WINDCREST

3121

Apple Care Manual

Diagnosing and Maintaining Your Apple® II+, IIe and IIc Computers

Chris Morrison and Teresa S. Stover

APPLE CARE MANUAL

Diagnosing and Maintaining Your Apple *II*+, *II*e and *II*c Computers

Chris Morrison & Teresa S. Stover

WINDCREST

Trademark List

Apple, Apple II Plus, Apple IIe, Apple IIe, and Applesoft are registered trademarks of Apple Computer, Inc.

CompuServe is a registered trademark of CompuServe Consumer Information Service.

Dow Jones News/Retrieval is a registered trademark of Dow Jones & Company, Inc.

ImageWriter and ImageWriter II are trademarks of Apple Computer, Inc.

Panasonic is a registered trademark of Matsushita Electrical Industrial Co., Ltd.

The Source is a trademark of Source Telecomputing Corporation.

VT100 is a trademark of Digital Equipment Corporation.

Published by **Windcrest Books**
FIRST EDITION/FIRST PRINTING

Library of Congress Cataloging-in-Publication Data

Morrison. Chris.
Apple care manual : diagnosing and maintaining your Apple II+, IIe, and IIc computers / by Chris Morrison and Teresa S. Stove.
p. cm.
Includes index.
ISBN 0-8306-3121-6 (pbk.)
1. Apple II+ (Computer—Maintenance and repair. 2. Apple IIe (Computer)—Maintenance and repair. 3. Apple IIc (Computer)--Maintenance and repair. I. Stover, Teresa S. II. Title.
TK7889.A663M37 1989
621.391'6—dc19 88-36587
CIP

TAB BOOKS Inc. offers software for sale. For information and a catalog, please contact TAB Software Department, Blue Ridge Summit, PA 17294-0850.

Questions regarding the content of this book should be addressed to:
Windcrest Books
Division of TAB BOOKS Inc.
Blue Ridge Summit, PA 17294-0850

Ron Powers: Director of Acquisitions
Heilan Yvette Grimes: Technical Editor
Katherine Brown: Production
Jaclyn B. Saunders: Series Design

Contents

Program Listings

Acknowledgments

We'd like to thank the following treasures of humanity and intelligence for their invaluable assistance in this book:

Craig A. Stover for providing the technical and editorial review of the manuscript, shooting the photographs, and instilling the "Stoverized" insistence for perfection.

Pat A. Mahony for the thorough editorial review of the manuscript, encouragement and enthusiasm, support and friendship.

Ilva Klar for creating the manual line drawings and attaining new pinnacles of excellence.

Robert Dierkes for testing the program and catching those pesky bugs.

Acknowledgments also go to Qubix Graphic Systems, Inc. for allowing Chris to create the electronic line drawings on the Qubix Designer Workstation.

Introduction

You will find that your Apple II computer is quite a marvel. You might use the Apple II+, IIe, or IIc computer to help manage a small business, do schoolwork, budget home finances, keep records, write, or play games. The Apple can help you to become more efficient and organized. It can provide you with information, resources, and capabilities formerly unavailable.

But just when you're really rolling, you get your first silicon slap-in-the-face: your computer breaks down! Only at that moment do you realize that your computer is not a magical wonder—it's an electromechanical applicance, similar to your television, stereo, or automobile. It has moving mechanical parts that wear out. It has integrated circuit chips and other electronic components that short out and fail. And there are a number of other little problems that can cause big worries.

If your computer is beginning to show signs of age and wear, you may have already experienced trips to a computer service center, with the attendant high-priced repair bills. Although you may not be a computer technician, and may not know the difference between DC and AC electronics, there are ways to assist you in having more control over the care of your computer, thereby saving yourself time and money. If you have a conscientious desire to take better care of your Apple II system, to diagnose problems as they surface, and to make simple

repairs, this book will help you attain that goal. You need not be technically or mechanically inclined in order to use the material in this book successfully. You also do not need to know how to write or understand computer programs. All that is required is that you are able to use a screwdriver, and read and follow the prompts in a program.

This book will provide procedures and techniques for the troubleshooting, maintenance, and repair of your Apple II system. A unique system diagnostic program is also provided to help you with the troubleshooting tasks. This BASIC program tests for and diagnoses various subsystem problems, and exercises the system as a whole. In addition, this program is structured in a modular fashion to allow you to expand and customize it for your particular system configuration.

MAINTENANCE AND REPAIR TECHNIQUES

Chapter 1 covers general computer maintenance, troubleshooting, and repair techniques. It provides the foundation necessary to make use of the specific procedures and instructions found in the subsequent subsystem chapters under the "Troubleshooting and Repair Guidelines" sections.

The repairs described in Chapter 1, as well as in the later subsystem chapters, do not require knowledge or expertise in electronics. You do not need to know how to solder, nor do you need to know how to troubleshoot a series of integrated circuit chips. This book details the steps necessary for diagnosing a problem, and once this is done, for effecting a repair. You are even given suggestions as to when the repair of a subassembly should not be attempted, and when it should be taken in for professional servicing.

THE SYSTEM DIAGNOSTIC PROGRAM

The System Diagnostic Program provided with this book assists in troubleshooting. This BASIC program includes eight modules, each designed to perform a series of tests on a different component or aspect of the system. The modules that make up the System Diagnostic Program are as follows:

- Main Menu Module
- Keyboard Module
- Monitor Module
- Printer Module
- Disk Drive Module
- Serial Communication Interface Module
- The System Exerciser Module
- The Game Paddle Module

You can buy the program on a diskette (refer to the last page of this book), or you can key it in yourself (refer to the Appendix).

Chapter 2 explains how to load and run this program. In addition, the Main Menu module is explained, just as the subsystem modules are explained in subsequent chapters. A brief overview of the Applesoft BASIC commands used in the System Diagnostic Program is also included.

THE SUBSYSTEMS

Chapters 3 through 8 each specialize in a particular Apple II subsystem:

- system unit
- keyboard
- monitor
- printer
- disk drive
- serial communication interface

As it pertains to the specific subsystem, each of these chapters includes:

- a brief theory of operation
- specific preventive maintenance procedures
- a list of possible problems
- troubleshooting and repair techniques
- an explanation of the diagnostic program module

Subsystem Theory of Operation

The theory of operation provides a general idea of how the subsystem does its job. When you know what a subsystem is supposed to do, and how it does it, you have more of the information needed to successfully troubleshoot and diagnose any subsystem problems that might arise.

Subsystem Preventive Maintenance

Preventive maintenance can go a long way to help you to avoid system problems. Following these procedures will help you maintain the computer in its best possible condition.

Subsystem Troubleshooting and Repair Guidelines

If a problem with your computer does surface, refer to the appropriate subsystem chapter. For example, if the printer is not working correctly, refer to Chapter

6, The Printer. Within that chapter, refer to the "Troubleshooting and Repair Guidelines" section. Then look up the name of the problem that fits what the system is experiencing. Follow the troubleshooting and repair instructions in the order given. This becomes the computer's treatment plan, helping you to isolate the problem and then take the appropriate action.

You can often fix the problem yourself by following these instructions. If it's a more complex problem that warrants taking your system to a computer service center, you can do so with a more in-depth explanation, a diagnosis of the problem, and a specific focus of what must be done. This saves the technician repair time, and thus saves you money.

The Subsystem Diagnostic Module

The System Diagnostic Program lets you gather information 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 its own section, along with illustrations of expected test results. After performing 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 program logic. This can also help you modify the program for your own specific purposes, or create new modules for additional subsystems and peripherals.

If you are not familiar with BASIC, you can easily skip these 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.

THE SYSTEM EXERCISER

Chapter 9 covers the System Exerciser Module, which burns in and thoroughly tests the system when it is new, or after a repair. All system functions are exercised and run continuously with the Exerciser. This is much like test-driving your car after a repair. The Exerciser checks whether your computer has an inherent problem, or whether your problem has indeed been repaired as expected.

Your computer is a system of many interrelated components, and as such, when something goes wrong in one part of the computer, it might actually be caused by something in another part of your computer. The Exerciser Module of the System Diagnostic Program ensures that the system as a whole is working properly.

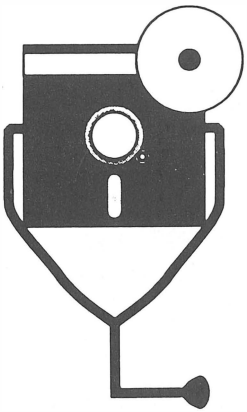
WRITING YOUR OWN DIAGNOSTICS

Finally, Chapter 10 details the techniques used in developing diagnostic modules for other peripheral devices, such as digitizer tablets and joysticks, not already provided for in the System Diagnostic Program.

By demonstrating how the game paddle diagnostic module was developed, Chapter 10 presents a systematic diagnostic approach that starts by helping you learn how to predict problems that can occur with the device. Once you learn the types of problems that can occur, and you have the information needed to diagnose these problems, you can write a new routine appropriate to such a troubleshooting activity.

The procedures and techniques covered throughout this book will help you better understand the internal workings of your system. You will be able to maintain a well-running system less likely to experience long periods of downtime. You will find that you are saving time by not having to deal with a broken computer, and saving money with fewer trips to a computer service center. You might also gain a new understanding of BASIC programming, and will certainly gain more confidence resulting from the knowledge gained about your computer.

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.



1

Maintenance and Repair

This chapter will help you better understand the components that make up the Apple II series personal computer system. It will also cover general preventive maintenance, troubleshooting, and repair techniques that will provide a good foundation for the computer care details about which you will learn in subsequent chapters.

SYSTEM OVERVIEW

The typical Apple II computer system consists of the following components:

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

Although your particular system might include other peripherals, this is a basic configuration, resembling the system shown in Fig. 1-1.

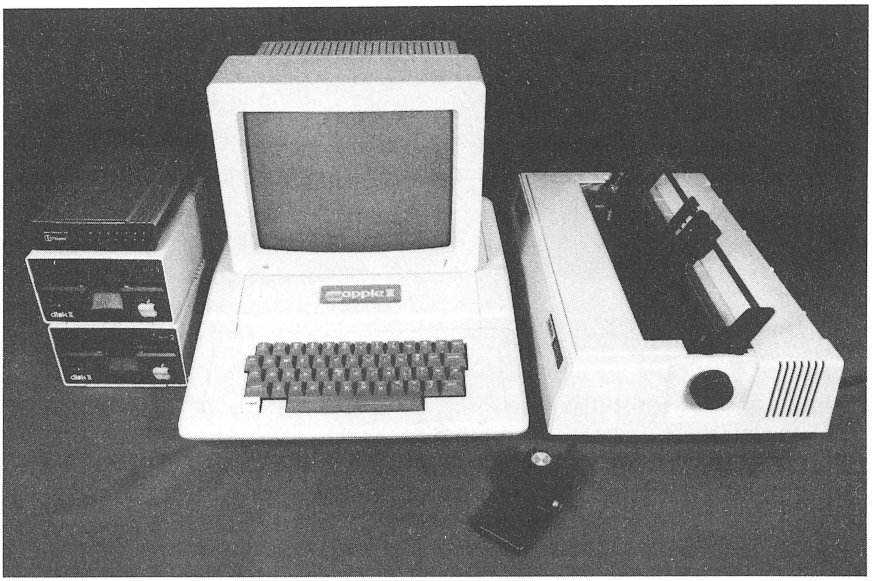


Fig. 1-1. The Apple II configuration.

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

System Unit

The system unit is the actual computer, and as such, is the single most important component of the computer system. On the Apple II systems, the system unit is enclosed in the same assembly with the keyboard.

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 as well as coordinating the various input and output devices. It transfers and processes data to and from the outside world via ports.

The CPU microprocessor chip resides on the *motherboard,* as do the video, cassette drive, and game paddle controllers. The various versions of the Apple II system — the II Plus, IIe, and IIc — have different CPU configurations.

The Apple II Plus system unit microprocessor consists of the 6502 CPU chip with 48 kilobytes of memory. It also includes eight expansion slots for additional peripheral boards such as a disk drive controller or serial communication interface board.

The system unit microprocessor for the Apple IIe includes the 6502B CPU chip, and 64 kilobytes of memory. Like the II Plus, the Apple IIe has eight expansion slots.

The Apple IIc uses the 65C02 microprocessor with 128 kilobytes of memory. The IIc is contained in a different housing than the II Plus and IIe systems. The case itself is smaller and more compact. There are two serial I/O ports for a printer and modem, a mouse interface, and two disk drive controllers: one for an internal drive, and one for an external drive. Further information about the system unit is found in Chapter 3.

Keyboard

The keyboard is the computer's primary input device. Typed characters form commands, move the cursor, and enter data. The keyboard controller generates the code of the characters entered, the system unit interprets and processes these keyboard signals, and the monitor displays key entries and processing results. Further information about the keyboard is provided in Chapter 4.

Monitor

The monitor, also known as a "cathode ray tube" (CRT) or "video display," is the visual output for work input on the Apple. It shows the input data from the keyboard, modem, mouse, game paddle, or other input device; it also displays the results of system processing performed within the system unit. The Apple II usually uses a color monitor, which can be either a standard television set or a computer monitor. The IIc also supports both 40-column or 80-column displays. Further information about the monitor is found in Chapter 5.

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 printer allows data and processing results to be transferred from the system and typed using a dot-matrix or letter-quality printer. To learn more about the printer, refer to Chapter 6.

Disk Drive(s)

The disk drive is used to store and retrieve software programs as well as data, recording information in electronic form on floppy diskettes. Chapter 7 covers disk drives.

Serial Communication Interface

The serial communication interface is the means used to enable the Apple II 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 via telephone lines. It can also interface with a printer, transmitting data to be printed, or interface directly with another computer. Further information about the serial communication interface can be found in Chapter 8.

TOOLS AND RESOURCES

Troubleshooting and repairing computers may seem to be a difficult task. However, most computer technicians go about their work methodically. They test various aspects of the system to gather information about the symptoms of the problem. Then they use this information to make logical deductions as to the nature of the malfunction. The system repair is effected by making adjustments and replacing faulty components.

To perform such troubleshooting and repair activities, computer technicians have specific knowledge and training about the particular systems they service, as well as complete sets of pertinent documentation and schematics. Technicians use a good set of tools, and have spare parts available to them. Additionally, they have various information resources upon 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 processes and techniques. Then, when your computer malfunctions or breaks down, you will be better prepared to analyze and then solve the problem 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 and its peripherals. This documentation provides the necessary specifics regarding your particular Apple II configuration. Such information becomes vital when installing and setting up a new device, or if the computer experiences a malfunction.

If you have inadvertently lost or thrown out important documentation, another copy can probably be obtained from your Apple II dealer or device manufacturer. You can also check with your library or Apple II user group for documentation to borrow or buy.

Tools

Tools let you open, bend, cut, insert, and connect things in the computer. The following tables list required tools and cleaning supplies. The tools make the job

of computer troubleshooting and repair possible. The cleaning supplies are vital, as cleaning is a major aspect of preventive maintenance.

When buying tools, try to obtain good quality tools. This does not necessarily mean you should buy the most expensive tools on the market, but they must 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. 1-2 illustrates these tools. Table 1-2 lists cleaning supplies, and Fig. 1-3 illustrates them. The tables list the functions, uses, and approximate costs for the tools and cleaning supplies.

Table 1-1. Tools.

Required Tools	Use
flat-blade screwdrivers	to tighten and untighten slotted screws
phillips screwdrivers	to tighten and untighten cross-slotted screws used for holding bolts down
needle-nose pliers	to grip and/or bend small objects as an extension or a "helping hand"
pliers	to grip and/or bend small objects, to hold larger nuts
wire strippers	to strip away plastic insulation
wire cutters	to cut wire
non-conductive screwdriver "tuning wand"	to tighten and untighten screws in an electrically charged environment
key extractor	to remove keys from the key post on the keyboard
chip extractor	to remove a socketed IC chip from its socket
metal flat file	to smooth or grind down a plastic surface such as a keycap

Table 1-2. Cleaning Supplies.

Cleaning Supplies	Use
compressed air with nozzle and extension	pressurized air used to blow out dust and dirt
freon or contact cleaner with nozzle and extension	solution used to clean contacts and switches on printed circuit board components
denatured isopropyl alcohol	for general cleaning of electronic hardware, floppy disk drive heads, and print wheels
cotton swabs	for cleaning hardware and components in tight places; used with isopropyl alcohol
sponges and lint-free paper or cloth	for cleaning hardware components with isopropyl alcohol
emery paper or pencil eraser	for cleaning oxidation from edge connectors and cable pins

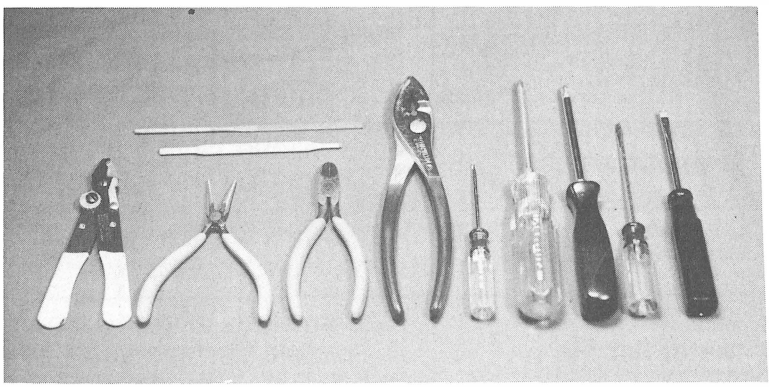


Fig. 1-2. Tools.

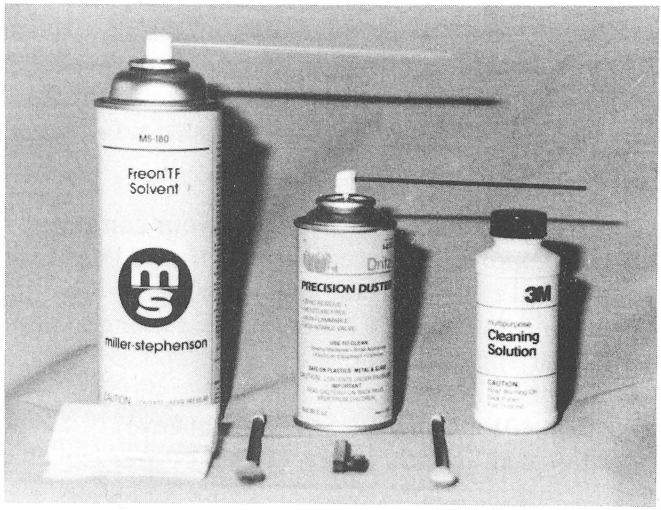


Fig. 1-3. Cleaning supplies.

Spares

Professional technicians usually have an inventory of spare parts associated with the equipment they work on. The spares help them in troubleshooting, when they 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 your own inventory of spare parts for the Apple II could be costly and impractical. An alternative is to find someone (a friend, a colleague at work, an acquaintance in a user group) who owns the same type of Apple II system. Work out a deal with this person in which you can temporarily swap parts with each other when one of your systems is malfunctioning. You can use this person's system for troubleshooting by temporarily swapping suspect parts to find exactly where the problem is.

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

Find out who has 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 the back pages of most computer magazines, and even the daily newspaper, are good sources for used parts. Computer flea markets and

swap meets are also good sources for used parts at low prices. The Apple II systems have been popular computers for several years. As a result, used parts and peripherals should be readily available.

User Resources

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

If you still cannot find the source of a problem after trying all the troubleshooting steps, call your local Apple II dealer. Many stores and service outlets will assist with valuable information if you can ask specific questions.

That friend of yours with an Apple II system might have knowledge you do not. Exchange information with any other Apple II users of your acquaintance. Everyone has a different baseline of computer experience and knowledge.

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

Read the Apple II computer magazines such as *A+* (Ziff-Davis Publishing Company, Ziff Communications Company, One Park Avenue, New York, NY 10016), *NIBBLE* (Micro SPARK, Inc., 52 Domino Drive, Concord, MA 01742), and *inCider* (IDG Communications/Peterborough, Inc., 80 Elm Street, Peterborough, NH 03458). Such magazines provide good advice, information, and further resources.

General Troubleshooting and Repair Guidelines

This book is broken down in chapters according to major computer subsystems. If you're having trouble with the keyboard, for example, refer to Chapter 4, "The Keyboard." The chapter describes specific preventive maintenance for the keyboard, possible keyboard problems, and steps to take in troubleshooting and repairing them. Part of this troubleshooting process includes running one of the diagnostic modules. The chapter explains 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 the source of the problem, 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 and even repairing diagnosed problems on the Apple II.

If you run one of the diagnostic tests and the results indicate a problem, you might wish to double-check the diagnostic on an identical system to ensure that the diagnostic program is providing accurate results before spending any time and money on repair. This is especially true if you have modified the program in any way.

Limitations

Never try to fix something for which you do not have the proper knowledge, documentation, tools, or parts. A little knowledge on your part can be a dangerous thing to your computer, especially when you know just enough to get into trouble and possibly make the problem worse.

Never attempt any procedure you feel you cannot handle. Do 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 place your computer in needless jeopardy.

Don't try any troubleshooting or repair procedure on a working system just because it sounds like fun and you want to experiment. If it works, don't fix it. Preventive maintenance procedures are the only procedures that should be performed on a well-running system.

MAINTAINING YOUR APPLE II SYSTEM

Preventive maintenance and good working habits are the two keys to minimum system *downtime* and maximum system longevity. Many system problems can be avoided if the system is set up properly in a non-threatening environment, if good working habits are established; if preventive maintenance procedures are performed regularly; and if basic troubleshooting and repair techniques are followed.

Safety

Certain safety precautions must be followed whenever working with any kind of electrical device.

Before disconnecting cables or opening any part of the system, unplug the ac connector from its source, usually the electrical outlet or ac strip. Never work with an open system if it is "live," or electrically powered.

Even when the electricity is disconnected from an open system, ground yourself by touching metal before touching the system. When you are grounded, electrical charges are not attracted through your body, and you are prevented from electrical shock.

When you make adjustments on one of the printed circuit boards, use a tuning wand, which is a non-conductive screwdriver. This prevents any electrical currents residing in the board from being conducted into your hand and through your body.

Never open the monitor, because it contains dangerous capacitors and high voltage that is stored and can be discharged through you, causing severe electrical shock. If the system has a problem that requires opening the monitor, take it in for servicing, and have a professional do it for you. Such technicians are certified in high-voltage electrical work.

Setup

When you set up your system for the first time, be sure to follow the installation instructions provided in the Apple II user guide. Situate the computer in a clean, cool, relatively dust-free environment. Place the computer in as permanent a location as possible. The less the computer is moved, the fewer problems it will have.

Set all parts of your system on a flat, secure surface. For example, don't teeter any part of the system on the edge of a table. Don't have loose cords hanging where someone could trip and hurt themselves as well as the computer.

Do not place your computer near any power supplies, appliances, or any other items that contain some type of motor. A motor creates a magnetic field, which can interfere with the data written magnetically on your diskettes. This magnetic interference could easily erase or garble your programs and data.

Several interface cables can be connected to your system unit at the rear, to provide pathways from the controller circuits to the peripheral devices. The cable connectors plugged into the system unit receptacles can work loose when a part of the system is moved. If the cable loosens enough to where it's not making the proper connection, the peripheral will stop working.

Make sure that the keyboard/system unit is closed. The system unit cover attaches to the unit with interlocking plastic fasteners. Make certain these fasteners are properly snapped down, and replace them if they aren't working correctly. If the unit pops open, it can cause improper contacts while working.

Everyday Working Habits

Good everyday working habits are vital to the health of your computer. Measures such as using dust covers, an antistatic mat, and a surge protector can go a long way in preventing system problems.

Using 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. Contacts — the exposed wires, or legs, of the integrated circuit (IC) chips on the system's printed circuit boards — have electric current flowing through them,

which can attract particles of dust. The dust, in turn 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.

In addition, dust and dirt accumulating under the keycaps of the keyboard can cause contact problems. In this case, you might press a key and have nothing happen, because there is so much dirt that the pressed key can no longer send the signal to the keyboard controller.

A piece of dust inside the disk drive can cause problems there as well. The drive head travels mere microns above the surface of the diskette. If a piece of dust gets in the way of the drive head, it can cause data to be read or written incorrectly.

Accumulated dust inside the printer mechanism can cause poor-quality printing.

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 or other damage caused by dust. Get into the habit of covering the system when not in use.

Preventing Static Electricity. Static electricity is another computer foe. Static electricity, also known as electrostatic discharge, refers to the giving off of electrical charges that are contained in a particular object, or produced by friction, for example. Static electricity can present a serious problem to your computer. This is because the IC chips throughout the system contain minute electronic circuits that are particularly vulnerable to damage by large voltages inherent in static electricity. If you live in a dry, cool climate, the potential for static electricity is greater, so your computer stands an even greater risk of getting "zapped."

Most devices within the computer run on power of up to about 15 volts, and are protected from electrical damage up to 2-5 kilovolts. However, the electrostatic discharge from a human can be more than 20 kilovolts. This means that you could easily be the source of static electricity that can damage your system's components.

There are several ways to protect your computer from static electricity. Spray the floor or carpet around your system with an anti-static solution. Additionally, lay down an anti-static mat under your computer. Anti-static spray as well as the mat can be obtained at an office or computer supply store. The least expensive solution is to simply touch something metal that's not part of the system, perhaps a desk lamp or file cabinet, before touching the computer. This grounds you and prevents the transfer of static electricity to the computer.

Do not touch the metal edge connectors, when handling a printed circuit board within the system. This will discharge electricity in your body through the printed circuit board to the nearest IC chip.

Power Surge Protection. The electricity supplied at the electrical wall outlet 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 nominal. You've probably experienced drops in electricity when the lights flicker or dim in intensity. This *brown-out* effect might cause the 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 the power supply's filter circuits by giving it more peak voltage than it can effectively suppress. The Apple II 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 or spike, it is not capable of suppressing it. This instantaneous spike is sent through to the chips, damaging or even destroying them. The spike not only damages circuits but also causes the system to fail, wiping out everything you're working on currently stored in system memory.

Because boards are expensive to repair and many chips are expensive to replace, and because your work and your time are valuable, protect the computer from these indiscriminate voltage surges by using a surge protector, which isolates the voltage surges and spikes in order to provide the system with a constant 110/120-volt supply. You can buy a surge protector at a computer or electronics supply store. Connect the AC cord into the surge protector, and plug the surge protector into the AC outlet. Many power strips have built-in surge protectors as well. Using a surge protector with the computer will extend the life of its integrated circuits, and will also prevent loss of data while you're working.

Food and Drink. Don't eat, drink, or smoke near your computer. For example, if you spill or drip something into the keyboard, you will have an emergency cleaning job on your hands, or 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 system components out and causing permanent damage.

Ashes, cookie crumbs, sandwich particles, dinner bits, can all end up inside the keyboard and system unit if you smoke or eat while working. This can destroy the keyboard contacts and render the keyboard useless. It can also damage components within the system unit.

Preventive Maintenance Techniques

Preventive maintenance consists of those procedures which are performed as part of periodic general system care. Preventive maintenance procedures help keep the system running properly, and prevent breakdowns. The following preventive maintenance procedures are applicable to the system as a whole. More detailed procedures about individual subsystems are covered in the "Preventive Maintenance" section found in each subsystem chapter.

These preventive maintenance procedures should be performed regularly — at least once every three months. Do them more often if the computer is situated in a harsher environment than normal.

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. 1-4), and printer mechanism. Also, open the system unit cover and carefully vacuum or blow around the inside of the computer. This clears out any dust and dirt that works its way into the system in spite of the dust cover.

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

External Fan. If you use the Apple II Plus or IIe for more than a couple hours at a time, obtain an external fan. Sometimes the system experiences overheating, which can cause malfunctions in programs being run. An external fan clips onto the rear of the system unit and forces air through the vents to the inside of the system to keep it cool.

Disk Drive Head. The floppy disk drive head can accumulate a layer of oxidation after continued use. To remove this oxidation, use a disk drive head cleaning kit. These kits are available at most computer stores, and include a

Fig. 1-4. Dusting with compressed air.

special diskette permeated with isopropyl alcohol. Follow the instructions that come with the cleaning kit. This will consist of inserting the cleaning diskette into the disk drive, and entering a command that causes the diskette to rotate. In this manner, the read/write heads are flash-cleaned.

Printer. Dried ink and bits of paper can accumulate on daisy wheels and dot-matrix print heads. A lint-free gauze cloth moistened with denatured isopropyl alcohol (not rubbing alcohol) can be used to clean these.

General Cleaning. Denatured isopropyl alcohol can also be used to clean the exterior or interior of the system where dirt and dust have accumulated. Use cotton swabs lightly moistened with isopropyl alcohol to clean hard-to-access areas.

Troubleshooting Techniques

Troubleshooting is another word for problem-solving. When there is a computer malfunction, troubleshooting is a systematic approach with which you can discover what caused the problem so that you can effect a repair.

When a subsystem or even the entire computer isn't working, check the easy things first, such as items having to do with power and cabling.

- Does it have power?
- Are all power switches turned to the ON position?
- Is the unit plugged into an AC supply?
- Is the AC supply controlled by a light switch?
- Is the AC supply controlled by a power strip switch?
- Is the fuse blown?
- Is there a problem in the fuse box or circuit breaker?
- Are the cables properly seated?

If the power and cabling are in order, then try swapping the malfunctioning subsystem with one you know is working. First try another subsystem on your computer. If it works, then you know that your subsystem is defective. If it does not work, this probably indicates a problem with your system unit. To double-check this, try your malfunctioning subsystem on another system unit. If it works, this confirms a problem with your system unit.

Other general troubleshooting techniques include reseating controller boards, cleaning oxidation from cable pins and board edge connectors, and checking switch settings. Procedures for each of these are covered in Chapter 3, under "System Unit Troubleshooting and Repair Guidelines."

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 problem is probably in the system unit itself. Refer to Chapter 3, "The System Unit," for system unit troubleshooting procedures.

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

Repair Techniques

There are various techniques and tricks that technicians learn after doing repair work for a period of time. The following repair techniques can help you take advantage of some of this experience, especially if you have never attempted any kind of computer repair.

Working With Tools. Always make sure you're using the proper tool for the job at hand. Don't try to force one tool to do another tool's job for the sake of convenience. You'll end up making more work for yourself, because fittings can break and screws can be stripped, making them useless. Use the proper screwdriver size for the screw. Use flatblade screwdrivers for slotted-head screws, and Phillips screwdrivers for Phillips-head screws (Fig. 1-5). When making adjustments on one of the printed circuit boards, use a tuning wand instead of a metal screwdriver. This prevents electrical components from accidental shorting or bridging. Electrical shorts between components can cause serious damage to a circuit.

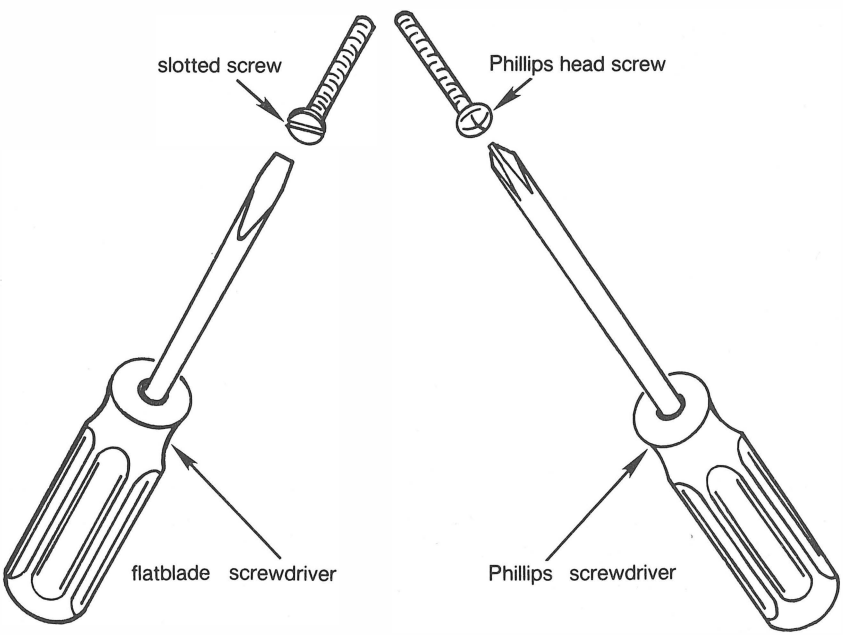


Fig. 1-5. Using the proper screwdrivers.

When disassembling a unit by removing the screws, keep all the screws in one place where they won't roll away and get lost. Putting them in a shallow bowl or ashtray is a good practice. If there are different types of screws coming out at different places, it's a good idea to keep these separate from each other. This will prevent you from inadvertently replacing the wrong screws in the wrong holes when reassembling.

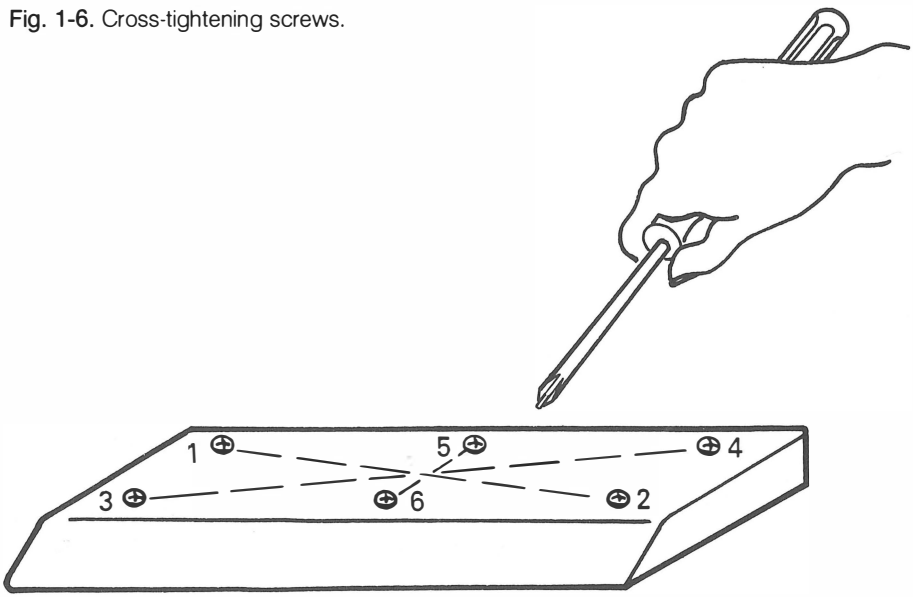
Remember, clockwise is on, counterclockwise is off. Turn the screw to the left to disassemble the unit. Turn the screw to the right to reassemble the unit.

When screwing a unit back together again, always place the screws into their holes with just a couple turns. This lets you situate and align the unit properly before tightening down. If a unit is being reassembled with more than three screws, tighten the screws down in a criss-cross fashion (Fig. 1-6). Start at one corner, then screw the opposite corner. Continue in this star manner until all the screws are tightened down. This allows for a properly aligned and tightened unit, and also prevents the screws from being stripped.

When replacing screws, do not force them into their holes. Turn the screw gently in its hole and wait until you feel the threads engage before tightening it down. If the screw should go in at an angle, and you force the screw in, this strips the screw threads, making the screw and its hole useless.

Never force anything when removing or replacing any part of the system. If the item is in its proper place, force is not necessary and will only cause damage.

Fig. 1-6. Cross-tightening screws.



Making Adjustments. When making adjustments with knobs or *potentiometers* (variable resistors), turn it exactly 1/4 turn at a time in a particular direction, clockwise or counterclockwise. Then, if you over-adjust, you will know how far in which direction to go back to correct the adjustment.

Working with PC Boards. When disconnecting cables in order to get to a PC board, note how the connector fits onto the receptacle. Most connectors are keyed so that they can go onto the receptacle only in one direction. Check each cable for orientation and keying before setting it aside. If it is not keyed, mark or tag it in some way so that you'll know how to plug it back in when reassembling the unit.

When disconnecting several cables at the same time, make sure you know where they are connected. If they each have different types of connectors, the job is easy. However, if they are all similar to one another, mark or tag the connectors in some way so that you'll know where to connect which cables.

The PC board can be removed once all the cables are disconnected. Hold it by its edges and gently rock the board out of its edge connector. Take care not to touch the underside of the board with its exposed IC pins, which could cut your fingers. Also, don't touch the metal edge connector. You could transfer electrical charges through all the circuits on the board and damage them.

When installing the board back into its edge connector, align it in place, then hold it by its edges and push it gently into the connector until it is firmly seated. Remember, never force anything.

Working with IC Chips. If your troubleshooting efforts determine that a particular IC chip is defective, you can easily replace the chip if it is socketed.

IC chips are either soldered onto the board, or socketed into chip sockets. *Socketed* chips can easily be removed and replaced. To do this, use a chip extractor to lift the chip off of its socket (Fig. 1-7).

There is usually a notch or dot on the chip as well as on the socket to help with alignment orientation when inserting a chip onto the socket. Have the notches of the chip and socket facing the same direction (Fig. 1-8). Carefully line the IC chip pins up with the socket holes, then gently press the chip into the socket as far as it will go. Be sure not to bend the IC pins while you're pressing it into the socket. This could render the chip useless.

If the chip needing replacement is soldered onto the PC board, take the board in to computer service. Soldering is a skill that takes a lot of practice, so have the professionals do it.

Cleaning. Whenever opening up a part of the system for troubleshooting or repair, take this opportunity to clean. Cleaning can often solve the current problem, as well as prevent future problems. Vacuum dust up, wipe with isopropyl alcohol on lint-free cloth or cotton swabs, and do any other cleaning appropriate to that unit.

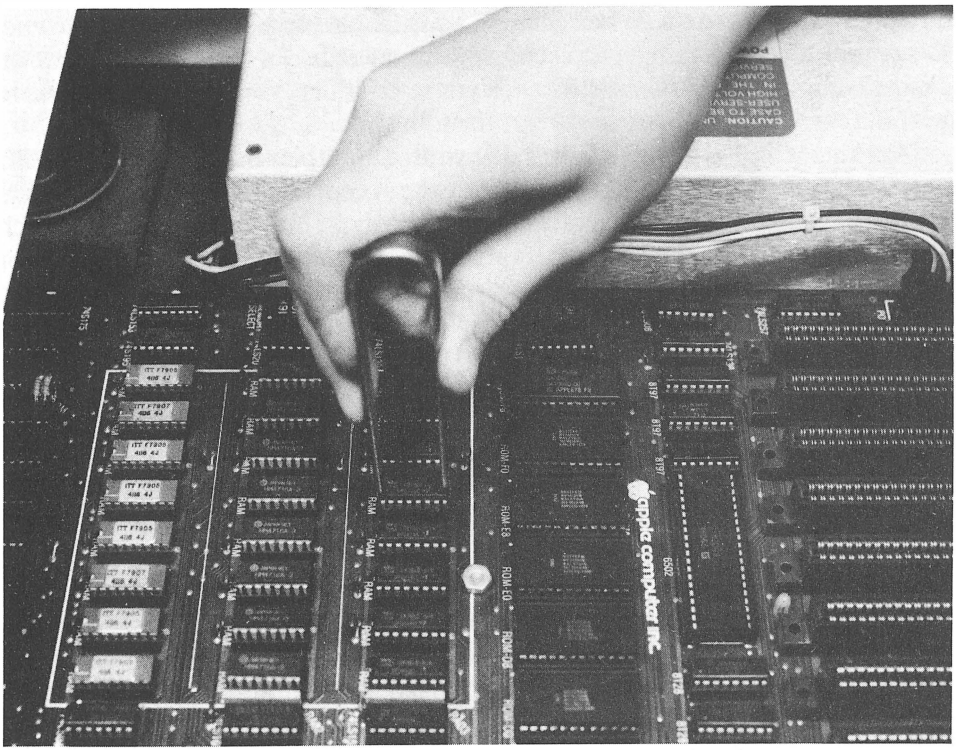


Fig. 1-7. Removing a socketed chip.

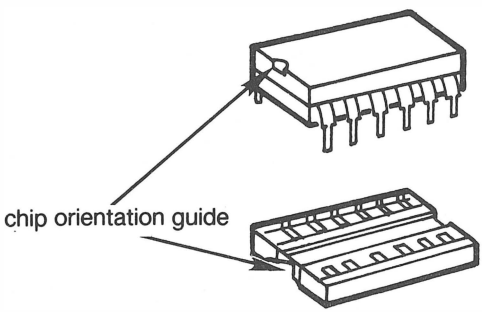


Fig. 1-8. IC chip orientation.

Replacing Subassemblies. When you suspect that a particular subsystem of the computer needs to be replaced, first swap the component with that of another system. This will confirm that the component really is defective, and that it is not a problem elsewhere in the system.

When you confirm that a subassembly must be replaced, take it with you to the computer dealer or electronics supply store. This will help you get an exact replacement. You can also find out if the store has a trade-in allowance for the old part.

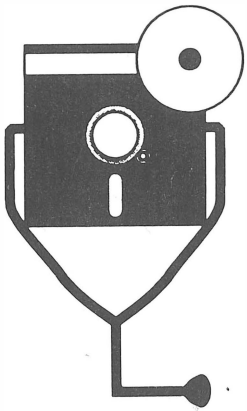
If the component is broken beyond repair, you might find it interesting to take it apart. There is nothing to lose at this point, so you can fool with it and find out how it ticks. This way, if the replacement component ever has a problem, you'll have a better idea of what to do with it, without the fear of damaging it further.

GETTING STARTED

To get going on general care of the Apple II system, preventive maintenance, troubleshooting, and repair, do the following:

- Read through this book to become familiar 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 the basic tools and cleaning supplies into a computer maintenance kit.
- Collect all technical documentation and find a central place for it, perhaps an area at your desk or bookcase, or in a binder.
- Contact someone who owns an identical Apple II 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 encounter 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 satisfied that the program is working as it should, make a backup copy.

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



2

The System Diagnostic Program

This chapter covers the System Diagnostic Program: what subsystem modules consist of, and how to load and run the program. The Main Menu module of the System Diagnostic Program is then described in detail to explain how the program actually works.

An overview on program modification and adaptation for your own purposes is covered here briefly, and treated in greater depth in Chapter 10.

For those unfamiliar with Applesoft BASIC programming, or who need a quick refresher, a final section describes how to interpret the module code explanations provided in each chapter. This section also provides short definitions of the Applesoft BASIC statements and commands used in the System Diagnostic Program.

SYSTEM DIAGNOSTIC PROGRAM SUMMARY

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

The block diagram in Fig. 2-1 outlines the structure of the System Diagnostic Program. The numbers within each of the blocks are the line numbers where the

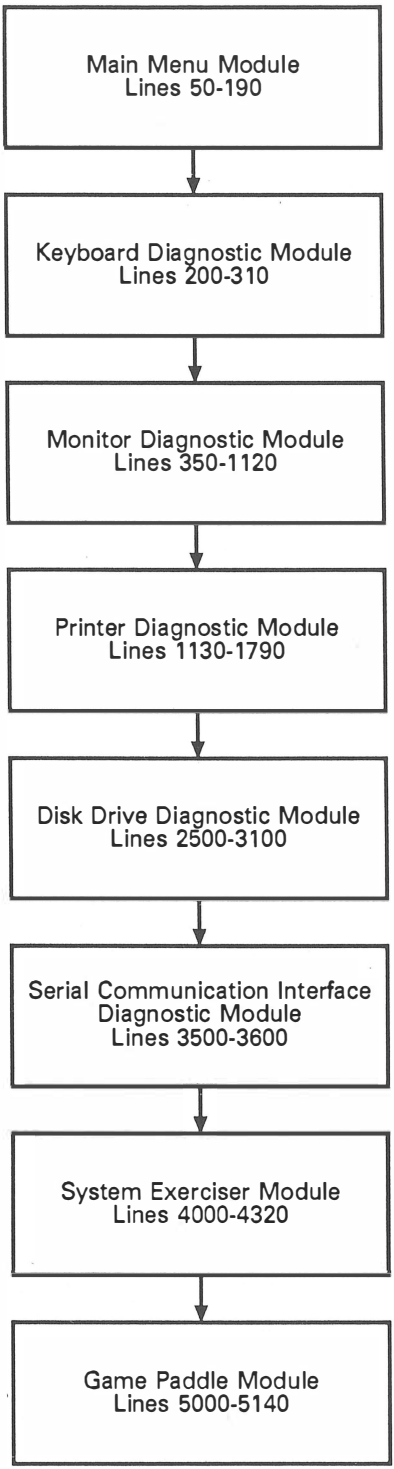


Fig. 2-1. System diagnostic program block diagram.

code for that subsystem module is located. Lines 10 through 30 are set aside for introductory program remarks.

The first module is the System Diagnostic Main Menu. The program code for the Main Menu module, residing in lines 50 through 190, will be 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 5, "The Monitor."

The second module in the program is the Keyboard Diagnostic module. This module runs a test that displays the character (echos) and gives the numeric code, or *ASCII value,* of any key that is pressed. 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 through 310.

The third module is the Monitor Diagnostic module, which consists of five different tests residing in lines 350 through 1120.

The Display Test fills the entire screen with a specified character, testing whether or not each location on the monitor is capable of displaying data. The Alignment Test displays a series of vertical and horizontal lines to check that the monitor is properly aligned. The Character Set Test displays the monitor's full set of available display characters. The Scroll Test provides a sliding character set to check that the monitor is scrolling correctly. Finally Low-Resolution Color Graphics Test checks the 16 colors available at each low-resolution memory location on the screen.

A sixth monitor test, the High-Resolution Color Graphics Test, is also provided as a separate program, due to Apple II memory allocation constraints pertaining to high-resolution graphics. It checks the six colors available at each high-resolution memory location on the screen. This program is provided on the same diskette with the System Diagnostic Program, or it can be copied from the listing provided in Appendix B.

The Printer Diagnostic module consists of five tests and is stored at lines 1130 through 1790.

