN KY$="f1":GOTO 600
470 IF ASC(A$)=134 THEN KY$="f3":GOTO 600
480 IF ASC(A$)=135 THEN KY$="f5":GOTO 600
490 IF ASC(A$)=136 THEN KY$="f7":GOTO 600
500 IF ASC(A$)=137 THEN KY$="f2":GOTO 600
510 IF ASC(A$)=138 THEN KY$="f4":GOTO 600
520 IF ASC(A$)=139 THEN KY$="f6":GOTO 600
530 IF ASC(A$)=140 THEN KY$="f8":GOTO 600
560 IF S=0 OR S=1 THEN 600
570 IF S=2 THEN KY$="C= ":GOTO 595
580 IF S=8 THEN KY$="alt ":GOTO 595
590 KY$="ctrl/"
595 KY$=KY$+K$(K)
600 GOSUB 630
610 PRINT TAB(5) A$;TAB(13) N;TAB(23) KY$;TAB(35) S1
620 GOTO 220
630 GOSUB 2990
640 V=PEEK(2604)
650 IF V=20 THEN S1=1:GOTO 680
660 S1=2:PRINT TAB(10) "Keyboard Diagnostic":PRINT
670 PRINT TAB(2) "Character  Value Key Combination Set":
    GOTO 700
680 PRINT TAB(10) "keyboard diagnostic module":PRINT
690 PRINT TAB(2) "character  value key combination set"
700 PRINT TAB(2)"---------  ----- -------------- ---":
    RETURN
710 GOSUB 630
720 PRINT "An illegal shift code was encountered!"
730 PRINT "your keyboard needs service": GOTO 220
740 POKE 2604,22:PRINT "{BLUE}":GOTO 50
800 GOSUB 2990:REM "Monitor Diagnostic Module"
810 PRINT TAB(6) "Monitor Diagnostic":PRINT
820 PRINT TAB(5) "1...Display Test":PRINT
830 PRINT TAB(5) "2...Alignment Test":PRINT
840 PRINT TAB(5) "3...Character Set Test":PRINT
850 PRINT TAB(5) "4...Color Test":PRINT
860 PRINT TAB(5) "5...Scroll Test":PRINT
870 PRINT TAB(5) "6...Sprite Test":PRINT
890 PRINT TAB(5) "Press CLR/HOME to End Diagnostic"
900 GOSUB 3000
1000 IF ASC(A$)=19 THEN 50
1010 IF A$="1" THEN 1090
```

```
1020 IF A$="2" THEN 1180
1030 IF A$="3" THEN 1320
1040 IF A$="4" THEN 1440
1050 IF A$="5" THEN 1650
1070 IF A$="6" THEN 1770
1080 GOSUB 3100:GOTO 900
1090 GOSUB 2990:REM Display Character Set Test
1100 PRINT TAB(14) "Display Test":PRINT
1110 PRINT "Enter Any Character to Fill Screen":PRINT
1120 PRINT "Press CLR/HOME to End Test"
1130 GOSUB 3000
1132 PRINT "{CLR/HOME}"
1135 IF ASC(A$)=19 THEN 800
1140 FOR I=1 TO 999
1150 PRINT A$;:NEXT
1155 IF EX=1 THEN 1360
1160 GOTO 1130
1180 GOSUB 2990:REM Alignment Test
1185 POKE 2604,20
1190 PRINT TAB(13) "alignment test":
1200 PRINT CHR$(176);:FOR I=1 TO 18: PRINT CHR$(99);:NEXT I
1210 PRINT CHR$(178);:FOR I=1 TO 18: PRINT CHR$(99);:NEXT I
1220 PRINT CHR$(174):
1230 FOR I=1 TO 10: PRINT CHR$(98);TAB(19)CHR$(98);
     TAB(38)CHR$(98):NEXT I
1240 PRINT CHR$(171);:FOR I=1 TO 18 PRINT CHR$(99);:NEXT
1250 PRINT CHR$(123);:FOR I=1 TO 18 PRINT CHR$(99);:NEXT
1260 PRINT CHR$(179)
1270 FOR I=1 TO 10:PRINT CHR$(98);TAB(19) CHR$(98);TAB(38)
     CHR$(98):NEXT
1280 PRINT CHR$(173);:FOR I=1 TO 18:PRINT CHR$(99);:NEXT I
1290 PRINT CHR$(177);:FOR I=1 TO 18:PRINT CHR$(99);:NEXT
1300 PRINT CHR$(189):GOSUB 3200
1310 POKE 2604,22:GOTO 800
1320 GOSUB 2990:REM "Character Set Test"
1330 PRINT TAB(11) "Character Set Test"
1340 PRINT:PRINT "Enter Character Set (1) or (2)-"
1350 GOSUB 3000
1360 IF A$="1" THEN POKE 2604,20
1370 FOR I=32 TO 255
1380 IF I=128 OR I=130 OR I=131 OR I=132 OR I=141
     I=143 OR I=144 THEN 1410
1390 IF I=145 OR I=146 OR I=147 OR I=148 OR I=156
     OR I=157 OR I=158 THEN 1410
1395 IF I=159 THEN 1410
1400 PRINT CHR$(I);+" ";
1410 NEXT
```