The Sliding Alpha Test checks each letter of the printer's available character set. The Character Set Print Test displays the primary character set that is recognized by the printer. The Echo Character Print Test checks that specified characters can be printed. The Horizontal Tab Test checks all horizontal tab positions across the width of the printer carriage. The Line Feed Test checks various line space gradations, including the feeding of a new continuous form page.

The Disk Drive Diagnostic module tests and exercises the two main functions of the disk drive: reading from and writing to the diskette. Lines 2500 through 3100 contain this disk drive test.

Lines 3500 through 3600 contain the Serial Communication Diagnostic module, which tests the serial communication interface. For this test, the user presses a key on the keyboard, and the character and its ASCII value is displayed on the

screen. This "loopback" test determines whether or not the modem interface is functioning accurately.

The System Exerciser module resides at lines 4000 through 4320. If you have just purchased a new Apple II system, or if you've just brought the 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 because most computer problems take place when the system is first set up, or when the computer is moved. The Exerciser module allows you to burn in and check system functions automatically and unattended. Further information about the System Exerciser can be found in Chapter 9.

The Game Paddle Diagnostic module, located at lines 5000 through 5140, is the example used in Chapter 10 to demonstrate how you can develop and integrate a new diagnostic module for any additional peripherals you own.

USING 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 to do, as with any new program diskette, is to make a backup copy and keep the original in a safe place.

Keying in the Program

If you have not purchased the diskette, enter the program code as listed in the Appendices. To do this:

1) Turn on the Apple system.
   Applesoft BASIC is automatically invoked.
2) Enter **NEW** to indicate to BASIC that this is a new program.
3) Enter the System Diagnostic Program code exactly as listed in Appendix A.
4) Save the program after each hundred lines, to protect from data loss in the event of a power or system failure. Use the command **SAVE DIAG**. That portion of the program is saved to the diskette.

   > **Note:** There is nothing special about the program name of "diag". Name it anything you like. Just made sure it is a descriptive name, and that there is no other program by that name on that diskette.

5) When the entire System Diagnostic Program is entered and saved to the diskette, enter **RUN DIAG** and test each program function to make sure there are no syntax errors. Such errors indicate typographical errors made while entering the program.
6) To correct any typographical errors, enter the line number containing the error, then simply retype the line correctly. This overwrites the incorrect

line. For further editing commands and techniques, refer to your *Applesoft BASIC Programmer's Reference Manual.* (Scott Kaming, Addison-Wesley Publishing Company, 1987).

7) Save the corrected version of the program.
8) Make another backup copy of the program.
9) Repeat these steps for keying in the High-Resolution Color Graphics Test program, as listed in Appendix B.

Note: It is important that the program is keyed in and properly working *before* a system problem surfaces. If you wait until after you suspect a malfunction, you might not be able to enter the program due to the malfunction. Even if you are able to enter and run the program, the test results might be inaccurate and misleading.

Running the Program

To load and run the System Diagnostic Program, follow these steps:

1) Turn power on to the system. Applesoft BASIC is loaded automatically, and the] prompt is displayed.
2) Place the diagnostic diskette into the disk drive.
3) Enter **RUN DIAG.** If you have two disk drives and you're loading the program from drive #2, enter **RUN DIAG,D2.** A beep indicates that the loading process has begun. A second beep indicates that the System Diagnostic Program has completed the load. The System Diagnostic Main Menu is then displayed.
4) Follow the same steps to run the High-Resolution Graphics Color Test. Enter **RUN HIRES.** The High-Resolution Graphics Color Test instructions will be displayed.

THE MAIN MENU

Once the program is loaded, the screen displays the System Diagnostic Main Menu. This menu presents several diagnostic options, as shown in Fig. 2-2.

A selection is made by pressing the one-character command representing the first letter of the name of the desired test. For example, if you wish to run the keyboard test, enter a **K** or **k.** The program is not character case-sensitive, but the Apple II Plus can only use the uppercase values.

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

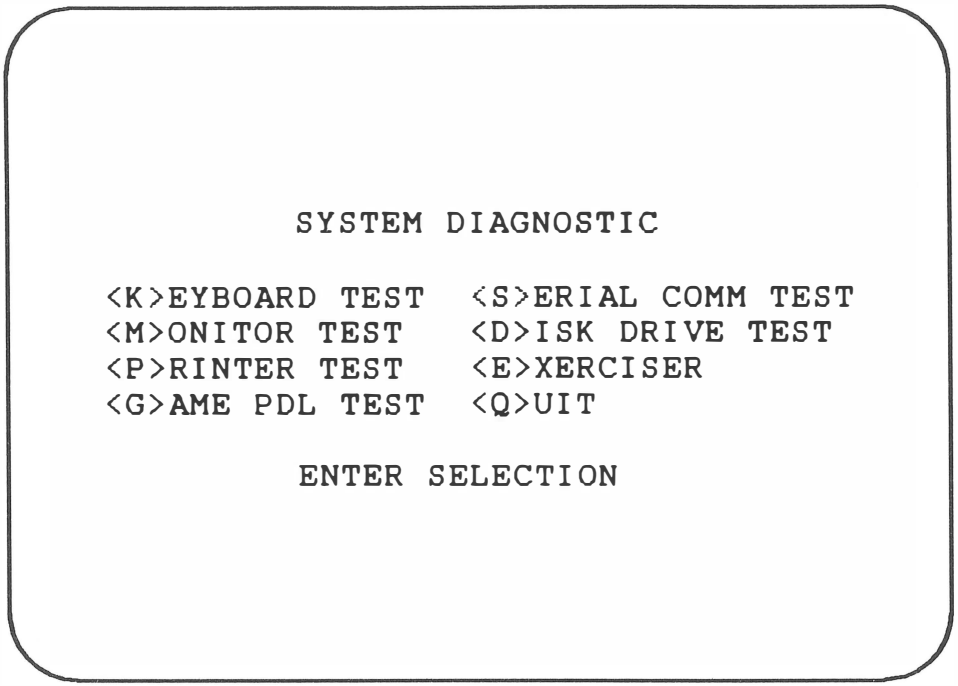


Fig. 2-2. System diagnostic main menu.

When you press Q, the program "quits," or ceases operation, and returns the computer to the Applesoft 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. 2-3.

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 the Apple II system.

Lines 10 through 30 consist of REMARK statements indicating the title and copyright information for the System Diagnostic Program.

```
10 REM APPLE ][ SERIES
20 REM SYSTEM DIAGNOSTIC PROGRAM
30 REM COPYRIGHT 1988 TAB BOOKS, INC.
```

```
10  REM  APPLE ][ SERIES
20  REM  SYSTEM DIAGNOSTIC PROGRAM
30  REM  COPYRIGHT 1988 TAB BOOKS INC.
50  REM  MAIN MENU MODULE
60  HOME : PRINT  TAB( 12)"SYSTEM DIAGNOSTIC": PRINT
70  PRINT  TAB( 3)"<K>EYBOARD TEST  <S>ERIAL COMM TEST":
    PRINT  TAB( 3)"<M>ONITOR TEST   <D>ISK DRIVE TEST"
80  PRINT  TAB( 3)"<P>RINTER TEST   <E>XERCISER":
    PRINT  TAB( 3)"<G>AME PDL TEST  <Q>UIT"
90  PRINT : PRINT  TAB( 12)"ENTER SELECTION"
100  GOSUB 8000: REM  GET A KEY
110  IF A$ = "K" OR A$ = "k" THEN  GOTO 200
120  IF A$ = "M" OR A$ = "m" THEN  GOTO 350
130  IF A$ = "P" OR A$ = "p" THEN  GOTO 1250
140  IF A$ = "S" OR A$ = "s" THEN  GOTO 3500
150  IF A$ = "D" OR A$ = "d" THEN  GOTO 2500
160  IF A$ = "E" OR A$ = "e" THEN  GOTO 4000
170  IF A$ = "G" OR A$ = "g" THEN  GOTO 5000
180  IF A$ = "Q" OR A$ = "q" THEN  END
190  PRINT CHR$ (7): GOTO 100
 .
 .
 .
8000  GET A$: RETURN
```

Fig. 2-3. Main menu module listing.

Line 50 is a **REMARK** statement indicating the beginning of the Main Menu module.

```
50 REM MAIN MENU MODULE
```

Line 60 clears the screen with the **HOME** instruction, and then prints the program title on the screen.

```
60 HOME : PRINT TAB( 12)"SYSTEM DIAGNOSTIC": PRINT
```

Lines 70 and 80 display the option menu which gives the user the various diagnostic module choices.

```
70 PRINT TAB( 3)"<K>EYBOARD TEST <S>ERIAL COMM TEST":
   PRINT TAB( 3)"<M>ONITOR TEST <D>ISK DRIVE TEST"
80 PRINT TAB( 3)"<P>RINTER TEST <E>XERCISER":
   PRINT TAB( 3)"<G>AME PDL TEST <Q>UIT"
```

Line 90 prompts the user to choose one of the modules by entering the first letter of the test name.

```
90 PRINT : PRINT TAB( 12)"ENTER SELECTION"
```

Line 100 calls the Get-a-key subroutine located at line 8000. This will get the key pressed by the user to indicate which test is being selected.

```
100 GOSUB 8000: REM GET A KEY
```

Lines 110 through 180 check for the value of the selected key. Depending on the key value, it branches to the location of the indicated test module. For example, if the user pressed K, line 110 causes the program to branch to line 200, which is the beginning of the Keyboard Diagnostic module. The lowercase values are only valid for the IIe and IIc systems.

```
110 IF A$ = "K" OR A$ = "k" THEN GOTO 200
120 IF A$ = "M" OR A$ = "m" THEN GOTO 350
130 IF A$ = "P" OR A$ = "p" THEN GOTO 1250
140 IF A$ = "S" OR A$ = "s" THEN GOTO 3500
150 IF A$ = "D" OR A$ = "d" THEN GOTO 2500
160 IF A$ = "E" OR A$ = "e" THEN GOTO 4000
170 IF A$ = "G" OR A$ = "g" THEN GOTO 5000
180 IF A$ = "Q" OR A$ = "q" THEN END
```

If none of these IF-THEN statements are matched to the key that was pressed by the user, then an invalid key has been pressed. The program runs through all the IF-THEN statements and falls through to line 190. Line 190 prints character string value 7, which causes a beep to sound, indicating the user error. The program branches back to line 100 and lets the user try again with another key.

```
190 PRINT CHR$ (7): GOTO 100
```

The Get-a-Key Subroutine

The Get-a-Key subroutine at line 8000 is used throughout the System Diagnostic Program, and is encountered first 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 in variable A$ for use elsewhere in the program.

```
8000 GET A$: RETURN
```

The RETURN instruction causes program control to return to the place in the code from which it branched.

PROGRAM MODIFICATION AND ADAPTATION

As you have seen, the System Diagnostic Program is written in Applesoft BASIC, and is structured to accommodate modification and additions. This means you are not restricted to simply testing the standard components of a typical Apple II system. Should you purchase a new peripheral device for the Apple II, such as a joystick, a mouse, or a digitizer pad, you can write a new diagnostic program for it. Chapter 10 covers such techniques, information, and references for writing new diagnostics and customizing the System Diagnostic Program for any new devices.

BASIC PROGRAMMING OVERVIEW

It is not necessary for you to be conversant in BASIC for the material presented in this book to be useful. You can skip the module explanations found in the subsystem chapters without detracting from your diagnostic, troubleshooting, or repair capabilities. However, this section is provided in case you would like to become more knowledgeable about BASIC, or for a brief review of BASIC commands. The information in this section will help you understand the code explanations better, and therefore understand how the program works.

If you are brand new to BASIC, and would like to learn it more thoroughly, obtain one of the many excellent books on learning BASIC. There are also BASIC classes available at many community colleges, adult education programs, and computer stores.

The Module Code Explanations

Each subsystem module of the System Diagnostic Program is listed and explained line-by-line, command-by-command, in the appropriate chapter. It is described in the chapter section entitled "How the Module Works."

Whether a program consists of 100, 3000, or 50,000 lines of code, when the program is running, only one command can be active, or executed, at a time. That command might only take a nanosecond or less to execute, but regardless of the speed of the program, it has to be successfully executed before the program can proceed to the next command.

The program code is like a puzzle, or more specifically, a maze. In the big picture, the program simply runs from line 1 at the beginning to the END

statement in a sequential manner. However, many many detours and circles can take place along the way in order to make the program do all the different functions for which it has been designed. Detours can be the result of user input, branching to subroutines to perform special functions, or decisions needing to be made based on present conditions.

The circles are process loops, which can be very simple and immediate, or can take place throughout the length of the program. There can even be loops within loops, as there can be decisions within decisions.

The subsystem module code explanations describe when each of these lines become active and are executed, and details exactly what is taking place. It explains specifically what the command is doing in this case, what values the command is using, and any loops and branches through which the program flows. And it describes the order in which these actions are taking place, so the direction of the program maze can be understood.

The code explanation describes the direct cause and effect of events taking place within the program. It also indicates the result of user actions.

The subsystem code explanation indicates where certain values, arrays, and variables reside. This can be helpful in the event you wish to modify, add to, or branch to a part of the program as a part of your own programming efforts.

BASIC Commands and Statements

This overview provides a cursory description of the specific BASIC statements used in Applesoft BASIC for the System Diagnostic Program. Only those commands used in the Apple II System Diagnostic Program are explained here.

There are other commands and structures available, which are described in the BASIC manual that came with your system. Two good Applesoft BASIC books are Applesoft BASIC Programmer's Reference Manual (Scott Kamins, Addison-Wesley Publishing Company, 1987), and Applesoft BASIC Toolbox (Larry G. Wintermeyer, Addison-Wesley Publishing Company, 1987). These books thoroughly cover each command available in Applesoft BASIC running on your Apple II system, and explain special programming techniques. Also included is an alphabetical reference of each available statement, its proper syntax and action, as well as applicable remarks and examples.

The following are brief descriptions and examples of statements used in the Applesoft System Diagnostic Program. The statements are listed in alphabetical order.

ASC returns the ASCII value of the first character in a string.

```
250 IF ASC (A$) = 27 THEN GOTO 50
```

In this example, if the ASCII value of the first letter contained in variable A$ string is equal to 27, the program is to branch to line 50.

COLOR sets the color for the low-resolution graphics mode, using a numeric representation.

```
5060 GR : COLOR= 13
```

In this example, the display is set to the low-resolution graphics mode, and the color of the display is set to 13, the numeric value for yellow.

END terminates the program, and returns the computer from the program environment to the Applesoft BASIC environment, displaying the right bracket (]) prompt.

FOR-NEXT is a two-part loop statement. It allows a series of instructions to be performed in a loop for a specified number of times before going on to the subsequent instruction.

The FOR statement opens a loop by specifying the number of loops to be run. It also specifies the loop counter.

The lines of code following the FOR statement are executed until the NEXT statement is encountered. The instructions located between the FOR and NEXT statements comprise the body of the loop, and are executed repeatedly as indicated in the FOR statement.

The NEXT statement closes the loop.

At this point, the counter is incremented by one, and this incremented number is compared with the final end value as specified in the FOR statement. If the counter is not greater than the specified end value, the program repeats another loop. If the counter is greater than the end value, the loop process is completed, and program execution goes on to execute the statement following the NEXT instruction.

```
2635 FOR L = 1 TO 430
2640 PRINT B$: NEXT L
```

In this example, L is the counter, and specifies that the loop is to run 430 times, starting at 1 and ending at 430. The loop consists of printing the value stored in B$. Each time A$ is printed, the NEXT instruction is encountered. The colon separates the two statements used within the same line of code. NEXT sends the program back to the FOR statement, and L is incremented by 1. Counter L checks to see whether its value is now more than 430. If it is not, it allows another loop to be run, and A$ is printed again. If counter L does have a value of more than 430, the program proceeds to the instruction immediately following the FOR-NEXT loop.

GET reads a single keystroke and stores it into a designated variable.

```
8000 GET A$
```

In this example, program execution pauses until the user presses a key. When a key is pressed, its value is stored in variable A$. Program execution now continues on to the next instruction.

GOSUB-RETURN is a two-part statement which branches the program to a subroutine, and then returns the program to the place in the code where it left off. When the program encounters the GOSUB statement, the program branches to the specified line number where the subroutine begins, and then the subroutine is executed. The RETURN statement resides at the end of the subroutine, which returns the program to the location within the code immediately following the GOSUB statement.

```
100 GOSUB 8000
110 IF A$ = "K" THEN GOTO 200
.
.
.
8000 GET A$: RETURN
```

In this example, the GOSUB statement in line 100 branches the program to line 8000 where a subroutine exists. Line 8000 consists of a single-line subroutine, in which user input is received from the keyboard. When this statement is executed, the program encounters the RETURN statement. The program returns and executes the statement immediately following line 100, which is line 110.

GOTO branches unconditionally to the specified line number within the program.

```
140 GOTO 3500
```

In this example, line 140 consists of a GOTO statement which transfers program control to line 3500.

GR switches the monitor display from the default 40-column by 24-row text mode to the 40-column by 48-row low-resolution graphics mode, which can display 16 colors.

```
550 GR : COLOR 15
```

In this example, the monitor display switches from the text mode to the low-resolution graphics mode, and the color is set to white.

HCOLOR sets the color for the high-resolution graphics mode, using a numeric representation.

```
200 HGR
220 HCOLOR= 2
```

In this example, the display is set to the high-resolution graphics mode, and the color of the display is set to 2, the numeric value for violet.

HGR switches the monitor display from the default 40-column by 24-row text mode to the 140 x 192-pixel high-resolution graphics mode, which can display six colors.

```
210 HGR
```

In this example, the monitor display switches from the text mode to the high-resolution graphics mode.

HLIN X,X1 AT Y is used in either the low-resolution or high-resolution graphics mode to draw a horizontal line from one point to another (X to X1) at the row specified by Y.

```
575 HLIN 0,39 AT 47
```

In this example, a horizontal line is drawn from column 0 to column 39 at row 47.

HOME clears the screen and positions the cursor at the top left corner of the screen. It is often used as the first statement beginning a new segment of a routine.

```
60 HOME : PRINT TAB( 12)"SYSTEM DIAGNOSTIC": PRINT
```

In this example, the HOME instruction clears the screen and positions the cursor at the top left corner of the screen before printing the program title.

HPLOT X,Y is used in the high-resolution graphics mode to draw a pixel in the current color at the specified X,Y coordinates.

```
250 HPLOT 235,69
```

In this example, a pixel is drawn on the screen at the 235th column and 69th row of the high-resolution pixel matrix.

IF-THEN is a two-part conditional statement, making a decision regarding program flow based on the result of a specified expression. When the IF statement is true, the THEN statement is executed. The THEN statement might consist of a line number for branching, or one or more statements to be executed. When the

IF statement is not true, the THEN statement is not executed; rather, program execution continues with the next executable statement.

```
160 IF A$="E" OR A$="e" THEN GOTO 4000
```

In this example, the IF statement is accompanied by the expression A$="E" OR A$="e". If either of these expressions is true, the program executes the THEN statement to go to line 4000. If neither of these expressions is true, the program proceeds to the next line.

IN# X switches input from the keyboard to the device whose controller resides in slot X, which would be slot 0 through 7.

```
3560 IN# 2
```

In this example, input is taken from the device whose controller resides in slot #2, perhaps a serial communication interface, rather than from the keyboard. IN# 0 switches input back to the keyboard.

INPUT allows user input from the keyboard during program execution. When the program encounters an INPUT statement, it usually prints a specified user prompt, then prints a question mark. The program waits until the user enters the prompted data, which is then placed into the indicated variable(s).

```
4020 INPUT "TEST MONITOR? (Y OR N)-";M$
```

In this example, the message between quotes is printed on the screen. The program waits until the user enters the requested data. This data is placed into variable M$.

ONERR GOTO disables the normal Applesoft error handling routine, and lets a different error trap within the program take over. Ordinarily, when Applesoft encounters an error, the program is halted. This allows you to provide more detailed error handling within your program, so that when an error is encountered, the program can branch to the indicated line number for a program-specific error-handling routine.

```
2600 ONERR GOTO 3000
```

In this example, when an error is encountered in the previous routine, the program is to branch to line 3000.

PDL returns a value representing the position of the game paddle control wheel. This value can range from 0 (extreme left) to 255 (extreme right).

```
5090 LET X = PDL
```

In this example, the numeric value indicating the current location of the game paddle control wheel is placed in variable X.

PEEK reads a byte from the specified memory location.

```
3000 ER = PEEK (222)
```

In this example, memory location 222 is read, and the value residing there is placed into variable ER. On the Apple II, memory location 222 controls error handling. Placing this value into a variable makes the errors usable in the program.

PLOT X,Y is used in the low-resolution graphics mode to draw a block in the current color at the specified X,Y coordinates.

```
5110 COLOR= 13 PLOT 29,20
```

In this example, a block is drawn on the screen at the 29th column and 20th row of the low-resolution column/row matrix. The color of the block is specified by the COLOR instruction.

POKE writes a specified byte into a specified memory location.

```
1060 POKE -16302,0
```

In this example, a value of 0 is written into memory location -16302. On the Apple II, memory location -16302 controls the four text lines available in the graphics modes. Placing a value of 0 into this byte disables these text lines.

PR# X switches output from the screen to the output device whose controller resides in slot X, which would be slot 0 through 7.

```
1320 PR# 1
```

In this example, output is sent to the output device, probably a printer, whose controller resides in slot #1. PR #0 switches output back to the monitor display.

PRINT displays the specified data on the screen. If there is no data specified between quotes with the PRINT statement, a blank line is printed on the screen.

```
2515 PRINT "DISK DRIVE DIAGNOSTIC"
2518 PRINT
```

In this example, line 2515 prints the words between quotes on the screen. Line 2518 then prints a blank line.

A semicolon is used in PRINT statements to cause items to be printed close

together. A ; is placed at the end of a PRINT statement suppresses the automatic printer carriage return that usually takes place when the computer reaches the end of a PRINT instruction.

A comma can be used in PRINT statements to space the items to be printed out in a predetermined manner across the page.

PRINT D$ is seen throughout the Disk Drive Diagnostic module, but is actually caused by an ASCII(4) D$. ASCII(4) D$ must precede any command to the disk drive, and is listed as PRINT D$.

```
2620 PRINT D$;"OPEN TEST"
.
.
.
2650 PRINT D$;"CLOSE TEST": PRINT D$
```

In this example, PRINT D$ in line 2620 enables the file "TEST" to be opened on the diskette for subsequent use by the program. Likewise, the same statement in line 2650 with "CLOSE TEST" closes the opened file called "TEST" because it is no longer needed by the program. The PRINT D$ with no subsequent command string indicates that the system is to return to using the keyboard for input and the monitor for output.

REM indicates a non-executable remark statement, provided only for information to the reader of the program listing. Remark statements provide program documentation within the code for the benefit of the programmer and others who wish to read and understand the program. A REM statement is output exactly as entered when the program is listed. However, it has no effect on the program execution itself. As soon as the program encounters REM, it ignores the rest of the statement and proceeds to the next command.

```
660 REM LOW RES COLOR GRAPHICS TEST
```

In this example, the REM tells the reader of the program listing that the subsequent lines constitute the Low-Resolution Color Graphics Test routine.

STEP is used in conjunction with a FOR-NEXT loop. If the loop is not to increment by 1, the STEP statement sets the increments of the loops.

```
560 FOR I = 1 TO 45 STEP 8
570 HLIN 0,39 AT I - 1
580 NEXT I
```

In this example, the FOR-NEXT loop executes the drawing of a series of horizontal lines starting at the top of the screen and ending at the bottom of the screen.

The number of rows on the screen is indicated by the value of 45 in line 560. However, a line is not required at each row. Instead, it is to be drawn every eighth row. The STEP 8 instruction in line 560 makes sure that the I counter is incremented in eighths, so that this is executed as desired.

TAB is used with the PRINT command, and positions the item to be printed at the specified column number. It controls the horizontal spacing on the print line, the way the TAB key controls spacing on a typewriter.

```
60 PRINT TAB (12) "SYSTEM DIAGNOSTIC"
```

In this example, the words between quotes begin printing 12 spaces from the left side of the screen.

TEXT switches the monitor display from one of the graphics modes back to the default 40-column x 24-row text mode.

```
550 GR
.
.
.
650 TEXT
```

In this example, the monitor display is switched from the default text mode to the low-resolution graphics mode in line 550. A low-resolution graphics routine takes place, and then at line 650, the TEXT instruction returns the display to the text mode.

VAL translates and returns a string value to the numeric value it is representing. A string value, even if it is a number, cannot be used as a number. It must have its numeric value assigned back to it before it can be used as a number, for example, in arithmetic operations.

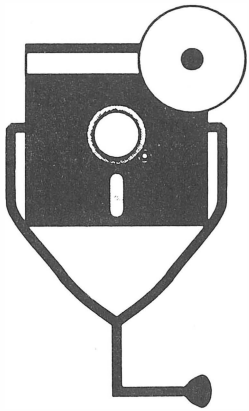
```
5040 A = VAL (A$)
```

In this example, numeric variable A is made equal to the string variable A$.

VLIN Y,Y1 AT X is used in either the low-resolution or high-resolution graphics modes to draw a vertical line from one point to another (Y to Y1) at the column specified by X.

```
620 VLIN 0,47 AT 39
```

In this example, a vertical line is drawn from row 0 through row 47 at column 39.



3

The System Unit

The system unit is the brains of the Apple II system, controlling and coordinating all activities taking place within the computer. The system unit issues commands, processes input and output, and directs the activities of peripheral devices. The Apple II series personal computers have the system unit and keyboard contained within the same housing (Fig. 3-1).

DESCRIPTION AND FUNCTION OF THE SYSTEM UNIT

The system unit consists of:

- microprocessor chip
- random-access memory (RAM)
- Applesoft BASIC in read-only memory (ROM)
- interfaces and controller boards
- expansion slots for optional peripheral devices
- power supply

The system unit is the very center of the computer. Within it are the motherboard and power supply. The motherboard, sometimes also called the system board, contains the microprocessor chip, system memory chips, peripheral inter-

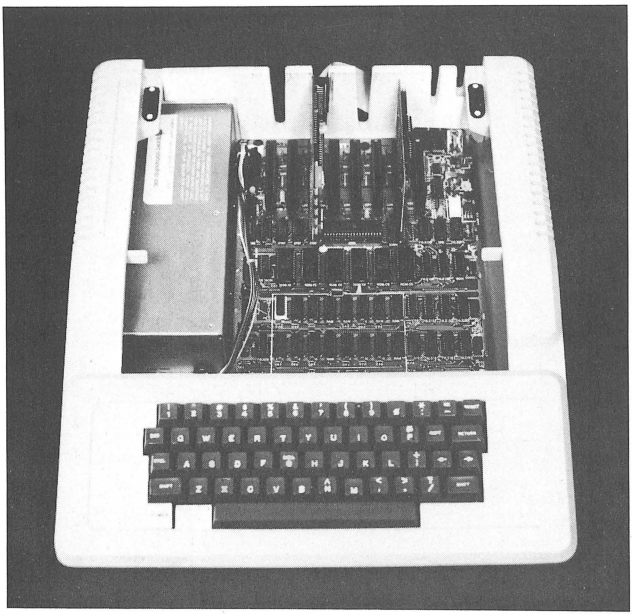


Fig. 3-1. The system unit.

face controllers, and expansion slots. This board is the foundation of the computer. The power supply outputs DC power to the motherboard to provide the required voltages to the integrated circuits.

The Central Processing Unit

The central processing unit (CPU) consists of three sections:

- arithmetic-logic
- general control
- storage

The CPU, located on the motherboard, is made up of several integrated circuit (IC) chips: the microprocessor chip and system memory chips (Fig. 3-2). The microprocessor performs the arithmetic-logic and general control CPU functions. The memory chips perform system data storage functions.

The Apple II Plus uses the 6502 microprocessor chip as part of its CPU; the IIe uses the 6502B; and the 65C02 microprocessor controls the IIc. These chips perform the arithmetic-logic and control functions for their respective computers.

Arithmetic-Logic Section. It has been said that the computer is nothing more than a counting machine. Although this is an oversimplification when discussing

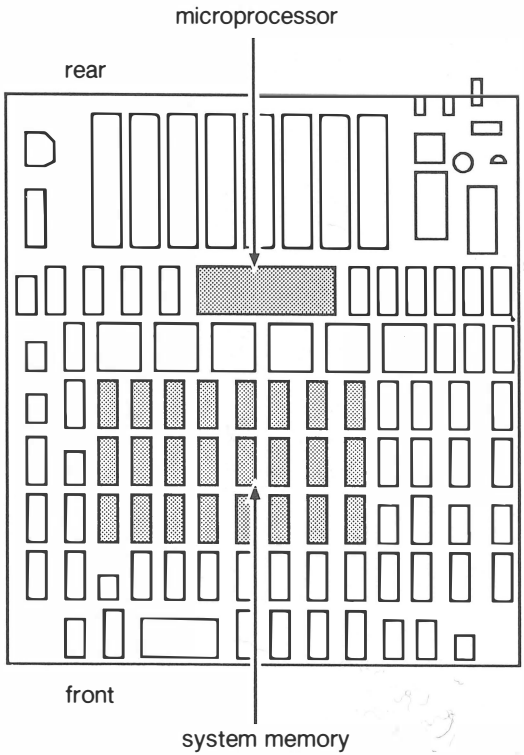


Fig. 3-2. The CPU on the motherboard.

overall computer functions, it is accurate when discussing the arithmetic-logic section of the CPU.

The arithmetic-logic section performs all counting functions, and the counting amounts to the actual processing of data. Special circuits within the microprocessor such as *registers, accumulators,* and *adders,* make all the required calculations. Comparisons are made of one number to another for logic functions using *comparers.*

Control Section. Every computer program instruction consists of at least two parts. The first part of the instruction indicates the command. The second part of the instruction indicates the address which locates the data to be acted upon by the command. The control section of the CPU interprets these program instructions, and then executes them. The control section works closely with the arithmetic-logic section to perform the necessary calculations or comparisons to carry it out.

Storage Section. General-purpose system memory, performed by the random-access memory (RAM) chips, stores the currently active program and the data being processed. Several RAM chips are provided on the motherboard.

Computer data is read, written, and measured by a series of ON and OFF bits: 1s and 0s. Eight of these ON and OFF bits constitute an Apple *byte,* also known as a data word or character. Most computer dimension terminology is measured in terms of bytes, particularly the amount of data that can be stored in system memory.

A kilobyte is 1024 bytes. When a computer is referred to as a 128-kilobyte machine, this indicates that its system memory can store approximately 131,000 bytes at one time. The Apple IIc is such a computer. On the other hand, the Apple II Plus has 48 kilobytes of system memory in RAM; while the Apple IIe has 64kB.

The term *random-access memory* indicates that the information currently stored in the chip can be directly and immediately retrieved for the user. The physical location of the data is not a factor in retrieval time.

However, the memory of the RAM chips is volatile. *Volatile* means that when power is removed from the computer, all programs and data stored in RAM are cleared. When the computer is powered on again, the RAM is empty, and the required software and data must be once again loaded into RAM from a mass storage device such as a disk drive or cassette tape drive. RAM is used as temporary storage for programs and data while the system is powered-on.

A second type of memory is also provided as part of the CPU, called *read-only memory* (ROM). ROM chips have programs permanently stored in its memory which is retained whether or not the computer system has power. When the system is turned on, the program can be invoked immediately, without having to load it from an outside storage device. Data cannot be written to ROM chips by the user; they are preprogrammed and can only be read, hence the name. In the Apple II systems, BASIC is permanently stored in ROM. Basic input/output functions are also handled in the ROM chips. In addition, some Apple II systems have a machine language monitor in ROM.

Controllers and Ports

The different Apple II systems have different combinations of device controller circuits, ports, and expansion slots on the motherboard. All three systems include the following controllers on the motherboard:

- monitor controller
- keyboard controller
- cassette drive controller
- game paddle controller

These controllers exist as separate circuits on the motherboard, each with its appropriate external connector with which to connect the monitor, keyboard, cassette drive, and game paddle (Fig. 3-3).

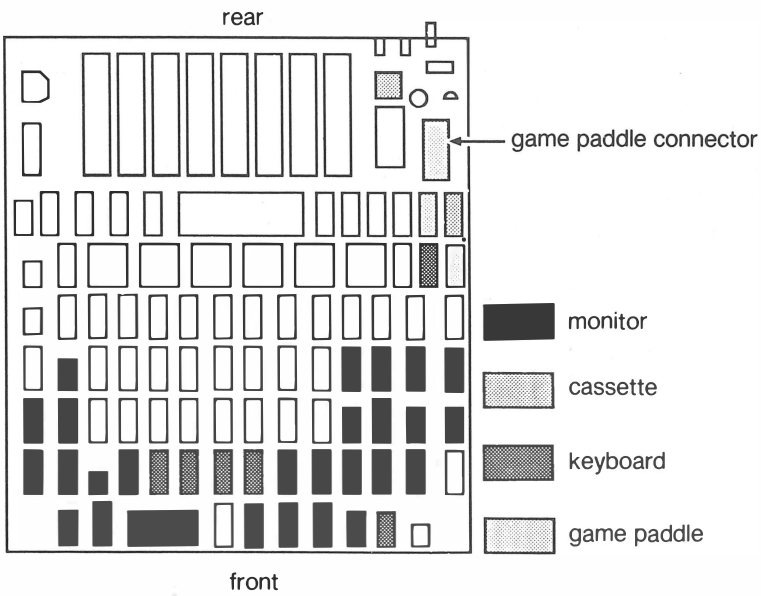


Fig. 3-3. System unit controllers and interfaces.

Additionally, the Apple II Plus and IIe both include eight expansion slots to for other optional peripheral controller boards. These options include the disk drive controller, or a serial communication interface board that would allow the use of either a printer or modem through the serial communication port.

The IIc motherboard has the greatest variety of functionality, including a mouse interface, two serial I/O ports for a printer and modem, and two disk drive controllers: one for an internal drive, and one for an external drive. No expansion slots are necessary, since the IIc motherboard includes all necessary controller circuits as standard equipment.

The Power Supply

The power supply provides DC (direct current) voltage to the computer. It consists of a transformer, rectifier, and filter, and also includes a cooling fan to dissipate the heat generated by the input and output voltages.

The computer's power cord, which is plugged into the AC wall outlet, ensures that 115 volts of alternating current (VAC) goes from the standard electrical supply into the computer's power supply. The power supply then rectifies 115 VAC into the 5 or 12 volts direct current (VDC) required for the computer's electronic circuitry. The power supply is an external unit on the Apple IIc. On the II Plus and IIe systems, the power supply is internal, housed within the system unit (Fig. 3-4).

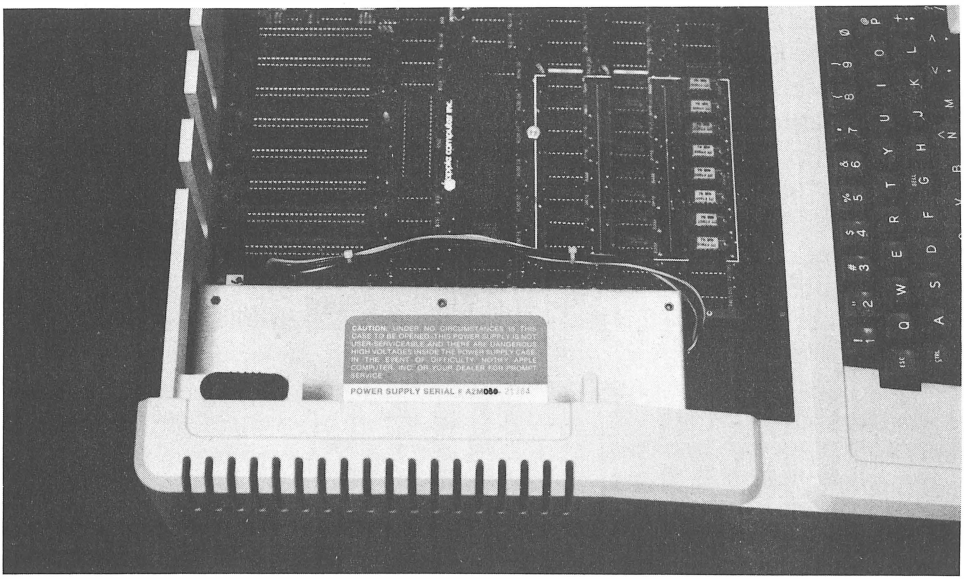


Fig. 3-4. Apple II Plus/IIe internal power supply.

SYSTEM UNIT PREVENTIVE MAINTENANCE

Place a dust cover on the system unit/keyboard whenever it isn't being used. Dust and dirt can work their way down inside the system unit and corrode the ICs. rendering them defective. Food particles can interfere with circuits, and if a liquid is ever spilled onto the system unit/keyboard, the whole unit will need to be replaced.

About once a month, use a hand-held vacuum cleaner or compressed air to vacuum or blow around the keyboard keys. This clears out any dust and dirt that has gotten onto the keyboard in spite of the dust cover, and prevents it from going any further down into the system unit.

If you ever need to open up the system unit for troubleshooting or repair, carefully vacuum dust and dirt off the boards. Wipe the inside of the casing with a lint-free cloth moistened with isopropyl alcohol. While the system is open, disconnect all cables from their boards and remove oxidation from the cable connector pins and board edge connectors. Oxidation is caused by metal contact with air, moisture, or a combination of the two. Oxidation tarnishes the metal, and can look like either a dull gray or black film.

Oxidation can form on board edge connectors and cable pins and reduce their conductivity. Oxidation can be removed by rubbing the surface with a pencil eraser (Fig. 3-5). You can also use an emery cloth, or wipe it with lint-free cloth moistened with isopropyl alcohol. The film of oxidation is visibly removed by any of these methods.

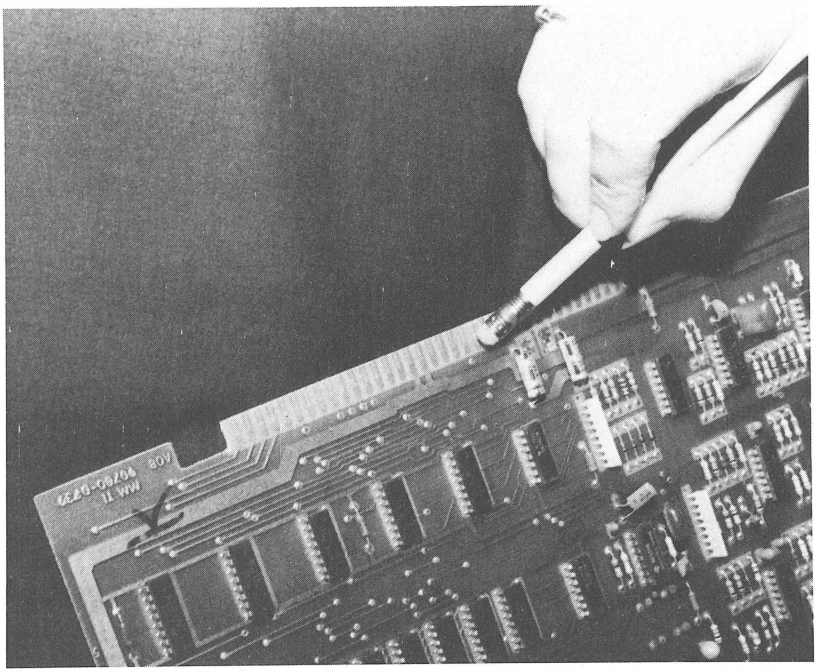


Fig. 3-5. Erasing oxidation.

After cleaning the cable pins and board edge connectors, reconnect them properly, and tighten the cable screws if applicable. If the boards are seated in slots, make sure they are aligned and completely installed in the slots for a proper connection.

SYSTEM UNIT TROUBLESHOOTING AND REPAIR GUIDELINES

This section describes the problems that the system unit could experience, what causes these problems, and details the steps needed to be taken to fix them.

After reading subsequent chapters covering troubleshooting procedures for each of the subsystems, it will become evident that if initially nothing happens, the system unit is the suspect component to troubleshoot. And, if you try all the troubleshooting suggestions for a particular subsystem, but still cannot find the problem, then again it is probably the system unit. You will be directed back to this section to find and repair the system unit problem.

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.

Problem: The System Unit Has No Power

If the system unit is not working at all, first check whether or not it has power. To do this, try the following solutions.

Solutions:

- Check whether or not the system unit's power light located on the lower left of the system unit/keyboard is illuminated. If it is not lit, then the system unit is not getting power.

  Check the 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. On the Apple IIc, this is a fairly simple procedure, because the power supply is external to the system. Remove the power supply and swap it on another Apple IIc system. If that system also fails to power up, then buying and installing a new power supply will solve the problem.

  Do not tamper with the internal power supply if you own the II Plus or IIe. Take the system unit to an Apple II service outlet and have the technician check the power supply.
- If the power light is illuminated, but the keyboard does not work, check the monitor for the standard right bracket prompt. This prompt indicates that the system is in the Applesoft BASIC environment. If the prompt is absent, turn the computer off, make sure the power supply cable is securely connected to the system, then turn the computer on again.

Problem: Oxidation Needs to Be Removed

If there is a problem with the system unit, but it is receiving sufficient power, then a dirty connection causing ineffective contact might be indicated. Try the following solution.

Solution:

- Remove oxidation from the expansion board edge connectors and cable pins. To do this:

  1) Power off the system.
  2) Disconnect the power supply (if it is a IIc system).
  3) Disconnect all attached peripheral devices (cassette drive, disk drive, printer, monitor, game paddle, etc.).
  4) Remove the top cover to open the system unit (Fig. 3-6).
  5) Remove the cable connector from each expansion board one at a time,

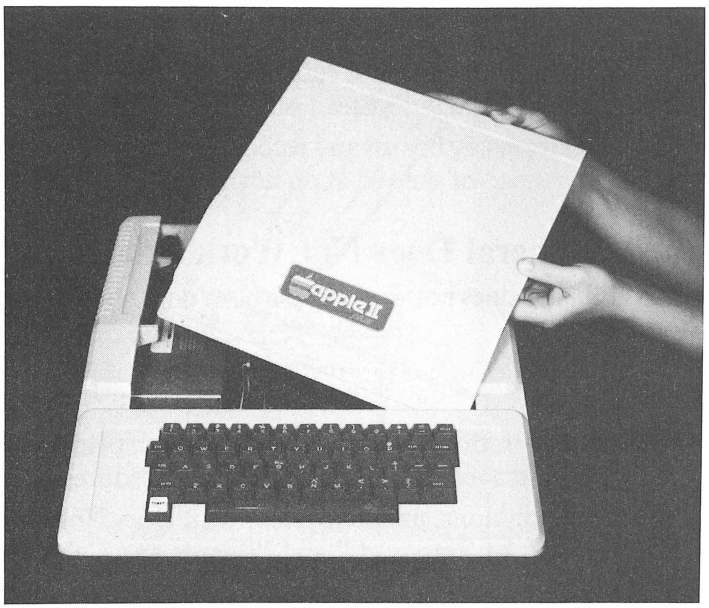


Fig. 3-6. Opening the system unit.

and check the cable pins for oxidation — the gray or black film on the metal.

6) If there is oxidation, remove it with a pencil eraser, emery cloth, or lint-free cloth moistened with isopropyl alcohol.
7) Remove the expansion board from the slot (refer to "Working with PC Boards" in Chapter 1 for further instructions on handling printed circuit boards). Check the edge connector for oxidation, and remove it if necessary.
8) Insert the expansion board back in the slot, and reconnect its cable.
9) Go on to the next expansion board. Repeat the process of cleaning oxidation from the cable pins and edge connectors.
10) Close the system unit, reconnect the power supply and monitor, and power-on the system unit.

Problem: Boards and Cables Need to be Reseated

If there is a problem with the system unit, and it is receiving sufficient power, and the connections are clear of oxidation, then a faulty connection may be indicated. Try the following solutions.

Solutions:

- Remove each board from its slot or cable and carefully reseat them. Make sure the board goes in straight, instead of skewed at an angle.
- Remove all cables from any boards and reconnect them. Make sure the cable goes on straight, instead of skewed at an angle.

Problem: A Peripheral Does Not Work Properly

If one of the peripherals does not work properly, or does not work at all, try the following solutions.

Solutions:

- Refer to the chapter in this book covering that particular peripheral, and follow the suggested troubleshooting and repair procedures.
- Refer to the above solutions, under "System Unit Does Not Have Power," "Oxidation Needs to be Removed," and "Boards and Cables Need to be Reseated," and follow those suggestions.
- If the peripheral still does not work, refer to the peripheral documentation. There might be a DIP-switch on the controller board that needs to be set, particularly on the serial communication interface board. The printer itself might have DIP-switches.

  A DIP-switch is a dual inline plastic switch containing a number of microswitches mounted on a board (Fig. 3-7), each of which can be set to ON or OFF. The different microswitches can each control a different setting, mode, or configuration; can work in various combinations; or the entire switch can have a specified ON/OFF sequence of the microswitches which indicate a particular setting.

  Switch settings are read by the system unit when it is powered on, and determine whether or not a certain configuration or setting is enabled. An incorrect switch setting for your particular Apple II configuration can cause a malfunction in the peripheral or even in the entire system. Information about switch settings is found in your peripheral documentation.

Problem: System Has Crashed

When the Apple II system is powered on, Applesoft BASIC is automatically loaded from ROM. A successful load is indicated by a right bracket (]) prompt on the screen. If the screen suddenly clears and is replaced by a series of hexadecimal numbers, and if the right bracket prompt changes to an asterisk (*) prompt, this indicates that the system has just crashed and placed the computer in the machine monitor program environment.

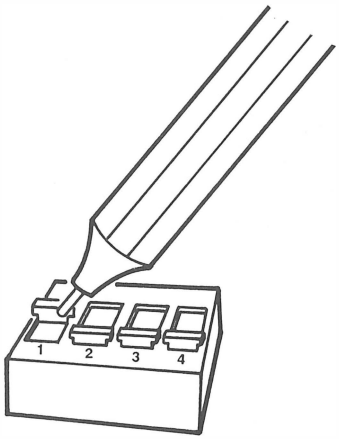


Fig. 3-7. DIP-switch.

Solution:

- To recover from the crash, press RESET or CTRL RESET. The environment returns to Applesoft BASIC as indicated by the right bracket prompt.

  This is a simple solution, but, unfortunately, work stored in system memory at the time of the crash is probably lost. To prevent losing work due to a system crash (or power outage, for that matter), save your work to disk about every ten minutes.
- If you were working within a language other than BASIC, you might need to reload the language board.
- If pressing RESET does not return the computer to the Applesoft BASIC environment, turn the computer off, wait a few seconds, and then turn it on again.
- If the system still does not access the Applesoft BASIC environment with the right bracket prompt, this indicates a more serious problem. The BASIC ROM chip might have blown, and this chip will need to be replaced.

Problem: System Unit Needs to Be Serviced

If the solutions described above fail to solve the system unit problem, it is time to take the unit in for servicing.

First call the service center and talk to the technician. Describe the computer and its configuration, and explain the problem it's experiencing. Also describe what you have already tried to do to find and fix the problem, and indicate your results. Find out if there are any solutions they can suggest over the phone. Try all their phone solutions. One of them might fix the problem.

If the problem cannot be repaired over the phone, ask the technician how much of the system you should bring to the shop. In the case of the system unit, you will probably have to take the entire system in, including monitor, interfaces, printer, game paddle, and everything else that is hooked up to the computer while it is experiencing the problem. The technician will need to look at the system as a whole.

Pack everything carefully in the original cartons and packing material. Load the boxes securely in your car so that they won't fall or be jarred during a hard turn or fast stop.

Provide the technician with all the information you have. Find out if the technician has any ideas on what the problem might be, and how long and how much it might cost to fix. Be sure to get a copy of the work order.

Before you pay for and accept your computer, have the repair technician hook everything up and demonstrate that the problem has actually been fixed and your computer is working properly. If you're satisfied, then pack the system up and take it home. Set your system up again and refer to Chapter 9, "The Exerciser," for instructions on *burning in* the system after repair.

4

The Keyboard

Commands and data are input to the system unit for processing via the keyboard, which is the primary external link to the computer.

DESCRIPTION AND FUNCTION OF THE KEYBOARD

The keyboard is used to type characters to enter data, form commands, make selections, and move the cursor while working within a program or application. The characters are sent to the CPU, which interprets and processes them, causing the appropriate results to be displayed on the monitor (Fig. 4-1).

The Apple II 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. 4-2). The key set found on the Apple II keyboards consists of the "QWERTY" typewriter keys (named for the first six alphabetic characters found on the second row a typewriter), cursor control keys, and a numeric keypad. The IIc keyboard has additional function keys such as the CONTROL, CAPS LOCK, ESCAPE, and DELETE keys. There are also two special "Apple" keys, one open, and the other solid.

Note: If desired, the IIc can be reconfigured as a "Dvorak keyboard," an alternative keyboard layout often preferred for high-speed typing. If you wish to

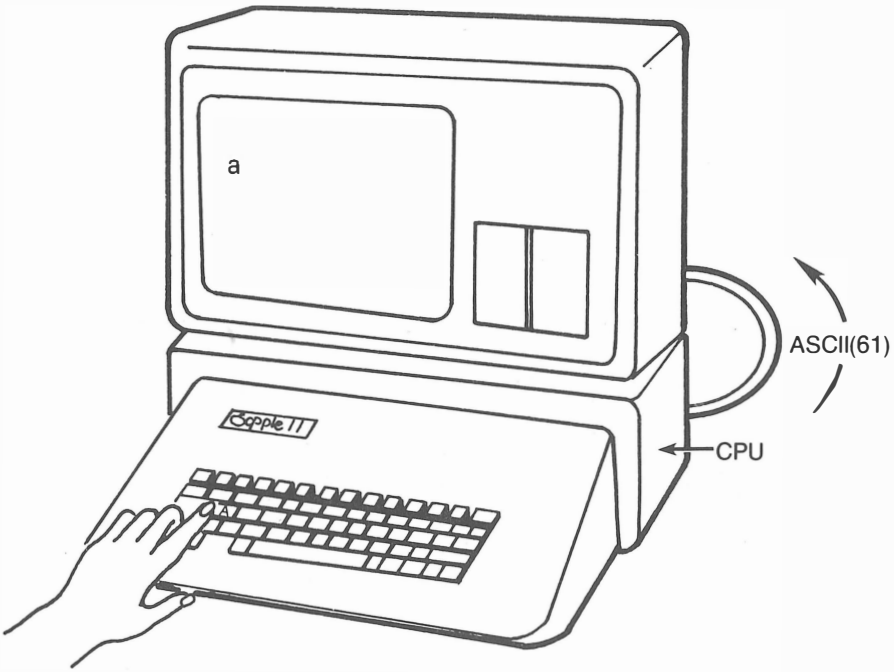


Fig. 4-1. Pressing a key.

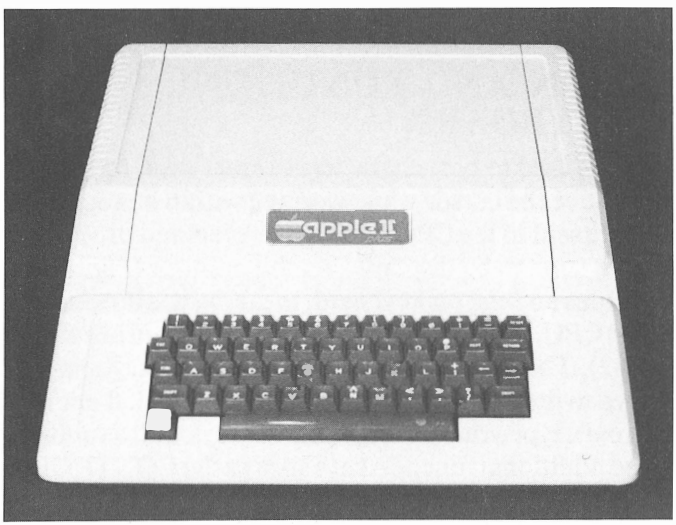


Fig. 4-2. The integral keyboard/system unit.

switch to the Dvorak layout, refer to the IIc user manual or technical reference manual for instructions.

Even though a mouse, joystick, game paddle, and digitizer pad are available, the keyboard is the Apple II system's primary input device.

How the Keyboard Functions

Three actions take place when a keyboard key is pressed.

1) A column/row value indicating the key's location in the keyboard matrix is generated in the keyboard controller.
2) A unique ASCII value, based on the column/row value, is generated to represent the value of the pressed key.
3) The ASCII value is sent from the keyboard controller to the CPU, which processes the value and determines what it is supposed to do. The results are then sent to the screen, if applicable, for the appropriate display or functional action.

Key Location. Pressing a key on the keyboard causes an electromechanical process to take place, similar to toggling a light switch or pressing a button on a television to change channels. Each key post is positioned over a wire matrix, with each post being directly over the crossover of an X (horizontal) wire, and a Y (vertical) wire. The contact with this crossover point generates a column/row value indicating the originating keyboard location.

The keyboard location information, which is calculated by the keyboard column/row matrix, is sent to an encoder on the keyboard controller. This encoder is a counter circuit with a built-in ROM containing the ASCII character set. Based on the matrix value of the pressed key, and based on whether or not the SHIFT or CTRL key was pressed in conjunction with the key, an ASCII value representing the pressed key is generated.

ASCII Value. *ASCII* (American Standard Code for Information Interchange) values are used to represent the standard display characters, such as A through Z, as an eight-bit binary (1s and 0s) code which can be interpreted and processed by the computer. For example, the letter "A" translates into the ASCII value of 11000000, with a decimal equivalent of 64.

ASCII values also represent control functions like SHIFT, CTRL, cursor movement, special characters, and symbols. The Apple II systems recognize 128 such ASCII code values.

Directly related to the ASCII code is the character string value, or CHR$ code. The CHR$ code 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 Apple II, the BASIC command PRINT CHR$(61) displays the lower-

case "a". If you were to examine the code listings for the System Diagnostic Program, you would see several instances in which a CHR$ value is invoked.

Because CHR$ and ASCII values are related to one another on a one-to-one basis, CHR$(61) and ASCII 61 both represent the lowercase "a". The difference between the uses of the two code value schemes is that CHR$ is used to invoke letters and functions within a BASIC program, while ASCII is used for processing of keyboard input in the CPU. The ASCII value is also used to convert (using the BASIC ASC instruction) numeric string numbers into numeric values that can be used as such.

Key Interpretation. After the ASCII value is generated, it is sent to a special program residing in ROM which processes all input and output functions. This program reads the ASCII value and interprets what function the key is expected to perform. If the key is to be displayed on the screen, further processing transfers its value to the character generator ROM, which converts the ASCII value into a form that can be used by the monitor. If the value represents a particular key combination, such as CTRL G, further processing in the I/O ROM causes the designated function of CTRL G to be executed.

KEYBOARD PREVENTIVE MAINTENANCE

Preventive maintenance for the keyboard is simple and important, consisting of three items:

- Use a dust cover on the computer when it is not in use
- Do not eat or drink near the computer
- Vacuum the keyboard regularly

It is particularly important to use a dust cover on the keyboard whenever it is not being used. The key contacts can quickly become defective due to an accumulation of dust and dirt. Bits of food falling around the sides of the keys will work their way down under the key and interfere with the contacts, and cause keys to malfunction due to poor contact. If a drink is ever accidentally spilled into the keyboard, the keyboard will probably need to be replaced. This is particularly expensive for the Apple II systems, because replacing the keyboard means replacing the entire system unit.

Use a hand-held vacuum cleaner or, if necessary, a can of compressed air to vacuum or blow around the keyboard keys. This removes any dust and dirt that has gotten into the keyboard in spite of the dust cover.

If you have to remove a keycap for a repair take the opportunity to clean the inside of the keycap with a cotton swab moistened with denatured isopropyl alcohol (IPA). Clean the key contact with spray contact cleaner. These cleaning measures will prevent future problems with that key.

KEYBOARD TROUBLESHOOTING AND REPAIR GUIDELINES

This section describes various keyboard problems, explains what causes them, and describes how to fix them.

Problem: The Keyboard Doesn't Function At All

If, after pressing the keys on the keyboard nothing displays on the screen, then the keyboard is not making contact with the CPU. The character signals are not being processed by the CPU to be displayed on the screen.

Solutions:

- The first thing to check is whether the system unit's power light, located on the lower left of the keyboard, is illuminated. If the light is off, the system unit is not getting power.

  Check the electrical outlet, power strip (if applicable), wall switch (if applicable), and the building's fuse box or circuit breaker. If these are in order, check the system's power supply. This is a fairly simple procedure on the Apple IIc, because the power supply is external to the system. First, disconnect and remove the power supply and swap it with one from another IIc system. If the other system fails to power up, then buying and installing a new Apple IIc power supply will solve the problem.

  If you own the Apple II Plus or Apple IIe, which have internal power supplies, take the system unit to an Apple II service center and have the power supply checked.

- If the white keyboard light is illuminated, but the keyboard does not work, check the monitor for the standard screen prompts. These prompts, displaying the right bracket cursor, indicate that the computer is in Applesoft BASIC.

  1) Turn the computer off.
  2) Make sure the power supply cable is securely connected to the system.
  3) Turn the computer on again.

  If the screen display still does not look right, refer to Chapter 3, "The System Unit" for system unit troubleshooting procedures.

- If the power light is illuminated and the system prompts are displayed correctly on the screen, but the keyboard still does not work, then there is a problem with the keyboard circuitry. Take the system in for servicing.

Problem: An Individual Key Does Not Work

If after pressing a character key, its character does not display on the screen, then that key might have a problem. If you press a function key, but it does not perform its specified function, that key might be defective as well. For one of several reasons, the key's ASCII code is not getting to the CPU for processing. Try the following solution.

Solution:

- Use a key extractor to lift the key from the keyboard (Fig. 4-3). Then lightly spray contact cleaner on the key post (Fig. 4-4). 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's contact switch might be damaged. Take the unit in for servicing.

Problem: A Key Sticks When Pressed

A sticking key stays down after being pressed, causing its character to fill up the whole screen until you physically lift the key up again. There are three remedies for this type of problem.

Solutions:

- Use the key extractor to lift the key cap off. Then spray contact cleaner on the key post as described above. This removes dust and dirt that might be obstructing the free movement of the key.

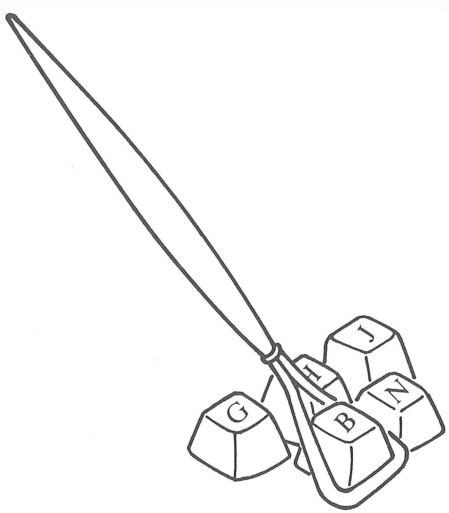


Fig. 4-3. Removing a key.

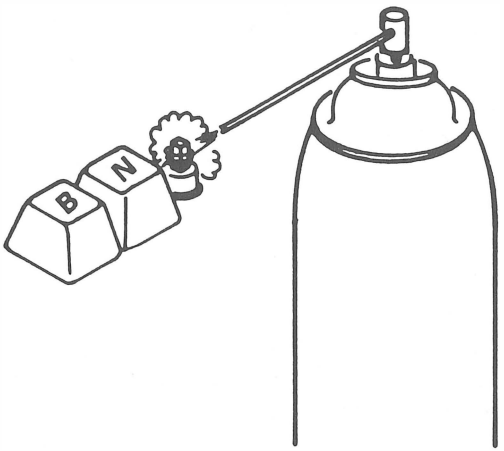


Fig. 4-4. Spraying contact cleaner on the key post.

- If the key still sticks after being pressed, remove the key cap again, and use a small metal file to lightly file the edges of the key cap itself (Fig. 4-5). This increases the distance between the key and key case so that the keycap can move up and down more freely.

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 lose tension as they age.

Solutions:

- Call computer retailers to find keyboard springs for your keyboard. Lift off the key cap and remove the spring from its post (Fig. 4-6). Replace the old spring with the new one, and replace the key cap.

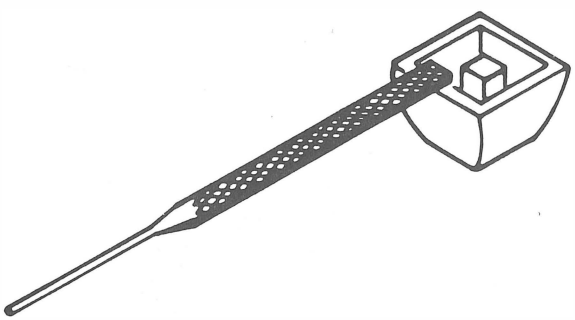


Fig. 4-5. Filing the edges of the key.

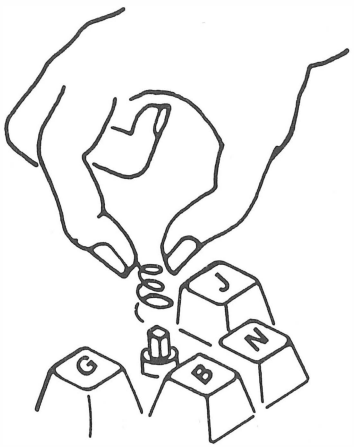


Fig. 4-6. Removing the key spring from the post.

- If you cannot find a replacement spring, use one from a keyboard key that is rarely used. It could be the backslash (\) key, or maybe the plus/equal sign (+/=) key.

Problem: Keyboard Needs to Be Serviced

If the above solutions fail to solve the keyboard problems described above, call the Apple II service center and talk to the technician. Describe the make, model, and configuration of your computer and keyboard, and then describe the problem it is experiencing. Also tell them what you have already tried to do to find and fix the problem, and describe your results.

Provide the technician with all the information you have. Don't leave out any details. Find out whether the technician has any ideas on what the problem might be. If the technician is unable to help you over the phone, find out how much of the system you should bring. You will probably only need to bring the keyboard/system unit itself.

Have the technician provide an estimate as to how long the work might take, and how much it might cost. Be sure to get a copy of the work order.

Before accepting your computer have the technician demonstrate that the problem has been repaired. Test your system after you set it up and reconnect it. Refer to Chapter 9, "The Exerciser," for instructions on burning in your system after repair.

RUNNING THE KEYBOARD DIAGNOSTIC MODULE

The Keyboard Diagnostic module consists of the Echo Key Test. This test displays, or echos, any key pressed on the keyboard along with its ASCII value. If

you press a special character or key combination, the name of the character or combination is displayed, along with its ASCII value.

Run the Keyboard Diagnostic module, and then press a key. This test will read and displays the key, and let you check whether or not it, or even the entire keyboard, is working. The Keyboard Diagnostic module also displays the key's ASCII value, letting you check whether the keys are sending the proper code to the computer to display the proper text character, graphic symbol, special character, or function. The *Apple II Reference Manual,* provided with your Apple II system, includes a table that shows which key and/or function each ASCII value is supposed to provide.

Follow the instructions provided in "Running the Diagnostic Program" in Chapter 2 to run the Keyboard Diagnostic module. Then press K or k to initiate the Keyboard Diagnostic module. The Echo Test screen is displayed, together with a prompt to press the key to be echoed. Any key on the keyboard can be tested: alphanumeric keys, special character keys, and cursor keys. The keycap legend of the pressed key is displayed, along with its ASCII code value and any applicable key combination (Fig. 4-7).

If a character other than the one pressed is displayed, or if no ASCII code value is displayed, then the first troubleshooting step has been accomplished: there is

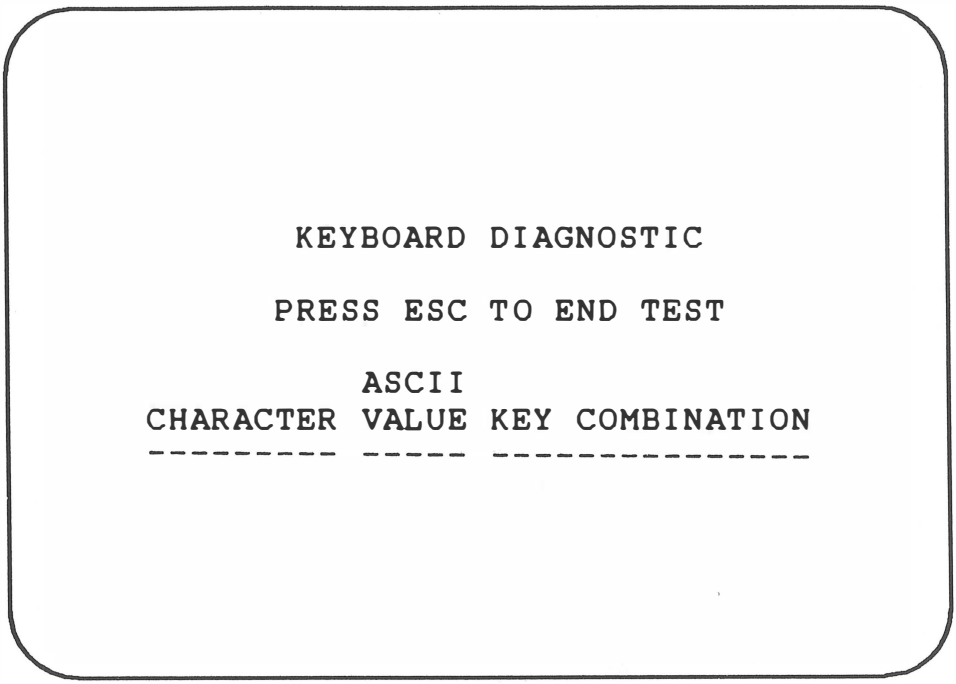


Fig. 4-7. The keyboard diagnostic test.

something wrong with the keyboard. Refer to "Troubleshooting and Repair Guidelines" earlier in this chapter to get the keyboard working properly again.

After testing is completed, press the ESC key to return the program to the System Diagnostic Main Menu.

HOW THE KEYBOARD MODULE WORKS

When you press K or k from the main menu, the System Diagnostic Program branches to line 200, where the Keyboard Diagnostic module begins. Figure 4-8 is a complete listing of the Keyboard Diagnostic module.

Line 200 is the REMARK statement indicating that this is the beginning of the Keyboard Diagnostic module.

```
200 REM KEYBOARD DIAGNOSTIC MODULE
```

Line 210 clears the screen and positions the cursor with the HOME instruction. Then a switch, called S, is set to 0, indicating that the module has just been initialized. When the switch is set to 0, the program branches to line 210 to begin displaying the test title and prompts. After this is done, the switch is set to 1.

```
205 HOME :S = 0: GOTO 210
```

Line 207 sends the program to the Get-a-key subroutine at line 8000. The key that you press for testing is used later in the program, after the test title and

```
200  REM  KEYBOARD DIAGNOSTIC MODULE
205  HOME :S = 0: GOTO 210
207  GOSUB 8000
210  HOME : PRINT  TAB( 7)"KEYBOARD DIAGNOSTIC": PRINT
215  PRINT  TAB( 9)"PRESS ESC TO END TEST": PRINT
220  PRINT  TAB( 13)"ASCII": PRINT TAB( 3)
     "CHARACTER VALUE KEY COMBINATION":
     PRINT  TAB(3)"--------- ----- ---------- -----"
230  IF S = 1 THEN  GOTO 250
235  S = 1
240  GOSUB 8000
250  IF  ASC (A$) = 27 THEN  GOTO 50
260  IF ASC (A$)  < 32 THEN  GOTO 305
300  PRINT  TAB( 6)A$;: PRINT  TAB( 15) ASC (A$);:
     GOTO 207
305  K$ = CHR$ (ASC (A$) + 64)
310  PRINT  TAB( 6)A$; TAB( 15) ASC (A$); TAB( 23)
     "CTRL ";K$: GOTO 207
```

Fig. 4-8. Keyboard diagnostic module listing.

prompts are displayed. Once this is done, the program can return to line 207 for repeated tests of different keys.

```
207 GOSUB 8000
```

Lines 210 and 220 display the test title, prompt, and test setup display.

```
210 HOME : PRINT TAB( 7)"KEYBOARD DIAGNOSTIC": PRINT
215 PRINT TAB( 9)"PRESS ESC TO END TEST": PRINT
220 PRINT TAB( 13)"ASCII": PRINT TAB( 3)
    "CHARACTER VALUE KEY COMBINATION":
    PRINT TAB(3)"--------- ----- ---------- -----"
```

Line 230 checks switch S to see whether it is set to 1. If it is, the program branches to line 250. If this is the first run through the test, S starts out at 0, as set in line 205, in which case the program branches to line 235.

```
230 IF S = 1 THEN GOTO 250
```

Line 235 then sets the switch to 1, indicating that the initial pass through the program has been completed. The test screen is displayed.

```
235 S = 1
```

The program then proceeds on to line 240, which goes to the Get-a-key subroutine at line 8000.

```
240 GOSUB 8000
```

Line 250 checks whether or not the ESC key has been pressed. The ESC key is ASCII value 27. If the value in A$, which is the Get-a-key variable, is equal to 27, then you must have pressed ESC, and the program branches back to line 50, to end the Keyboard Diagnostic module and display the System Diagnostic Main Menu again.

```
250 IF ASC (A$) = 27 THEN GOTO 50
```

Line 260 checks whether the ASCII value of A$ is less than 32, indicating that a control sequence such as CTRL D has been pressed. If this is the case, the program branches to line 305.

```
260 IF ASC (A$) < 32 THEN GOTO 305
```

Line 300 prints the character that has been pressed as well as its ASCII value onto the test screen display. The program then branches back to line 207, which waits for you to press another key to test.

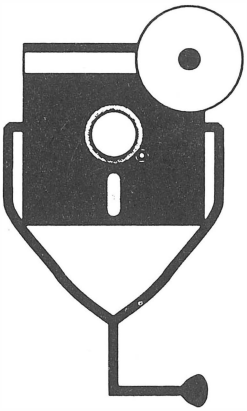
```
300 PRINT TAB( 6)A$;: PRINT TAB( 15) ASC (A$);:
    GOTO 207
```

If a control sequence was encountered at line 260, the program branches around line 300 to line 305. Line 305 sets variable K$ equal to the CHR$ of the ASCII value held in A$, and adds 64, to calculate which key was struck in combination with the CTRL key.

```
305 K$ = CHR$ (ASC (A$) + 64)
```

Line 310 prints the character stored in A$, its ASCII value, the word CTRL, and then prints the key that was pressed together with the CTRL key, as calculated in line 305. When this is done, the program branches to line 207 to wait for another key to be tested.

```
310 PRINT TAB( 6)A$; TAB( 15) ASC (A$); TAB( 23)
    "CTRL ";K$: GOTO 207
```



5

The Monitor

Visual feedback, or output, is necessary when working with your computer. You need to see the input as you enter it, as well as processing results as soon as the computer produces them. Just as the keyboard is the primary input device, the monitor is 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 currently being used (Fig. 5-1).

The Apple II systems all support the 40-column x 24-row screen display format. In addition, the 80-column x 24-row display mode is built into the IIe and IIc, and is available as an option on the II Plus.

The Apple II systems support both low-resolution and high-resolution graphics modes. The low-resolution mode consists of 40 x 48 screen locations which can be displayed in 16 different colors. The high-resolution mode supports 140 x 192 pixel locations in six colors, or 280 x 192 pixel locations in black and white.

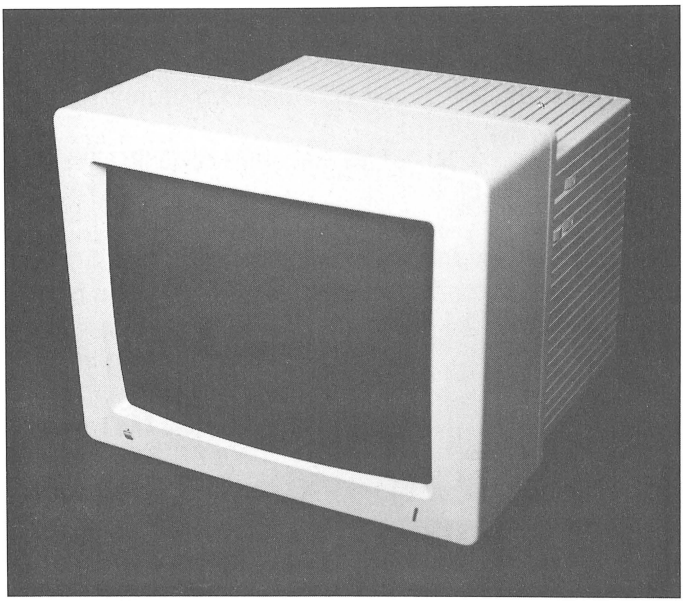


Fig. 5-1. The monitor.

Types of Monitors

There are several types of monitors that can work with the Apple II system. They include:

- a television set (color or black-and-white)
- a high-resolution monochrome graphics monitor
- a high-resolution color graphics monitor

The monitor you use with your Apple must be able to use the RCA phono connector. Any computer monitor can accommodate this type of cable.

Television Set. The Apple II series computers, like many home computers, can use a television set as its display monitor. An external radio frequency (RF) converter is attached to the television via the television's VHF antenna leads. If your television is cable-ready, a coaxial cable connector can be used instead of the antenna leads. The Apple II cable attaches from the internal RF converter (Fig. 5-2) to an RF converter on the television using a composite video cable. The television RF converter includes a switch to choose between using the television as a computer monitor (COMPUTER) and using the television to view broadcasts (TV). The television channel selection must also be set to channel 33 when working with the Apple II.

The television set monitor can display graphics and color, but only in the 40 x

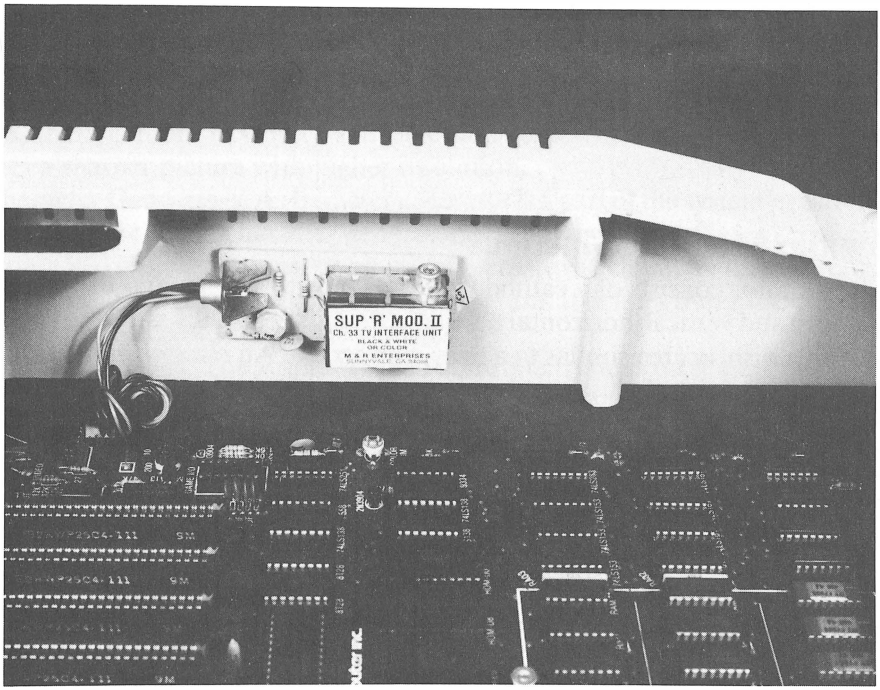


Fig. 5-2. The RF converter.

24 text format (as contrasted with the 80 x 24 text format on a standard computer monitor).

The low-resolution graphics mode divides the screen into a matrix of 40 x 48 blocks. Each of these blocks can be assigned one of the 16 available colors, and can draw low-resolution graphics.

High-Resolution Graphics Monitor. Some software programs utilize the high-resolution graphics mode, which illuminates every speck of light, or *pixel* picture element, in different combinations to form pictures. These pixels allow the creation and display of 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 that can be accommodated on the screen at one time. The Apple graphics monitors accommodate 280 x 192 monochrome pixels. The various display modes are selected through the software program currently in use, or through BASIC commands.

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 the computer to do business graphics or design work. The Apple graphics monitors accommodate 140 x 192 pixels with six colors.

If you use a color television as the computer monitor, colors are already available to you.

How the Monitor Functions

The monitor consists of a cathode ray tube, power supply, and the contrast, brightness, and vertical/horizontal hold adjustments. The CRT is a large vacuum tube with a display area similar to a television screen. An *electron gun* inside the vacuum tube collects, focuses, and emits an electron beam for controlled contact on the back of the picture tube, which is coated with *phosphors.* As the electrons strike the phosphor atoms, the atoms begin to glow. This reaction forms a small, clearly defined light spot (Fig. 5-3).

A special matrix of these light spots are the dots that form each of the characters. On a monochrome monitor, about seven of these light spots positioned in a certain manner make up one character. On a high-resolution monitor, these light spots are the pixels which can densely form the characters, graphic elements, and drawings. The series of dots used to build characters is called the *dot matrix.*

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

In the text mode, the monitor's display works on a character matrix. The display is divided into 24 horizontal rows which are intersected by 40 or 80

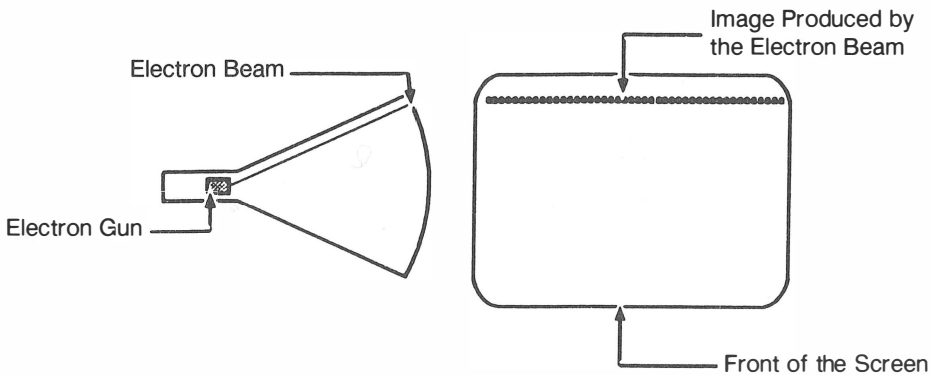


Fig. 5-3. The electron beam and display image.

vertical columns. The position where a column and row intersect is one character location (Fig. 5-4).

The high-resolution or color display is a matrix as well, but a more detailed matrix, working at a pixel level rather than a character level. Since an individual pixel is much smaller than a character, and since 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). The character generator actually produces the series of pixel ON and OFF representations, or bit map, to create the character that is accessed either through the keyboard or through a program. It defines the entire character set that can be accessed from the system.

Video Controller. The Apple II video controller resides on the motherboard. These circuits control the display mode: text, low-resolution, and high-resolution. The character generator ROM is a part of the video controller. A series of RAM chips comprise video memory, which provide the memory locations on the screen for a character, low-res block, or high-res pixel to be displayed at a given location on the screen. A color trim resistor is also included with the video controller circuitry, allowing the adjustment of monitor color.

The Monitor Interface. When using a television as the monitor, the composite cable with the round single-pin RCA phono jack is used to create the monitor interface between the two RF converters. On the Apple side, there is the built-in

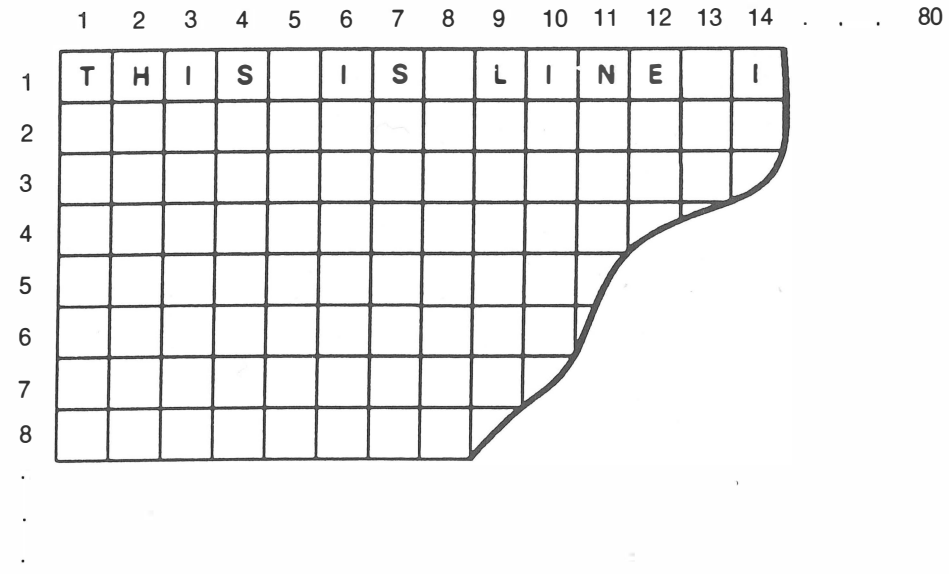


Fig. 5-4. The character location matrix.

RF converter inside the system unit. On the television side, the RF converter connects to the VHF antenna terminals. The composite cable connects between the two RF converters to form the television monitor interface. The RF converter receives the computer's digital signals and converts them into the analog signals required by the television.

If you use a computer monitor instead of a television as the video output device, note that the II Plus and IIe can only use composite-type computer monitors with an RCA phono jack. On the other hand, the IIc provides two types of video connectors: RCA as well as RGB.

The video interface circuit resides on the motherboard in all Apple II systems. This circuit includes trim potentiometers for adjusting color and horizontal synchronization.

MONITOR PREVENTIVE MAINTENANCE

Monitor preventive maintenance consists of some everyday working habits, and some periodic cleaning procedures.

Etching

Always be sure to turn the monitor off or turn the brightness down whenever you leave the computer for a while. This will easily prevent *etching,* which can be an expensive problem.

Etching, or ghosting, takes place on the screen when one particular format, such as a text editor menu or database screen, is left 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 can permanently burn, or etch, that image into the screen.

After etching occurs, even when something else is displayed, a ghost image of the etched display can still be seen. At the same time, the display you want to see is fainter and less defined. Once it happens, nothing can erase the image etched into the display. In other words, etching cannot be fixed, outside of buying 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.

Dust Cover

Always place a dust cover over your monitor whenever it is not in use. The monitor, especially the screen itself, attracts a lot of dust. Because of the high-voltage electrical charge, dust can also fall through the cooling vents, settling on and ultimately corroding sensitive monitor components.

Cleaning

At least once a month, wipe the monitor case down with a lint-free cloth dampened with denatured isopropyl alcohol. Use a hand-held vacuum cleaner or compressed air to clean dust accumulated around the ON/OFF switch, brightness control, contrast control, and any other controls and switches on the monitor.

Clean the monitor screen with a special optical cleaner. Household glass cleaner is not always appropriate. Check the monitor documentation or a computer retailer for screen cleaning recommendations.

Monitor Interface Connector

Check the monitor interface connector and receptacle and clean out any dust or dirt. Remove any oxidation with an eraser, denatured isopropyl alcohol, or an emery cloth; whichever is most convenient and effective.

MONITOR TROUBLESHOOTING AND REPAIR GUIDELINES

This section describes problems that can occur with your monitor, and how to troubleshoot and fix them. Many of the troubleshooting steps include running the Monitor Diagnostic module. Further instructions and explanations of this module are provided in the next section entitled "Running the Diagnostic Module."

Problem: Nothing Displays on the Monitor

If nothing is displayed on the monitor, try the following solutions.

Solutions:

- Make sure the monitor is turned on. Check the brightness and contrast knobs, and make sure they are turned at least two-thirds of the way up. The controls may have been turned down to prevent the screen from etching (see "Etching" above).
- Check that the monitor is properly connected. If the monitor is a television set plugged into a different socket than the computer, check to see whether the outlet is controlled by a light switch. If it is, make certain the light switch is turned on.

  Note: It's not a good idea to use a light switch controlled outlet for the computer or monitor. Someone might accidentally snap the switch off while you are using your computer, causing the system to crash and data to be lost.

- Make certain the electrical outlet is active and working. Test this by checking whether or not another electrical device plugged into the socket works. If it doesn't, then you have an electrical supply problem. Check the fuse box or circuit breaker.
- Check the cabling to ensure that the monitor is properly attached to the system unit monitor interface on the motherboard.
- After 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. 5-5). The monitor's power supply might be defective if the screen remains blank. Take the monitor in for servicing.
- Besides the monitor simply not working, if it's making a crackling noise, it might indicate a problem with the monitor interface. A missing cursor also indicates an interface problem.

  First try swapping a different monitor or television onto your system. If the new monitor works, then your monitor needs servicing. If the swapped monitor fails to work, then you might have a problem with the system unit. Refer to Chapter 3 for system unit troubleshooting procedures.

Problem: Static on the Television Screen

If you are using a television monitor, and it displays static, or snow, rather than a picture, it could indicate a problem with the television set, the RF converter, or the system unit.

Solutions:

- First, check your television. Switch the RF converter to the TV mode, or disconnect it altogether. Set the television channel selector to a known working television station. For example, if channel 2 television broadcasts are usually received in your area, switch it to channel 2. If channel 2 does not come in, nor does any other channel, this indicates a problem with the

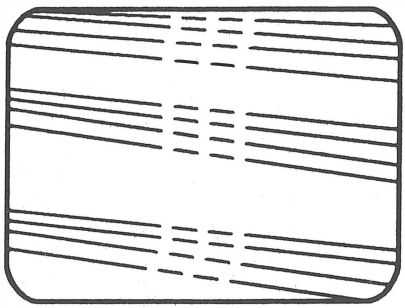


Fig. 5-5. High brightness lines.

television's tuning or receiving circuitry, and it should be taken to a television repair shop for servicing.

- If television broadcast pictures are received with no problem, then the television is working fine, and the RF converter should be tested.

  The RF converter has two switch positions: COMPUTER and TV. Make sure that the converter is switched to the COMPUTER position. Also, check that the converter is properly connected to the VHF rather than UHF antenna leads.

  Next, make certain the television channel selector is set at Channel 33 to work with your computer.

  If the screen still just shows static, replace the RF converter. You can purchase one at a stereo or electronics store.

- If you try all these tests with your television set and RF converter, and there is still static on the television screen, then the system unit is probably malfunctioning. Refer to Chapter 3 on how to troubleshoot system unit problems.

Problem: Misaligned Display

Sometimes the monitor display can become skewed, or it can appear compressed. It can also go out of alignment (Fig. 5- 6). Follow these suggestions to check for proper alignment.

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. These are labeled as Horizontal,

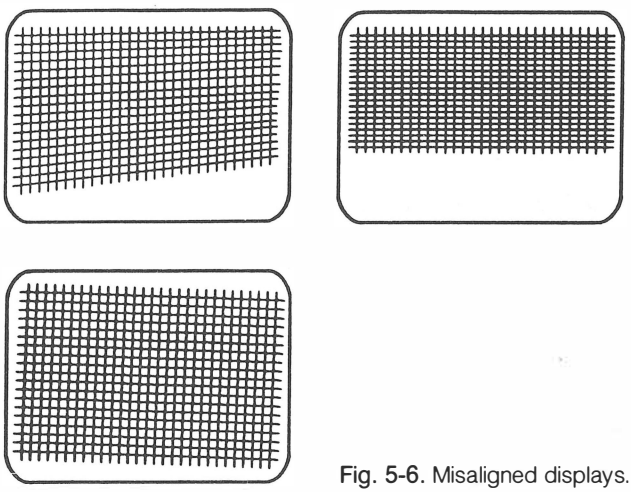


Fig. 5-6. Misaligned displays.

Vertical, Brightness, and Contrast adjustments, and serve to deflect the electron beam or change its intensity.

Use the tuning wand (do not use a metal screwdriver) to gently turn the Horizontal and Vertical potentiometers ("pots") either clockwise or counterclockwise to bring the screen back into alignment. Note the direction you're turning the pot, and turn only one-quarter turn at a time, so you can easily return the pot to its original position if you over-adjust.

- If adjusting these pots does not bring the monitor back into the proper alignment, take the monitor in for servicing.
- If the monitor is a television set, adjust the horizontal and vertical hold controls until the alignment grid looks straight.

Problem: Defective or Missing Character

If you press a character key on the keyboard, but do not see it displayed on the monitor, run the Character Set Test of the Monitor Diagnostic module. See if the character is displayed in that environment. If it is missing, this confirms a problem with the character generator ROM.

If you notice that part of a character is missing or malformed, run the Display Test to display that character across the entire screen. If the defect appears in every instance of the character throughout the screen, this also indicates a problem with the character generator.

To solve this problem, you can either replace the character generator ROM (chip 2513 on the Apple II Plus and IIe), or take the motherboard to an Apple II service outlet and have them replace it for you.

If you run the Display Test and do not see all the expected characters, or if the display is incorrect or garbled, then there is a problem with the system unit itself. Refer to Chapter 3 on how to troubleshoot the system unit.

Problem: Display Pages Do Not Scroll Properly

When the screen is completely filled with text and you enter more characters on the next line, the entire screen display moves 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.

Run the Scroll Test to check this. If the display does not scroll properly, then there is a problem with the display section of the system unit. Refer to Chapter 3 on how to troubleshoot the system unit.

Problem: Missing Text Mode Character Location

If you're entering characters in the text mode, and you notice that there is a location on the screen into which you are not able to display any character, then there is a problem with the text mode character location. Try the following solutions.

Solutions:

- Run the Display Test of the Monitor Diagnostic module. This test fills the entire screen with an entered character. Any blank locations indicate a problem with those text character locations.

  If the Display Test reveals no blank locations, then the problem is not with the monitor, but with the software you were using when the blanks occurred. Also check to see if there's a problem with a specific character, rather than a character screen location.

  If the test reveals blank locations, then there is a problem with the video memory circuitry for that text character location. Each character represents a video memory location. If the character does not display at a certain location, but it does display at other locations, it means the memory location, controlled by one of the RAM chips, is defective.

  Each RAM chip is labeled RAM beside its socket on the floor of the motherboard. Unplug your system, open it up, and carefully remove one of the RAMs using a chip extractor (refer to "Working with IC Chips" in Chapter 1). Take the chip to an electronics supply store and buy one or two exact replacements. Replace the new RAM in the original socket, and check if the problem is still present.

- If the location is still blank, unplug your system again, open it up, and replace the old RAM in its original socket. Remove a different RAM and replace it will the new one. Turn the system on again and check for the problem. Continue in this manner, *shotgunning* the bank of RAM chips until the malfunctioning location is present once again.

- If the character location remains blank after replacing all the RAMS, take the motherboard in for servicing.

Problem: Missing Low-Resolution Location

If you're running an application using the low-resolution graphic mode, and certain sections of the screen appear blank, this indicates a problem with the low-resolution memory locations. Try the following solutions.

Solutions:

- Run the Low-Resolution Test of the Monitor Diagnostic module. This will confirm whether or not there is a problem with the low-resolution memory locations. If any part of the screen is blank instead of filled, then there is a problem with those memory locations.

  If the Low-Resolution Test reveals no blank spots, then the problem is not with the monitor, but with the software you were using when the blanks occurred.

  If the test reveals blank spots, then there is a problem with video memory as controlled by one of the RAM chips. Each RAM chip is labeled RAM beside its socket on the floor of the motherboard. Unplug your system, open it up, and carefully remove one of the RAMs using a chip extractor (refer to "Working with IC Chips" in Chapter 1). Take the chip to an electronics supply store and buy one or two exact replacements. Replace the new RAM in the original socket, and check if the problem is still present.
- If the location is still blank, unplug your system again, open it up, and replace the old RAM in its original socket. Remove a different RAM and replace it will the new one. Turn the system on again and check for the problem. Continue in this manner, shotgunning the bank of RAM chips until the malfunctioning location is present once again.
- If the character location remains blank after replacing all the RAMS, take the motherboard in for servicing.