```
1415 IF EX=1 THEN POKE 2604,22:goto 1440
1420 PRINT:GOSUB 3200
1430 POKE 2604,22:GOTO 800
1440 GOSUB 2990:REM "Color Test"
1450 PRINT TAB(15) "Color Test":PRINT
1460 "C$="XXXXXXXXXXXXXXXXXXX"
1470 PRINT "{CTRL 1} Black";TAB(12) C$
1480 PRINT "{CTRL 2} White";TAB(12) C$
1490 PRINT "{CTRL 3} Red";TAB(12) C$
1500 PRINT "{CTRL 4} Cyan";TAB(12) C$
1510 PRINT "{CTRL 5} Purple";TAB(12) C$
1520 PRINT "{CTRL 6} Green";TAB(12) C$
1530 PRINT "{CTRL 7} Blue";TAB(12) C$
1540 PRINT "{CTRL 8} Yellow";TAB(12) C$
1550 PRINT "{C= 1} Orange";TAB(12) C$
1560 PRINT "{C= 2} Brown";TAB(12) C$
1570 PRINT "{C= 3} Lt. Red";TAB(12) C$
1580 PRINT "{C= 4} Gray 1";TAB(12) C$
1590 PRINT "{C= 5} Gray 2";TAB(12) C$
1600 PRINT "{C= 6} Lt. Green";TAB(12) C$
1610 PRINT "{C= 7} Lt. Blue";TAB(12) C$
1620 PRINT "{C= 8} Gray 3";TAB(12) C$
1625 IF EX=1 THEN PRINT "{C= 7}":GOTO 1680
1630 PRINT:GOSUB 3200
1640 PRINT "{C= 7}":GOTO 800
1650 GOSUB 2990:REM "Scroll Test"
1660 PRINT TAB(15) "Scroll Test":PRINT
1670 PRINT "Enter the Number of Repetitions-";A:PRINT
1680 N=31
1690 FOR I=1 TO A
1700 N=N+1:IF N>71 THEN N=32
1710 FOR X=1 TO 40
1720 PRINT CHR$(N);:N=N+1
1730 IF N>71 THEN N=32
1740 NEXT X
1750 NEXT I
1755 PRINT
1757 IF EX=1 THEN 1770
1760 GOSUB 3200
1765 GOTO 800
1770 GOSUB 2990:REM "Sprite Test"
1780 PRINT TAB(15)"Sprite Test"
1790 FOR I=3584 TO 4095
1800 POKE I,255:REM "Define Sprite Shape"
1810 NEXT
1820 FOR X=1 TO 8
1830 SPRITE X,1,2,1
```

```
1840 MOVSPR X,24,152
1850 MOVSPR X,90#5
1860 FOR A=1 TO 1200:NEXT A
1870 SPRITE X,0
1880 NEXT X
1915 IF EX=1 THEN RETURN
1920 PRINT:GOSUB 3200
1930 GOTO 800

2900 PRINT "{HOME/CLEAR}":RETURN:REM "Clear Screen Routine"
3000 GET A$:IF A$="" THEN 3000:REM "Get a Key Routine"
3005 K=PEEK(213):S=PEEK(211)
3010 RETURN
3100 PRINT CHR$(7):RETURN:REM "Beep Routine"
3200 PRINT TAB(7)"press any key to end test";
3210 GOSUB 3000
3220 RETURN
3300 PRINT TAB(7)"press any key to continue":RETURN

3500 REM "Printer Diagnostic Module"
3510 GOSUB 2900:PRINT TAB(11) "Printer Diagnostic":PRINT
3515 PRINT TAB(3)"1...Sliding Alpha Test":PRINT
3517 PRINT TAB(3)"2...Display Character Print Test":PRINT
3518 PRINT TAB(3)"3...Echo Character Print Test":PRINT
3519 PRINT TAB(3)"4...Horizontal Tab Test":PRINT
3520 PRINT TAB(3)"5...Line Feed Test":PRINT
3530 PRINT TAB(5)"Press CLR/HOME to End Diagnostic
3535 OPEN 1,4,7:REM "Open as Upper/Lower case"
3540 GOSUB 3000:IF ASC(A$)<>19 THEN 3560
3550 CLOSE 1:GOTO 50
3560 IF A$="1" THEN 3630
3570 IF A$="2" THEN 3750
3580 IF A$="3" THEN 3920
3590 IF A$="4" THEN 4010
3600 IF A$="5" THEN 4070
3610 GOSUB 3100:GOTO 3540
3630 REM "Sliding Alpha Test"
3640 GOSUB 2990:PRINT TAB(11) "Sliding Alpha Test":PRINT
3645 INPUT "Enter the Number of Repetitions-";X
3650 N=31
3660 FOR I=1 TO X
3670 L$="":N=N+1:IF N>111 THEN N=32
3680 FOR A=1 TO 80
3690 L$=L$+CHR$(N):N=N+1
3700 IF N>111 THEN N=32
3710 NEXT A
3720 PRINT#1,L$;:NEXT I
```

```
3730 IF EX=1 THEN GOTO 3760
3735 CLOSE 1
3740 GOTO 3500
3750 REM Display Character Print Test
3760 GOSUB 2990:PRINT TAB(6)"Display Character Print
     Test":L$="":N=1
3770 CLOSE 1
3780 OPEN 1,4:REM "Open as Upper Case Only
3790 FOR I=32 TO 255
3800 IF N=80 THEN PRINT#1,L$;:N=1:L$="":GOTO 3820
3810 L$=L$+CHR$(I)+" ":N=N+1
3820 NEXT I
3830 CLOSE 1
3840 OPEN 1,4,7:REM "Open as Upper/Lower case"
3850 FOR I=32 TO 255
3860 IF N=80 THEN PRINT#1,L$;:N=1:L$="":GOTO 3880
3870 L$=L$+CHR$(I)+" ":N=N+1
3880 NEXT I
3890 IF L$<>"" THEN PRINT#1,L$
3900 IF EX=1 THEN GOTO 3950
3905 CLOSE 1
3910 GOTO 3500
3915 REM "Echo Character Print Test"
3920 GOSUB 2990:PRINT TAB(7)"Echo Character Print Test":
     PRINT
3925 PRINT TAB(5)"Enter the Character to Echo":PRINT
3935 PRINT "Press CLR/HOME to End Test"
3940 GOSUB 3000:IF ASC(A$)=19 THEN CLOSE 1:GOTO 3500
3950 L$=""
3970 FOR I=1 TO 80
3980 L$=L$+A$:NEXT I
3990 PRINT#1,L$
4000 GOTO 3940
4010 REM Horizontal Tab Test
4020 GOSUB 2990:PRINT TAB(10)"Horizontal Tab Test"
4030 FOR I=1 TO 80
4040 PRINT #1,TAB(I)"*":NEXT I
4050 IF EX=1 THEN GOTO 4080
4060 CLOSE 1:GOTO 3500
4070 REM "Line Feed Test"
4080 GOSUB 2990:PRINT TAB(13)"Line Feed Test"
4100 FOR I=2 TO 10
4110 FOR A=1 TO I
4115 PRINT#1,"":NEXT A
4120 N$=STR$(I-1)
4130 PRINT#1,"There have been ";N$;" line feeds";:NEXT I
4140 IF EX=1 THEN RETURN
```