Problem: Missing High-Resolution Location

If you're running a game or application, such as a spreadsheet or graphics program, that uses the high-resolution graphics mode, and certain characters, or a part of the background appears to have tiny dark spots in them, then there are pixels missing due to a problem with the high-resolution memory locations. Try the following solution.

Solution:

- Run the High-Resolution Test Program, the HIRES program on the diagnostic diskette, but separate from the System Diagnostic Program. This test checks all high-resolution memory locations.

  If no blank spots are revealed by running the High-Resolution Test, the problem is not with the monitor, but with the software you were using when the blanks occurred.

  If any part of the screen is blank instead of filled, then there is a problem with the high-resolution memory locations as controlled by one of the RAM chips. Each RAM chip is labeled RAM beside its socket on the floor of the

motherboard. Unplug your system, open it up, and carefully remove one of the RAMs using a chip extractor (refer to "Working with IC Chips" in Chapter 1). Take the chip to an electronics supply store and buy one or two exact replacements. Replace the new RAM in the original socket, and check if the problem is still present.

- If the location is still blank, unplug your system again, open it up, and replace the old RAM in its original socket. Remove a different RAM and replace it with the new one. Turn the system on again and check for the problem. Continue in this manner, shotgunning the bank of RAM chips until the malfunctioning location is present once again.
- If the character location remains blank after replacing all the RAMS, take the motherboard in for servicing.

Problem: Colors Do Not Display Correctly

When using a color monitor (either a color television or a high-resolution color monitor), another possible problem is that no colors display at all, or that the colors display the incorrect tones. Check the colors with the Low-Resolution Color Test of the Monitor Diagnostic module, as well as the High-Resolution Color Test program.

Solution:

- First make certain that the monitor is actually a color television or a color monitor instead of a black-and-white television or monochrome monitor. Some monitors have a control that allows switching between a monochrome mode and a color mode. Make sure it is in the color mode.
- If the monitor is displaying colors, but they're not the correct colors, there are two adjustments that can rectify this problem.

  For a television monitor, first check a normal television picture, and see if the color is adjusted properly. If it doesn't look right, for instance, if faces look too red, or green, then the color and tint controls on the television set simply need to be adjusted.

  Once the television broadcast picture displays 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, 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 still are not correct, there is a variable resistor on the motherboard that serves to optimize video color.

Follow these steps to use this color trim pot:

1) Turn system power off, disconnect all peripheral cables, and remove the system unit cover.
2) Reconnect the monitor and disk drive cables.
3) Turn system power on again. While the system is open and power is on, do not touch anything. When making adjustments, use only a non-conductive (plastic) tuning wand. Do not use a metal screwdriver.
4) Load and run the System Diagnostic Program. Choose the Monitor Diagnostic, and then the Low-Resolution Color Test (refer to "Running the Monitor Diagnostic Module" later in this chapter for further instructions).
5) Locate the color trim variable resistor on the motherboard (Fig. 5-7).
6) Use the tuning wand to adjust this resistor in small increments until you see satisfactory colors on the screen display.
7) Turn the system off again and disconnect the peripheral cables.
8) Replace the system unit cover.
9) Connect all peripheral and power supply cables.

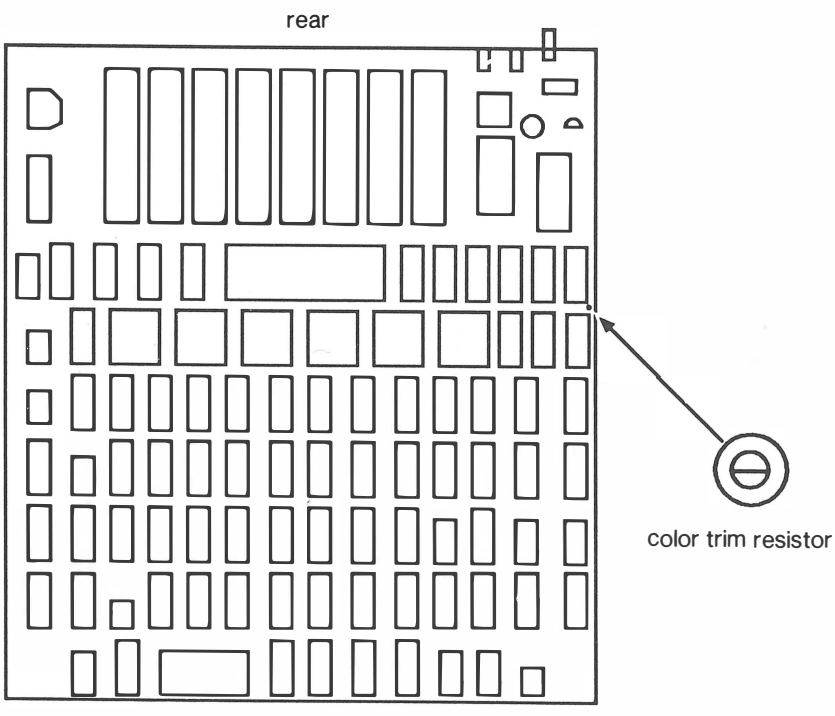


Fig. 5-7. Color trim variable resistor.

10) Power up the system and run several passes of the Exerciser program (see Chapter 9, The System Exerciser) to ensure that the system is functioning properly.

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

Problem: Monitor Needs to Be Serviced

If the troubleshooting procedures described above fail to solve the monitor problem, call your Apple II service center and speak with a technician. Describe the make of your monitor, and the problem it is experiencing. Also describe what you have already tried to do to find and fix the problem, and report the results. Find out if there are any solutions they can suggest over the phone.

If the monitor cannot be repaired over the phone, ask the technician if you should bring in just the monitor, or if the system unit should be brought in as well.

After supplying all available information about the monitor malfunction, find out if the technician has any ideas yet on what the problem might be, and how long and how much it might cost to fix. Be sure to get a copy of the work order.

Before you pay for and accept your monitor, have the repair technician demonstrate that the problem has actually been fixed. When you get the monitor home again, connect everything together, and refer to Chapter 9,"The Exerciser," for instructions on burning in the system after repair. The Exerciser includes an extensive monitor burn-in procedure that should prove very helpful.

RUNNING THE MONITOR DIAGNOSTIC MODULE

To run the Monitor Diagnostic module, follow the instructions provided in "Running the Diagnostic Program" in Chapter 2. Then press **M**. The screen clears, and the Monitor Diagnostic Menu is displayed (Fig. 5-8). To activate one of these tests, press the corresponding number. To end the Monitor Diagnostic module, press the **ESC** key, whereupon the Diagnostic Main Menu will be displayed again.

Display Test

To run the Display Test, press 1. The screen will clear and then display the test instructions (Fig. 5-9). Press a character on the keyboard. (Function keys and the cursor arrow keys do not work for this test, because they do not have a graphic display.) The character will fill the entire screen at every available character location (Fig. 5-10). Any character locations that do not display the character will be very obvious.

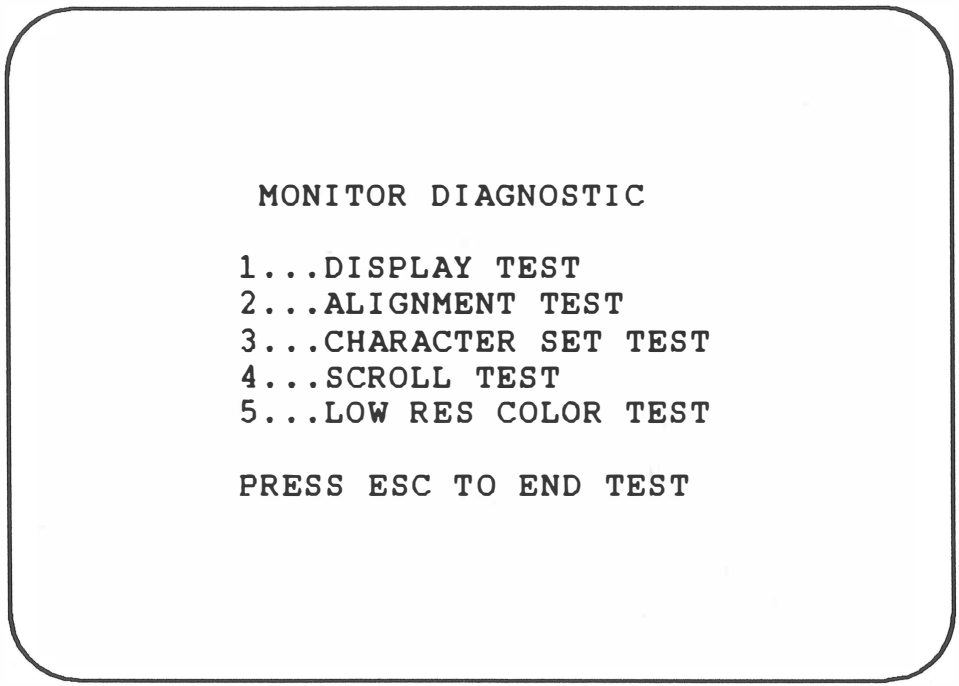


Fig. 5-8. The monitor diagnostic menu.

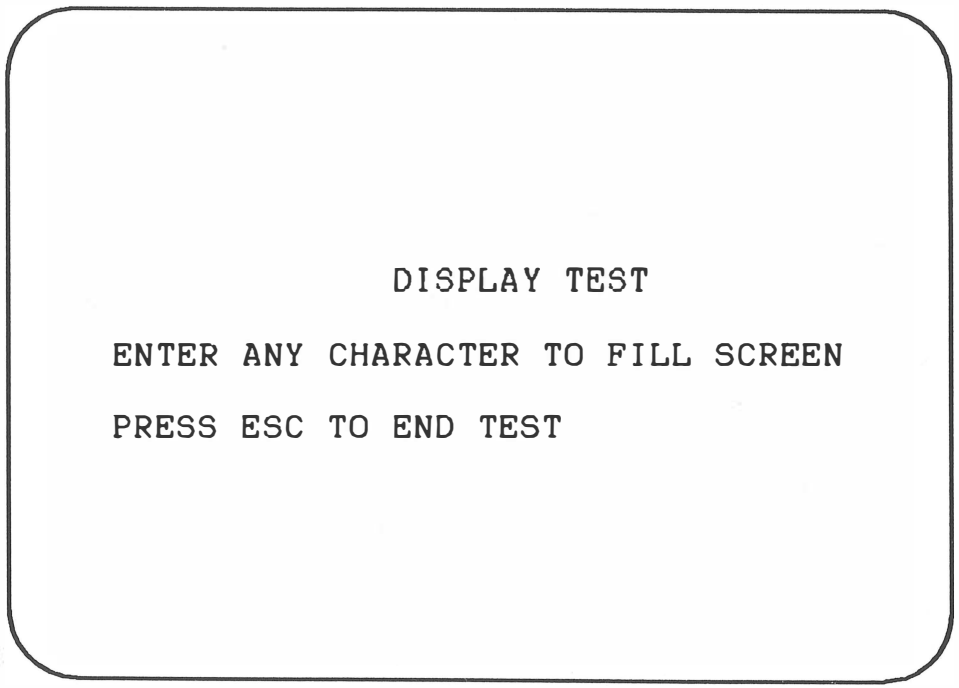


Fig. 5-9. Display test instructions.

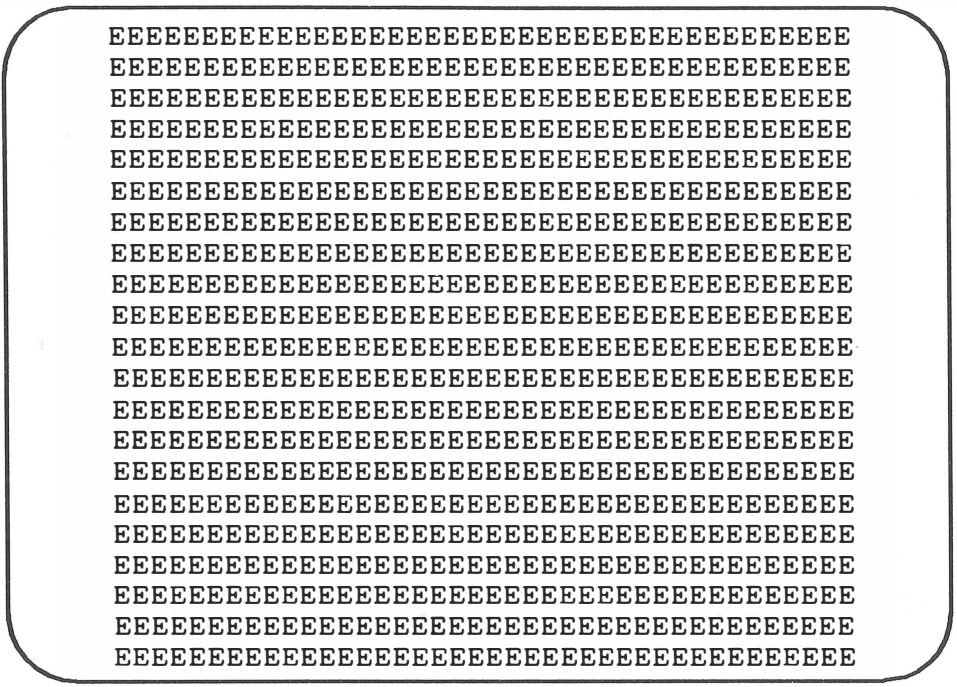


Fig. 5-10. Display test results.

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

Alignment Test

Pressing 2 from the Monitor Diagnostic Menu activates the Alignment Test, which displays the alignment box with horizontal and vertical lines as shown in Fig. 5-11. The alignment grid illustrates any alignment or skew problems. The display should look symmetrical, with straight lines. There should be no distortion. To end this test and return to the Monitor Diagnostic Menu, press the ESC key.

Character Set Test

Press 3 to invoke the Character Set Test. This test checks whether any characters are missing. All 128 display characters are displayed on the screen (Fig. 5-12). Pressing ESC ends this test and displays the Monitor Diagnostic Menu once again.

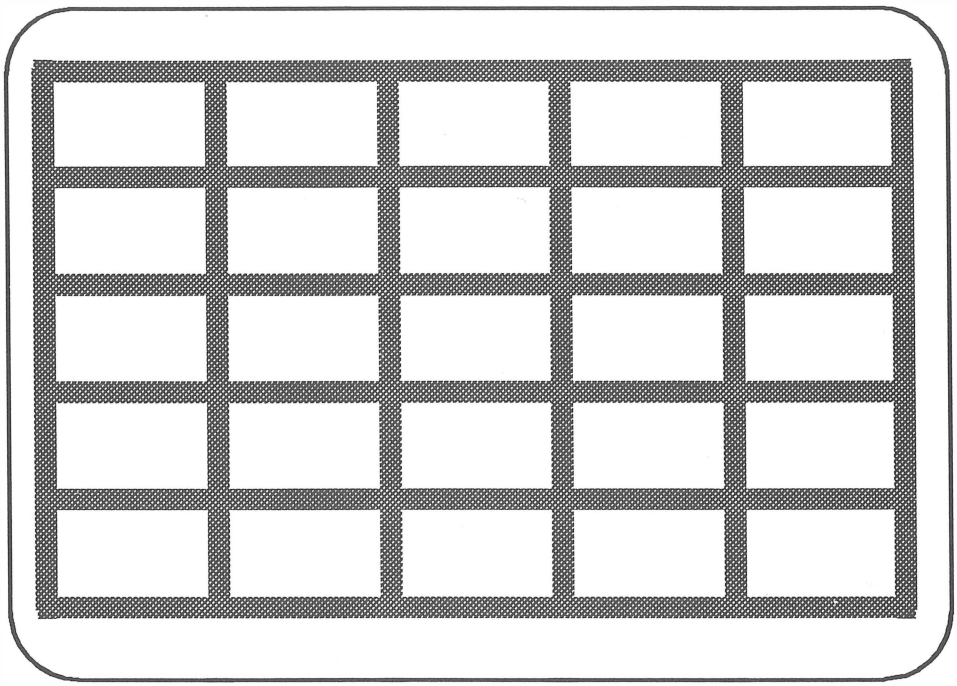


Fig. 5-11. Alignment test results.

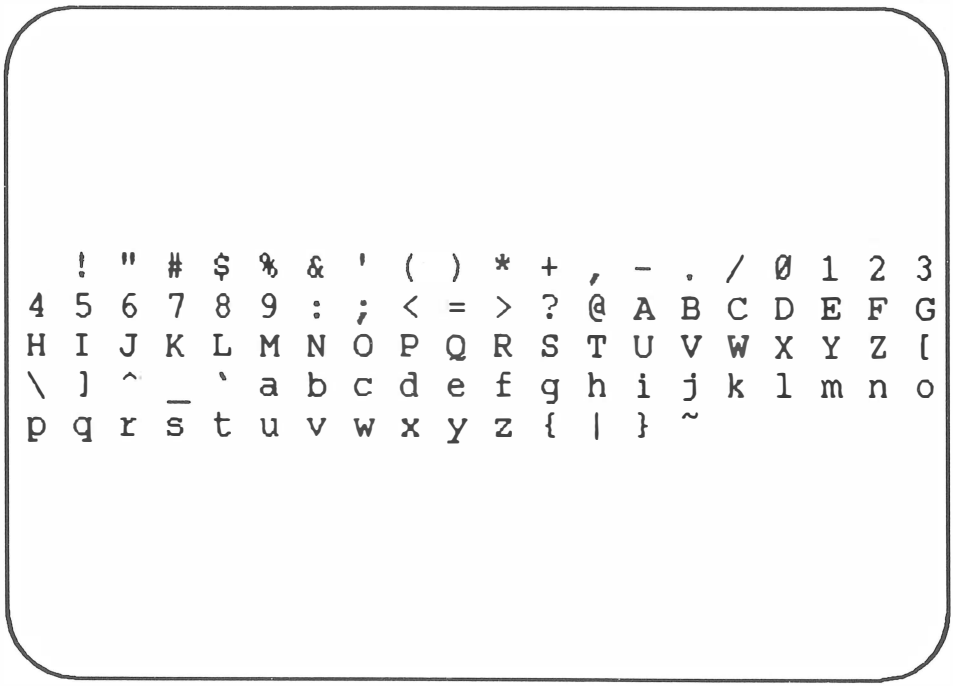


Fig. 5-12. Character set test results.

Scroll Test

To test display page scrolling, press 4 from the Monitor Diagnostic Menu. The Scroll Test prompts for the number of repetitions for the test. If you enter a number less than 35, the program defaults to scrolling 35 lines. The test displays the same line for the specified number of repetitions in a sliding character set display on the screen.

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 down and across the screen, shifting to the right, one character at a time. If it scrolls properly, the screen resembles Fig. 5-13.

Low-Resolution Color Graphics Test

To test the low-resolution color mode, press 5 from the Monitor Diagnostic Menu. The screen displays a menu of the sixteen colors available in the low-resolution mode. Enter the color name exactly as it is shown in the menu, using all uppercase letters. As soon as you enter the name of the color to test, the screen is completely filled with low-resolution blocks at each location. Even if you are not

```
              SCROLL TEST

ENTER NUMBER OF REPETITIONS-35
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno
"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !
#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"
$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#
%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$
&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%
'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&
()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'
)*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'(
*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'()
+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'()*
,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'()*+
-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'()*+,
./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'()*+,-
/0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'()*+,-.
0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'()*+,-./
123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'()*+,-./0
23456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'()*+,-./01
3456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'()*+,-./012
456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'()*+,-./0123
56789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'()*+,-./01234
6789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno !"#$%&'()*+,-./012345
PRESS ESC TO END TEST
```

Fig. 5-13. Scroll test results.

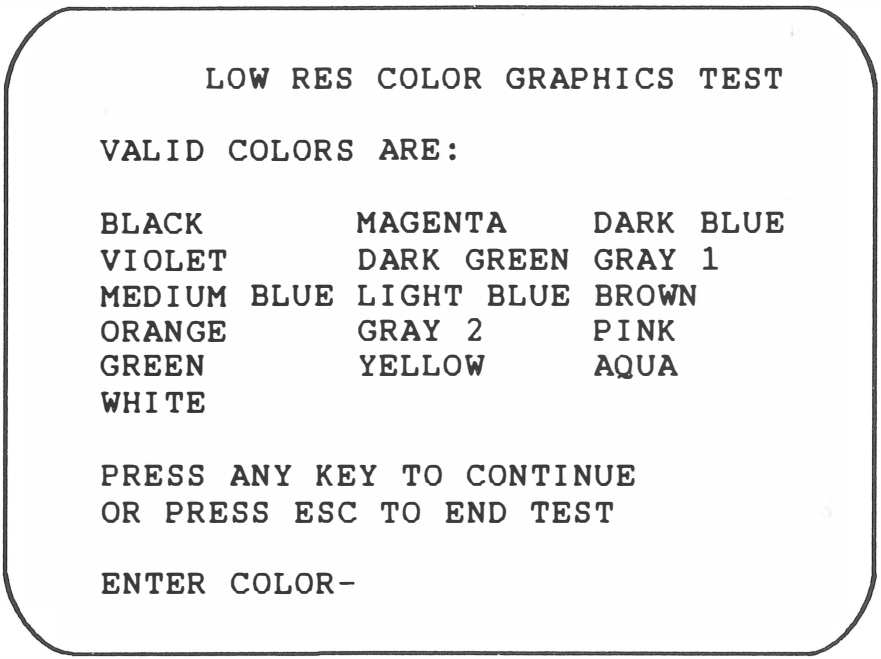


Fig. 5-14. Low-resolution test instructions.

using a color monitor, this is a good test for the low-resolution memory locations (Fig. 5-14).

If there are any locations that are not filled with one of the blocks, then there is a video memory problem. Also, if the selected color is not correctly represented on the screen, then there is a color adjustment problem. Leave this test displayed on the screen, make the necessary adjustments, and view all the colors until they're displaying correctly. Refer to "Problem: Colors Do Not Display Correctly" earlier in this chapter for the adjustment procedure.

Press any key to end the test. The screen will clear and return to the color menu, and another color can be tested if desired. When finished testing colors, press ESC, and the Monitor Diagnostic Main Menu will be displayed again.

High-Resolution Color Graphics Test

To test colors in the high-resolution graphics mode, load the High-Resolution Color Test Program, available as a separate program called "HIRES" on the System Diagnostic Program diskette. If you do not have this diskette, you can key it in yourself following the listing provided in Fig. 5-17.

After this program is loaded, a menu of the six high-resolution colors is displayed (Fig. 5-15). Enter the name of the color to be tested in uppercase letters.

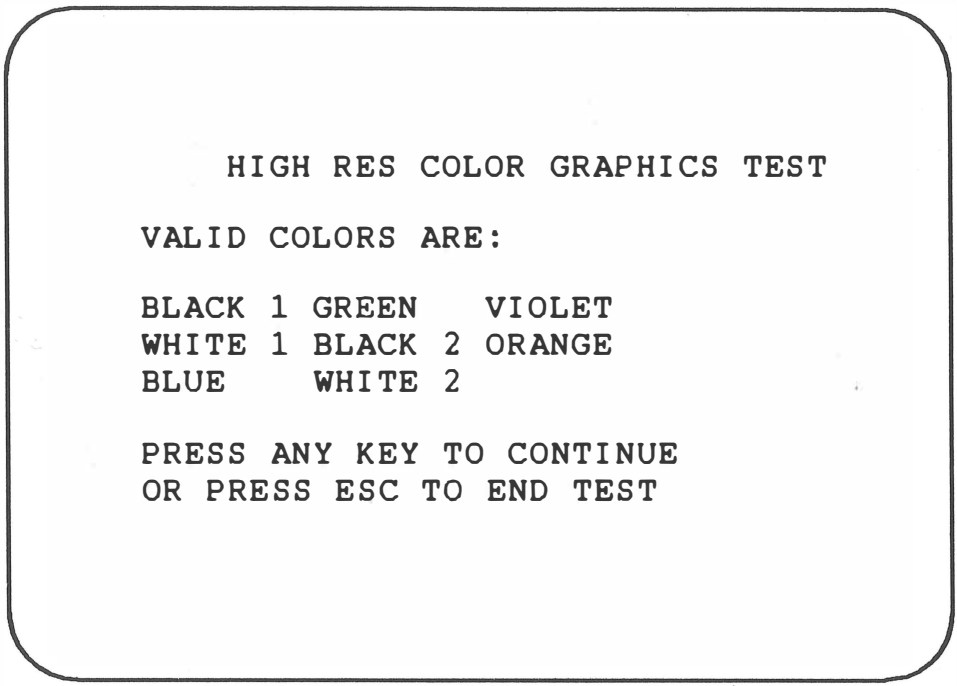


Fig. 5-15. High-resolution color graphics test instructions.

Once this is done, the screen is completely filled with a series of high-resolution pixels in that color at each screen memory location. Any location that is not filled with the color pixel indicates a high-resolution video memory location problem.

Press any key to end the test. The screen will clear and return to the color menu, so another color can be tested if desired. When finished testing colors, press ESC, and the program quits to the Applesoft BASIC environment displaying the right bracket prompt.

HOW THE MONITOR DIAGNOSTIC MODULE WORKS

Selecting M (or m) from the System Diagnostic Main Menu causes the program to branch to line 350, which is the start of the Monitor Diagnostic module consisting of five monitor tests (Fig. 5-16).

Line 350 uses a REMARK statement to indicate the beginning of the Monitor Diagnostic module.

```
350 REM MONITOR DIAGNOSTIC MODULE
```

```
350  REM  MONITOR DIAGNOSTIC MODULE
360  HOME : PRINT  TAB( 6)"MONITOR DIAGNOSTIC":
     PRINT : PRINT  TAB( 5)"1...DISPLAY TEST":
     PRINT  TAB( 5)"2...ALIGNMENT TEST"
370  PRINT  TAB( 5)"3...CHARACTER SET TEST":
     PRINT  TAB( 5)"4...SCROLL TEST": PRINT  TAB( 5)
     "5...LOW RES COLOR TEST"
380  PRINT : PRINT  TAB( 5)"PRESS ESC TO END TEST"
390  GOSUB 8000: IF  ASC (A$) = 27 THEN  GOTO 50
400  IF A$ = "1" THEN  GOTO 470
410  IF A$ = "2" THEN  GOTO 540
420  IF A$ = "3" THEN  GOTO 660
430  IF A$ = "4" THEN  GOTO 730
440  IF A$ = "5" THEN  GOTO 860
465  PRINT  CHR$ (7): GOTO 390
470  REM  DISPLAY TEST
475  HOME : PRINT  TAB( 14)"DISPLAY TEST": PRINT
480  PRINT "ENTER ANY CHARACTER TO FILL SCREEN":
     PRINT : PRINT "PRESS ESC TO END TEST"
490  GOSUB 8000
495  HOME
500  IF  ASC  (A$) = 27 THEN  GOTO 350
510  FOR I = 1 TO 959
520  PRINT A$;: NEXT I
525  IF EX = 1 THEN  GOTO 680
530  GOTO 490
540  REM  ALIGNMENT TEST
550  GR : POKE  - 16302,0: CALL - 1998: COLOR= 15
560  FOR I = 1 TO 45 STEP 8
570  HLIN 0,39 AT I - 1
575  HLIN 0,39 AT 47
580  NEXT I
590  FOR I = 1 TO 40 STEP 8
600  VLIN 0,47 AT I - 1
610  NEXT I
620  VLIN 0,47 AT 39
640  GOSUB 8000
650  TEXT : GOTO 350
660  REM  CHARACTER SET TEST
670  HOME : PRINT  TAB( 11)"CHARACTER SET TEST":
     PRINT : PRINT "PRESS ESC TO END TEST": PRINT
680  FOR I = 0 TO 128
690  PRINT  CHR$ (I) + " ";: NEXT I
695  IF EX = 1 THEN  GOTO 750
700  GOSUB 8000
```

Fig. 5-16. Monitor diagnostic module listing.

```
710  GOTO 350
720  REM  SCROLL TEST
730  HOME : PRINT  TAB( 15)"SCROLL TEST": PRINT
740  INPUT "ENTER NUMBER OF REPETITIONS-";A
750  IF A < 35 THEN A = 35
760  N = 31
770  FOR I = 1 TO A
780  N = N + 1: IF N > 71 THEN N = 32
790  FOR X = 1 TO 40
800  PRINT  CHR$ (N);:N = N + 1
810  IF N > 71 THEN N = 32
820  NEXT X
830  NEXT I
840  IF EX = 1 THEN  RETURN
850  PRINT : PRINT "PRESS ESC TO END TEST": GOSUB 8000:
     GOTO 350
860  REM  LOW RES COLOR GRAPHICS TEST
870  HOME : PRINT  TAB( 10)"LOW RES COLOR GRAPHICS
     TEST": PRINT
872  PRINT "VALID COLORS ARE:": PRINT : PRINT "BLACK";
     TAB(13)"MAGENTA"; TAB( 24)"DARK BLUE": PRINT
     "VIOLET"; TAB( 13)"DARK GREEN"; TAB( 24)"GRAY 1"
874  PRINT "MEDIUM BLUE"; TAB( 13)"LIGHT BLUE";
     TAB( 24)"BROWN": PRINT "ORANGE"; TAB( 13)"GRAY 2";
     TAB( 24)"PINK": PRINT "GREEN"; TAB( 13)"YELLOW";
     TAB( 24)"AQUA": PRINT "WHITE"
875  PRINT
876  PRINT "PRESS ANY KEY TO CONTINUE": PRINT "OR PRESS
     ESC TO END TEST": GOSUB 8000: IF ASC (A$) = 27
     THEN  GOTO 350
878  PRINT
880  INPUT "ENTER COLOR-";C$
890  IF C$ = "BLACK" THEN C = 0: GOTO 1060
900  IF C$ = "MAGENTA" THEN C = 1: GOTO 1060
910  IF C$ = "DARK BLUE" THEN C = 2: GOTO 1060
920  IF C$ = "VIOLET" THEN C = 3: GOTO 1060

930  IF C$ = "DARK GREEN" THEN C = 4: GOTO 1060
940  IF C$ = "GRAY 1" THEN C = 5: GOTO 1060
950  IF C$ = "MEDIUM BLUE" THEN C = 6: GOTO 1060
960  IF C$ = "LIGHT BLUE" THEN C = 7: GOTO 1060
970  IF C$ = "BROWN" THEN C = 8: GOTO 1060
980  IF C$ = "ORANGE" THEN C = 9: GOTO 1060
990  IF C$ = "GRAY 2" THEN C = 10: GOTO 1060
1000  IF C$ = "PINK" THEN C = 11: GOTO 1060
```

Fig. 5-16. (continued)

```
1010  IF C$ = "GREEN" THEN C = 12: GOTO 1060
1020  IF C$ = "YELLOW" THEN C = 13: GOTO 1060
1030  IF C$ = "AQUA" THEN C = 14: GOTO 1060
1040  IF C$ = "WHITE" THEN C = 15: GOTO 1060
1050  PRINT : PRINT  CHR$ (7): PRINT "INVALID COLOR! -
      RE-ENTER": PRINT : GOTO 880
1060  GR : POKE  - 16302,0: CALL - 1998
1070  COLOR= C
1080  FOR Y = 0 TO 47
1090  FOR X = 0 TO 39
1100  PLOT X,Y: NEXT X
1110  NEXT Y
1120  GOSUB 8000: TEXT : GOTO 870
```

Fig. 5-16. (continued)

Line 360 clears the screen and positions the cursor with the HOME command. Together with lines 370 and 380, the Monitor Diagnostic Menu and prompts are displayed.

```
360 HOME : PRINT TAB( 6)"MONITOR DIAGNOSTIC":
    PRINT : PRINT TAB( 5)"1...DISPLAY TEST":
    PRINT TAB( 5)"2...ALIGNMENT TEST"
370 PRINT TAB( 5)"3...CHARACTER SET TEST":
    PRINT TAB( 5)"4...SCROLL TEST": PRINT TAB( 5)
    "5...LOW RES COLOR TEST"
380 PRINT : PRINT TAB( 5)"PRESS ESC TO END TEST"
```

Line 390 branches to the Get-a-key subroutine at line 8000 to get your menu selection. If you press the ESC key, indicated by an ASCII value of 27 in A$, the Monitor Diagnostic ends and the program branches back to line 50 to display the System Diagnostic Main Menu.

```
390 GOSUB 8000: IF ASC (A$) = 27 THEN GOTO 50
```

Lines 400 through 440 branch to the appropriate monitor test, depending on the user input stored in variable A$. Valid keys are 1 through 5.

```
400 IF A$ = "1" THEN GOTO 470
410 IF A$ = "2" THEN GOTO 540
420 IF A$ = "3" THEN GOTO 660
430 IF A$ = "4" THEN GOTO 730
440 IF A$ = "5" THEN GOTO 860
```

If you press a key other than 1 through 5, the program falls through to line 465. A beep will be sounded to indicate the error. Then the program will branch back to line 390 to allow you to try again with another key.

```
465 PRINT CHR$ (7): GOTO 390
```

Line 470 is the beginning of the Display Test, as indicated by the REMARK statement. The Display Test prompts you to enter a key, and then fills the screen with that character.

```
470 REM DISPLAY TEST
```

Line 475 clears the screen, positions the cursor, and prints the test title.

```
475 HOME : PRINT TAB( 14)"DISPLAY TEST": PRINT
```

Line 480 prompts you to enter the character to fill the screen, and then to press the ESC key to end the test.

```
480 PRINT "ENTER ANY CHARACTER TO FILL SCREEN":
    PRINT : PRINT "PRESS ESC TO END TEST"
```

Line 490 branches to the Get-a-key subroutine to get the user input of the character to test.

```
490 GOSUB 8000
```

Line 495 clears the screen and positions the cursor again.

```
495 HOME
```

Line 500 checks whether or not the ESC key, as ASCII value 27, has been pressed. If it has, the Display Test ends and the program returns to line 350 to display the Monitor Diagnostic Menu again.

```
500 IF ASC (A$) = 27 THEN GOTO 350
```

Lines 510 and 520 constitute the FOR-NEXT loop which fills the screen with the character input you select. This input is stored in variable A$. There are 40 columns and 24 rows, for 960 locations. The FOR-NEXT loop stops at 959,

leaving the last location blank. If this final location were to be filled, it would cause the display to scroll up.

```
510 FOR I = 1 TO 959
520 PRINT A$;: NEXT I
```

Line 525 checks whether or not the Exerciser currently has program control. If it does, EX is equal to 1, in which case the program branches to line 680. If it does not have control of the program, EX is equal to 0, in which case the program proceeds through to line 490 to wait for another key to be tested.

```
525 IF EX = 1 THEN GOTO 680
530 GOTO 490
```

Line 540 begins the Alignment Test with the REMARK statement. In this test, an alignment grid is built in the low-resolution graphics mode, and is used to check whether the monitor's display is properly aligned horizontally and vertically.

```
540 REM ALIGNMENT TEST
```

Line 550 uses the GR instruction to set the screen for operation in the low-resolution graphics mode. The POKE statement and the CALL statement are used in conjunction with one another to ensure that the entire screen page is utilized. Without these two instructions, the last four lines of the screen would consist of a small text window, which would not be desirable for painting the alignment grid. The color instruction sets the color of the screen to white.

```
550 GR : POKE - 16302,0: CALL - 1998: COLOR= 15
```

Line 560 is the start of the FOR-NEXT loop which draws a horizontal line for the alignment grid. The FOR-NEXT loop is run 45 times in steps of eight. This means that a line is drawn on every eighth row.

```
560 FOR I = 1 TO 45 STEP 8
```

Lines 570 and 575 use the HLIN instruction to actually draw a horizontal line from column 0 through column 39. I - 1 indicates the row in which the line is to be drawn, and is related to the current position in the FOR-NEXT loop. Line 575 draws the line on the 47th row to complete the horizontal portion of the alignment grid.

```
570 HLIN 0,39 AT I - 1
575 HLIN 0,39 AT 47
```

Line 580 is the NEXT for the FOR-NEXT loop begun at line 560. This loop is run 45 times in steps of eight, until all the horizontal lines are drawn on the screen from top to bottom.

```
580 NEXT I
```

Line 590 begins a similar FOR-NEXT loop, this time drawing the vertical lines to complete the alignment grid. The loop runs through 40 times, for the 40 columns in a row, drawing a line every eighth row.

```
590 FOR I = 1 TO 40 STEP 8
```

Line 600 actually draws the vertical line, using the VLIN instruction. The line is drawn from row 0 through row 47 at the column relating to where the program is currently in the FOR-NEXT loop.

```
600 VLIN 0,47 AT I - 1
610 NEXT I
```

Line 620 is the final vertical line drawn to close up the grid at column 39.

```
620 VLIN 0,47 AT 39
```

Line 640 branches to the Get-a-key subroutine at line 8000 to wait for the user to press a key to end the alignment test.

```
640 GOSUB 8000
```

Once this is done, the TEXT instruction in line 650 switches the monitor from the low-resolution graphics mode back to the text mode. The program then branches back to line 350 to display the Monitor Diagnostic Menu.

```
650 TEXT : GOTO 350
```

When you press a 3 from the Monitor Diagnostic Menu, the program will branch to line 660 and begin the Character Set Test. This test displays the monitor's full character set on the screen.

```
660 REM CHARACTER SET TEST
```

Line 670 clears the screen and positions the cursor with the HOME instruction. The test title is displayed, as is the prompt for ending the test.

```
670 HOME : PRINT TAB( 11)"CHARACTER SET TEST":
    PRINT : PRINT "PRESS ESC TO END TEST": PRINT
```

Line 680 begins a FOR-NEXT loop which builds and collates the 128 display characters available on the Apple II systems.

```
680 FOR I = 0 TO 128
```

Line 690 prints the character string value of I and adds a blank. The character of CHR$(0) is printed first to satisfy the first loop. The second loop increments variable I to 2, to create CHR$(2). This character is printed. NEXT I loops it back to line 680 again, incrementing variable I to 3. Line 690 reads CHR$(3), and that character is printed. This continues for the full 128 character set, until all characters are displayed on the screen.

```
690 PRINT CHR$ (I) + " ";: NEXT I
```

Line 695 checks whether or not the Exerciser is in effect. If it is, the program branches to line 750. If it is not set, the program proceeds to line 700.

```
695 IF EX = 1 THEN GOTO 750
```

Line 700 branches to the Get-a-key subroutine at line 8000. This waits for you to press any key to end the Character Set Test.

```
700 GOSUB 8000
```

When you press a key, the Character Set Test ends and the program returns to line 350 to display the Monitor Diagnostic Menu.

```
710 GOTO 350
```

When you press 4 from the Monitor Diagnostic Menu, the program branches to line 720, which begins the Scroll Test with a REMARK statement. The Scroll Test builds a line of display characters and prints it at least 35 times to let you check the monitor's scrolling capabilities.

```
720 REM SCROLL TEST
```

Line 730 prints the test title on the screen.

```
730 HOME : PRINT TAB( 15)"SCROLL TEST": PRINT
```

Line 740 is an INPUT statement prompting you to enter the number of repetitions for the scroll line to be printed. Your answer is stored in variable A.

```
740 INPUT "ENTER NUMBER OF REPETITIONS-";A
```

Line 750 checks if A is less than 35. If so, the repetitions are reset to 35. This is done so that you will see enough repetitions of the line across the screen to see scrolling take place.

```
750 IF A < 35 THEN A = 35
```

Line 760 sets variable N to 31, which is 1 less than the first display character, ASCII (32), a blank. This is used in the following FOR-NEXT loop.

```
760 N = 31
```

Line 770 begins the FOR-NEXT loop to build the scroll line.

```
770 FOR I = 1 TO A
```

Line 780 then sets N equal to N+1 as it pertains to the loop. Line 760 had set N to 31, so now it is 32, for the first display character, a space. N begins with a value of 31. Because there are 40 columns in a line across the screen, and only one line is needed for this test, 40 is added to 31 to obtain the value of 71. When N has incremented to a value greater than 71, it is reset back to 32 to start over again.

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

Line 790 begins an X FOR-NEXT loop nested within the I loop. This loop indicates the 40 characters that can be printed across one line on the screen, and does the work of building the actual line.

```
790 FOR X = 1 TO 40
```

Line 800 prints the characters as specified by the two FOR-NEXT loops. First CHR$(32), a blank, is printed. Then N is incremented to 33 for the X loop, and CHR$(33) is printed.

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

Line 810 checks whether N is greater than 71. If it is, N is reset to 32 for the X loop.

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

Line 820 increments the X loop.

```
820 NEXT X
```

Line 830 increments the I loop, and the specified repetitions of the scroll line are printed across the screen until it scrolls.

```
830 NEXT I
```

Line 840 checks the Exerciser flag. If the Exerciser is in operation, indicated by a 1, it returns to the Exerciser module, as it is now finished with the monitor Exerciser tests.

```
840 IF EX = 1 THEN RETURN
```

If the Exerciser is not in effect, the program proceeds to line 850. This line displays a user prompt to end the test, and the response is received through the Get-a-key subroutine at line 8000. The Scroll Test ends and the program branches back to line 350 to display the Monitor Diagnostic Menu.

```
850 PRINT : PRINT "PRESS ESC TO END TEST": GOSUB 8000: GOTO 350
```

When you press a **5** from the Monitor Diagnostic Menu, the program branches to line 860 to begin the Low-Res Color Graphics Test. This test fills every low-resolution character location on the screen with a user-specified color, so that you can check that every memory location is operational, and that the colors are displaying accurately.

```
860 REM LOW RES COLOR GRAPHICS TEST
```

Line 870 clears the screen and positions the cursor with the HOME instruction. The test title is printed on the screen.

```
870 HOME : PRINT TAB( 10)"LOW RES COLOR GRAPHICS TEST": PRINT
```

Lines 872 and 874 display the names of the 16 available low-resolution colors.

```
872 PRINT "VALID COLORS ARE:": PRINT : PRINT "BLACK";
    TAB(13)"MAGENTA"; TAB( 24)"DARK BLUE": PRINT
    "VIOLET"; TAB( 13)"DARK GREEN"; TAB( 24)"GRAY 1"
874 PRINT "MEDIUM BLUE"; TAB( 13)"LIGHT BLUE";
    TAB( 24)"BROWN": PRINT "ORANGE"; TAB( 13)"GRAY 2";
    TAB( 24)"PINK": PRINT "GREEN"; TAB( 13)"YELLOW";
    TAB( 24)"AQUA": PRINT "WHITE"
```

Line 875 feeds an extra line to separate the columns of color names from the subsequent user prompts.

```
875 PRINT
```

Line 876 displays prompts asking you to either press any key to continue the test, or to press ESC to end the test. The program branches to the Get-a-key subroutine at line 8000 to wait for the user response to these prompts. If the response is an ESC key, as indicated by a value of 27 in A$, the Low-Res Color Graphics Test ends and the program branches back to line 350 to display the Monitor Diagnostic Menu.

```
876 PRINT "PRESS ANY KEY TO CONTINUE": PRINT "OR PRESS
    ESC TO END TEST": GOSUB 8000: IF ASC (A$) = 27
    THEN GOTO 350
```

Lines 878 prints a blank line between the prompts and the next input line to be printed at line 880.

```
878 PRINT
```

Line 880 prompts you to enter a color in uppercase letters, exactly as it appears in the valid color list displayed on the screen. The entered color is placed into variable C$.

```
880 INPUT "ENTER COLOR-";C$
```

Lines 890 through 1040 check for the valid colors. Each color has a corresponding numeric value. For example, if you enter BLACK in C$, it is translated

to a value of 0, which is placed into variable C. Variable C is then used to paint the color in line 1070.

```
890 IF C$ = "BLACK" THEN C = 0: GOTO 1060
900 IF C$ = "MAGENTA" THEN C = 1: GOTO 1060
910 IF C$ = "DARK BLUE" THEN C = 2: GOTO 1060
920 IF C$ = "VIOLET" THEN C = 3: GOTO 1060
930 IF C$ = "DARK GREEN" THEN C = 4: GOTO 1060
940 IF C$ = "GRAY 1" THEN C = 5: GOTO 1060
950 IF C$ = "MEDIUM BLUE" THEN C = 6: GOTO 1060
960 IF C$ = "LIGHT BLUE" THEN C = 7: GOTO 1060
970 IF C$ = "BROWN" THEN C = 8: GOTO 1060
980 IF C$ = "ORANGE" THEN C = 9: GOTO 1060
990 IF C$ = "GRAY 2" THEN C = 10: GOTO 1060
1000 IF C$ = "PINK" THEN C = 11: GOTO 1060
1010 IF C$ = "GREEN" THEN C = 12: GOTO 1060
1020 IF C$ = "YELLOW" THEN C = 13: GOTO 1060
1030 IF C$ = "AQUA" THEN C = 14: GOTO 1060
1040 IF C$ = "WHITE" THEN C = 15: GOTO 1060
```

If you enter an invalid color, or misspell a color, the program will proceed through all the IF-THEN statements to line 1050. Line 1050 uses CHR$(7) to sound a beep indicating the error. Then an error message is printed with a prompt for you to try again. The program will return to line 880 so that you can input the name of another color.

```
1050 PRINT : PRINT CHR$ (7): PRINT "INVALID COLOR! - RE-ENTER": PRINT : GOTO 880
```

Line 1060 uses the GR instruction to switch to the low- resolution graphics mode. The POKE and CALL instructions make sure that the four-line text window is not displayed at the bottom of the graphics screen.

```
1060 GR : POKE - 16302,0: CALL - 1998
```

Line 1070 issues a COLOR command, using variable C to set the color according to your previous specification.

```
1070 COLOR= C
```

Line 1080 begins a FOR-NEXT loop which sets the Y coordinate between 0 and 47 for the 48 rows available in the low-resolution graphics mode.

```
1080 FOR Y = 0 TO 47
```

Line 1090 begins a second FOR-NEXT loop which sets the X coordinate between 0 and 39 for the 40 columns available in the low-resolution graphics mode.

```
1090 FOR X = 0 TO 39
```

Line 1100 uses a PLOT statement which paints a box in the specified color in each one of the low-resolution graphic locations. This statement uses the X and Y variables set in the FOR-NEXT loops in lines 1080 and 1090. For example, in the first pass, X is equal to 0, and Y is also equal to 0. This paints a box at location 0,0 on the screen. The color has already been set in line 1070, so the box is painted in the specified color. The NEXT X instruction then increments X to 1, while Y remains at 0. This paints a box at location 1,0, directly to the right of the first box in the row.

```
1100 PLOT X,Y: NEXT X
```

When all 40 locations in the row are filled, the NEXT Y instruction at line 1110 increments Y to 1, and the boxes begin painting at row 1, column 0.

```
1110 NEXT Y
```

Line 1120 branches the program to the Get-a-key subroutine at line 8000 to wait for the user to press a key to end the test. After this happens, the TEXT command switches the monitor from the low-resolution graphics mode back to the text mode. Then the program branches back to line 870 to display the color names and the user prompts again. At this point, the user can make another entry to test another color.

```
1120 GOSUB 8000: TEXT : GOTO 870
```

How the High-Res Color Graphics Test Works

The High-Resolution Color Graphics Test is not included with the System Diagnostic Program, because of memory allocation constraints pertaining to high-resolution graphics. However, it is provided as a separate program on the System Diagnostic Program diskette, and the program listing is found in Fig. 5-17.

```
  1  REM  APPLE ][ SERIES
  5  REM  HIGH-RES COLOR GRAPHICS TEST
  8  REM  COPYRIGHT 1988 TAB BOOKS INC.
 10  HOME : PRINT  TAB( 5)"HIGH RES COLOR GRAPHICS
     TEST": PRINT
 20  PRINT "VALID COLORS ARE:": PRINT
 30  PRINT "BLACK 1"; TAB(9)"GREEN"; TAB(17)
     "VIOLET"
 40  PRINT "WHITE 1"; TAB(9)"BLACK 2"; TAB(17)
     "ORANGE"
 50  PRINT "BLUE"; TAB(9)"WHITE 2"
 60  PRINT : PRINT "PRESS ANY KEY TO CONTINUE":
     PRINT "OR PRESS ESC TO END TEST": PRINT
 70  GET A$: IF  ASC (A$) = 27 THEN END
 80  INPUT "ENTER COLOR-";C$
 90  IF C$ = "BLACK 1" THEN C = 0: GOTO 180
100  IF C$ = "GREEN" THEN C = 1: GOTO 180
110  IF C$ = "VIOLET" THEN C = 2: GOTO 180
120  IF C$ = "WHITE 1" THEN C = 3: GOTO 180
130  IF C$ = "BLACK 2" THEN C = 4: GOTO 180
140  IF C$ = "ORANGE" THEN C = 5: GOTO 180
150  IF C$ = "BLUE" THEN C = 6: GOTO 180
160  IF C$ = "WHITE 2" THEN C = 7: GOTO 180
170  PRINT : PRINT  CHR$ (7);: PRINT "INVALID COLOR!
     - RE-ENTER": GOTO 80
180  HGR : POKE - 16302,0: HCOLOR= C
230  FOR Y = 0 TO 191
240  FOR X = 0 TO 279
250  HPLOT X,Y: NEXT X
260  NEXT Y
270  GET A$: TEXT : GOTO 10
```

Fig. 5-17. High-resolution color graphics test program listing.