```
4150 CLOSE 1:GOTO 3500

4200 REM "Cassette Diagnostic Module"
4210 GOSUB 2900
4220 PRINT TAB(10) "Cassette Diagnostic":PRINT
4230 PRINT TAB(10) "1...Write/Read Test":PRINT
4240 PRINT TAB(10) "2...Read Only Test":PRINT
4250 PRINT TAB(10) "Enter Command"
4260 GOSUB 3000
4270 IF A$="1" THEN 4310
4280 IF A$="2" THEN 4420
4290 GOSUB 3100
4300 GOTO 4260
4310 GOSUB 2990:PRINT TAB(13)"Write/Read Test":PRINT
4320 PRINT "Place a blank tape in the cassette drive"
4330 PRINT "and press RECORD.":PRINT
4340 GOSUB 3300:GOSUB 3000:PRINT
4350 OPEN 1,1,1, "Test-Tape 1"
4360 FOR I=1 TO 100
4370 PRINT#1,CHR$(170):NEXT
4380 CLOSE 1:OPEN 1,1,1,"Test-Tape 2"
4390 FOR I=1 TO 100
4400 PRINT#1 CHR$(85):NEXT
4405 CLOSE 1
4410 PRINT:PRINT "Rewind the tape and press PLAY"
     PRINT:GOSUB 3300:gosub 3000
4415 GOTO 4460
4420 GOSUB 2990
4430 PRINT TAB(13) "Read Only Test":PRINT
4440 PRINT "Place the test tape in the cassette drive"
4450 PRINT "and press PLAY"
4455 PRINT:GOSUB 3300:GOSUB 3000:PRINT
4460 OPEN 1,1,0,"Test-Tape 1"
4470 FOR I=1 TO 100
4480 INPUT#1,A$
4490 IF A$<>CHR$(170) THEN E=E+1
4495 NEXT
4500 CLOSE 1
4510 OPEN 1,1,0,"Test-Tape 2"
4520 FOR I=1 TO 100
4530 INPUT#1,A$
4540 IF A$<>CHR$(85) THEN E1=E1+1
4550 NEXT I
4560 CLOSE 1:PRINT
4570 PRINT TAB(3) "There were ";E;" High Value errors"
     :PRINT
```

```
4580 PRINT TAB(3) "There were ";E1;" Low Value errors"
     :PRINT
4590 GOSUB 3200
4600 GOTO 50

5000 REM: "Disk Drive Diagnostic Module"
5010 FOR I=1 TO 8:HV$=HV$+CHR$(170):NEXT
5020 LV$="":FOR I=1 TO 8:LV$=LV$+CHR$(85):NEXT
5030 GOSUB 2990:PRINT TAB(10)"Disk Drive Diagnostic":PRINT
5035 INPUT "Which Drive Do You Want To Test-";D:PRINT
5036 IF D<8 OR D>15 THEN PRINT "Invalid Disk Number":
     GOTO 5035
5037 INPUT "Enter Drive Type 1-1541 2-1571";T
5038 IF T<>2 THEN T=1
5040 BF$=HV$:NM$="High Values"
5050 FOR I=1 TO 2
5060 OPEN 1,D,2,"TEST,w"
5070 PRINT:PRINT"Writing ";NM$;" to Disk"
5072 IF T=2 THEN Z=33600
5074 IF T=1 THEN Z=16800
5075 FOR A=1 TO Z
5077 B1$=""
5080 PRINT#1,BF$
5090 NEXT A
5095 CLOSE 1
5100 OPEN 1,D,2, "TEST,s,r"
5110 PRINT "Reading ";NM$;" from Disk":PRINT
5115 FOR A=1 TO Z
5120 INPUT#1,B$
5140 IF B$<>BF$ THEN PRINT "Error Reading ";NM$:E=E+1
5150 NEXT
5160 CLOSE 1:SCRATCH "TEST"
5165 OPEN 15,8,15,"s0:TEST":CLOSE 15
5170 BF$=LV$:NM$="Low Values":NEXT I
5172 CLOSE 1:SCRATCH "TEST"
5175 IF EX=1 THEN RETURN
5180 PRINT:PRINT "There were ";E;" errors encountered"
     PRINT
5182 PRINT TAB(5)DS$:PRINT
5185 IF EX=1 THEN RETURN
5190 GOSUB 3200
5200 GOTO 50

6000 GOSUB 2990:REM "Serial Comm Module"
6010 PRINT TAB(3) "Serial Communications Diagnostic":
     PRINT
```

```
6020 PRINT TAB(7)"Press CLR/HOME to End Test":PRINT
6025 GOSUB 3300
6030 GOSUB 3000:GOSUB 2990
6040 OPEN 2,2,3,CHR$(6)+CHR$(0)
6050 GET #2,A$
6060 IF A$="" THEN 6070
6065 PRINT A$:
6070 GET A$:IF A$="{CLR/HOME}" THEN 6100
6075 IF A$="" THEN 6050
6080 PRINT#2,A$;:GOTO 6050
6100 CLOSE 2:GOTO 50

7000 D$="":DIM D$(10):GOSUB 2990:
     REM "Joystick Diagnostic Module"
7002 FOR I=1 TO 8
7004 READ D$(I):NEXT
7006 DATA "NORTH","NORTHEAST","EAST"
7007 DATA "SOUTHEAST","SOUTH"
7008 DATA "SOUTHWEST","WEST","NORTHWEST"
7010 PRINT TAB(10) "Joystick Diagnostic":PRINT
7020 PRINT TAB(7) "Which Port to Test 1 or 2-";P
7030 IF P<>2 THEN P=1
7040 PRINT:PRINT "Press Fire Button to End Test"
7050 A=JOY(P):IF A=128 OR A>128 THEN 50
7060 PRINT D$(A):GOTO 750

7200 EX=1:GOSUB 2990:REM "Exerciser Module"
7210 PRINT TAB(10)"Exerciser Diagnostic":PRINT
7220 INPUT "Test Monitor (Y or N)-";MN$:PRINT
7230 INPUT "Test Printer (Y or N)-";PT$:PRINT
7240 INPUT "Test Disk Drive(s) (Y or N)-";DD$
7250 IF DD$="N" OR DD$="n" THEN 7310
7260 PRINT:INPUT "How many drives to test-";DD
7265 DN="":DIM DN(DD):DIM DT(DD)
7270 FOR I=1 TO DD
7280 PRINT:INPUT "Enter drive number for drive-";DN(I)
7290 PRINT:INPUT "Enter drive type 1-1541 2-1571-";DT(I)
7300 NEXT
7310 IF MN$="N" OR MN$="n" THEN 7330
7320 A$="X":A=50:GOSUB 1140
7340 IF PT$="N" OR PT$="n" THEN 7380
7350 OPEN 1,4,7
7360 A$="E":X=10:GOSUB 3650
7370 CLOSE 1
7380 IF DD$="N" OR DD$="n" THEN 7310
7390 FOR I=1 TO DD
7395 GOSUB 2990
```