When you load the HIRES program from the diskette under Applesoft BASIC, the program begins with the three identifying REMARK statements about the program.

```
1 REM APPLE ][ SERIES
5 REM HIGH-RES COLOR GRAPHICS TEST
8 REM COPYRIGHT 1988 TAB BOOKS INC.
```

Line 10 clears the screen and positions the cursor with the HOME command. Next the test title is displayed on the screen.

```
10 HOME : PRINT TAB( 5)"HIGH RES COLOR GRAPHICS
   TEST": PRINT
```

Lines 20 through 50 print the colors available under the high-resolution graphics mode.

```
20 PRINT "VALID COLORS ARE:": PRINT
30 PRINT "BLACK 1"; TAB(9)"GREEN"; TAB(17) "VIOLET"
40 PRINT "WHITE 1"; TAB(9)"BLACK 2"; TAB(17) "ORANGE"
50 PRINT "BLUE"; TAB(9)"WHITE 2"
```

Line 60 displays prompts asking you to either press any key to continue, or press ESC to end the test. The final PRINT instruction feeds an extra line.

```
60 PRINT : PRINT "PRESS ANY KEY TO CONTINUE":
   PRINT "OR PRESS ESC TO END TEST": PRINT
```

Line 70 gets your input from the keyboard and places it in variable A$. The IF-THEN statement checks whether the value stored in A$ is equal to 27, which is the ASCII value for the ESC key. If it is, the program ends, and the Applesoft BASIC right-bracket prompt is displayed on the screen.

```
70 GET A$: IF ASC (A$) = 27 THEN END
```

Line 80 displays another prompt asking you to enter the color for the test. The color name must be entered in uppercase letters, exactly as shown in the color menu, or it will be returned as invalid. The entered color name is stored in variable C$.

```
80 INPUT "ENTER COLOR-";C$
```

Lines 90 through 160 checks for which color was entered, translating the color name to the appropriate color code. Then the program branches to line 180 to continue with the test.

```
 90 IF C$ = "BLACK 1" THEN C = 0: GOTO 180
100 IF C$ = "GREEN" THEN C = 1: GOTO 180
110 IF C$ = "VIOLET" THEN C = 2: GOTO 180
120 IF C$ = "WHITE 1" THEN C = 3: GOTO 180
```

```
130 IF C$ = "BLACK 2" THEN C = 4: GOTO 180
140 IF C$ = "ORANGE" THEN C = 5: GOTO 180
150 IF C$ = "BLUE" THEN C = 6: GOTO 180
160 IF C$ = "WHITE 2" THEN C = 7: GOTO 180
```

If the value in C$ does not match any of the strings listed in lines 90 through 160, the program falls through to line 170 as an invalid entry. Line 170 feeds another line, and then uses character string 7 to sound a beep indicating to you that there was an error. The error message is printed, and the program branches back to line 80 to let you try entering a color name again.

```
170 PRINT : PRINT CHR$ (7);: PRINT "INVALID COLOR!
    - RE-ENTER": GOTO 80
```

Line 180 uses the HGR command to clear the screen and switch from the text mode to the high-resolution graphics mode. The POKE statement clears the four lines ordinarily set aside for text in the high-res graphics mode. Because all possible high-res graphic locations are to be tested, these text lines must be cleared and set to the high-res mode as well. The HCOLOR instruction then sets the high-resolution color equal to variable C, the value derived in lines 90 to 160 from the color you entered in line 80.

```
180 HGR : POKE - 16302,0: HCOLOR= C
```

Line 230 begins the FOR-NEXT loop which sets the loop for the depth of the screen (there are 192 vertical pixels available in the high-resolution graphics mode).

```
230 FOR Y = 0 TO 191
```

Line 240 begins a second FOR-NEXT loop, setting the width of the screen (there are 280 horizontal pixels available in the high-resolution graphics mode).

```
240 FOR X = 0 TO 279
```

Line 250 draws each individual dot in the specified color with the high-resolution plot, or HPLOT, command. Together with the two FOR-TO loops, a pixel is drawn in each high- resolution location on the screen. This statement uses the X and Y variables set in the FOR-NEXT loops in lines 230 and 240. For example, in the first pass, X is equal to 0, and Y is also equal to 0. This paints a pixel at location 0,0 on the screen. The NEXT X instruction then increments X to 1, while Y

remains at 0. This paints a pixel at location 1,0, directly to the right of the first pixel in the row.

```
250 HPLOT X,Y: NEXT X
```

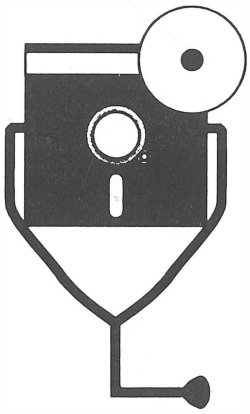
When all 280 horizontal locations are painted, the program proceeds to line 260 for the NEXT statement to the Y loop. Y is incremented to 1, and the pixels begin painting at row 1, column 0.

```
260 NEXT Y
```

When the X and Y loops are completed, the screen will be fully painted at each location with the specified color. The program will wait until you press a key to continue. Line 270 handles this input, storing the key in variable A$. The TEXT instruction switches the screen display from the high-resolution graphics mode back to the text mode.

Finally, the program branches back to line 10, at which point you can either test another color, or press ESC to end the test and return to the Applesoft BASIC environment.

```
270 GET A$: TEXT : GOTO 10
```



6

The Printer

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

DESCRIPTION AND FUNCTION OF THE PRINTER

The printer is ordinarily an optional device to the computer, since a printer is not needed to run programs or process data. But in order to put the results on paper instead of only on the monitor screen, you must have a printer (Fig. 6-1).

Types of Printers

There are many printers available compatible with the Apple II systems. Most Apple II printers use a serial interface connector, although some printer controllers include an on-board parallel interface.

The two basic types of Apple II printer choices consist of the dot-matrix printer and the letter-quality printer. Laser printers are also available for the Apple II, but they are not covered in this book.

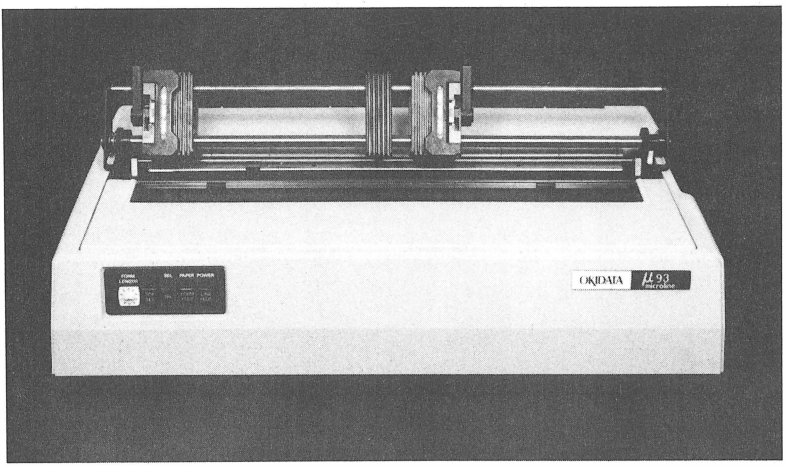


Fig. 6-1. The printer.

Dot Matrix Printers. If a character on the monitor screen were to be magnified, you would see that instead of being made up of solid lines, the character is in fact composed of a series of dots created with light. 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 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, these print head wires punch out the shape of the character through the ink ribbon to create its impression on the paper (Fig. 6-2).

The dot-matrix printer provides very quick output, and is the least expensive printer type available. Speed is measured in characters per second (cps), referring to how fast the printer can place characters on paper. A dot-matrix printer can range in speed from 30 to 200+ cps.

The wide range of possible speeds is attributed to several factors, including the speed at which the computer can feed data to the printer; whether the printer interface is a serial or parallel connection; whether there is a printer buffer; the number of dots used to create a character; and whether a printer is *unidirectional* or *bidirectional.*

Some printers print only left to right, and are called unidirectional printers. Others can print both left to right and right to left, and are called *bidirectional* printers. They put characters on a page much faster, since no time is wasted in sending the print head back to the beginning of a new line.

Many dot matrix printers offer several qualities of print, including 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

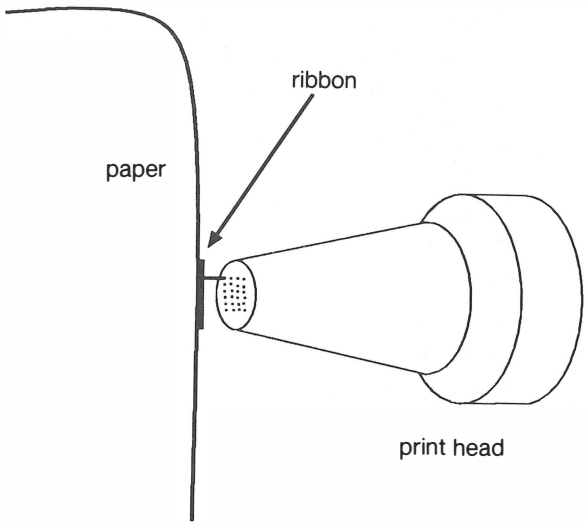


Fig. 6-2. Dot-matrix print head.

modes use multiple print passes and additional 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 syles and point sizes.

Certain dot-matrix printers, called graphic printers, can render graphic characters. The Apple ImageWriter is an example of such a graphic printer. A graphic printer can address just as many ink points (which correspond to the monitor's pixels) as your monitor resolution can display. It can reproduce graphic characters, as well as drawings and business graphics. Any shape can be created with a series of dots. This makes the dot-matrix printer highly flexible as to the output that can be created.

There are also dot-matrix printers that can print in different colors. The major difference lies in the printer using multicolored ribbon. The ImageWriter, Panasonic, and several other popular printers have multicolored printer ribbons available for their printers.

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

The letter-quality printer is often preferred by individuals who use their computers to do mass mailings as part of their business. Letters printed on a letter-quality printer appear as though they were individually typed on a typewriter, rather than mass-produced with a computer. In fact, any document that a company or individual wants to look particularly handsome is best done with a

letter-quality printer. Letter-quality output simply looks cleaner, with sharper characters than dot-matrix output.

The letter-quality printer uses a daisy wheel to create the characters. On the end of each *petal* of the daisy wheel is a character (Fig. 6-3).

The daisy wheel includes the entire character set for a given font family and point size. Instead of being formed by a single print head with a series of dots, these characters are individually and solidly embossed onto the daisy wheel. A printer motor spins the daisy wheel around until the desired character spins into position, at which point the print hammer strikes the character against the ink ribbon to form the character on paper. It works on a principle similar to the typewriter, although much faster.

Although the daisy wheel is the most common print element used in letter-quality printers, character *thimbles* and *cups* are used by some printers. They work in the same general manner as the daisy wheel. No matter which element is used with a letter-quality printer, they all provide the capability of printing different font families at a higher quality.

While dot-matrix printers can run at speeds of 30 to 200+ cps, letter-quality printers run at much slower speeds of 10 to 25 + cps. This is because the print head mechanism must position the wheel or thimble at the correct character and then energize a solenoid — a much longer process than staying in one location to form characters out of a series of dots.

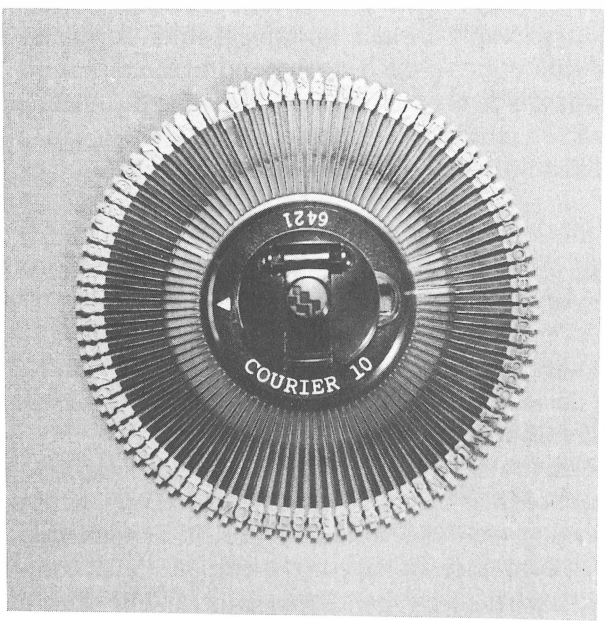


Fig. 6-3. Daisy wheel.

Printer Interfaces

The printer is physically connected, or *interfaced,* to the Apple II system through an optional serial interface board. A parallel interface board is also available. The interface board is plugged into one of the expansion slots on the motherboard, and includes a receptacle that is exposed at the rear of the system unit to accept the printer cable (Fig. 6-4).

A serial interface sends data one bit at a time, in a *series* (Fig. 6-5). This is contrasted with a parallel interface, which accepts 7 or 8 bits of data sent simultaneously by a parallel printer.

PRINTER PREVENTIVE MAINTENANCE

Protect the printer with a dust cover whenever it is not being used.

Once a month, use a hand-held vacuum cleaner or compressed air to vacuum or blow around the printer mechanism. This removes any dust and dirt that

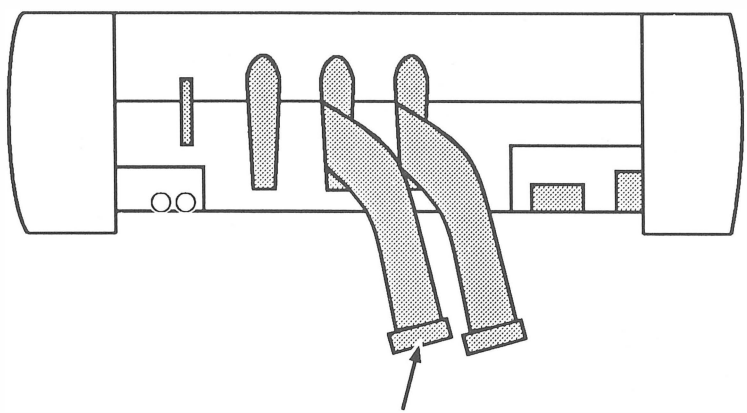


Fig. 6-4. Serial printer interface.

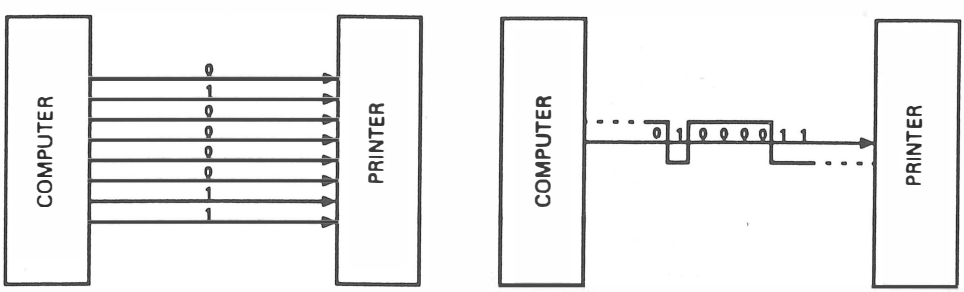


Fig. 6-5. Parallel and serial data transfer.

might have accumulated. It also removes the paper and ink particles created during printing. Make sure the printer power is off before cleaning.

About once a month, moisten a piece of lint-free paper with denatured isopropyl alcohol and clean the daisy wheel character petals on a letter-quality printer, or the print head on a dot-matrix printer. This gently cleans off the dried ink and paper bits that can accumulate.

Also, use lint-free paper moistened with denatured isopropyl alcohol to wipe down the printer *platen,* the cylinder around which the paper rolls. Rotate the platen with the platen control knob to clean all sides.

Wipe down the printer case with lint-free paper and denatured isopropyl alcohol to complete your printer preventive maintenance.

PRINTER TROUBLESHOOTING AND REPAIR GUIDELINES

The following sections describe potential printer problems, how to diagnose them, and the steps necessary to repair them.

Problem: The Printer Does Not Work

There are a number of different problem situations which could cause your printer not to work.

Solutions:

- First check that the printer is on. 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 outlet is controlled by a light switch or a switch on the power strip.
- 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. It might also have a LOCAL mode which allows physical manipulation of the printer: rolling the paper, preparing the printer for a different form, or adjusting the platen. To get the printer operating again, switch it back to the ONLINE mode.
- Check the cabling. Make sure the right kind of cable is being used for the printer, and that it is inserted into the proper receptacles, both at the printer and at the computer. Ensure that the connectors are sufficiently secured and tightened down between the printer and the printer interface port on the system unit.

- 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 the application in use.

  The driver is activated by a printer installation program provided with your application software. This special utility program prompts you to enter the make of printer 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 a wide range of popular 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, which would in turn cause the printer not to work.

- Certain option switch settings within the printer might make it impossible to print. These switch settings control parameters such as baud rate and word length. Refer to the printer documentation for more information about these internal switch settings. For more information about terms such as baud rate and word length, refer to Chapter 8, "The Serial Communication Interface."

Problem: Poor Print Quality

Another printer problem can result in poor print image quality. Though the characters are recognizable, they look blurred, light, or washed out. A related problem can occur when printing on a multiple copy form. In this instance, 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 program 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. Manually advance the ribbon with the advance knob on the cartridge, or install a new ribbon cartridge.
- Some printers have a forms adjustment, which enables the print head to be brought further back from the carriage to accommodate papers of heavier thicknesses, or multi-part forms. Use the forms adjustment to move the head closer to the carriage. Run the print sample and adjust the head until the print image is clear and sharp.

- 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 line across the paper, starting at position 1.

Solutions:

- If the tabs do not print in this diagonal line, check whether something is obstructing the platen or the print head. Often simply blowing through the print head, or pulling out a bit of paper hung up on it will solve the problem. Use compressed air or a hand-held vacuum cleaner to help unclog tight areas in the printer.
- If these solutions fail to take care of 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, similar to carriage returns on a typewriter. When working with special forms like checks and invoices, the application program sets up the line spacing to occur in a specific pattern that is more complex than single- or double-spacing. Also, you might enter a form feed command at a 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. The specified number of lines between text might not feed, or the paper might bunch up and jam as it's rolling to the next page.

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 it feeds the paper up to the next page.
- If this does not take place as expected, check that the paper has been correctly inserted into the printer.

- If 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 loosens the platen when rolling the paper in. Then 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 could have lodged into the platen or into the paper feed area. If you cannot reach it with your fingers, remove it with tweezers, compressed air, or a hand-held vacuum cleaner.
- 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 in between the printed lines, there may be a faulty option switch setting that's causing the extra line feeds. Check the printer documentation for more information about switch settings.
- If none of these suggestions correct the line spacing problem, refer to the printer documentation or call the manufacturer for 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 the 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.

Solutions:

- If you're working with a dot-matrix printer, run the Display Character Print or Echo Character Test of the Printer Diagnostic. If there is a 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, it cannot print graphics characters.

  It may be confusing to see the graphics character on the monitor display, but not get it to print. This is because different makes of printers support different character sets. Some character sets are more limited than others. Be sure to send only valid characters to your printer.

- In the case of a letter-quality printer, the daisy wheel is limited in size and can usually only accommodate about 100 upper- and lowercase characters. Because of this, it is likely that the daisy wheel does not include any special

graphics characters. (However, special graphics daisy wheels are available which include such characters as part of their character sets. Consult the printer manufacturer or a printer retail outlet.)

- If your letter-quality printer is having trouble with regular alphanumeric characters, the problem is entirely different.

  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 lowercase "a" are missing. If this is true, the physical character position on the daisy wheel or cup has broken off its stem. Daisy wheels and cups are fragile, and characters can break off. Remove the daisy wheel and examine it to confirm that it is indeed damaged. Buy a new daisy wheel to replace the broken one. It's a good practice to keep at least one backup daisy wheel on hand.

Problem: Certain Characters Do Not Print Completely

A particular dot-matrix printer problem is that while all the characters 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", "r", or "m", might be displayed, and the same segment of the character is missing in each of the characters, for example, upper left, or lower right.

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 obtain sample output.
- If the ribbon is in good condition, then there is a problem in the print head. One of the solenoids that actuates the dot matrix pins in the head has probably malfunctioned or worn out. Take the printer in for servicing, and ask to have the faulty solenoid repaired or replaced.

Problem: Installing a New Printer

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

For instance, you might have been printing to a dot-matrix printer with no problems. The day may come when you decide to obtain a letter-quality printer. You disconnect the old printer and connect the new printer to the Apple, and think you're finished. But upon running an application and attempting to print

the results, you discover that either the printer does not print at all, or that there are incorrect results.

The reason this happens is that as far as the Apple II system is concerned, the new printer is a completely foreign device. It is not addressable, therefore, the computer and new printer do not know how to effectively communicate with one another.

Particular rules, or *protocols* must be followed in order for the various system components to communicate with one another and send data back and forth. The protocols that the original printer understood may be quite different from the protocols that the new printer requires.

Many software applications include a program called a *device driver* that is established for a particular printer. A device driver is a program that recognizes a peripheral device such as a letter-quality printer. Many applications, especially word processors, need to know the 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 you connect your printer to your Apple II, ensure that any software program that uses the printer is set up to the proper interface. This information should be provided by the user documentation: the computer system, the printer, and the software documentation are all vital for specific configuration.

You may encounter other problems when setting up a new printer. Again, read the printer documentation thoroughly before installing. The software user guides can also be a good resource. Consult the printer manufacturer for further information.

Problem: Printer Needs to Be Serviced

If the solutions described above fail to solve your printer problem, write down every symptom of the problem. Also indicate what you have already checked and the results. Then call your Apple II service center and talk to a technician. Describe the type of printer and computer you have, and the problem it is experiencing. Explain what you have already tried to do to find and fix the problem, and indicate the results, based on your notes. Find out if there are any solutions they can suggest over the phone.

If the technician cannot suggest any further solutions over the phone, ask whether you should bring in just the printer, or include other parts of the computer system as well.

Find out if the technician has any ideas yet on what the problem might be, and how long and how much it might cost to fix. Be sure to get a copy of the work order.

Before you pay for and accept your printer, have the repair technician demonstrate that the problem has actually been fixed. Then pack the printer up and take it home. Connect everything together, and refer to Chapter 9, "The Exerciser," for instructions on burning in the system after repair. The Exerciser includes several tests for the printer which can be run continuously.

RUNNING THE PRINTER DIAGNOSTIC MODULE

To run the Printer Diagnostic module, first turn the printer on, then follow the instructions provided in "Running the Diagnostic Program" in Chapter 2 to load and run the program. Press P to initiate the Printer Diagnostic module. The screen will clear and the Printer Setup routine will be displayed.

Printer Setup

There are two basic types of printers between which the Printer Diagnostic module must distinguish. Because of this, the Printer Setup routine is run before the Printer Diagnostic menu is even displayed. Two prompts are given (Fig. 6-6).

The first prompt asks whether the printer can print up to 80 or 132 characters in one line. The default value is 80 characters. The second prompt asks for the slot

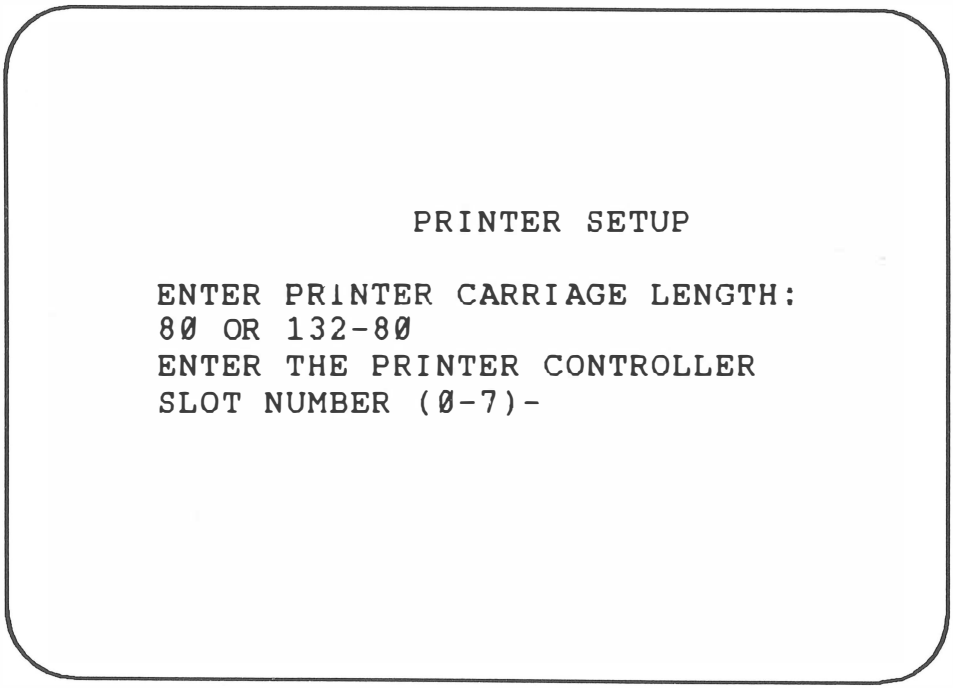


Fig. 6-6 Printer setup routine.

number in which the printer controller board resides. This information allows the program to properly address the board while running the printer tests. Enter the slot number at the prompt, and press **RETURN**. If you don't know the slot number, open the system unit, and locate the printer controller board. The slot number is labeled on the motherboard in front of the slot itself.

After answering these prompts and pressing **RETURN**, the module is set up for your printer, and the Printer Diagnostic Menu is displayed as shown in Fig. 6-7. To activate one of the printer tests, press the corresponding number, 1 through 5.

Sliding Alpha Test

The Sliding Alpha Test is very similar to the sliding alpha routine used in the Scroll Test of the Monitor Diagnostic module. It serves two functions: to fully exercise the printer, and to get a sample of the full alphanumeric display character set available on the printer.

When you press 1, you are prompted to enter the number of lines you wish to run. If you enter a number less than 35, the program defaults to running 35 scrolling lines. The printer begins printing a sliding alpha pattern. It types a line full of as many different characters of the character set as will fit on a line. It then

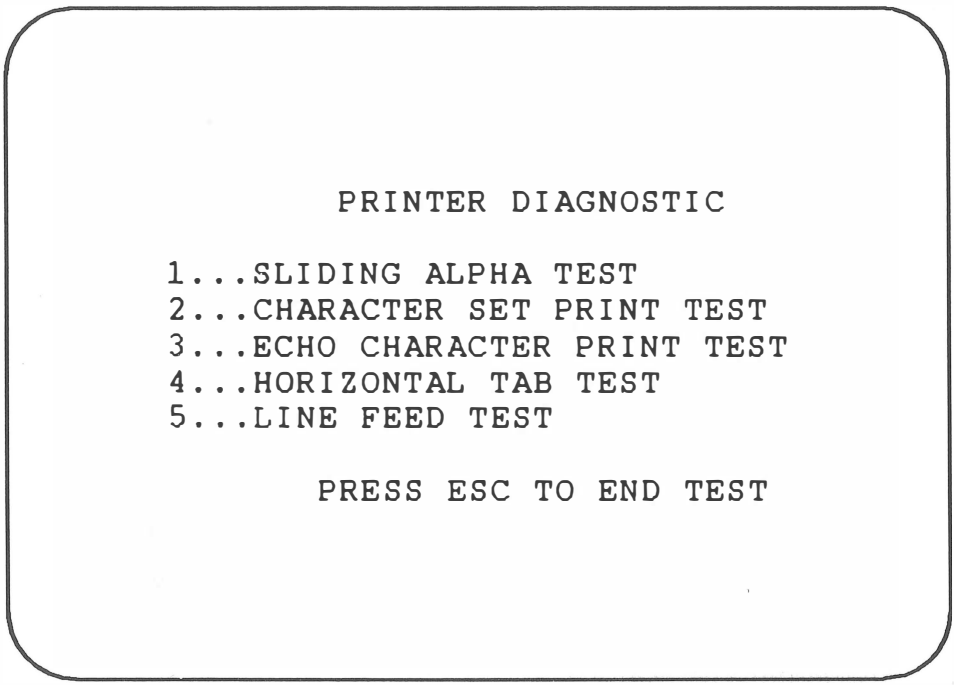


Fig. 6-7. The printer diagnostic menu.

feeds up another line, slides one character to the left, and prints the line of characters. The program continues to do this for the specified number of lines (Fig. 6-8).

Character Set Print Test

The Character Set Print Test displays all characters available within the printer's character set, and checks whether all the characters are printing satisfactorily. To run the Character Set Print Test, press 2. All characters in the character set will be printed (Fig. 6-9).

After printing the character set, the Printer Diagnostic Menu is displayed once again.

Echo Character Print Test

The Echo Character Print Test prints, or echos, an entered character across all 80 or 132 columns of the printer. This test ensures that a certain character is

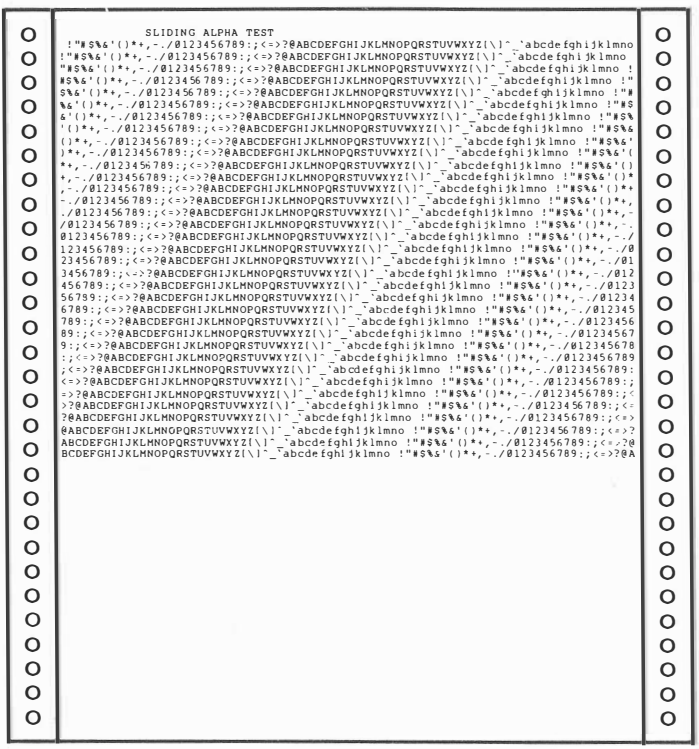


Fig. 6-8. Sliding alpha test results.

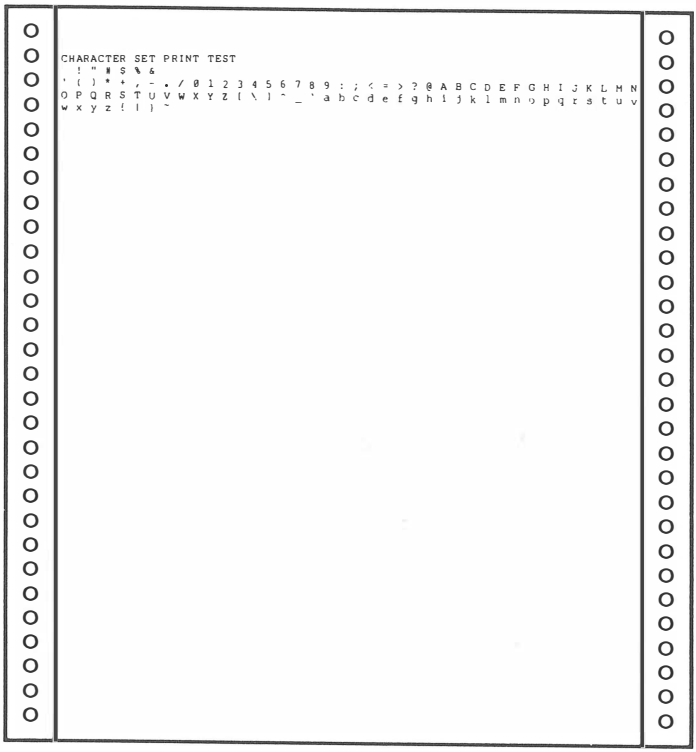


Fig. 6-9. Character set print test results.

printing, and if it is, it lets you check its appearance, making sure it's a complete character, and that it's not broken or faded. This test also lets you check whether all columns along the printer carriage are accepting characters.

Press 3 to run the Echo Character Print Test. You will be prompted to enter the character to be echoed. Function keys and cursor keys are disabled for this test. Only valid character keys can be echoed. The printer prints this character at all positions along the carriage, columns 1 through 80 (Fig. 6-10). When it's finished, press any other character to have that character echoed.

After you press the ESC key, the Printer Diagnostic Menu will be displayed once again.

Horizontal Tab Test

The Horizontal Tab Test checks all combinations of tabbing across the printer carriage.

Press 4 to run the Horizontal Tab Test. 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,

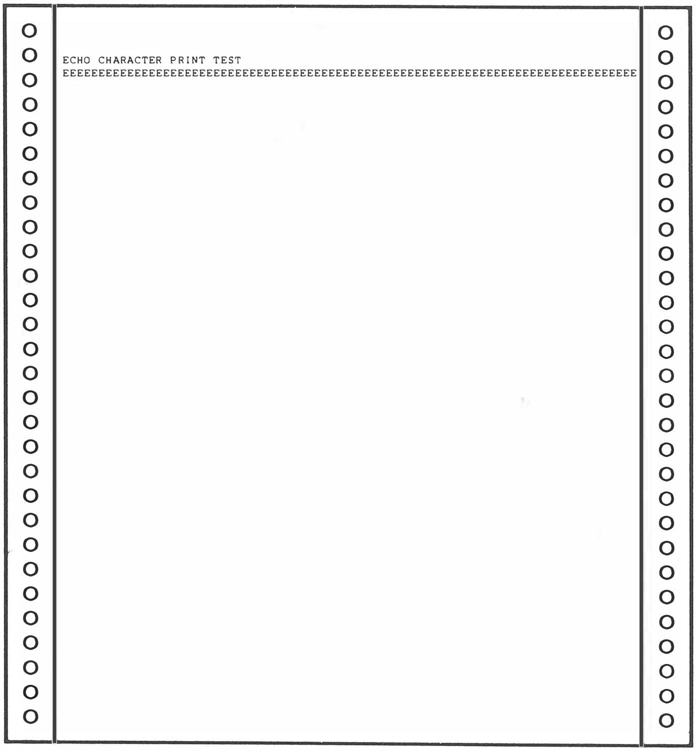


Fig. 6-10. Echo character print test results.

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. 6-11).

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

Line Feed Test

Line feed problems are manifested by text printing on top of previous lines because the printer failed to feed a line at the proper time. 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

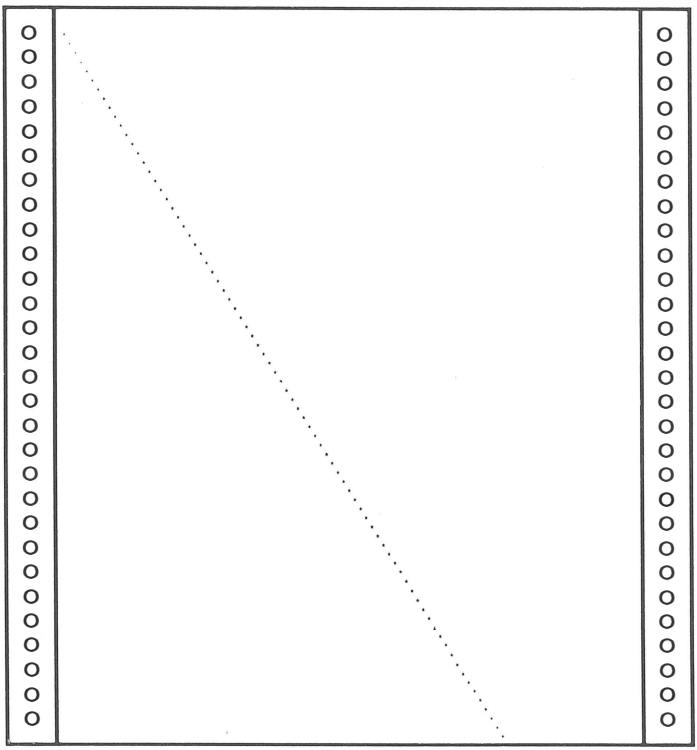


Fig. 6-11. Horizontal tab test results.

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, then types a fifth line. Finally a form feed command is issued, and the printer rolls up to the next page and prints a line (Fig. 6-12).

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 from the System Diagnostic Main Menu causes the program to branch to line 1250, which is the beginning of the printer setup routine to prepare for the Printer Diagnostic module (Fig. 6-13). The Printer Diagnostic module consists of five tests for printer problems.

Line 1130 begins the Printer Diagnostic module with a REMARK statement.

```
1130 REM PRINTER DIAGNOSTIC MODULE
```

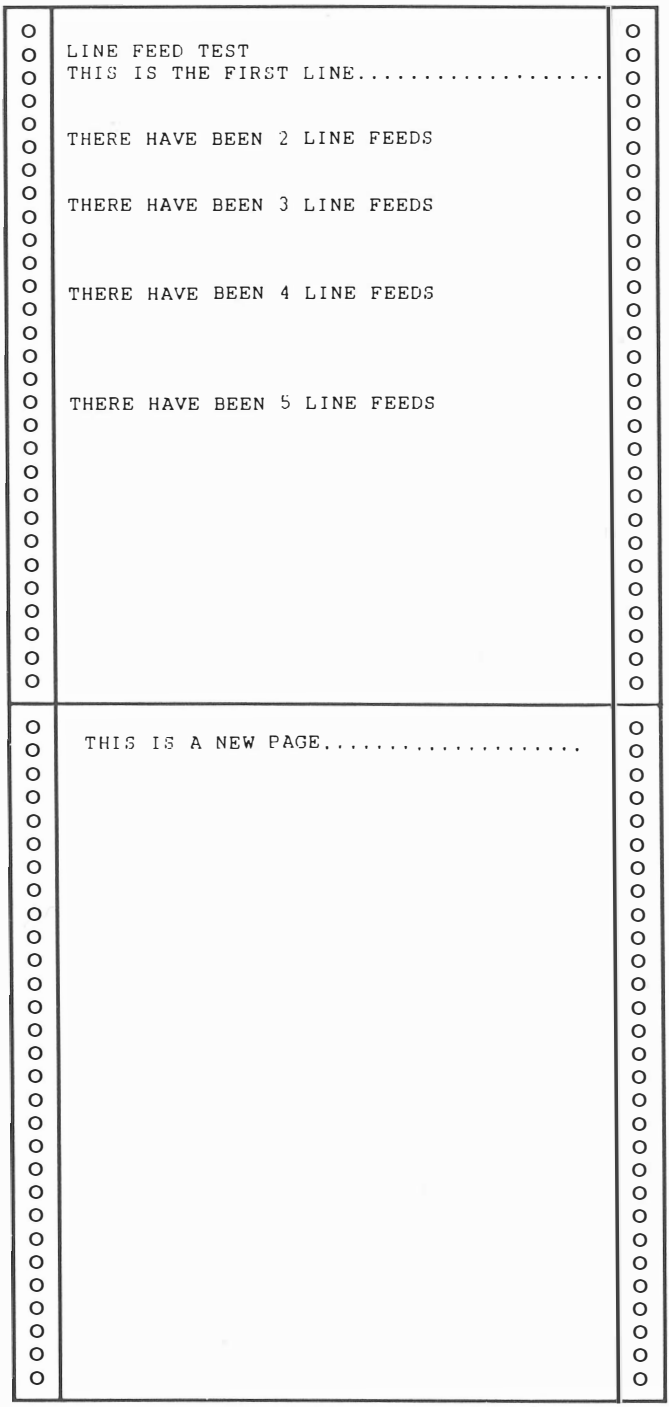


Fig. 6-12. Line feed test results.

```
1130  REM  PRINTER DIAGNOSTIC MODULE
1140  HOME : PRINT  TAB( 11)"PRINTER DIAGNOSTIC":
      PRINT : PRINT  TAB( 3)"1...SLIDING ALPHA TEST"
1150  PRINT  TAB( 3)"2...CHARACTER SET PRINT TEST":
      PRINT  TAB( 3)"3...ECHO CHARACTER PRINT TEST"
1160  PRINT  TAB( 3)"4...HORIZONTAL TAB TEST":
      PRINT  TAB( 3)"5...LINE FEED TEST": PRINT
1170  PRINT  TAB( 10)"PRESS ESC TO END TEST"
1180  GOSUB 8000: IF  ASC (A$) = 27 THEN 50
1190  IF A$ = "1" THEN  GOTO 1280
1200  IF A$ = "2" THEN GOTO 1430
1210  IF A$ = "3" THEN  GOTO 1520
1230  IF A$ = "4" THEN  GOTO 1610
1240  IF A$ = "5" THEN  GOTO 1680
1245  PRINT CHR$ (7): GOTO 1180
1250  HOME : PRINT  TAB( 13)"PRINTER SETUP"
1260  PRINT : PRINT "ENTER PRINTER CARRIAGE LENGTH:":
      INPUT "80 OR 132-";W
1265  IF W <  > 132 THEN W = 80
1270  PRINT "ENTER THE PRINTER CONTROLLER":
      INPUT "SLOT NUMBER (0-7)-";SL
1275  GOTO 1140
1280  REM  SLIDING ALPHA TEST
1290  HOME : PRINT  TAB( 11)"SLIDING ALPHA TEST": PRINT
1300  INPUT "ENTER THE NUMBER OF REPETITIONS-";X
1310  IF X < 35 THEN X = 35
1320  PR# SL
1325  PRINT "SLIDING ALPHA TEST"
1330  N = 31
1340  FOR I = 1 TO X
1350  L$ = "":N = N + 1: IF N > 111 THEN N = 32
1360  FOR A = 1 TO W
1370  L$ = L$ +  CHR$ (N):N = N + 1
1380  IF N > 111 THEN N = 32
1390  NEXT A
1400  PRINT L$: NEXT I
1410  IF EX = 1 THEN  PR# 0: GOTO 1440
1420  PR# 0: GOTO 1130
1430  REM  CHARACTER SET PRINT TEST
1440  HOME : PRINT  TAB( 6)"CHARACTER SET PRINT TEST":
      L$ = ""
1445  PR# SL: REM  ACTIVATE PRINTER
1447  PRINT "CHARACTER SET PRINT TEST"
1450  FOR I = 32 TO 128
1460  IF N = W THEN  PRINT L$:L$ = "":N = 1
1470  L$ = L$ +  CHR$ (I) + " ":N = N + 2
1480  NEXT I
```

Fig. 6-13. Printer diagnostic module listing.

```
1490  IF  LEN (L$) > 0 THEN  PRINT L$
1500  IF EX = 1 THEN  PR# 0: GOTO 1555
1515  PR# 0: GOTO 1130
1520  REM  ECHO CHARACTER PRINT TEST
1530  HOME : PRINT TAB( 8)"ECHO CHARACTER PRINT TEST"
1540  PRINT : PRINT  TAB( 8)"ENTER CHARACTER TO ECHO":
      PRINT : PRINT  TAB( 8)"PRESS ESC TO END TEST"
1550  GOSUB 8000: IF  ASC (A$) = 27 THEN  GOTO 1130
1555  L$ = ""
1560  FOR I = 1 TO W
1570  L$ = L$ + A$: NEXT I
1575  PR# SL: PRINT "ECHO CHARACTER PRINT TEST"
1580  PRINT L$: PR# 0
1590  IF EX = 1 THEN  GOTO 1620
1600  GOTO 1530
1610  REM  HORIZONTAL TAB TEST
1620  HOME : PRINT  TAB( 10)"HORIZONTAL TAB TEST"
1625  PR# SL
1627  PRINT "HORIZONTAL TAB TEST"
1630  FOR I = 1 TO W
1640  PRINT  TAB( I)"*": NEXT I
1650  PR# 0
1660  IF EX = 1 THEN  GOTO 1690
1670  GOTO 1130
1680  REM  LINE FEED TEST
1690  HOME : PRINT  TAB( 13)"LINE FEED TEST"
1695  PR# SL: PRINT "LINE FEED TEST"
1700  PRINT "THIS IS THE FIRST LINE....................."
1710  FOR I = 2 TO 5
1720  FOR A = 1 TO I
1730  PRINT CHR$ (10);: NEXT A
1740  N$ = STR$ (I)
1750  PRINT "THERE HAVE BEEN ";N$;" LINE FEEDS";:
      NEXT I
1760  PRINT  CHR$ (12);: PRINT "THIS IS A NEW PAGE....
      ................."
1770  PR# 0
1780  IF EX = 1 THEN  RETURN
1790  GOTO 1130
```

Fig. 6-13. (continued)

Line 1140 uses the HOME statement to clear the screen and position the cursor. Then, together with lines 1150 and 1160, the Printer Diagnostic Menu is displayed on the screen.

```
1140 HOME : PRINT TAB( 11)"PRINTER DIAGNOSTIC":
     PRINT : PRINT TAB( 3)"1...SLIDING ALPHA TEST"
1150 PRINT TAB( 3)"2...CHARACTER SET PRINT TEST":
     PRINT TAB( 3)"3...ECHO CHARACTER PRINT TEST"
1160 PRINT TAB( 3)"4...HORIZONTAL TAB TEST":
     PRINT TAB( 3)"5...LINE FEED TEST": PRINT
```

Line 1170 displays the user prompt to press ESC when ready to end the test.

```
1170 PRINT TAB( 10)"PRESS ESC TO END TEST"
```

Line 1180 branches to the Get-a-key subroutine at line 8000 to wait for you to make a choice from the Printer Diagnostic menu. If you press the ESC key, a value of 27 is placed into variable A$, indicating that you wish to end the Printer Diagnostic. When this happens, the program branches to line 50, where the System Diagnostic Main Menu will be displayed.

```
1180 GOSUB 8000: IF ASC (A$) = 27 THEN 50
```

Lines 1190 through 1240 check for other entries you might have made. Valid entries are 1 through 5, corresponding to the printer tests from which to choose.

```
1190 IF A$ = "1" THEN GOTO 1280
1200 IF A$ = "2" THEN GOTO 1430
1210 IF A$ = "3" THEN GOTO 1520
1230 IF A$ = "4" THEN GOTO 1610
1240 IF A$ = "5" THEN GOTO 1680
```

If the entry is not the ESC key, and is not 1 through 5, then you have made an invalid selection. In this case, the program falls through to line 1245. Printing CHR$(7) causes a beep to sound to indicate the error. The program then branches to line 1180 to let you try again.

```
1245 PRINT CHR$ (7): GOTO 1180
```

When you select P from the System Diagnostic Main Menu, the program branches to line 1250, where the Printer Setup routine is located. You must set up the printer parameters before the Printer Diagnostic Menu is even displayed.

```
1260 HOME : PRINT TAB( 13)"PRINTER SETUP"
```

Line 1260 prompts you to input the length of the printer carriage, either 80 or 132 columns. The answer is stored in variable W.

```
1260 PRINT : PRINT "ENTER PRINTER CARRIAGE LENGTH:"
     INPUT "80 OR 132-";W
```

Line 1265 checks whether or not W is greater than or less than 132. If your entry is any value but 132, W is automatically set to 80. This simply means that an 80-column printer is the program default.

```
1265 IF W <> 132 THEN W = 80
```

Line 1270 prompts you to enter the number of the slot in which the printer controller board resides. This way the program knows where to find and address the printer during the program. Your input is stored in variable SL.

```
1270 PRINT : INPUT "ENTER THE PRINTER CONTROLLER":
     INPUT "SLOT NUMBER (0-7)-";SL
```

Line 1275 sends the program back to line 1140, which displays the Printer Diagnostic Menu to let you begin actual testing.

```
1275 GOTO 1140
```

Line 1280 is the beginning of the Sliding Alpha Test, as indicated with the REMARK statement. The Sliding Alpha Test prints a sliding alpha line down a page for at least 35 repetitions. This lets you get a good-sized sample printout to check print quality and printer operation.

```
1280 REM SLIDING ALPHA TEST
```

Line 1290 uses the HOME instruction to clear the screen and position the cursor. The test title will then be displayed on the screen.

```
1290 HOME : PRINT TAB( 11)"SLIDING ALPHA TEST": PRINT
```

Line 1300 prompts you to enter the number of repetitions for the sliding alpha line. The response is stored in variable X.

```
1300 INPUT "ENTER THE NUMBER OF REPETITIONS-";X
```

Line 1310 checks if X is less than 35. If it is, X is automatically set equal to 35.

```
1310 IF X < 35 THEN X = 35
```

Line 1320 switches output from the screen to the printer. It does this with the PR# instruction, using variable SL to indicate where the printer controller resides. Subsequent output is directed to the printer. Incidentally, output still is displayed on the screen as well.

```
1320 PR# SL
```

Line 1325 prints the test title on the printer.

```
1325 PRINT "SLIDING ALPHA TEST"
```

Line 1330 sets variable N equal to 31. This is done so that when 1 is added to N in line 1350, the result is CHR$(32), which is the first display character, a blank.

```
1330 N = 31
```

Line 1340 begins a FOR-NEXT loop, using the number of repetitions specified in line 1300 as its upper limit.

```
1340 FOR I = 1 TO X
```

Line 1350 begins building the line for the sliding alpha. Variable L$ is set equal to null. Then N is set equal to N plus 1 for the I loop. The final instruction on this line indicates that if N is greater than 111, N is to be reset for the I loop to 32. The value of 111 is 31 (the value of N at line 1330) plus 80 (the minimum number of printer columns).

```
1350 L$ = "":N = N + 1: IF N > 111 THEN N = 32
```

Line 1360 begins a second FOR-NEXT loop, cycling A for the length of the carriage as indicated in variable W. This builds the character string in a single line across the printer column width.

```
1360 FOR A = 1 TO W
```

Line 1370 sets variable L$, the sliding alpha string, equal to L$ plus CHR$(N). This builds the string with incrementing characters. N is set equal to N + 1 to increment the character for the A loop.

```
1370 L$ = L$ + CHR$ (N):N = N + 1
```

Line 1380 checks whether N is greater than 111. If it is, N is reset to 32 for the A loop.

```
1380 IF N > 111 THEN N = 32
```

Line 1390 is the NEXT statement for the A loop. This loop continues until a full line is built for the length of the printer carriage.

```
1390 NEXT A
```

When the full line is built in L$, line 1400 prints it. Once this is done, the NEXT statement increments the I loop to start the next line.

```
1400 PRINT L$: NEXT I
```

Line 1410 is the Exerciser flag. If EX is equal to 1, then the Exerciser has control of the program. If this is the case, PR# 0 switches output control from the printer back to the screen. Then the program branches to line 1440 for the next printer Exerciser test.

```
1410 IF EX = 1 THEN PR# 0: GOTO 1440
```

If EX is not equal to 1, the program proceeds to line 1420. This line also uses the PR# 0 instruction to reset output control from the printer back to the screen. The program then branches to line 1130 to display the Printer Diagnostic Menu, now that the Sliding Alpha Test is complete.

```
1420 PR# 0: GOTO 1130
```

Line 1430 begins the Character Set Print Test, as indicated by the REMARK statement. The Character Set Print Test prints all available characters existing in the printer's character set. This allows you to check whether all available characters are printing properly.

```
1430 REM CHARACTER SET PRINT TEST
```

Line 1440 clears the screen, positions the cursor, and displays the test title on the screen. It also sets variable L$, the sliding alpha line built in the Sliding Alpha Test, equal to null.

```
1440 HOME : PRINT TAB( 6)"CHARACTER SET PRINT TEST":
     L$ = ""
```

Line 1445 switches output control from the screen to the printer.

```
1445 PR# SL: REM ACTIVATE PRINTER
```

Line 1447 prints the test title on the printer.

```
1447 PRINT "CHARACTER SET PRINT TEST"
```

Line 1450 begins the FOR-NEXT loop which builds a string of the printer's character set.

```
1450 FOR I = 32 TO 128
```

Line 1460 prints the character set line as soon as N is equal to the carriage length. Then the line is reset to null, and N is set back to 1 to begin the next line.

```
1460 IF N = W THEN PRINT L$:L$ = "":N = 1
```

If N is not equal to W, the program proceeds to line 1470. These statements build the line with succeeding characters in the printer's character set, each one separated with a blank. Variable N is then set equal to N + 2.

```
1470 L$ = L$ + CHR$ (I) + " ":N = N + 2
```

Line 1480 is the NEXT instruction which increments the I loop.

```
1480 NEXT I
```

Line 1490 checks whether the length of L$ is greater than 0, now that the FOR-NEXT loop is completed. If it is, the line is printed. This is done because even though the loop is completed, it's possible that a partial line might be built in L$, and it needs to be flushed out before proceeding further.

```
1490 IF LEN (L$) > 0 THEN PRINT L$
```

Line 1500 checks the Exerciser flag. If it is equal to 1, the PR# 0 instruction returns output control to the screen. Then the program branches to line 1555 for the next printer Exerciser test.

```
1500 IF EX = 1 THEN PR# 0: GOTO 1555
```

If the Exerciser is not in effect, the program proceeds to line 1515. PR# 0 resets output control from the printer to the display. Then the program branches back to display the Printer Diagnostic Menu.

```
1515 PR# 0: GOTO 1130
```

Line 1520 is the REMARK statement beginning the Echo Character Print Test. This test takes a character input from the user and prints it across one row on the printer.

```
1520 REM ECHO CHARACTER PRINT TEST
```

Line 1530 clears the screen, positions the cursor, and displays the test title on the screen.

```
1530 HOME : PRINT TAB( 8)"ECHO CHARACTER PRINT TEST"
```

Line 1540 displays the user prompts for the test, telling you to press any character for the Echo Character Print Test, and to press ESC to end the test.

```
1540 PRINT : PRINT TAB( 8)"ENTER CHARACTER TO ECHO":
     PRINT : PRINT TAB( 8)"PRESS ESC TO END TEST"
```

Line 1550 branches the program to the Get-a-key subroutine at line 8000. If you press ESC, a value of 27 is placed in variable A$. In this case, the program branches back to line 1130 to end the Echo Character Print Test and display the Printer Diagnostic Menu.

```
1550 GOSUB 8000: IF ASC (A$) = 27 THEN GOTO 1130
```

Line 1555 then sets L$, the line string, to a null again. Because L$ is used several times throughout this diagnostic to build a line to be printed, it needs to be cleared before being used again.

```
1555 L$ = ""
```

Line 1560 begins a FOR-NEXT loop which will serve to build the line containing the entered character to be echoed.

```
1560 FOR I = 1 TO W
```

Line 1570 sets L$ equal to L$ plus A$, which is the character you have entered. The NEXT statement loops it back through again to add another value of A$ onto L$. This loops either 80 or 132 times to fill up all the printer columns in one line.

```
1570 L$ = L$ + A$: NEXT I
```

Line 1575 activates the printer with the PR# SL command, then outputs the test title to the printer.

```
1575 PR# SL: PRINT "ECHO CHARACTER PRINT TEST"
```

Line 1580 prints the line that was built in L$. Once this is done, output control is switched from the printer back to the screen with PR# 0.

```
1580 PRINT L$: PR# 0
```

Line 1590 checks the Exerciser flag. If it is equal to 1, the program branches to line 1620 to continue with the next printer Exerciser test.

```
1590 IF EX = 1 THEN GOTO 1620
```

If the EX flag is not equal to 1, line 1600 branches the program back to line 1530 to wait for you to press another character to be echoed, or ESC to end the Echo Character Print Test.

```
1600 GOTO 1530
```

Line 1610 begins the Horizontal Tab Test with a REMARK statement. The Horizontal Tab Test checks every tab location across the printer carriage width.

```
1610 REM HORIZONTAL TAB TEST
```

Line 1620 clears the screen and positions the cursor with the HOME instruction, then prints the test title.

```
1620 HOME : PRINT TAB( 10)"HORIZONTAL TAB TEST"
```

Line 1625 redirects output to the printer with the PR# instruction.

```
1625 PR# SL
```

Line 1627 outputs the test title to the printer as a heading.

```
1627 PRINT "HORIZONTAL TAB TEST"
```

Line 1630 begins the FOR-NEXT loop that will tab and print a character at every column across the carriage width.

```
1630 FOR I = 1 TO W
```

Line 1640 uses the value of I as the increment to tab, then prints an asterisk at that location. Variable I, as determined in the FOR-NEXT loop, will be a value from 1 to 80 or 1 to 132, depending on the value of W. After the asterisk is printed at the appropriate tab location, there is a line feed and then the NEXT instruction increments the loop. This creates a diagonal of asterisks from the top left to the bottom right of the paper, each line stepping the asterisk one column further to the right until it reaches its upper limit of the carriage length.

```
1640 PRINT TAB( I)"*": NEXT I
```

Line 1650 resets the output from the printer back to the screen.

```
1650 PR# 0
```

Line 1660 checks whether or not the Exerciser has control of the program. If it does, the program branches to line 1690 for the next printer Exerciser test.

```
1660 IF EX = 1 THEN GOTO 1690
```

If the Exerciser is not on, the program proceeds to line 1670, which ends the Horizontal Tab Test and branches back to display the Printer Diagnostic Menu.

```
1670 GOTO 1130
```

Line 1680 begins the Line Feed Test with the REMARK statement. The Line Feed Test feeds a varying number of lines to check whether the feed mechanism is working properly.

```
1680 REM LINE FEED TEST
```

Line 1690 prints the test title on the screen.

```
1690 HOME : PRINT TAB( 13)"LINE FEED TEST"
```

Line 1695 activates the printer, and outputs the test title to the printer.

```
1695 PR# SL: PRINT "LINE FEED TEST"
```

Line 1700 prints the first line of the test.

```
1700 PRINT "THIS IS THE FIRST LINE....."
```

Line 1710 begins the FOR-NEXT loop which will increment the next four line feeds and print the accompanying lines.

```
1710 FOR I = 2 TO 5
```

Line 1720 begins a second, nested FOR-NEXT loop. This loop controls the number of line feeds in each loop, based on the value in I in the I loop.

```
1720 FOR A = 1 TO I
```

Line 1730 prints CHR$(10), which is a line feed. Then the NEXT statement increments the A loop, to feed the number of lines according to the value in A, either 2, 3, 4, or 5.

```
1730 PRINT CHR$ (10);: NEXT A
```

Line 1740 sets N$ equal to the string value of I.

```
1740 N$ = STR$ (I)
```

Line 1750 takes the value of N$ and prints the message of the number of line feeds to the printer. The I loop is incremented with the NEXT I statement.

```
1750 PRINT "THERE HAVE BEEN ";N$;" LINE FEEDS";:
     NEXT I
```

Line 1760 prints CHR$(12), which causes a form feed. Then it prints the message indicating this has been done.

```
1760 PRINT CHR$ (12);: PRINT "THIS IS A NEW PAGE....."
```

Line 1770 switches output control from the printer back to the screen.

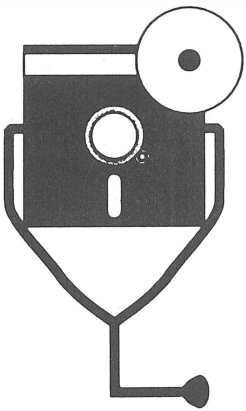
```
1770 PR# 0
```

Line 1780 checks the Exerciser flag. If the Exerciser is in control, the program returns to the Exerciser module, having completed the printer Exerciser tests.

```
1780 IF EX = 1 THEN RETURN
```

If the Exerciser is not in control, the program proceeds to line 1790, which branches the program back to line 1130 to display the Printer Diagnostic Menu.

```
1790 GOTO 1130
```



7

The Disk 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 sophisticated with the computer, the facilities of one more function are needed: mass storage, as provided by the disk drive.

The Apple II Plus CPU can store 48 kilobytes of information at one time. The IIe can store 64k, and the IIc provides the most memory, with 128k. This capacity is not adequate to hold software programs and all your data. Two or three application programs could easily fill this capacity, leaving no room for additional software or data with which to use the programs.

In addition, most microcomputers, the Apple II systems included, have a volatile memory. This means that in order for the random-access memory (RAM) chips to retain their data, power must be continuously supplied to their circuitry. Once the computer is turned off, everything that was stored in that 48, 64, or 128kB is now erased.

To provide an unlimited storage capacity, and to prevent the loss of valuable programs and data, the floppy diskette drive is used. The disk drive provides the additional storage necessary to supplement system memory. When you command the computer to save a file, the program currently in system memory is written onto the diskette. The file is written in *digital* form, as a series of ON and OFF bits, or 1s and 0s.

Likewise, when you command the computer to retrieve a saved file, the system recalls the information stored on the diskette. It places it into system memory just as it was when last saved. The disk drive also allows you to buy prerecorded programs and load them into your system.

Saving and retrieving files points out the two storage functions the diskette drive provides: recording, or *writing,* information from the computer onto the diskette; and playing back, or *reading,* information from the diskette into computer memory.

DESCRIPTION AND FUNCTION OF THE DISK DRIVE

A disk drive provides a means for high-speed storage and retrieval for programs and data, and as such, it is one of the most useful peripheral devices for the Apple II (Fig. 7-1). Spreadsheet software, database programs, and word processing applications can be quickly stored and accessed using a disk drive. The disk drive can also store the data on which these programs operate: the numbers and table data in a spreadsheet, the field and record data in a database, and the words in a word processing file.

The terms disk, diskette, disk drive, and drive are used in this chapter. *Disk* is the general 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.

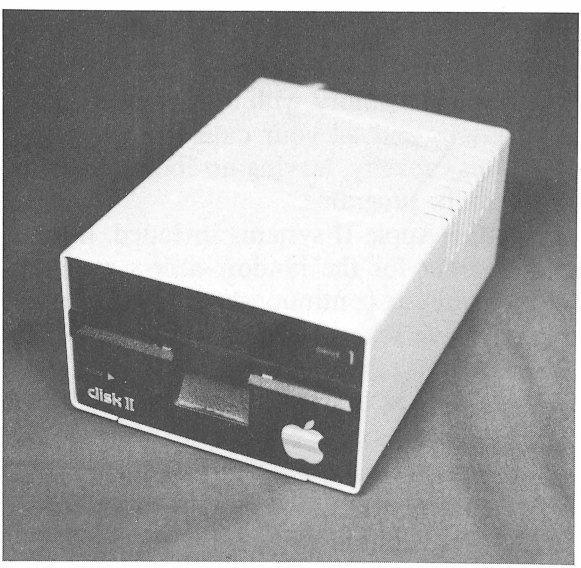


Fig. 7-1. The disk drive.

Types of Disk Drives

The Apple II disk drive can store 143 kilobytes of data on a diskette. It is an external disk drive, meaning it resides in its own housing external to the system unit, and has its own power supply.

In addition to the external drive, the Apple IIc also provides an internal disk drive, with the diskette door on the right side of the system unit.

The first computers with disk drives were the mainframe computers used in large corporations. The storage media used were 14-inch metal platters (Fig. 7-2). These rigid platters, or disks, were stacked like phonograph records in the mainframe's disk drive.

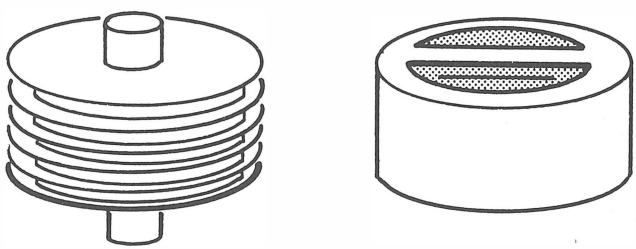
When the personal computer was invented in the '70s, a new kind of storage disk was introduced: the eight-inch floppy diskette. They were called floppy, because unlike their metal predecessors, the recording media consisted of a smaller platter made of thin, flexible plastic, which was encased in a protective plastic covering. The technology brought about by the microcomputer revolution had miniaturized everything about the computer, including the disk drive.

The microcomputer's components continued to shrink in various ways. Among these components were the disk drive and diskette. Soon the 5 ¼-inch mini-floppy diskette with its 5 ¼-inch disk drive became the standard. There are even micro-floppies, at 3 ½ inches (Fig. 7-3). These diskettes are smaller and easier to use than the 8-inch versions. The Apple II systems use 5 ¼-inch floppy diskettes.

The Disk Drive System

The disk drive system consists of three basic components:

- floppy diskette
- disk drive controller
- disk drive mechanism



disk platters disk pack

Fig. 7-2. 14-inch mass storage platters.

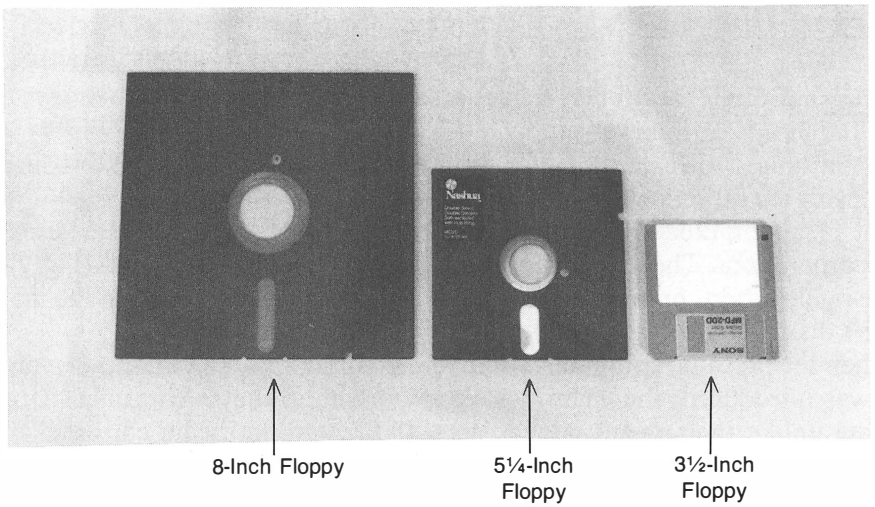


Fig. 7-3. Floppy diskettes.

Floppy Diskette. The floppy diskette is a magnetic recording medium coated with iron oxide, similar to the material used on cassette tapes. The iron oxide covers the surface of the diskette like a film, or emulsion, and is used because it holds a magnetic charge.

The magnetic media used by the Apple II systems is the 5 ¼-inch single-sided, double-density floppy diskette. The Apple II system can write 143 kilobytes of data to this diskette.

Disk Drive Controller. The disk drive connects to the computer system unit and is operated by means of the disk interface controller (Fig. 7-4). The disk drive and controller are optional on the II Plus and IIe. The disk drive is external, and conventionally its controller board is installed in expansion slot #6 on the motherboard.

An internal disk drive and controller are standard on the Apple IIc. The disk drive is built into the IIc system unit, and the disk drive controller consists of a single chip on the motherboard (Fig. 7-5). A connector for a second external drive is found at the rear of the IIc system unit.

Each disk drive controller has connectors for two disk drives. This means that for every disk drive controller the system contains, it can control two drives, designated as D1 and D2. An Apple II system can support a maximum of three disk drive controllers for a total of six disk drives (Fig. 7-6).

The disk drive controller instructs the disk drive when and how to read and write information, as directed by the CPU and software commands.

Disk Drive Mechanism. The disk drive itself consists of a motor, drive head, and access arm (Fig. 7-7). When you insert a diskette into the disk drive and close

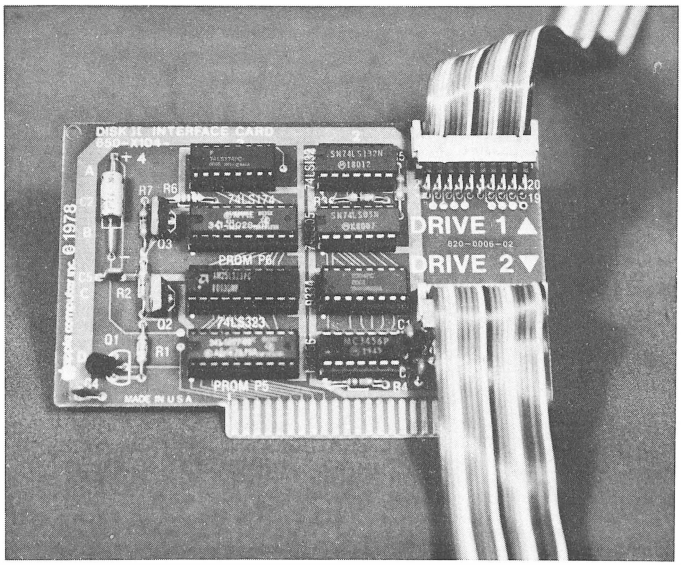


Fig. 7-4. Disk drive controller board.

the drive latch, a switch inside the drive tells the system that a diskette has been inserted. The drive motor then spins the diskette at a constant speed of 300–400 revolutions per minute (rpm).

The small hole located near the center of the diskette is used by the disk drive circuitry for timing and synchronization. This hole senses when the diskette is

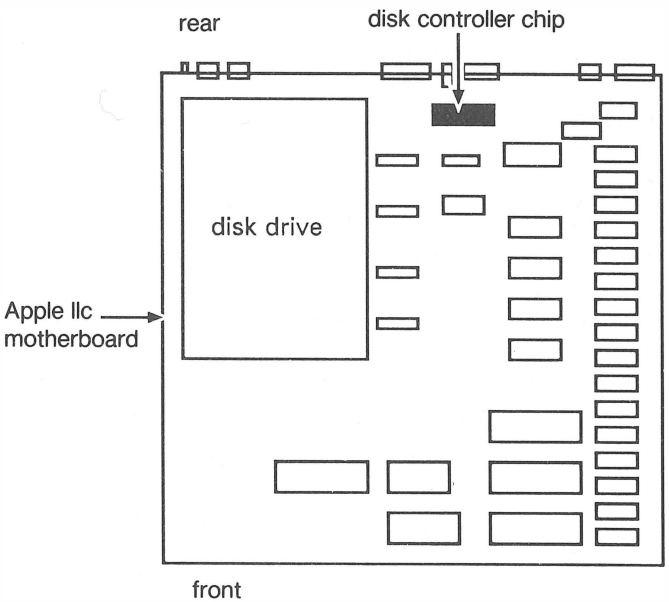


Fig. 7-5. Apple IIc disk drive controller chip.

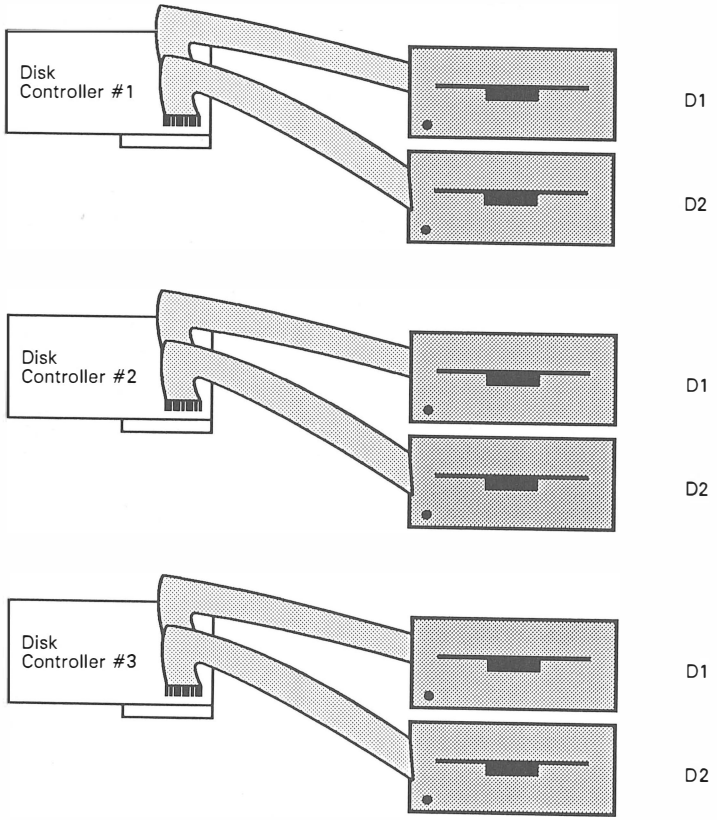


Fig. 7-6. Maximum disk drive configuration.

spinning at the correct rpm necessary for reading or writing data. As long as the disk drive motor is rotating the diskette at the correct speed, data can be written to and read from the surface of the diskette.

The head-arm assembly locates, reads, and writes data to and from the disk according to the instructions coming from the disk drive controller.

The head is a magnetic device on the end of a mechanical access arm (Fig. 7-8). When a save command is given, this head device magnetizes the diskette emulsion, thereby writing new data onto the diskette. The head can also locate and sense a magnetic pattern already existing 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 is positioned a few microns above it on a cushion of air. Software and hardware logic position the head at the proper location for reading or writing.

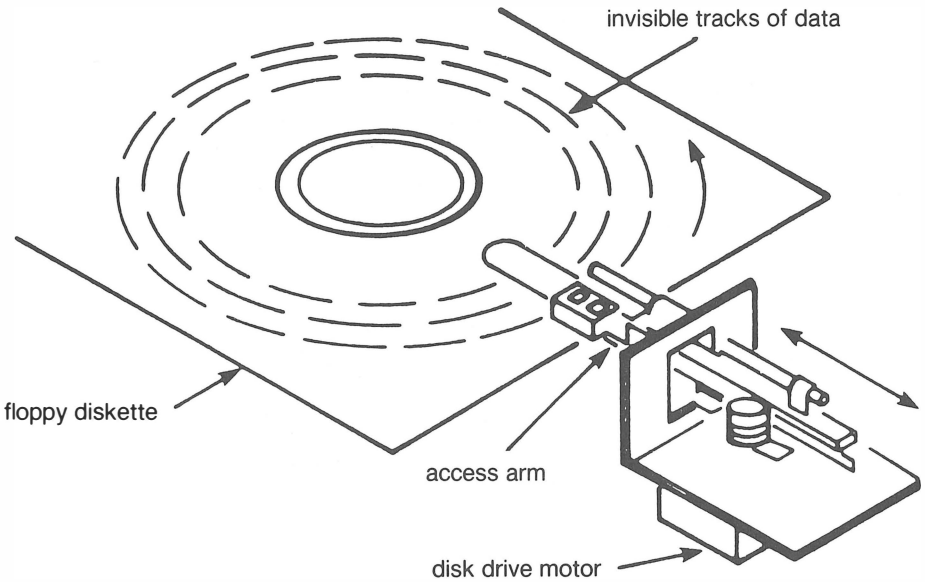


Fig. 7-7. Disk drive mechanism.

How a Disk Drive System Functions

The components described above: the floppy diskette, the disk drive controller board, and the disk drive mechanism itself, all work together to perform the functions of the disk drive system. These functions consist of:

- locating and grouping data on the diskette
- writing data from system memory to the diskette
- reading data from the diskette to system memory

Locating and Grouping Data. When data is being written, the timing and

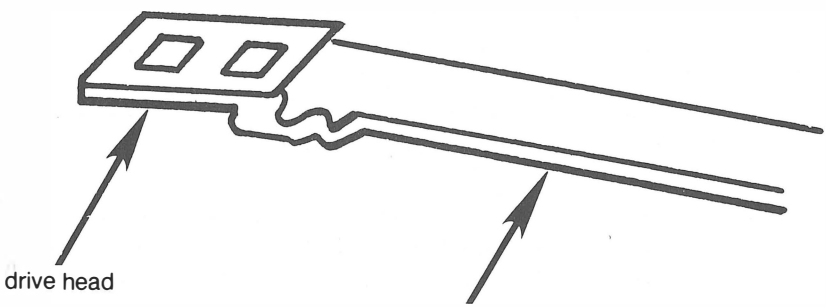


Fig. 7-8. Drive head.

synchronization hole on the diskette is used as a location reference point. This hole marks the starting point for information to be written onto the diskette.

The disk drive controller knows the locations and contents of each group of data stored on the diskette. When the controller receives a read or write command from the CPU, it locates the required data, and directs the head to the proper place for reading.

The diskette, like a phonograph record or a compact laser disk, stores specific information in specific areas. For example, just as the phonograph record stores different songs on different tracks, a diskette stores information on various tracks and sectors. The diskette is divided into a series of concentric circles, with each circle being one *track.* There are 35 tracks on the Apple II 143-kilobyte disk drive (Fig. 7-9). When the diskette is formatted, each track is divided into *sectors* (Fig. 7-10). Each Apple II disk sector contains 256 bytes. In any read or write operation, exactly one sector's worth of information is located and transferred. The program may need to manipulate only 20 bytes of that sector, but the computer always transfers all the information in the entire sector.

Writing Data. When the user gives a save command, data currently residing in a disk buffer in system memory is written onto the magnetic emulsion of the diskette. With the magnetic drive head positioned over the proper track and sector, the data is transferred, written, and stored as tiny magnetic fields on this surface (Fig. 7-11). A 1 data bit is magnetized in one direction on the diskette, while a 0 data bit is magnetized in another direction. A group of eight of these data bits constitutes one byte of data. Many bytes together constitute a record, or file.

The floppy diskette has a method for protecting the programs and data stored there from being overwritten or erased. The diskette includes a small cutout on the side called a write-protect notch. As long as the write-protect notch is exposed,

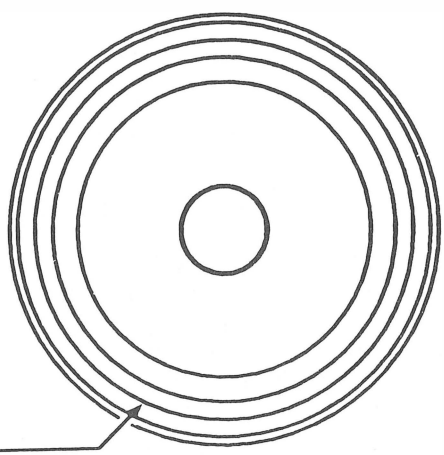


Fig. 7-9. Diskette tracks.

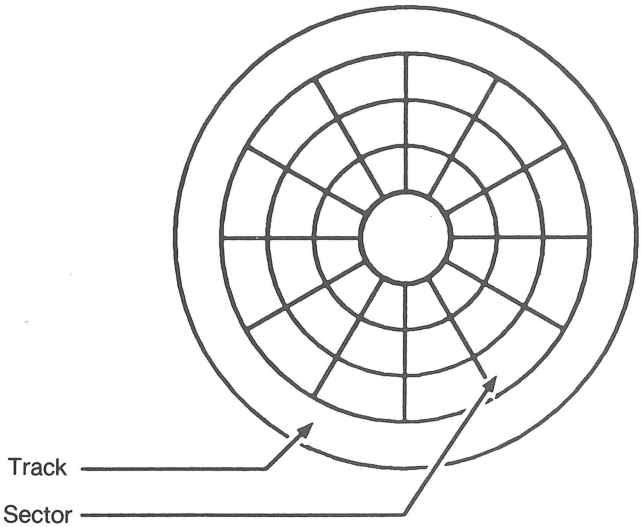


Fig. 7-10. Diskette sectors.

the disk drive head is able to write information onto the diskette as well as read from it (Fig. 7-12). New boxes of diskettes always include a package of small 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 dis-

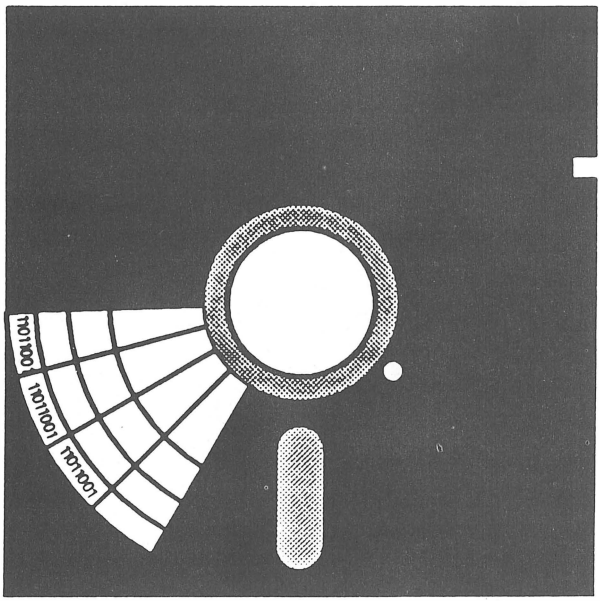


Fig. 7-11. Data written on the diskette.

abled. The tape blocks the write logic sensor in the disk drive, preventing data from being written onto the diskette.

Reading Data. When the operating system, application program, or user gives a recall or read command for a specific record or file, the CPU sends these instructions to the disk drive controller. The controller locates the requested data and sends its instructions to the disk drive head-arm assembly.

The head is positioned over the proper track and sector as directed by the drive controller. The tiny magnetic fields stored in the diskette emulsion at that location induce electrical impulses into the drive head, allowing the head to copy the data from the sector and transfer it to a disk buffer in system memory. At that point, the data can be used as directed by the program and the user.

DISK DRIVE PREVENTIVE MAINTENANCE

There are three major components involved in disk drive preventive maintenance: the disk drive, the disk drive controller board, and floppy diskettes.

The Disk Drive

The floppy disk drive requires perhaps a greater degree of preventive maintenance than the other subassemblies that make up the Apple II system. It is a very

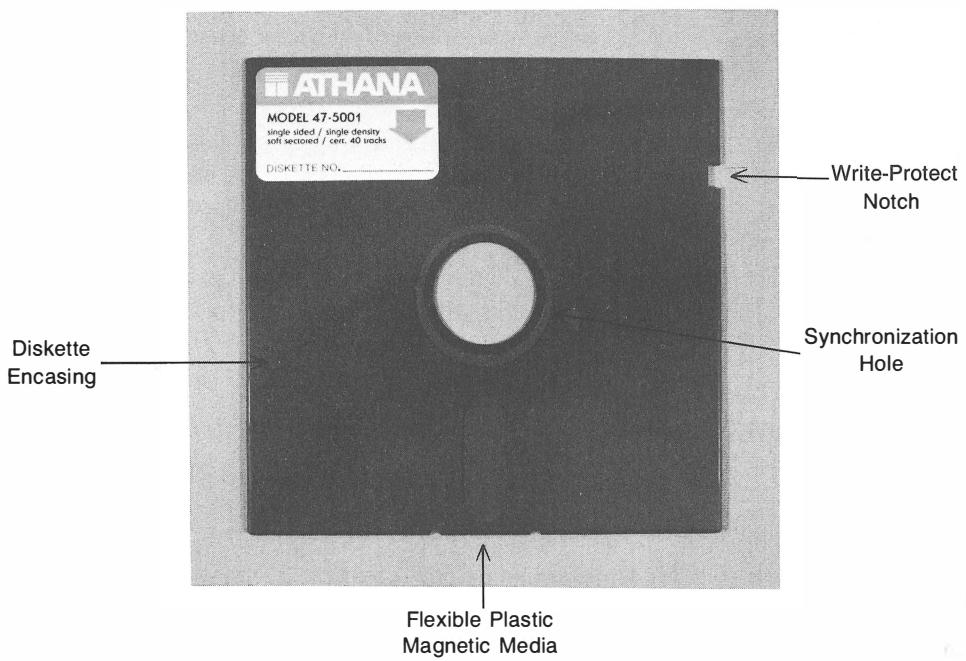


Fig. 7-12. 5 1/4-inch diskette.

delicate mechanism, in which a single piece of dust in the wrong place can prove catastrophic to programs and data, causing the disk drive head to crash onto the surface of the floppy diskette.

Make sure that the disk drive doors are closed when not in use, and cover the disk drive with its dust cover. Both of these measures prevent dirt and dust from collecting on the disk drive head mechanism.

When you move your system, insert the cardboard diskette that comes with the drive, and close the door. This protects the head mechanism from being jolted, thereby preventing it from going out of alignment during transport.

Purchase a disk drive head cleaning kit from a computer store, and clean the drive head at least once a month. Follow the instructions that come with the cleaning kit. Often these kits are supplied with a short program that positions the head over the cleaning diskette. The head is cleaned as the IPA impregnated cleaning diskette flashes off any material that might have accumulated on it.

The Disk Drive Controller Board

The disk drive controller is a separate board installed in one of the expansion slots on the motherboard. Whenever you open the system unit for any preventive maintenance, troubleshooting, or repair purposes, perform some preventive maintenance on this board as well.

Pull the board off (refer to "Maintaining Your Apple II System" in Chapter 1 for board handling and cleaning procedures) and clean any oxidation from the edge connector. Make certain the board is properly seated in its slot for an optimum connection. If there are one or two drives, make sure the controller is installed in the slot labeled as slot 6.

The Floppy Diskettes

The diskettes themselves also require special care and handling. Often what might appear to be a disk drive or controller problem is nothing more than a media problem that could have been prevented. When not in use, store each diskette in a diskette envelope, and place all your diskettes in a diskette box or case. This prevents dust and dirt from gathering on the diskette, and also protects the diskette from damage by heat or magnetic fields.

Just like a phonograph record, heat can warp a diskette, and render it unusable. If you place a warped diskette is placed into the disk drive, it will damage the drive head assembly. Magnetic fields caused by power supplies, appliance motors, magnets, or transformers will erase or change the data stored on the diskette surface.

As soon as you obtain (or write) a new program, immediately make a backup of it, and write-protect both copies. If one of the program diskettes wears out or

encounters a defect, the backup will keep you working. Write-protecting the program diskettes will prevent accidental erasure of the programs.

Keep current backups of data diskettes as well. Get into the habit of making daily or weekly backups of files, depending on how much work you do in a given time period. If a particular data file gets to a point where is is finished, and will not be changed anymore, make a final backup and write-protect both diskettes. Having backups and write-protecting when possible protects the data from accidental loss from a power failure, system failure, or inadvertent overwrite.

DISK DRIVE TROUBLESHOOTING AND REPAIR GUIDELINES

Disk drive problems can be cause by the diskette, the disk drive controller board, the disk drive, or the motherboard. When you troubleshoot any disk drive problem, check for diskette problems first. If the diskette is good, then check the disk drive controller. If the controller is functioning normally, troubleshoot the disk drive itself. If the disk drive is operational, finally troubleshoot the motherboard.

If your troubleshooting efforts point to a problem in the disk drive controller board, make sure it is seated properly in the correct expansion slot, and see that the contacts are clean.

If you find a problem in the disk drive, take it in for servicing. The disk drive can require many precision adjustments that should be performed only by computer repair professionals.

If a disk drive problem is caused by the motherboard, refer to Chapter 3 for system unit troubleshooting procedures.

Problem: The Disk Drive Does Not Work

If your disk drive cannot read from or write to a diskette, try the following solutions.

Solutions:

- Check how the diskette was inserted. If it's inserted upside-down, the drive cannot read from or write to it. The Apple II systems use horizontal disk drives, in which the diskette should be inserted face up, or diskette label up.
- The disk drive has an in-use indicator light next to the drive door. This light indicates that the drive is currently executing a read or write operation. If your system boots up displaying the cursor on the screen, and the fan is running but the disk drive in-use indicator light is not illuminated, then there's a good chance that no power is being supplied to the disk drive.

Another indicator is if no program loads from the diskette, and only the Applesoft BASIC prompt is displayed, this indicates that the drive has no power.

- Check that power is being supplied to the drive. Its power comes from the system unit power supply. Check that the drive is properly connected to the system unit.
- If the power checks out, but the drive indicator is not on, and the drive still cannot be accessed, there is a problem with the cable or controller. Take the drive in for servicing.
- If the operating system does not load, and you do not even see the Applesoft BASIC right bracket prompt, check that the disk drive interface cable is properly connected between the drive and the motherboard.
- If your Apple II configuration includes more than one disk drive, there might be an addressing problem from the software. Make sure that the disk is being properly addressed by the program in use.

  The disk address is based on the expansion slot in which the disk drive controller resides, such as slot #4. Because two disk drives can be connected to each controller, the address also must include the drive connector number: D1 or D2. An example of a disk drive address would be "Slot 4 D1," or "Slot 6 D2," and so forth.

Problem: Cannot Load a Program or Read a Data File

If your attempts to boot up the system, load a program, or read a file only cause a read error message to be returned, then there is probably a problem with the disk drive's read circuitry.

Solution:

- If your system includes a second disk drive, place the program diskette in the other drive, and read the program from there. If your system cannot read from the other drive, then you could have a bad program diskette. Load the backup diskette.

  If it does load, run the Disk Drive Diagnostic module on a blank, formatted diskette. The diagnostic writes a specific pattern to the disk, reads it back, and compares the two to make certain they match. This checks the read/write logic. Run the diagnostic on the drive that would not load the program.

  If the test runs successfully, and the test completion message is returned from the diagnostic, then nothing was wrong with your disk drive. It was probably a diskette problem. Try a different diskette in the drive, and it should work.