```
7400 D=DN(I):T=DT(I):GOSUB 5040
7410 NEXT
7420 PS=PS+1:GOTO 7310
7430 GOSUB 2990:CLOSE 1
7440 PRINT TAB(10)"Exerciser Diagnostic":PRINT
7450 PRINT TAB(3)"Total Passes through Exerciser-";PS
7460 GOSUB 3200
7470 EX=0:GOTO 50
```

Appendix C

Commodore 128 BASIC Programming

This book provides the System Diagnostic Program in two versions of Commodore BASIC: one for the Commodore 64, and the other for the Commodore 128. Although the two versions are very similar to one another, there are some fundamental differences in command availability. The Commodore 128 has a larger BASIC command set, which is summarized in this appendix. This alphabetical summary can be used to help you convert Commodore 64 programs to run successfully on the Commodore 128. Unless otherwise indicated, the command is not available on the Commodore 64.

APPEND—Allows a disk file to be opened at EOF in order to add new records.
AUTO—Allows auto-numbering of BASIC programs.

BACKUP—Allows an entire disk to be copied.
BANK—Selects one of the 16 memory banks.
BEGIN/BEND—Allows a multiline set of instructions to be used with an IF-THEN condition check.
BLOAD—Allows a binary file to be loaded from disk.
BOOT—Loads and runs a machine-language program from disk.
BOX—Draws a square or rectangle on a high-resolution screen. The Commodore 64 requires numerous PEEK and POKE instructions to accomplish this.
BSAVE—Saves a binary file to disk.
BUMP—Detects collisions between sprites. The Commodore 64 requires the examination of a collision register located at 53278. Each sprite is represented as a bit in the collision register. A value of 1 for the sprite's bit indicates a collision.

CATALOG—Lists programs on the current disk.
CHAR—Prints a string of characters in a given location on the screen.
CIRCLE—Draws all or part of a circle or ellipse.
COLLECT—Frees up available sectors on a disk.
COLLISION—Calls up a subroutine when a sprite collision is detected. See *BUMP* for the manner in which the Commodore 64 handles collisions.
COLOR—Sets screen, background, border, and foreground color. The Commodore 64 requires the POKE 53270, PEEK (53270) OR 16 instruction in conjunction with PRINT CHR$ (X), where X is the numeric value for a particular color.
CONCAT—Combines one file into two.
COPY—Makes a copy of a disk file.

DCLEAR—Closes open disk files and clears disk channels.
DCLOSE—Closes disk files. The Commodore 64 uses the CLOSE command only.
DEC—Converts a hexadecimal number to decimal.
DELETE—Deletes a disk file.
DIRECTORY—Displays a disk's directory.
DLOAD—Loads a BASIC program from disk. The Commodore 64 uses the instruction LOAD "filename", X where X is the drive number.
DO—Starts a DO/LOOP. Uses WHILE or UNTIL as the conditional breaks.
DOPEN—Opens a disk file. The Commodore 64 uses the OPEN command, as in OPEN 1,8,2, "filename,W".
DRAW—Draws a line between coordinates X and Y. The Commodore 64 uses POKE instructions to define bitmapped graphics.
DS—Returns the disk error code.
DSAVE—Saves the current program to disk. The Commodore 64 uses the SAVE command, as in SAVE "program", X where X is the drive number.
DS$—Returns the disk status.
DVERIFY—Verifies DSAVEd disk to the program in memory. The Commodore 64 uses VERIFY to perform this function.

EL—Contains the line number of the most recent error.
ENVELOPE—Defines the shape of a sound. The Commodore 64 requires POKE instructions to SID registers.
ER—Returns the most recent error code.
ERR$—Returns the most recent error message.
EXIT—Marks the end of a DO loop.

FAST—Causes the CPU clock to double in speed.
FETCH—Passes values from expansion memory to main memory.
FILTER—Creates sound effects. The Commodore 64 uses POKE instructions into SID registers.

GETKEY—Waits for a key to be pressed, and places the value into a variable. The Commodore 64 uses a GET loop, as in GET A$:IF A$-"" THEN 10.
GO 64—Puts the system into Commodore 64 mode.
GRAPHIC—Sets the screen display mode. The Commodore 64 requires direct POKE instructions into memory in order to set the modes.
GSHAPE—Displays a graphic shape in high-resolution mode.

HEADER—Formats a blank disk. The Commodore 64 requires a disk channel command.
HELP—Highlights the line where an error was encountered.
HEX$—Converts a decimal number into hexadecimal.

INSTR—Finds the position of a substring within a larger string.

JOY—Returns the current state of a joystick. The Commodore 64 requires direct PEEK instructions into memory to return these values.

KEY—Controls function keys.

LOCATE—Moves the pixel cursor to the desired XY coordinates. The Commodore 64 requires a direct POKE instruction into high-resolution memory.
LOOP—See *DO*.

MONITOR—Turns on the machine-language monitor.
MOVSPR—Positions or moves a sprite on the screen. The Commodore 64 requires a direct POKE instruction into sprite registers.

PAINT—Fills an area of the screen with a specific color. The Commodore 64 requires direct POKE instructions into high-resolution memory.
PEN—Returns the coordinates of the light pen. The Commodore 64 requires direct PEEK instructions into memory.
PLAY—Plays a musical note. The Commodore 64 requires direct POKE instructions into the SID registers.
POINTER—Returns the memory address of a variable.
POT—Returns information about a game paddle. The Commodore 64 requires direct PEEK instructions into memory.

RCLR—Reads color settings. The Commodore 64 requires PEEK instructions into memory.
RDOT—Returns the X or Y coordinate of the pixel cursor.
RECORD—Sets the record pointer to a specific record in a relative record file.
RENAME—Renames a disk file.
RENUMBER—Renumbers a BASIC program.
RESUME—Continues program execution after error trapping.
RGR—Returns the current graphic mode.
RIGHT$—Returns the right portion of a string.
RREG—Reads the 8502 registers.
RSPCOLOR—Reads the color of a sprite. The Commodore 64 requires a PEEK instruction into the sprite register.
RSPPOS—Returns the location XY coordinates and speed of a sprite. The Commodore 64 requires a PEEK instruction into the sprite registers.
RWINDOW—Returns the screen format information.

SCALE—Sets the high-resolution coordinate system to values given.
SCNCLR—Clears the screen. The Commodore 64 uses the PRINT {CLR/HOME}" instruction.
SCRATCH—Deletes a disk file.
SLEEP—Pauses program execution for a specified number of seconds.

SLOW—Sets the CPU clock speed to one megahertz.
SOUND—Plays a musical note. The Commodore 64 requires a direct POKE statement into the SID registers.
SPRCOLOR—Sets the color of sprites. The Commodore 64 requires a direct POKE instruction into the sprite registers.
SPRDEF—Allows definition of sprite shapes.
SPRITE—Activates and deactivates sprites. The Commodore 64 requires direct POKE instructions into sprite registers.
SPRSAV—Transfers sprite shapes between different sprites and string variables.
SSHAPE—Places a high-resolution shape into a string variable.
STASH—Copies a section of memory into expansion memory.
SWAP—Exchanges two sections of memory.

TEMPO—Controls the time elapsed between tones being sounded. The Commodore 64 requires POKE instructions into the SID registers.
TRAP—Prevents errors from halting a program.
TRON—Activates the BASIC trace facility used for debugging.
TROFF—Deactivates the BASIC trace facility.

UNTIL—Is used with a Do loop.
USING—Is used with PRINT to define a format for strings or numbers.

VOL—Sets the music volume. The Commodore 64 requires direct POKE instructions into SID registers.

WHILE—Is used with a Do loop.
WIDTH—Sets the width of a point plotted on the high-resolution screen.
WINDOW—Defines a screen window.

XOR—Performs an exclusive OR on two numbers.

Index

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.
Blue Ridge Summit, PA 17214-9988

COMMODORE™ CARE MANUAL:

Diagnosing and Maintaining Your 64 or 128 System

CHRIS MORRISON AND TERESA S. STOVER

Easy-to-follow, professional advice on money-saving maintenance, troubleshooting, and simple repair techniques!

This book is your guide to the care and repair of your Commodore 64 or 128. You need no previous experience! *PC Care Manual: Diagnosing and Maintaining Your 64 or 128 System* provides you with three basic approaches to Commodore care: preventive maintenance, problem diagnostics, and general repair.

PREVENTIVE MAINTENANCE: General practices that will help keep your Commodore healthy are outlined. Periodic maintenance procedures that can ward off system problems, and troubleshooting techniques to help you isolate problems, are explained. All of the major Commodore subsystems are addressed.

DIAGNOSTICS: A "System Diagnostics Program" written in BASIC is presented in both C64 and C128 versions. This program automatically checks the components of the system and reports on their status.

REPAIR GUIDELINES: Specific problems are discussed within chapters on the individual components. These "Troubleshooting and Repair" sections list step-by-step instructions that will often enable you to fix problems all by yourself.

Chris Morrison is currently employed as customer support manager for a computer-aided publishing manufacturer in San Jose, CA. He also manages the company's technical publications group which writes and produces user guides and marketing literature.

Teresa S. Stover is a technical writer who has written manuals for various Silicon Valley firms, including business computer systems consultants, circuit board designers, and software developers. She has also published articles in several nationally distributed trade journals.

TAB

TAB BOOKS Inc.
Blue Ridge Summit, PA 17294-0850

FPT > $16.95

ISBN 0-8306-3141-0

PRICES HIGHER IN CANADA 0688