If the diagnostic test fails, or returns a lengthy series of read errors, then either the disk drive or the read circuit on the disk drive controller board is defective.

- Swap the disk drive on another system. If it does not work, then your drive is faulty and should be taken in for servicing.
- If the disk drive does work, swap your disk drive controller with another system. If it does not work, then a certain chip on the drive controller board is faulty and should be replaced. Refer to "Working with IC Chips" in Chapter 1 for further instructions on replacing chips. On the II Plus and IIe, use the chip extractor to carefully remove the MC3470 and CA3146 chips, which control the board's read circuitry. On the IIc, remove the IWM chip from the motherboard. Take the chip(s) to an electronics supply store and obtain a replacement(s). Carefully install the new chip(s) in the socket.
- If both your disk drive and controller work on the other system, but they do not work on yours, then you have a faulty system unit. Refer to Chapter 3 for system unit troubleshooting procedures.

Problem: Read Data Is Garbled

In addition to wearing out, diskettes can be damaged and the data on them garbled if not handled properly. Since diskettes are magnetic media, magnetism can adversely affect the data on a diskette. If a diskette is placed next to an unshielded power supply, or near an appliance with a motor, the motor's magnetic field will confuse the data.

Although the 5 ¼-inch diskette is enclosed in plastic, 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.

Problem: Cannot Write to the Diskette

A common disk drive problem is that the system cannot write data to the diskette. The first thing you should check is the diskette itself.

Solutions:

- If your system is not writing to a brand new diskette, first make sure that it is the correct type of diskette for the drive. Make sure you are using a single-sided, soft-sectored diskette in your drive. Check the disk drive documentation for any additional diskette requirements.
- If the diskette is the correct type for the drive, the problem might be that the diskette is not *initialized,* or formatted. There are many diskette manufac-

turers, and the same type of diskettes can be used with a variety of computers. A diskette is made to work on your particular computer by inserting the diskette into the disk drive and running the operating system's initialization program, which for the Apple II is the INIT command. This must be done to all new diskettes in order for them to work with your computer. The initialization format program places operating system information and a disk directory program onto the new diskette.

Initializing also tests the diskette for bad sectors. If any bad sectors are found, the INIT 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, examine the diskette to see whether it is write-protected.
- If you've been using a particular diskette, and now find that the computer can no longer write to it, it may have 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 long-term use. This is another reason why keeping a current backup copy of each diskette is so important. If a diskette wears out, you can simply use its backup.

  You can also try re-initializing the diskette. When a diskette is first initialized, any bad sectors are blocked out. Sometimes a bad sector can develop after initialization. In this case, the diskette can be salvaged with re-initialization. Again, all bad sectors are blocked out, including the new ones, so the diskette is like new again. Remember, however, that when a diskette is reformatted, all data on it is erased.
- If you still cannot 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 was defective. Use its backup.
- If you cannot write to the second diskette, then you probably have a problem with your drive. To check this, try the diskette on another system. If it does not write on the system, then you have a bad diskette. However, if it does write on the other system, then you have a problem with either your disk drive controller or the drive itself.
- Open the system unit and locate the disk drive controller board (this only applies to Apple II Plus and IIe systems). Pull the board out of the slot (see Chapter 1 for board handling and cleaning procedures) and clean oxidation from the edge connector. Carefully reseat the board in its slot. If there are only one or two drives, made sure the controller is installed in slot #6.
- If your drive still does not write, swap your disk drive controller on another Apple II system. If you use an Apple IIc, either swap the entire motherboard

or the drive controller chip. If the other disk drive works, your system has a faulty disk drive.

If the other disk drive does not work with your controller, then a certain chip on the drive controller board is faulty and should be replaced. Refer to "Working With IC Chips" in Chapter 1 for further instructions on handling and replacing chips.

On the II Plus and IIe, use the chip extractor to carefully remove the MC3470, CA3146, 74LS125, and ULN2003 chips from the disk drive controller board. On the IIc, temove the IWM chip from the motherboard. Take the chip(s) to an electronics supply store and obtain a replacement(s). Carefully install the new chips(s) in the socket(s).

- If the other system worked with your drive controller, then you have a faulty disk drive. Swap the drive on another system. If your drive does not work, it is defective, and should be taken to an Apple II service outlet for further troubleshooting and repair. If your drive does work on another system, and you have already checked out the drive controller, then you have a problem with the motherboard. Refer to Chapter 3 for system unit troubleshooting procedures.

Problem: Data Mismatch — Read/Write Circuitry is Defective

If a large number of data mismatch errors are encountered when you run the Disk Drive Diagnostic module as indicated by the "THERE WERE X ERRORS ENCOUNTERED DURING TEST" message displayed at the end of the test, this means that the records that were written by the test do not match with the records that were read. This points to a problem with the read/write circuitry in the drive controller.

Solutions:

- Replace one or more chips on the drive controller board. Refer to "Working With IC Chips" in Chapter 1 for further instructions on handling and replacing chips.

  On the II Plus and IIe, use the chip extractor to carefully remove the MC3470, CA3146, 74LS125, and ULN2003 chips from the disk drive controller board. On the IIc, remove the IWM chip from the motherboard. Take the chip(s) to an electronics supply store and obtain a replacement(s). Carefully install the new chip(s) in the socket(s).

  Run the Disk Drive Diagnostic again. There should be no errors returned from the Read/Write Test. If the errors persist, contact an Apple II service

outlet. The drive controller or the motherboard might need more in-depth troubleshooting.

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, or if the disk drive has been shaken or dropped. A head crash can also be caused by trying to use a warped diskette, or by contamination on the diskette or in the drive mechanism (Fig. 7-13).

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 will scrape up against the head and cause a head crash. Don't try to use a warped diskette, because if it causes a head crash, the disk drive will become misadjusted, and will have to be taken in for servicing.

A head crash is indicated if the system cannot read from, write to, or initialize a disk. Another indication is a grinding, scratching sound heard when a diskette is inserted into the drive.

If 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: I/O Error Message

If an I/O Error message displays when you're trying to access the disk drive, then there are a number of different disk drive read-write problems. It might mean that the diskette is defective, write-protected, not initialized, or not even present. The error message might also mean that the drive is not able to locate the identifying header portion of a sector that the system is trying to access. It could

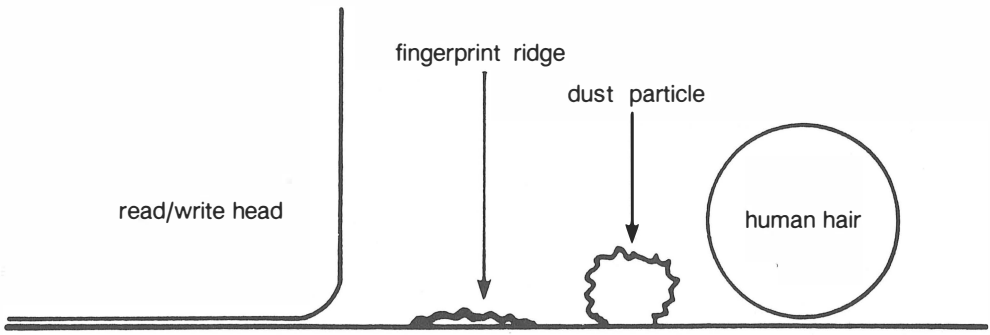


Fig. 7-13. Disk contamination.

indicate that the drive cannot find the synchronization marker for the track the system is trying to access.

Solutions:

- First check the diskette. See that there is a diskette present in the drive. Make certain that the disk drive door is closed.
- Remove the diskette from the drive, and check whether or not the write-protect notch is covered with an adhesive write-protect tape. If it is, remove the tape and insert the diskette into the drive again. The system should now be able to write information onto the diskette.

  However, remember that you probably originally placed the write-protect tape on the diskette in order to protect data already written on the diskette. Make sure that the system does not overwrite valuable data or programs, now that the diskette is write-enabled.
- Try reading from and writing to another diskette that you know is formatted. If you are successful with this diskette, and the error message is not returned, then the original diskette probably was not formatted.
- If the error message is returned when requesting access to a certain file, give the CATALOG command to see if the file is present on the diskette. If the file is present, then there is a problem with the diskette itself. Try the diskette's backup.
- Run the Disk Drive Diagnostic module on a blank, formatted diskette. If the diagnostic does not display any error messages with this diskette, then the original diskette was defective, and its backup should be used.

  If further error messages are displayed with the new diskette, the disk drive controller or the drive itself might be defective.
- Swap the disk drive controller on another system. If it does not work, then a certain chip on the drive controller board is faulty and should be taken in for servicing, or replaced. Refer to "Working with IC Chips" in Chapter 1 for further instructions on replacing chips.

  On the II Plus and IIe, if it's a read problem, use the chip extractor to carefully remove the MC3470 and CA3146 chips. If it's a write problem, remove those two chips in addition to 74LS125 and ULN2003. On the IIc, remove the IWM chip from the motherboard. Take the chip(s) to an electronics supply store and obtain a replacement(s). Carefully install the new chip(s) in the socket(s).
- If the disk drive controller works on the other system, troubleshoot the disk drive. Swap your disk drive on another system. If the I/O Error Message is still returned, then the drive is faulty and should be taken in for servicing.
- If the disk drive and controller work on the other system, but they do not

work on yours, then you have a faulty motherboard. Refer to Chapter 3 for system unit troubleshooting procedures.

Problem: "The Diskette is Write-Protected" Error Message

If the error message "The Diskette is Write-Protected" is returned while running the Disk Drive Diagnostic module, stop the program and check the contents of the diskette. Find out why it was write-protected. Replace it with a blank, formatted diskette, or remove the write-protect tape and resume the test.

Problem: "File Not Found" Error Message

If the error message "File Not Found" is returned while running the Disk Drive Diagnostic module, then there is a problem with the read circuitry. Refer to "Problem: Cannot Load a Program or Read a Data File" above for solutions.

Problem: "Volumn Mismatch" Error Message

If the error message "Volumn Mismatch" is returned while running the Disk Drive Diagnostic module, then there is a problem either with the read circuitry on the disk drive controller, or with the CPU. Refer to "Problem: Cannot Load a Program or Read a Data File" above for solutions.

Problem: "Try a New Diskette" Error Message

If the error message "Try a New Diskette" is returned while running the Disk Drive Diagnostic module, then you have a defective diskette. Try using a new, freshly formatted, blank diskette. If the error persists, refer to "Problem: I/O Error Message" above.

Problem: "Out of Space on Disk, Try an Empty One" Error Message

If the error message "Out of Space on Disk, Try an Empty One" is returned while running the Disk Drive Diagnostic module, then the scratch diskette being used for the test is not blank. Either check the contents of the diskette to make sure you no longer need the files and reinitialize it, or use another formatted diskette that is empty.

Problem: "No Buffers Available—Reboot System" Error Message

If the error message "No Buffers Available—Reboot System" is returned while running the Disk Drive Diagnostic module, then the user-controllable

buffers have been deallocated. Reset the system to reset the buffer defaults, and resume testing.

Problem: Disk Drive Needs to Be Serviced

If the solutions described above fail to solve your disk drive problem, and you need to take it in for servicing, first call the Apple II service center and talk to the repair technician. Describe the computer's configuration, and explain the disk drive problems. Tell them what troubleshooting measures you have already tried, and the results. Find out if there are any solutions they can provide over the phone.

If the disk drive cannot be repaired over the phone, ask the technician how much of your system you should bring. You will need to take in the disk drive and its controller board. The technician will probably also need the system unit.

When you take the disk drive in, find out if the technician has any ideas on what the problem might be, and how long and how much it might cost to fix. Be sure to get a copy of the work order.

Before you pay for and accept your disk drive and controller, have the repair technician demonstrate that the problem has actually been repaired. When the system is all together again, refer to Chapter 9, The Exerciser, for instructions on burning in the system after repair. The Exerciser has an extensive section on burning in the disk drives.

RUNNING THE DISK DRIVE DIAGNOSTIC MODULE

Follow the instructions provided in "Running the Diagnostic Program" in Chapter 2 to run the Disk Drive Diagnostic module. Then press `D` to initiate the Disk Drive Diagnostic module. Remove the System Diagnostic Program diskette and insert a blank, formatted diskette to be used for the test.

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, an error message will displayed on the screen. These error messages are referenced in the "Disk Drive Troubleshooting and Repair Guidelines" section above, and list remedies and responses to the error.

When you run the test, use a recently formatted, blank diskette. Do not use a diskette that contains data on it, because the test will run out of room on the diskette, and an error message will be returned.

The Disk Drive Diagnostic begins with two prompts. The first prompt asks for the slot number in which the disk drive controller resides, 0 through 7. A second prompt asks for the specific drive on that controller to be tested, 1 or 2 (Fig. 7-14).

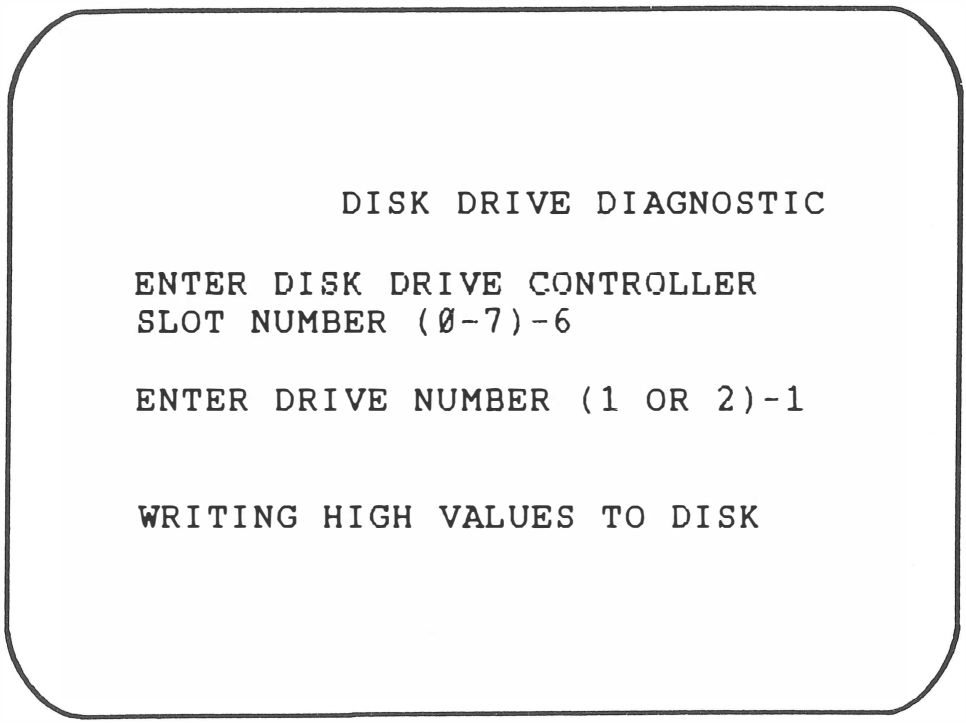


Fig. 7-14. Disk drive diagnostic module.

The diagnostic creates a file called "TEST," and makes four distinct passes through the specified disk.

Running the Disk Drive Diagnostic takes approximately 20 minutes, with five minutes per pass through the disk. If you wish to halt the program during this test, press CTRL C. The System Diagnostic Program will be halted, and the Applesoft BASIC environment will be displayed. Depending on when the program was halted, the "TEST" file might still be on the test diskette, and you might wish to delete it.

The first pass writes a series of 254-byte records that contain a 254-character string of CHR$ value 170. CHR$ value 64 is a character that has a bit string of 1000000. The diagnostic writes 239 records of CHR$ value 64 to fill up the disk. While the records are being written to disk, the message "WRITING HIGH VALUES TO DISK" is displayed. It takes about five minutes for the disk to be filled with these records. Once this is done, 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 64. While the records are being read and compared, the message "READING HIGH VALUES FROM

DISK" is displayed. It takes about five minutes for all the records to be read and compared.

If there is a discrepancy between what was written and what is now being read, an error is noted and added to the counter. The total number of errors is displayed on the screen at the end of the test.

After the second pass of the test is finished, the file will be closed and deleted. The third pass recreates the file, this time with a series of 254-character records of CHR$ value 63, which has a bit string of 0111111, opposite of the CHR$ value 64 done in the previous pass. The diagnostic writes 239 records of this value to fill the disk again. The message "WRITING LOW VALUES TO DISK" is displayed for approximately five minutes. When the diskette is full, again, the file is closed.

Because CHR$(63) and CHR$(64) have an alternating bit pattern, this ensures that all possible bits on the disk are written, creating a thorough test of the disk.

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$(63). The message "READING LOW VALUES FROM DISK" is displayed for approximately five minutes.

The error counter is incremented if there are any discrepancies between what was written and what is now being read. Then the file is deleted, and the test is complete.

Data mismatch errors are counted and a final total is displayed at the end of the test. Such errors are not fatal errors, and the test will continue. If there are 1 or 2 mismatch errors, you might want to test the drive more often, to see if it gets worse. But one or two errors do not warrant your taking action.

If a greater number of mismatch errors are found by the diagnostic, then you have a problem in the read/write circuitry in the drive controller. Refer to "Problem: Data Mismatch — Read/Write Circuitry is Defective" above for suggestions on correcting this problem.

If any fatal errors are found during the test, the error message will be displayed and the test will halt. Each error message is referenced under "Disk Drive Troubleshooting and Repair Guidelines" above. Follow the indicated suggestions and run the test again.

HOW THE DISK DRIVE DIAGNOSTIC MODULE WORKS

When the user presses D or d from the System Diagnostic Main Menu, the program branches to line 2500. A REMARK statement indicates the beginning of the Disk Drive Diagnostic module (Fig. 7-15).

```
2500 REM DISK DRIVE DIAGNOSTIC MODULE
```

```
2500  REM  DISK DRIVE DIAGNOSTIC MODULE
2510  HOME :D$ =  CHR$ (4):H$ = "":L$ = ""
2515  PRINT  TAB( 10)"DISK DRIVE DIAGNOSTIC": PRINT
2520  FOR I = 1 TO 239
2530  H$ = H$ +  CHR$ (64):L$ = L$ +  CHR$ (63): NEXT I
2540  IF EX = 1 THEN  GOTO 2590
2550  PRINT "ENTER DISK DRIVE CONTROLLER": INPUT "SLOT
      NUMBER (0-7)-";S
2560  IF S < 0 OR S > 7 THEN S = 6
2570  PRINT : INPUT "ENTER DRIVE NUMBER (1 OR 2)-";D:
      PRINT
2580  IF D < 1 OR D > 2 THEN D = 1
2590  B$ = H$:N$ = "HIGH VALUES"
2600  ONERR  GOTO 3000
2605  FOR I = 1 TO 2
2610  PRINT : PRINT "WRITING ";N$;: TO DISK"
2620  PRINT D$;"OPEN TEST,S";S;",D";D
2630  PRINT D$;"WRITE TEST"
2635  FOR L = 1 TO 430
2640  PRINT B$: NEXT L
2650  PRINT D$;"CLOSE TEST"
2670  PRINT : PRINT D$: PRINT "READING ";N$;" FROM
      DISK": PRINT
2675  PRINT D$;"OPEN TEST,S";S;",D";D
2680  PRINT D$;"READ TEST"
2685  FOR L = 1 TO 430
2690  INPUT B1$
2700  IF B1$ <  > B$ THEN  PRINT D$: PRINT "ERROR
      READING ";N$:E = E + 1
2710  NEXT L
2720  PRINT D$;"CLOSE TEST"
2730  PRINT D$;"DELETE TEST"
2740  B$ = L$:N$ = :LOW VALUES": NEXT I
2750  PRINT D$: PRINT : PRINT "THERE WERE ";E;" ERRORS
      ENCOUNTERED": PRINT "DURING TEST"
2760  IF EX = 1 THEN  POKE 216,0 : RETURN
2770  PRINT : PRINT "PRESS ESC TO END TEST":
      GOSUB 8000: POKE 216,0: GOTO 50
3000  ER =  PEEK (222)
3002  IF ER = 5 THEN  GOTO 2720
3005  PRINT D$;"CLOSE TEST": PR# 0: IN# 0
3010  PRINT : PRINT  TAB( 5)"A FATAL ERROR WAS
      ENCOUNTERED!" : PRINT
3020  PRINT "ERROR CODE:";ER: PRINT
3030  IF ER = 4 THEN  PRINT "THE DISKETTE IS
      WRITE-PROTECTED": GOTO 3090
3040  IF ER = 6 THEN  PRINT "FILE NOT FOUND": GOTO 3090
```

Fig. 7-15. Disk drive diagnostic module listing.

```
3050  IF ER = 7 THEN  PRINT "VOLUMN MISMATCH": GOTO
      3090
3060  IF ER = 8 THEN  PRINT "I/O ERROR - TRY A NEW
      DISKETTE"
3070  IF ER = 9 THEN  PRINT "OUT OF SPACE ON DISK - TRY
      AN EMPTY ONE": GOTO 3090
3080  IF ER = 11 THEN  PRINT "NO BUFFERS AVAILABLE -
      REBOOT SYSTEM": PRINT "AND RUN DIAGNOSTIC AGAIN"
3090  PRINT : PRINT "PRESS ESC TO CONTINUE":
      GOSUB 8000
3100  GOTO 2750
```

Fig. 7-15. (continued)

Line 2510 clears the screen and positions the cursor with the HOME command. Variable D$ is set equal to CHR$ (4). This is an Apple convention, in which all disk commands must be preceded by CHR$ (4). You will notice the use of this convention throughout the Disk Drive Diagnostic. Line 2510 also sets variables H$ and L$ equal to nulls, clearing them from any values they might hold from previous tests in other modules. Variables H$ and L$ will store the high values and the low values to be written to and read from the disk throughout this test.

```
2510 HOME :D$ = CHR$ (4) :H$ = " ":L$ = " "
```

Line 2515 prints the diagnostic title on the screen.

```
2515 PRINT TAB (10)"DISK DRIVE DIAGNOSTIC": PRINT
```

Line 2520 begins a FOR-NEXT loop which will increment up to 239, which is the number of bytes contained in one Apple II disk sector. This loop serves to fill variable H$ with 239 high values, and variable L$ with 239 low values.

```
2520 FOR I = 1 TO 239
```

Line 2530 sets variable H$ equal to H$ plus CHR$ (64). CHR$ (64) is 1000000. L$ is also set equal to L$ plus CHR$ (63), which is 0111111. These two character strings will ensure that one ON bit and one OFF bit will be written to and read from each location on the disk. The loop then increments with the NEXT statement.

```
2530 H$ = H$ + CHR$ (64):L$ = L$ + CHR$ (63): NEXT I
```

Line 2540 checks the Exerciser flag. If the exerciser is in effect, as indicated by the flag being equal to 1, the program branches to line 2590, skipping over the user prompts to get right to the actual testing.

```
2540 IF EX = 1 THEN GOTO 2590
```

If the Exerciser does not control the program, line 2550 prompts the user to enter the slot number which contains the disk drive controller to be tested. This slot number would be between 0 and 7. The user's input is placed into variable S.

```
2550 PRINT "ENTER DISK DRIVE CONTROLLER": INPUT "SLOT
     NUMBER (0-7)-";S
```

Line 2560 checks whether the slot number in variable S is less than 0 or greater than 7, either of which would indicate an invalid entry. If there is an invalid entry in S, S is set equal to 6, which is the traditional location for the disk controller in an Apple II system.

```
2560 IF S < 0 OR S > 7 THEN S = 6
```

Line 2570 prompts the user to enter the drive number, because there can be two drives per controller. The user's response is stored in variable D.

```
2570 PRINT : INPUT "ENTER DRIVE NUMBER (1 OR 2)-";D:
     PRINT
```

Line 2580 checks whether D is less than 0 or greater than 2, either of which would indicate an invalid entry. If there is an invalid entry in D, D is set equal to 1.

```
2580 IF D < 1 OR D > 2 THEN D = 1
```

Line 2590 sets buffer B$ equal to H$, the high values variable. Then variable N$ is set equal to the words "HIGH VALUES", which will display on the screen for the duration of the disk testing with the high values.

```
2590 B$ = H$:N$ = "HIGH VALUES"
```

The ONERR statement in line 2600 is an error trap activated to catch drive errors during the test. If an error is detected, the program branches to line 3000 for appropriate error handling.

```
2600 ONERR GOTO 3000
```

Line 2605 begins the FOR-NEXT loop that is responsible for making two passes through the test, one for high values and one for low values.

```
2605 FOR I = 1 TO 2
```

Line 2610 prints a message on the screen to inform you that either high values or low values are being written to disk. The appropriate message is created with N$ as set in line 2590.

```
2610 PRINT : PRINT "WRITING ";N$;: "TO DISK"
```

Line 2620 opens a disk file called "TEST" at slot S, drive D, as was input by you earlier in this program.

```
2620 PRINT D$;"OPEN TEST,S";S;" ,D";D
```

Line 2630 prints D$, indicating that information will be written to the disk.

```
2630 PRINT D$;"WRITE TEST"
```

Line 2635 is the FOR-NEXT loop that increments to the number of sectors required to fill an Apple II diskette.

```
2635 FOR L = 1 TO 430
```

Line 2640 prints the contents of buffer B$. This causes the L loop to be activated, writing records to the disk to fill up 430 sectors.

```
2640 PRINT B$: NEXT L
```

When the loop is completed, and all the records are written, line 2650 closes the disk test file.

```
2650 PRINT D$;"CLOSE TEST"
```

Line 2670 switches output from the disk back to the screen with the PRINT D$ command. The message indicating that records are being read from the disk is displayed on the screen.

```
2670 PRINT : PRINT D$: PRINT "READING ";N$;" FROM
     DISK": PRINT
```

Line 2675 re-opens the file called "TEST" in which the records were written.

```
2675 PRINT D$;"OPEN TEST,S";S;",D";D
```

Line 2680 indicates that the program is reading, rather than writing.

```
2680 PRINT D$;"READ TEST"
```

Line 2685 begins the FOR-NEXT loop that loops for the number of records that have been written to the disk.

```
2685 FOR L = 1 TO 430
```

Line 2690 reads the records into variable B1$.

```
2690 INPUT B1$
```

Line 2700 compares the contents of B1$ to those of B$. In other words, the records being read are compared to the records which were written, to check whether they match. If they do not match, an error message is printed, and the error counter is incremented.

```
2700 IF B1$ <> B$ THEN PRINT D$: PRINT "ERROR
     READING ";N$:E = E + 1
```

Line 2710 increments the L loop for reading the records.

```
2710 NEXT L
```

Line 2720 closes the "TEST" file.

```
2720 PRINT D$;"CLOSE TEST"
```

Line 2730 deletes the file from the diskette.

```
2730 PRINT D$;"DELETE TEST"
```

Line 2740 sets buffer B$ equal to the low values variable L$, and N$ is set equal to the words "LOW VALUES." The I loop is incremented with the NEXT I statement. This causes the write and read routine to be repeated for low values.

```
2740 B$ = L$:N$ = :"LOW VALUES": NEXT I
```

Line 2750 uses PRINT D$ to switch output control back to the screen. A message is displayed to indicate the number of disk errors encountered during the test.

```
2750 PRINT D$: PRINT : PRINT "THERE WERE ";E;" ERRORS
     ENCOUNTERED": PRINT "DURING TEST"
```

Line 2760 checks whether the Exerciser routine is in control of the program. If EX is equal to 1, then the drive portion of the Exerciser is completed. The POKE statement halts the error trapping method used in this module, and then the program returns to the Exerciser module.

```
2760 IF EX = 1 THEN POKE 216,0: RETURN
```

If the Exerciser is not in control, the program proceeds to line 2770, which prompts the user to press ESC to end the test. The program branches to the Get-a-key subroutine at line 8000 to get your response. The POKE statement halts the error trapping method used in this module.

Finally, the GOTO statement returns the program to line 50 to display the System Diagnostic Main Menu.

```
2770 PRINT : PRINT "PRESS ESC TO END TEST":
     GOSUB 8000: POKE 216,0: GOTO 50
```

Lines 3000 through 3100 comprise the disk error checking routine. Line 3000 PEEKs into memory location 222, which is where error codes are maintained by the Apple II system. The contents of this location are placed into variable ER.

```
3000 ER = PEEK (222)
```

Line 3002 checks if the error code returned was a 5, indicating an END-OF-FILE condition. If the error code is a 5, the program branches to line 2720, which continues the Read test.

```
3002 IF ER = 5 THEN GOTO 2720
```

Line 3005 closes the "TEST" file. The PR instruction returns output from the disk drive to the monitor. The IN instruction returns input from the disk drive to the keyboard.

```
3005 PRINT D$;"CLOSE TEST": PR# 0: IN# 0
```

Line 3010 prints the first part of the error message.

```
3010 PRINT : PRINT TAB( 5) "A FATAL ERROR WAS ENCOUNTERED!" : PRINT
```

Line 3020 prints the number of the error code.

```
3020 PRINT "ERROR CODE:";ER: PRINT
```

Lines 3030 through 3080 translate the error code number into an informational error message which informs the user of what the problem is, and in some cases, what the remedy might be.

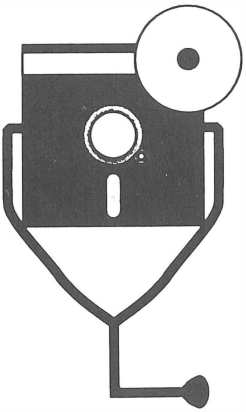
```
3030 IF ER = 4 THEN PRINT "THE DISKETTE IS WRITE-PROTECTED": GOTO 3090
3040 IF ER = 6 THEN PRINT "FILE NOT FOUND": GOTO 3090
3050 IF ER = 7 THEN PRINT "VOLUMN MISMATCH": GOTO 3090
3060 IF ER = 8 THEN PRINT "I/O ERROR - TRY A NEW DISKETTE"
3070 IF ER = 9 THEN PRINT "OUT OF SPACE ON DISK - TRY AN EMPTY ONE": GOTO 3090
3080 IF ER = 11 THEN PRINT "NO BUFFERS AVAILABLE - REBOOT SYSTEM": PRINT "AND RUN
     DIAGNOSTIC AGAIN"
```

Line 3090 prompts you to press any key to continue. The program then branches to the Get-a-key subroutine to wait for you to press a key.

```
3090 PRINT : PRINT "PRESS ANY KEY TO CONTINUE": GOSUB 8000
```

Line 3100 branches the program back to line 2750, which displays a final message about the number of errors encountered during the Disk Drive diagnostic, and then continues to end the test and return to the System Diagnostic Main Menu.

```
3100 GOTO 2750
```



8

The Serial Comm Interface

Using the proper equipment, the Apple II systems can talk to many other devices: a modem, printer, mouse, even another computer. Data and information are sent to and received from these peripherals. Communicating among different devices requires the services of a serial communication interface.

DESCRIPTION AND FUNCTION OF THE SERIAL COMM INTERFACE

The serial communication interface consists of three components:

- a serial communication circuit
- a serial port
- an interface cable

A fourth component is also necessary to establish communications: the device with which the Apple is to communicate.

The Serial Communication Interface Circuit

The serial communication interface circuit consists of a separate board on the Apple II Plus and the IIe, and is built into the motherboard on the Apple IIc. This

circuit is responsible for supplying, receiving, and interpreting various voltage levels, or analog signals, in order to control commands such as SEND DATA and DATA SET READY. Other voltage levels provide the signals of the sent or received data.

The Serial Communication Interface Port

The serial interface port on the Apple II Plus and IIe is a RS-232 25-pin connector located at the rear of the Apple II system unit. Although there are 25 pins, only 5 of these pins are used for serial communication signals, with pin assignments as follows:

2 - TRANSMIT DATA (output)
3 - READ DATA (input)
6 - DATA SET READY (input)
7 - POWER AND SIGNAL COMMON (ground)
20 - DATA TERMINAL READY (output)

The 20 remaining connector pins are not used.

The serial interface port on the Apple IIc is more efficient, using an actual RS-232 5-pin connector. Its pin assignments are as follows:

1 - DATA TERMINAL READY (output)
2 - TRANSMIT DATA (output)
3 - POWER AND SIGNAL COMMON (ground)
4 - READ DATA (input)
5 - DATA SET READY (input)

The Serial Communication Interface Cable

Serial communication with the outside world is achieved by means of an interface cable running from the Apple serial communication port to the other device with which communication is to take place (Fig. 8-1). It is through this cable that the appropriate signals and data are passed to and from the other device.

Serial Communication Devices

The Apple II serial communication interface is used to communicate with devices such as the modem, printer, or another computer.

Interfacing with a Modem. The modem is a device widely used for serial communication (Fig. 8-2). It allows transmission and reception of computer data using telephone lines.

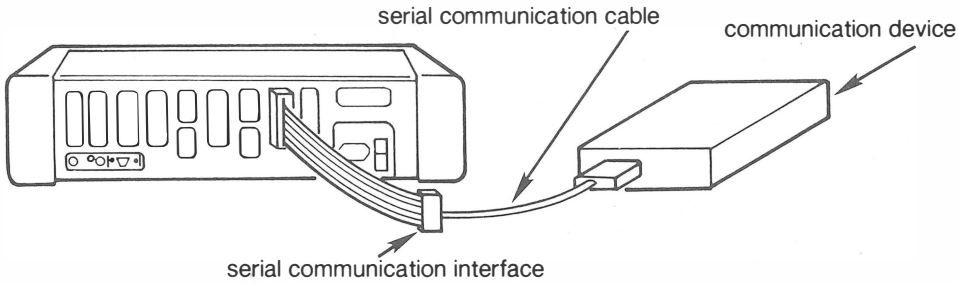


Fig. 8-1. Serial communication interface cable.

The word modem is derived from the term MODulator-DEModulator. The computer produces data and control signals in the form of electrical impulses of +5 volts, or 0 volts. These impulses are either ON or OFF, respectively. When the signal is ON, it is interpreted as a 1 bit. When the signal if OFF, it is interpreted as a 0 bit. All the information existing in computer programs and data is held in a series of 1s and 0s, or digital signals.

The modem is the intermediary for sending information from one computer to another. It converts the originating computer's digital signal into sound waves. These sound waves, or analog signals, allow for the transmission of information over telephone lines. This part of the conversion is referred to as modulation.

The modem at the receiving end converts the analog signal back into a digital signal. This part of the conversion is referred to as demodulation. Since the information is returned to 1s and 0s, it can be successfully interpreted by the computer connected to the receiving modem (Fig. 8-3).

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, including communication speed, data size, start bits, and other such parameters which will be discussed later in this chapter.

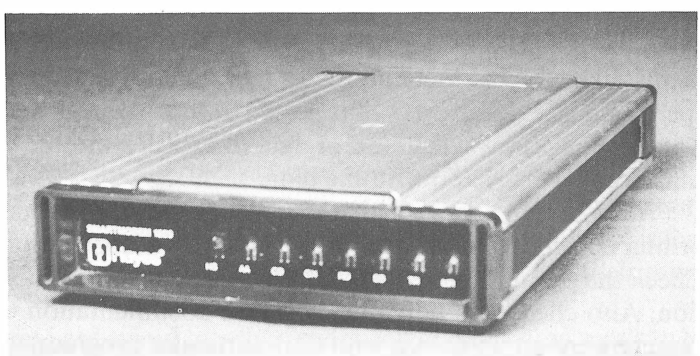


Fig. 8-2. The modem.

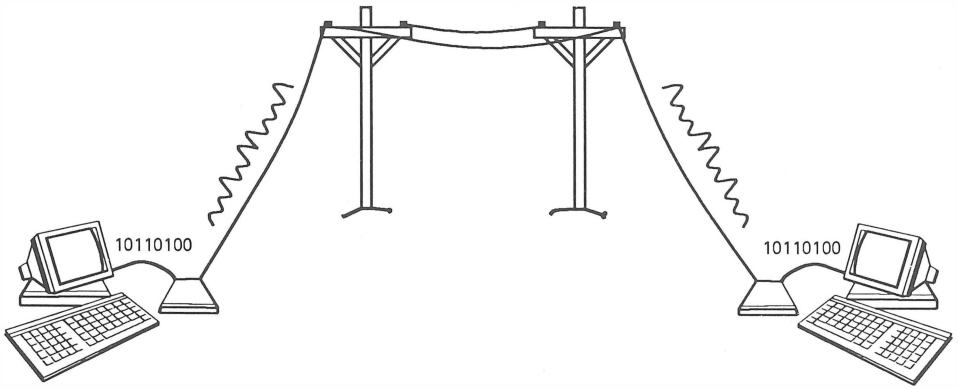


Fig. 8-3. Modem communication.

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 network 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 Apple II can be linked into a network's mainframe computer. The network service can then be used to research a wide range of information, as well as to communicate with other users. Online services can also provide you with banking, shopping, and airline reservation services through your Apple II serial communication interface and modem (Fig. 8-4).

The Apple II systems connect to external modems. 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 Apple II systems primarily use serial printers, meaning the Apple printer interface is a serial port. The Apple II systems have two serial ports: one generally used for the printer, and the other used for communication through a modem. Some serial communication boards can be used for both, and are controlled by exchanging the device interface cable and changing a switch setting on the board. It is possible to have as many serial ports as there are available expansion slots.

Just like with the modem, the printer must be configured with the same data speed and size parameters that are set within the Apple II. This is because the printer must know what kind of data to expect from the computer, and then it must know what to do with the data.

Printer parameters are either set through switches within the printer or on the printer controller board, and/or set through the application software being used. Check the printer's user manual to see if any switches must be set upon installation. Also check the individual software documentation to see how printer parameters are set up to run with that particular program.

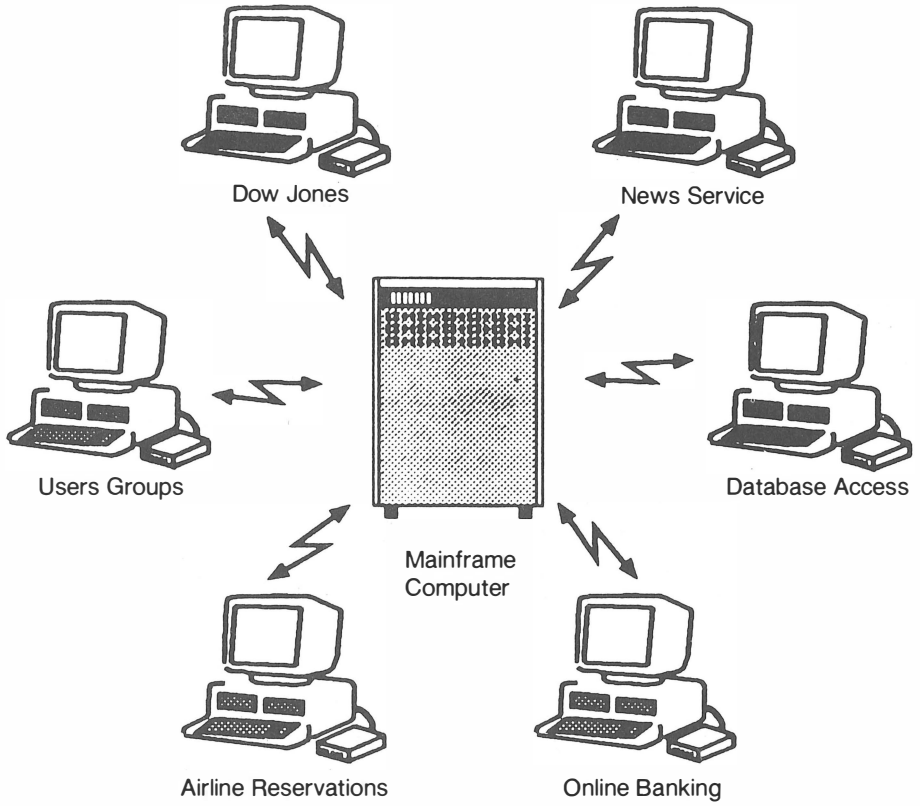


Fig. 8-4. Online services.

The printer is typically attached directly to the computer with an interface cable. 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 a local printer.

Interfacing with Other Devices. Other devices that connect to the computer through the serial communication interface include the mouse, plotter, digitizer tablet, system clock, and light pen.

Interfacing Directly with Another Computer. If your Apple II resides in an environment in which computers are less than 50 feet apart, these computers can be *hard-wired* together with an interface cable to connect them together directly via their serial communication ports. This connection lets them exchange data just as they would if they were hooked up through modems. The difference is that telephone lines are not needed in a hard-wire connection because of their proximity.

When computers are hard-wired together, the speed of data being sent can be

much faster. 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 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. These factors include:

- communication software
- data speed (baud rate)
- data size
- start and stop bits
- parity
- directionality
- emulation
- protocols
- interface port

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.

Most modems come supplied with communication software. The same is true for the mouse and other serial communication devices you might acquire.

Some available serial communication boards do not require separate software. DIP-switches let you set communication parameters, and special programs built into the board's ROMs provide communication command sets to perform all necessary communication software functions.

Communication Parameters. Since data is sent one bit at a time in serial communication, certain parameters must be specified when first configuring the communication software. These parameters pertain to data transmission speed, the size of the data characters being sent, whether start bits and stop bits are employed, and whether parity error checking is to take place.

The communication configuration serves to coordinate the communication and ensure that the data transfer is successful and accurate. The particular values specified in these parameters depend on the device with which the computer is

communicating. The modem documentation describes how to set up the parameters. Online service user documentation details the specific parameters needed to communicate with their computers.

The printer, modem, or other device with which the Apple II is going to communicate must have the communication interface variables set identically on their end in order that this communication link can 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 running communication software for the first time, access the setup mode for these communication parameters. Once they are set, these options are usually stored on the program disk so that they are automatically implemented each time the program is used.

Baud Rate. Baud rate 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, although not always, the number of bits that can be transmitted in a second. The differences in baud rates can be likened 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 at a much faster rate 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. A typical communication interface for the Apple II has a baud rate of 300 or 1200.

Higher speed communication in the range of 4800 and 9600 baud is available, but is used primarily for communicating with devices that are hard-wired to a system through an interface cable. A hard-wire interface does not require an intermediary such as a modem in order to establish communication. A printer or computer that is hard-wired to another computer is able to operate at the faster speeds. There are modems available that can operate at such high speeds, but they can be prohibitively expensive to the small business or home user.

Data Size. Data size refers to the size of the data being sent: how many bits make up a standard character or word. The Apple data character consists of eight bits. The communication software needs to know the character length or word length of the data being sent, to be able to send and receive data correctly (Fig. 8-5).

Start Bits and Stop Bits. In serial communication, the software generates 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

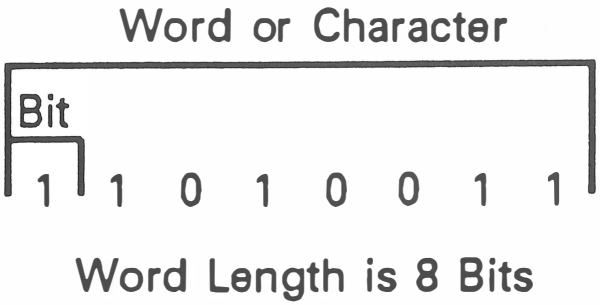


Fig. 8-5. Bits, characters, words.

both the start bit and stop bit, then ten bits are actually sent for each data character.

The start bit signals the beginning of a character, and flags the attention of the receiving device. It in essence informs the device to expect seven or eight more bits of data for processing. Likewise, the stop bit signifies the end of a character.

Both the transmitter and receiver in the communication line need to know whether start bits and stop bits are being employed (Fig. 8-6).

Parity. Parity 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.

For example, if the system is configured for 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 was lost during transmission. When parity is not achieved, this indicates an error in transmission. In this case, the communication software gives a prompt to try the data transmission again, or it restarts automatically.

Directionality. Depending on the device with which the Apple II is communicating, the data might be sent in one direction only, or the 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. On the other hand, when two computers are linked up through modems, the data is said to be bidirectional, since data can be sent to and received by each system.

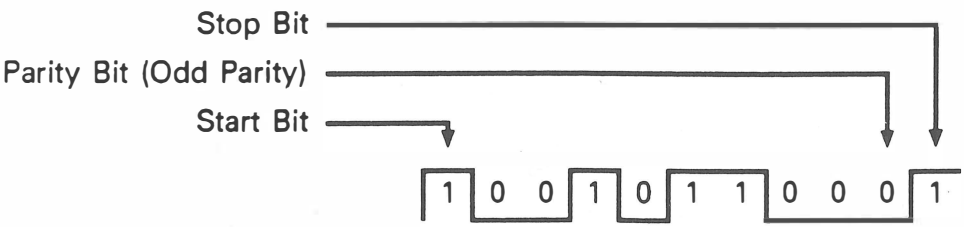


Fig. 8-6. Stop, parity, and start bits.

Whether the data is unidirectional or bidirectional depends on the type of device the data is sending to as well as the software being used for the communication transfer. Some system configuration parameters in the support software might need to be set. There might also be switches to set on the serial communication interface board. This would be detailed in the device documentation.

Emulation. Different types of computers have different keyboard control sequences, keyboard mapping, screen format display, text handling, and special functions. These aspects of unique computer design can hinder effective communication between two different computers. To solve this problem, an emulation program in the communication software can disguise the Apple II and make it simulate another computer of a different design. The DEC VT100 is a common emulation mode, available with communication software, in which the Apple II can emulate the VT100 terminal. When both types of computers are set up in the same emulation mode, 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 configuration parameters. The XON/XOFF protocol is optional. 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 as needed to prevent the communication buffers from overflowing, and then turns it on again to make sure that data is completely transferred.

For example, suppose the Apple II has a 128-byte communication buffer. In the process of receiving a file that might be 4,000 bytes long, for example, the communication buffer will be quickly filled. When this happens, the Apple II sends a suspend command to the other system, temporarily stopping data transfer. This suspension is the XOFF command. When the buffer is emptied through computer processing, such as displaying characters on the screen, the Apple II sends a resume command, which is XON. The transfer then continues where it left off. This process continues throughout the file in 128-byte chunks until all 4,000 bytes are successfully transferred.

Another protocol used by modems to establish communication is called the *handshaking* protocol. A series of signals is used to establish modem communication. The following is a listing of the handshaking protocol, including the procedure and data signals used to establish and maintain communication. Notice that some of the signals, such as Carrier Detect, Request to Send, and Clear to Send, are used only by the modem, and do not go through the serial communication interface itself.

1) Load and run the communication program. The modem issues the TR signal (Terminal Ready).
2) Dial the phone to call the other modem and computer system. The

modem switches into the data mode and issues the DSR signal (Data Set Ready).

3) The other system answers and responds. The CD (Carrier Detect) is issued, and the corresponding light is illuminated on the modem panel.
4) Press a key to begin sending a character code to the other system. The RTS (Request to Send) signal is sent from the computer to the modem, and the appropriate light is illuminated.
5) The modem transmits a signal back to the computer to indicate that it is prepared to accept data for transmission. The CTS (Clear to Send) signal is received.
6) The data is sent. The SD (Send Data) light is illuminated on the modem.

Interface Port. The serial communication interface board has a connector into which a cable is attached. This connection is where the functional signals and data are passed across the interface, one bit at a time, between the two communicating devices.

The serial communication interface transmits seven or eight bits of data sequentially, in serial form. These signals ensure that both machines are synchronized with one another, and are prepared to receive or send data. They also help detect breaks in the communication line and other communication errors, and return error signals to the programs when necessary.

The RS-232 serial communication standard determines what these functional signals consist of, and which cable wires are assigned to carry which types of signal or data.

SERIAL COMM INTERFACE PREVENTIVE MAINTENANCE

Preventive maintenance of the serial communication interface consists of keeping the serial communication circuit, port, cable, and device clean and free of dust.

Vacuum dust off the serial communication board (or circuit on the IIc motherboard) periodically. Check the board's edge connector for oxidation. Make sure board is seated securely in its slot.

Check the serial communication interface port and cable for oxidation, and clean it off if present. Make sure the cable is firmly connected.

SERIAL COMM TROUBLESHOOTING AND REPAIR GUIDELINES

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 two basic serial communication problems are that communication is not established, and that communication results are garbled or incorrect in some way.

Problem: Communication Cannot Be Established

The inability to establish communication might manifest itself in the inability to dial up, to make the connection after dialing, or to detect the carrier after connecting.

Solutions:

- Check that all connections are properly made and firmly seated. There are three interface connections: the interface board's ribbon cable connection on the board, the RS-232 cable connection to the ribbon cable, and the RS-232 cable connection to the communication device.
- 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 RS-232 cable.
- Swap your serial communication interface board with another Apple II system. (With the Apple IIc, this means swapping the entire motherboard.) If communications can be established with your board on the other system, the problem is with the serial communication port on your Apple II. Refer to "Problem: Oxidation Needs to be Removed" in Chapter 3, for instructions on cleaning the edge connector.

  If the board still cannot establish communications on the other system, the problem lies in the modem itself. Take the modem in for repair.
- 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?
- If possible, check the device with which you're trying to communicate. If it's another Apple II, run the diagnostic program on it to see whether its communication interface is working properly. If it's an online service, call their help line for further technical assistance.
- 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, expect different signals,

or have signals tied together. If this is the case, either have the cable modified to match your required pin configuration, or purchase another one with the correct pinouts. Consult the cable manufacturer for further information.

- Most modems provide additional assistance with troubleshooting, through its row of red indicator lights across its front status panel (Fig. 8-7). If your modem includes such an indicator panel, check that the signals are correct. For instance, when the modem is properly cabled into the Apple II, and the communication software is being run, the lights over TR (Terminal Ready) and DSR (Data Set Ready) should be illuminated. These lights indicate that the computer and modem are ready and that the interface signals are turned on and ready to communicate with one another. If these indicator lights are not on, there may 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 lit. When a key is pressed, these lights illuminate to show that the data for the key code are indeed being sent.

- The modem might also have a self-test mode (check the modem documentation). This self-test mode loops a data pattern back through the modem. It uses a pattern of characters from the computer, sends the pattern through the modem back to itself, and then returns them to the computer. This internal loopback test checks the modem to make sure that sending and receiving is indeed taking place. Use this self-test mode if the previous steps did not establish communication. If a problem is diagnosed through the modem self-test, refer to the modem documentation for further action.

- A carrier is a tone that the modem expects before it can send data. It's like saying "Hello" after the phone rings and you have picked up the receiver. After saying "Hello," communication can be established, and information from the caller can be transmitted.

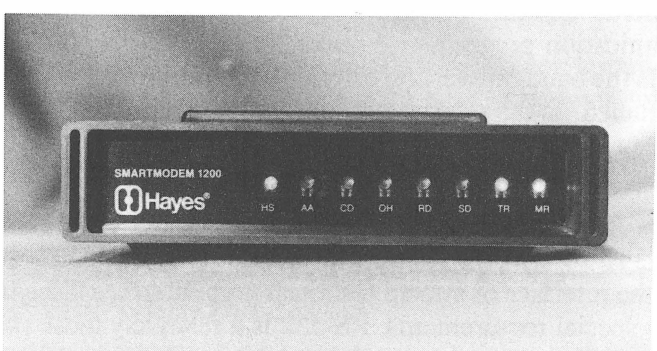


Fig. 8-7. Modem status panel.

This carrier can be heard when using a smart modem that dials the phone automatically. First you would hear the phone 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 indicate a wrong number, a busy signal, or a problem at the other end.

Problem: Communication Results Are Garbled

Garbled, confused, or miscommunicated 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. Access the communication software parameter setup mode again, and check all the settings.

A mixture of garbled characters and good characters indicates a poor telephone connection. For example, the other computer might send the word "login", but "loxxn" is received. In this case, 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 your configuration. Most printers can communicate using either a serial or 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, character length, parity, and so forth. This is set up with a DIP-switch on the printer interface board. The printer documentation details how to make the necessary settings.

Problem: Serial Comm Interface Needs to Be Serviced

If the solutions described above fail to solve the serial communication problem, and the system requires servicing, first call the Apple II service center and speak with the technician. Describe the computer configuration, type of modem, and the problem. Explain the troubleshooting measures you have already taken,

and the results. Ask if the technician can suggest any additional solutions over the phone.

If the unit still needs to be taken in, ask the technician how much of the system should be brought. The technician will probably need the system unit, interface, modem, and cable.

When speaking with the technician, provide all available information. Find out if the technician has any ideas on what the problem might be, and how long and how much it might cost to fix. Be sure to get a copy of the work order.

Before you pay and accept your system, have the repair technician demonstrate that the problem has actually been repaired. Then pack the system up and carefully take it home. Connect everything together, run the Serial Communication Interface Diagnostic module to make sure it's working. Then refer to Chapter 9, "The Exerciser," for instructions on burning in the system after repair.

RUNNING THE SERIAL COMM INTERFACE DIAGNOSTIC MODULE

The Serial Communication Diagnostic module checks that the interface and the interface address are both working properly. It also tests whether the system is receiving the same data that it is sending through the communication loop.

To run the Serial Communication Diagnostic module, first connect the RS-232 serial communications interface cable to the serial communications ribbon cable port coming out of the system unit. Then take the free end of the RS-232 cable, and instead of plugging it into the modem or other communicaiton device, attach a wire between pins 2 and 3 (Fig. 8-8).

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

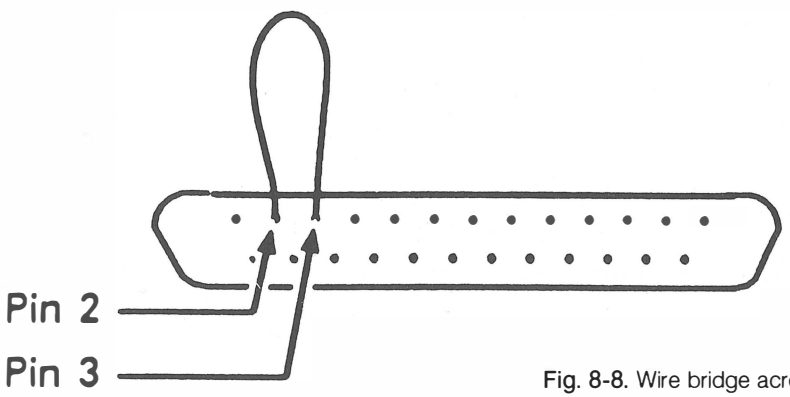


Fig. 8-8. Wire bridge across pins 2 and 3.

back and received. In other words, any character entered at the Apple II keyboard is sent out to the device, immediately received in through these pins and sent back to the computer. The character will be echoed back and displayed on the screen (Fig. 8-9).

There are test connectors available that already have pins 2 and 3 wired together. Called a 25-pin serial loopback connector, it can be purchased at most computer or electronic supply stores. The loopback connector fits directly onto

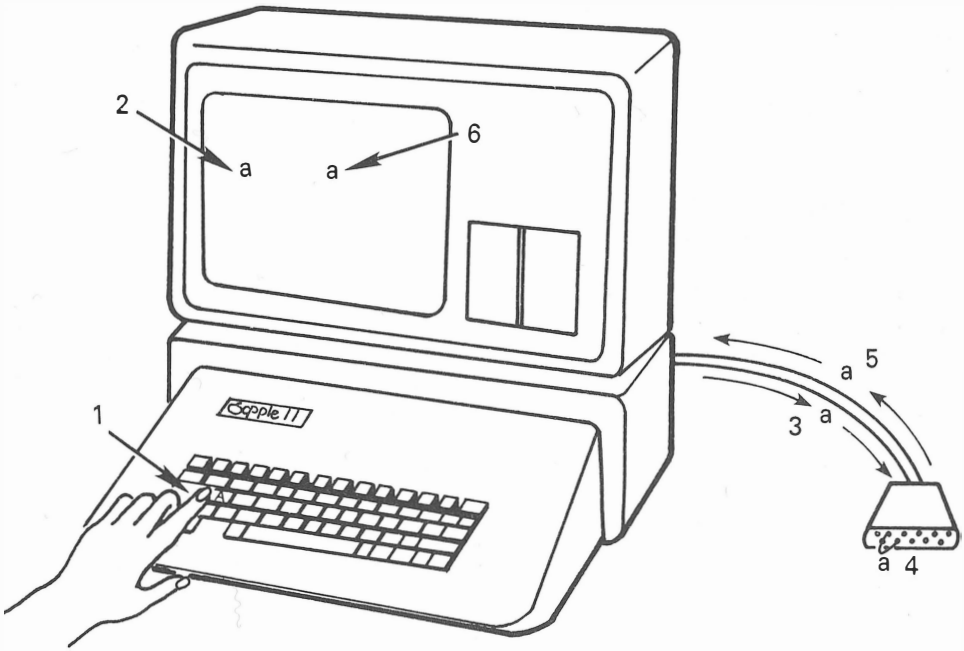


Fig. 8-9. Character loopback.

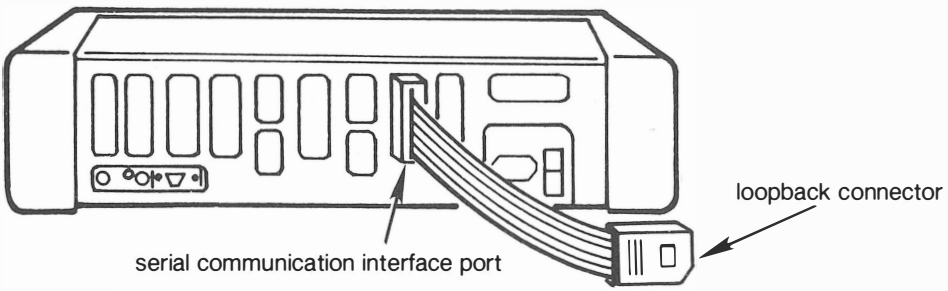


Fig. 8-10. Loopback connector.

the end of the RS-232 cable, and accomplishes the same loopback effect (Fig. 8-10).

Once the loopback is set up on the cable pins, follow the instructions provided in "Running the Diagnostic Program" in Chapter 2. Then press S to initiate the Serial Communication Diagnostic module. The Serial Communication Test instructions will be displayed (Fig. 8-11). At this point you are ready to run the Serial Communication Diagnostic, consisting of the Character Loopback Test.

The Character Loopback Test displays a prompt to enter a character on the keyboard. This character is sent through the communication loop created by the loopback wire. The loopback causes the character to be sent back immediately and received as output displayed on the monitor screen. This checks the functions of the serial communication interface by both transmitting and receiving data. The ASCII value of the character is also displayed on the screen (Fig. 8-12).

This test is particularly useful for testing function keys such as the carriage return, tab key, and cursor arrows. Through the display of ASCII values on the screen, you can check whether the communication interface is sending the correct codes for function keys as well as character keys.

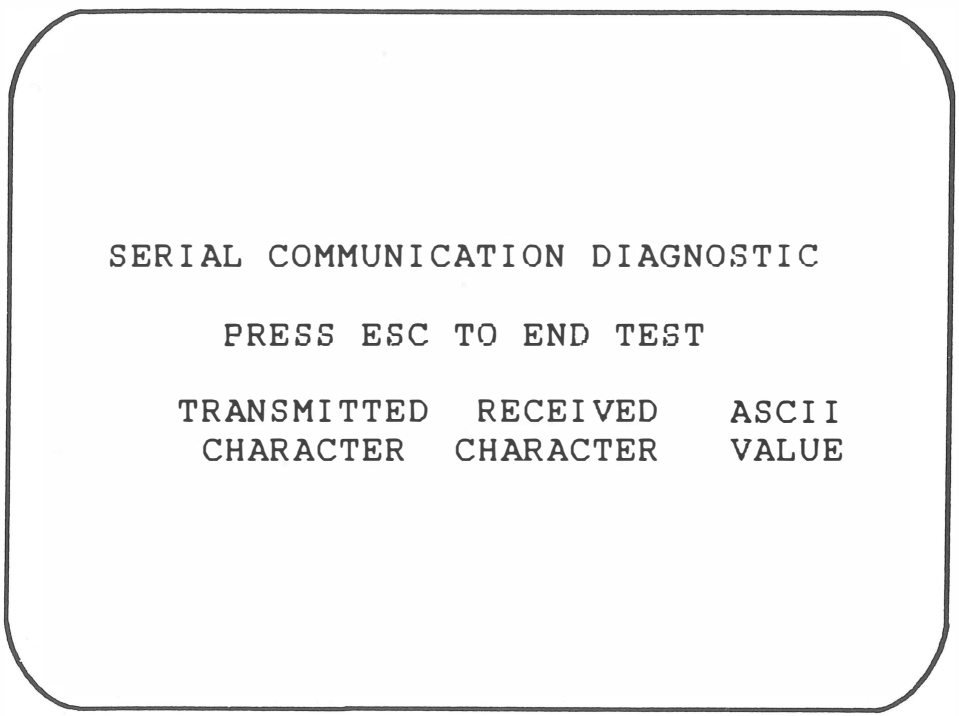


Fig. 8-11. Serial communication diagnostic test instructions.

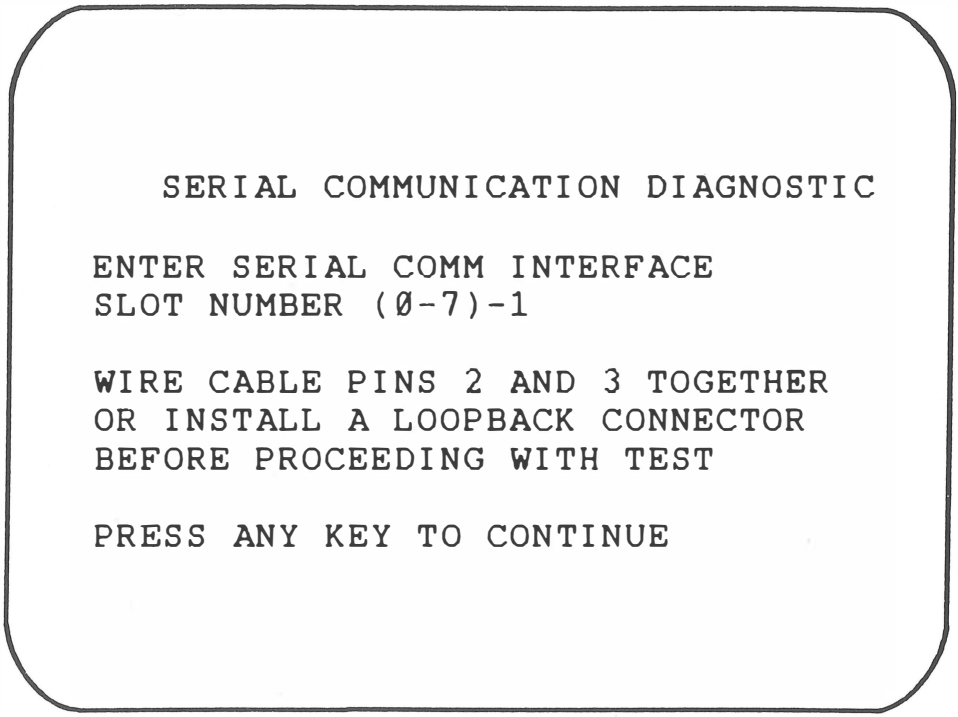


Fig. 8-12. Serial communication diagnostic test results.

HOW THE SERIAL COMM DIAGNOSTIC MODULE WORKS

Pressing S from the System Diagnostic Main Menu causes the program to branch to line 3500, where the Serial Communication Diagnostic module begins with a REMARK statement (Fig. 8-13).

```
3500 REM SERIAL COMMUNICATION DIAGNOSTIC MODULE
```

Line 3510 clears the screen, positions the cursor, and displays the diagnostic title on the screen.

```
3510 HOME : PRINT TAB( 7)"SERIAL COMMUNICATION DIAGNOSTIC" : PRINT
```

Line 3512 prompts you to enter the number of the slot which contains the serial communication interface board. This value is stored in variable SL. If the value of SL is less than 0 or greater than 7, then it is an invalid entry. If this is the case, SL is set to 2, the traditional location for the serial communication interface.

```
3500  REM  SERIAL COMMUNICATION DIAGNOSTIC MODULE
3510  HOME : PRINT  TAB( 7)"SERIAL COMMUNICATION
      DIAGNOSTIC" : PRINT
3512  PRINT "ENTER SERIAL COMM INTERFACE": INPUT "SLOT
      NUMBER (0-7)-";SL: IF SL < 0 OR SL > 7
      THEN SL = 2
3513  PRINT : PRINT "WIRE PINS 2 AND 3 TOGETHER":
      PRINT "OR INSTALL A LOOPBACK CONNECTOR":
      PRINT "BEFORE PROCEEDING WITH TEST": PRINT :
      PRINT "PRESS ANY KEY TO CONTINUE": GOSUB 8000
3515  S = 0: GOTO 3532
3520  B$ = ""
3530  GET A$: IF  ASC (A$) = 27 THEN GOTO 50
3532  HOME : PRINT  TAB( 7)"SERIAL COMMUNICATION
      DIAGNOSTIC": PRINT: PRINT  TAB( 9)"PRESS ESC TO
      END TEST": PRINT: PRINT  TAB( 7)"TRANSMITTED
      RECEIVED  ASCII"
3534  PRINT  TAB( 8)"CHARACTER  CHARACTER  VALUE":
      PRINT
3536  IF S = 1 THEN  GOTO 3540
3538  S = 1
3540  PR# SL
3550  PRINT  TAB( 12)A$;
3560  IN# SL
3570  GET B$
3580  IF B$ = "" THEN  GOTO 3520
3590  PR# 0
3600  PRINT  TAB( 24)B$; TAB( 32) ASC (B$): GOTO 3520
```

Fig. 8-13. Serial communication diagnostic module listing.

```
3512 PRINT "ENTER SERIAL COMM INTERFACE": INPUT "SLOT
     NUMBER (0-7)-";SL: IF SL < 0 OR SL > 7
     THEN SL = 2
```

Line 3513 prompts you to wire pins 2 and 3 together on the interface cable, or to attach a loopback connector. Then you are prompted to press any key to continue. The program branches to the Get-a-key subroutine at line 8000 to receive your character input.

```
3513 PRINT : PRINT "WIRE PINS 2 AND 3 TOGETHER":
     PRINT "OR INSTALL A LOOPBACK CONNECTOR":
     PRINT "BEFORE PROCEEDING WITH TEST": PRINT :
     PRINT "PRESS ANY KEY TO CONTINUE": GOSUB 8000
```

Line 3515 sets variable S equal to 0, then branches to line 3532, so that the screen test display is not repainted for each cycle of the test.

```
3515 S = 0: GOTO 3532
```

Line 3520 sets variable B$ equal to null in order to clear it of any values stored there from previous tests.

```
3520 B$ = ""
```

Line 3530 checks to see if the value stored in A$ from the Get-a-key subroutine is the ESC key: ASCII value 27. If it is, the Serial Communication Diagnostic is halted, and the program branches to line 50 to display the System Diagnostic Main Menu.

```
3530 GET A$: IF ASC (A$) = 27 THEN GOTO 50
```

If it is not the ESC key, the test display headings are printed on the screen.

```
3532 HOME : PRINT TAB( 7)"SERIAL COMMUNICATION DIAGNOSTIC": PRINT: PRINT TAB( 9)"PRESS
     ESC TO END TEST": PRINT: PRINT TAB( 7)"TRANSMITTED RECEIVED ASCII"
3534 PRINT TAB( 8)"CHARACTER CHARACTER VALUE": PRINT
```

Line 3536 checks whether S is equal to 1. If it is, the program branches to line 3540.

```
3536 IF S = 1 THEN GOTO 3540
```

Line 3538 sets S equal to 1, so that on subsequent passes through the test, certain lines need not be repeated.

```
3538 S = 1
```

Line 3540 switches output control to the serial communication interface.

```
3540 PR# SL
```

Line 3550 prints the character that the user pressed, which is stored in A$. The character is displayed under the TRANSMITTED heading.

```
3550 PRINT TAB( 12)A$;
```

Line 3560 switches input control to the serial communication interface board.

```
3560 IN# SL
```

Line 3570 reads the character that was transmitted and received back into variable B$.

```
3570 GET B$
```

Line 3580 checks whether B$ is a null. If it is, the program branches back to line 3520 and checks for another transmitted and received key.

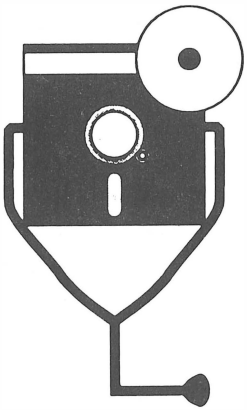
```
3580 IF B$ = "" THEN GOTO 3520
```

If B$ is not a null, line 3590 resets output control to the screen.

```
3590 PR# 0
```

Line 3600 prints the character stored in B$ under the RECEIVED heading on the screen, and the ASCII value of the received character is printed under the ASCII VALUE heading. Once this is done, the program branches back to line 3520 to let you press another key to be tested.

```
3600 PRINT TAB( 24)B$; TAB( 32) ASC (B$): GOTO 3520
```



9

Post-Repair Test and Burn-In

Chapters 3 through 8 have each covered individual components of the Apple II system. We have examined and treated problems that can occur within a particular subsystem. Chapter 9 provides a diagnostic module that tests not just a single subsystem, but rather exercises the Apple II system as a whole, testing the entire integrated computer for problems.

If something goes wrong with your computer and you repair it or take it in for professional servicing, check it out thoroughly after the repair is completed. This chapter describes and provides methods for testing the repaired system to make sure everything works as it should.

Have the service technician describe what was done, explain why it was done, and demonstrate that the problem is now fixed as a result. However, keep in mind that even while checking the components at the repair shop, not all system functions can be checked. The technician will probably run through just 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 taken only the system unit in for servicing, leaving the monitor, printer, disk drive, and modem behind. But the computer is a system, consisting of a number of devices and peripherals interacting to run applications. Because of this, when there is a problem in one device, it can actually manifest itself in another device.

So even after the technician has demonstrated that the system unit is working fine, the complete status will be unknown until you have the entire system assembled in its usual configuration.

The diagnostic module covered in this chapter is a program that thoroughly exercises the Apple II when it's new, when its configuration changes, or when it returns from servicing. This Exerciser module checks various components of the Apple II system and leaves it running in a continuous mode so that it can burn in for several hours, even overnight.

Burn-in is routinely practiced as an important segment of industrial manufacturing and product testing. Burn-in thoroughly tests and exercises the Apple II system and should be done for 12 to 24 hours. If there is a problem with your computer at this point, it's better to discover it during burn-in rather than while entering and processing valuable data.

TESTING THE APPLE II SYSTEM

There are three aspects to testing the Apple II system when it is new, or when it returns from repair:

- checking for the original problem
- running the subsystem diagnostic module
- burning in the system as a whole with the System Exerciser

This section describes these three aspects of testing, and suggests what should be done if testing reveals a problem.

Adhering to this testing sequence and following these suggestions will ensure system integrity. It will also help prevent the loss of any programs, data, and time when actual work is performed on the recently repaired system.

Checking for the Original Problem

Before accepting and paying for a repair, have the repair technician demonstrate that the problem has been repaired. While still at the shop, try the system out for yourself. This can save a lot of time, especially if the problem has not been fixed after all, or if there was an original miscommunication of the problem.

After the unit is back in place, securely cable the entire system together. Make sure the entire system is operating as normal.

Recreate the situation that originally caused the problem. This might consist of running a particular software program, or going through a specific sequence of events in a certain environment. If the problem does not occur now, then the problem has most certainly been repaired.

Running the Subsystem Diagnostic

If there are no problems at this point, load the System Diagnostic Program as described in Chapter 2. Run the particular module that pertains to the subsystem that was just repaired.

For example, if the printer has recently been repaired, run the Printer Diagnostic module. If the keyboard had a malfunction, run the Keyboard Diagnostic module. If the system unit has been serviced, run each module with various combinations to test all aspects of the system unit's functionality.

Burning In the System

Refer to "Running the System Exerciser" later in this chapter, and run the Exerciser module. This will thoroughly burn in the new or recently repaired computer as an entire system working together. Run all functions: the monitor, printer, and disk drives for at least three cycles. Let the monitor and disk drive portions of the Exerciser run for about 12 hours. Occasionally observe the results of the burn-in to see whether any errors have occurred.

When checking for the original problem, run the appropriate subsystem diagnostic module, and thoroughly run the Exerciser. If no old or new problems occur, you can be assured that your entire system is in an optimum state, and that the computer can be trusted with programs and data.

Responding to System Problems

However, if problems are found in this system checkout sequence, refer to the troubleshooting steps in the chapter pertaining to the malfunctioning subsystem. If that fails to solve the system problem, call the service technician and explain your discoveries. Chances are you will be invited to bring the system back right away to have the problem actually fixed at no extra charge.

FUNCTION OF THE EXERCISER MODULE

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 will be displayed. This menu prompts you to specify the devices to be tested, the monitor, printer, and/or disk drives. Any combination of devices can be specifed for inclusion in or exclusion 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 immediately begins to perform the continuous test on the specified devices. This means the Exerciser could be run for hours, even overnight if desired. The Exerciser runs nonstop until it encounters an error, or until it is stopped with CTRL RESET.

If the monitor is to be tested, the Display Test is run with the letter "X" across the screen, the Character Set Test is run, and finally the Scroll Test is run for 50 repetitions. For more information on these tests, refer to Chapter 5.

If the printer is to be tested, the Sliding Alpha Test is run for 35 repetitions, and then the Character Set Print Test is run. The Echo Character Print Test then runs with the letter "E" across one line. The Horizontal Tab Test and Line Space Test are each run through once. For more information on these tests, refer to Chapter 6.

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 7.

The Exerciser begins with the monitor diagnostics, then goes to the printer diagnostics, and finally to the disk drive diagnostics. When complete, the Exerciser cycles back to the monitor to repeat the process. This cycle continues until the CTRL RESET is pressed to interrupt the Exerciser. Upon interruption, the message, "TOTAL PASSES THROUGH EXERCISER-X", is displayed to indicate the number of cycles the Exerciser has run.

Because the Exerciser tests the three major components of the Apple II system: the monitor, printer, and disk drive(s), it can indicate whether there is a system 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 12-hour burn-in run of the Exerciser.

The System Exerciser module operates under BASIC, 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 instructions through BASIC, which is written in the CPU machine language.

RUNNING THE SYSTEM EXERCISER

When you are ready to run the System Exerciser on the Apple II, make sure that the components are ready for hours of testing. If the printer is part of the test, load it 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 inserted into each of the drives being tested. Although you might wish to dim the monitor display, there is no danger from etching (see Chapter 5 for more information about etching), because the display is continuously changing.

To run the System Exerciser, follow the instructions provided in "Running the Diagnostic Program" in Chapter 2. Then from the System Diagnostic Main Menu, press **E** to initiate the Exerciser module. The screen will clear and a prompt will ask you if the monitor is to be tested, yes or no. Enter **Y** or **N** and press **RETURN**.

A second prompt asks if the printer is to be tested. Again, enter **Y** or **N**. If the answer is yes, you are then prompted to enter the printer carriage length and the printer controller slot number.

A third 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. Using the Exerciser, all the disk drives on the Apple II can be tested sequentially. Enter the appropriate number between 1 and 6 to indicate the number of drives you wish to test.

The program then prompts you for the slot number of each of the drives to be tested. Enter the appropriate number, 0 through 7, as indicated on the expansion slot itself. If running a IIc, specify slot 6.

Finally, the program prompts you to enter which drive is to be tested. Each drive board can control two disk drives, drive 1 or 2. Enter the disk address for the drive to be tested for that board (Fig. 9-1).

```
            SYSTEM EXERCISER

TEST MONITOR? (Y OR N)-Y

TEST PRINTER? (Y OR N)-Y
   ENTER PRINTER CARRIAGE LENGTH:
   (80 OR 132)-80
   ENTER PRINTER CONTROLLER SLOT NUMBER
   (0-7)-1

TEST DISK DRIVE(S)? (Y OR N)-Y
   HOW MANY DRIVES TO TEST-2
   ENTER DRIVE CONTROLLER SLOT NUMBER
   (0-7)-6
   ENTER DRIVE NUMBER (1 OR 2)-1
   ENTER DRIVE CONTROLLER SLOT NUMBER
   (0-7)-6
   ENTER DRIVE NUMBER (1 OR 2)-
```

Fig. 9-1. System exerciser module prompts.

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 Xs for 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, the printer tests begin upon completion of the monitor testing. 35 repetitions of the Sliding Alpha Test are printed. It runs through the Character Set 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, providing you with a thorough printer test.

If the disk drive(s) are included in the Exerciser, the disk drive test begins upon completion of the printer testing. The write/read test is run on each disk drive specified for testing.

After this is completed, the program loops back and tests the monitor again, then the printer, and so forth. The Exerciser runs in this continuous loop, running the tests in a repeated cycle. If all three subsystems are included in the Exerciser, one cycle takes approximately 25 minutes, with 20 of those minutes devoted to the disk drive portion of the test.

Press CTRL RESET to halt the Exerciser. This ends the System Exerciser module, and as a result, halts the entire System Diagnostic Program. To return to the Exerciser, enter GOTO 4300 at the Applesoft BASIC right bracket prompt. The system returns to the System Diagnostic Program at line 4300 in the code.

At this point, a message is displayed indicating the number of passes the program has completed. The system then prompts to enter any character to end the diagnostic. Once this is done, the System Diagnostic Program displays the System Diagnostic Main Menu.

If the Exerciser is interrupted while running the disk drive test, there could be a file called "TEST" in the diskette's catalog listing. Delete this file from the diskette. Normally the diagnostic removes this file after its run is completed.

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 will stop at the point at which the problem took place. An error message generated by the Apple operating system or by the disk drive diagnostic error handling routine might be displayed, indicating the problem and the appropriate action to be taken. If it is a problem for which there is no error message, the program simply halts at the point where the error occurred, and you can probably see for yourself what the problem is.

HOW THE SYSTEM EXERCISER MODULE WORKS

When you choose E from the System Diagnostic Main Menu, the program branches to line 4000 for the beginning of the System Exerciser module (Fig. 9-2).

```
4000 REM SYSTEM EXERCISER
```

```
4000  REM  SYSTEM EXERCISER
4010  HOME :PS = 0:EX = 1: PRINT TAB( 10)"SYSTEM
      EXERCISER": PRINT
4020  INPUT "TEST MONITOR? (Y OR N)-";M$: PRINT
4030  INPUT "TEST PRINTER? (Y OR N)-";P$
4040  IF P$ = "Y" OR P$ = "y" THEN GOTO 4050
4045  GOTO 4060
4050  PRINT "   ENTER PRINTER CARRIAGE LENGTH":
      INPUT "   (80 OR 132)-";W: IF W <  > 132 THEN
      W = 80
4055  INPUT "   ENTER PRINTER CONTROLLER SLOT
      NUMBER-": INPUT "   (0-7)-";SL: IF SL < 0 OR
      SL > 7 THEN SL = 1
4060  PRINT : INPUT "TEST DISK DRIVE(S)? (Y OR N)-";
      DD$(0)
4070  IF DD$(0) = "N" OR DD$(0) = "n" THEN  GOTO 4120
4080  INPUT "   HOW MANY DRIVES TO TEST-";DD(0)
4090  FOR I = 1 TO DD(0)
4100  INPUT "   ENTER DRIVE CONTROLLER SLOT NUMBER":
      INPUT "   (0-7)-";SS(I): IF SS(I) < 0 OR
      SS(I) > 7 THEN SS(I) = 6
4105  INPUT "   ENTER DRIVE NUMBER (1 OR 2)-";DD(I):
      IF DD(I) < 1 OR DD(I) > 2 THEN DD(I) = 1
4110  NEXT I
4120  IF M$ = "N" OR M$ = "n" THEN  GOTO 4140
4130  A$ = "X":A = 50: GOSUB 510
4140  IF P$ = "N" OR P$ = "n" THEN  GOTO 4160
4150  A$ = "E":X = 35: GOSUB 1320
4160  IF DD$(0) = "N" OR DD$(0) = "n" THEN  GOTO 4200
4170  FOR Q = 1 TO DD(0)
4180  S = SS(Q):D = DD(Q): GOSUB 2500
4190  HOME : NEXT Q
4200  PS = PS + 1: GOTO 4120
4300  HOME : PRINT D$;"CLOSE TEST": PRINT D$: PRINT
      TAB( 10)"SYSTEM EXERCISER": PRINT
4310  PRINT "TOTAL PASSES THROUGH EXERCISER-";PS
4320  PRINT "PRESS ESC TO END TEST": GOSUB 8000:
      EX = 0: POKE 216,0: GOTO 50
```

Fig. 9-2. System exerciser module listing.

Line 4010 clears the screen and positions the cursor with the HOME instruction. Variable PS, which will count the number of cycles through the Exerciser, is set to 0. The EX Exerciser flag is set to 1, indicating throughout the various tests that the Exerciser has control of the program. Finally, the diagnostic title is displayed on the screen.

```
4010 HOME :PS = 0:EX = 1: PRINT TAB( 10)"SYSTEM EXERCISER": PRINT
```

Line 4020 through 4110 provides the prompts necessary for you to set up the Exerciser for the subsystems to be tested. Line 4020 asks whether or not the monitor is to be tested. Your Y or N response is placed into variable M$.

```
4020 INPUT "TEST MONITOR? (Y OR N)-";M$: PRINT
```

Line 4030 asks whether or not the printer is to be tested. Your Y or N response is placed into variable P$.

```
4030 INPUT "TEST PRINTER? (Y OR N)-";P$
```

If you choose to test the printer, line 4040 directs the program to line 4050 to prompt for the printer carriage length.

```
4040 IF P$ = "Y" OR P$ = "y" THEN GOTO 4050
```

If you chose not to test the printer, the program will proceed to line 4045. This line branches the program to line 4060 in order to skip the subsequent printer prompts.

```
4045 GOTO 4060
```

Line 4050 prompts you to enter the printer carriage length, either 80 or 132. The response is stored in variable W. If W is not equal to 132, it is set to 80.

```
4050 PRINT " ENTER PRINTER CARRIAGE LENGTH":INPUT " (80 OR 132)-";W:
     IF W <> 132 THEN W = 80
```

Line 4055 prompts you to enter the number of the slot which contains the printer controller. Your response is stored in variable SL. If you enter a number other than 0 through 7, SL is set to 1, which is the traditional slot for the printer controller.

```
4055 INPUT " ENTER PRINTER CONTROLLER SLOT NUMBER-": INPUT " (0-7)-";SL:
     IF SL < 0 OR SL > 7 THEN SL = 1
```

Line 4060 asks whether or not the disk drives are to be tested. Your Y or N response is stored in variable DD$(0).

```
4060 PRINT : INPUT "TEST DISK DRIVE(S)? (Y OR N)-"; DD$(0)
```

If you chose not to test the disk drive, line 4070 branches to 4120. This skips around the subsequent prompts for more disk drive test details.

```
4070 IF DD$(0) = "N" OR DD$(0) = "n" THEN GOTO 4120
```

If you did choose to test the disk drive, the program proceeds to line 4080. Line 4080 asks you for the number of drives to be tested. The response is stored in variable DD(0).

```
4080 INPUT " HOW MANY DRIVES TO TEST-";DD(0)
```

Line 4090 begins a FOR-NEXT loop which asks two more questions about the disk drive. The loop increments for as many times as there are drives to be tested, as indicated in variable DD(0). In other words, if you specified three drives to be tested in line 4080, the loop runs through three cycles.

```
4090 FOR I = 1 TO DD(0)
```

Line 4100 prompts for the slot number of the controller for the drive to be tested. The response, which should be between 0 and 7, is stored in variable SS(I). If the value in SS(I) is not between 0 and 7, it is automatically set to 6, which is the traditional slot location for the drive controller.

```
4100 INPUT " ENTER DRIVE CONTROLLER SLOT NUMBER": INPUT " (0-7)-";SS(I):
     IF SS(I) < 0 OR SS(I) > 7 THEN SS(I) = 6
```

Line 4105 prompts for the number of the drive to be tested. The response, which should be 1 or 2, is stored in variable DD(I). If the value in DD(I) is not 1 or 2, it is automatically set to 1.

```
4105 INPUT " ENTER DRIVE NUMBER (1 OR 2)-";DD(I):IF DD(I) < 1 OR DD(I) > 2 THEN DD(I) = 1
```

Line 4110 increments the I loop with the NEXT instruction.

```
4110 NEXT I
```

Line 4120 begins actual processing of the Exerciser. Variable M$ is checked first. If it is equal to N, the program branches down to line 4140.

```
4120 IF M$ = "N" OR M$ = "n" THEN GOTO 4140
```

If variable M$ is equal to Y, the program proceeds to line 4130 to process the monitor test portion of the Exerciser module. Monitor test variables are set. A$ is set equal to X, for the character to be used in the Display Character Test. Variable A is set equal to 50, for the number of repetitions in the Scroll Test. Then the program branches to line 510, which is the Monitor Diagnostic module. The monitor portion of the Exerciser is run.

```
4130 A$ = "X":A = 50: GOSUB 510
```

After the monitor Exerciser tests are completed, and the program encounters a RETURN, the program returns to line 4140. If variable P$ is equal to N, meaning the user chose not to test the printer, the program branches to line 4160.

```
4140 IF P$ = "N" OR P$ = "n" THEN GOTO 4160
```

If variable P$ is equal to Y, the program proceeds to line 4150 to process the printer test portion of the Exerciser module. A$ is set equal to E, for the Echo Character Print Test, and X is set equal to 35 for the Sliding Alpha Print Test. The program then branches to line 1320, which is the Printer Diagnostic module. The printer portion of the Exerciser is run.

```
4150 A$ = "E":X = 35: GOSUB 1320
```

When the printer Exerciser tests are completed, and the program encounters the RETURN instruction, the program returns to line 4160. If variable DD$(0) is equal to N, meaning you chose not to test the disk drives, the program branches to line 4200.

```
4160 IF DD$(0) = "N" OR DD$(0) = "n" THEN GOTO 4200
```

If variable DD$(0) is equal to Y, the program proceeds to line 4170 to process the disk drive test portion of the Exerciser module. Line 4170 initiates a FOR-NEXT loop which will loop through the disk drive tests for as many times as there are disk drives to test, as indicated in variable DD(0).

```
4170 FOR Q = 1 TO DD(0)
```

Line 4180 sets the disk drive test variables. S, the drive controller slot number, is set equal to SS(Q). Variable D, the drive number is set equal to DD(Q). These arrays hold the slot and drive numbers for the specific disk drives you wish to test. Then the program branches to line 2500, the Disk Drive Diagnostic module.

```
4180 S = SS(Q):D = DD(Q): GOSUB 2500
```

After the first loop through the disk drive test is completed, and the RETURN instruction is encountered, the program returns to line 4190. Line 4190 clears the screen and positions the cursor with the HOME instruction. The NEXT Q instruction increments the loop for the next drive to be tested. This loop continues until the specified number of drives have been tested.

```
4190 HOME : NEXT Q
```

After all the loops through the disk drive test have been completed, line 4200 increments the Exerciser pass counter PS. The program then branches to line 120 to begin another pass starting with the monitor tests again. This loop cycles continuously until you press CTRL RESET.

```
4200 PS = PS + 1: GOTO 4120
```

After you press CTRL RESET, the Exerciser halts, and the program quits to the Applesoft BASIC environment. The user then enters a GOTO 4300 in this environment, and the program is reinstated at this point. Line 4300 clears the screen and positions the cursor. The disk is closed, and the test title is displayed on the screen.

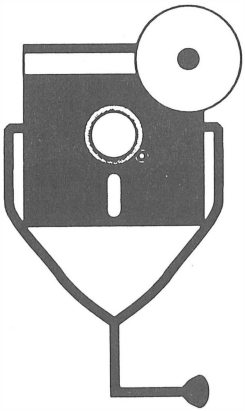
```
4300 HOME : PRINT D$;"CLOSE TEST": PRINT D$: PRINT TAB( 10)"SYSTEM EXERCISER": PRINT
```

Line 4310 displays a message indicating the number of passes made through the Exerciser, using the PS counter from line 4200.

```
4310 PRINT "TOTAL PASSES THROUGH EXERCISER-";PS
```

Line 4320 prompts you to press the ESC key to end the test, and then branches to the Get-a-key subroutine at line 8000 to wait for the key. The EX flag is set back to 0. The POKE statement turns the disk drive's special error handling routine off. Finally, the program branches back to line 50, to display the System Diagnostic Main Menu.

```
4320 PRINT "PRESS ESC TO END TEST": GOSUB 8000: EX = 0: POKE 216,0: GOTO 50
```



10

Writing Diagnostics for Other Peripherals

It is important to be able to isolate and identify the source of any problem in your Apple II system. The Apple II System Diagnostic Program provides the means for locating problems in your system unit, keyboard, monitor, printer, disk drives, or serial communication interface. Based on the results, you can then make repairs on the indicated subsystem.

However, if your system uses peripherals not covered by this program, it might be advantageous to write new diagnostic modules for them. In this way, the System Diagnostic Program can grow with your computer system, and new or different components will not be excluded from the diagnostic loop that tests the rest of the system.

Using the game paddle as an example, this chapter will demonstrate and explain how to research and identify the basic functions of a new device, as well as the types of problems you might experience using it. This information will then lead you to determine the kinds of tests necessary for an effective diagnostic module. Flowcharting and coding advice will be provided, along with integration and testing procedures.

If you are not familiar with BASIC, refer to Chapter 2, "The System Diagnostic Program." That chapter briefly describes the Applesoft BASIC commands used throughout the System Diagnostic Program. It also points out other resources for learning and using BASIC in more depth.

TYPES OF MICROCOMPUTER PERIPHERALS

There are numerous interesting peripheral devices available for the Apple II systems. These peripherals include digitizer tablets which allow drawing and drafting on the computer. Voice recognition devices can recognize voice patterns and even respond to verbal commands. Voice synthesizers can speak words, sentences, and can even carry on conversations.

Other popular peripheral devices include the bar code reader, light pen, mouse, and video cassette recorder. Apple game paddles and joysticks are used with various interactive games.

Additionally, more and more devices are being invented and marketed for the Apple II computer user.

DEVELOPING A NEW DIAGNOSTIC MODULE

Begin considering the development of a diagnostic module for the new peripheral as soon as you have acquired a degree of familiarity with it. After this new device becomes an integral part of your permanent system, and not just a new toy, then it is time that its own diagnostic should be developed. The diagnostic must be developed before a malfunction occurs.

If you are apprehensive about writing your own diagnostic module, particularly if you have never programmed before, just remember that programming is based on accomplishing a task in a very logical manner. You know where you want to go, you know what the objective is. If you don't know what you're trying to accomplish, there is no way you can write a program.

First determine the tasks your program has to accomplish. Then decide what needs to be displayed on the screen. Work backwards from these objectives and determine how to complete these tasks through a program, within the constraints of BASIC commands, syntax, and formats.

When professional software developers create a new software program, they have specific methods which help them determine the objectives of a program. These methods help them design and finally write the program so that it successfully meets the original objectives and accomplishes the necessary tasks. It is not a haphazard method with programmers quickly brainstorming about a new kind of program and immediately writing code. Rather, programming is an extremely logical process in which well-developed detail and organization skills are necessary for success. The process of developing a program includes the following phases:

- needs analysis
- system design
- program analysis (flowcharting)

- program preparation (coding)
- module testing
- integration
- integrated program testing

Professional software developed in a corporate environment, can have a development cycle lasting from three months to two years. However, a module consisting of simple diagnostic routines might only take a few days. Nonetheless, each development phase is necessary for a good program that accomplishes its objectives. People have been successfully programming computers in this manner for almost forty years.

Needs Analysis

A needs analysis consists of the research and subsequent determination of the particular problems to be solved, and the tasks to be accomplished. These requirements must be clearly defined well before any coding begins.

Performing a needs analysis for a new device diagnostic module consists of:

- reading the device documentation
- working with the device itself
- determining potential device problems

Read the Device Manual. Read the device documentation as soon as you obtain the new peripheral. This will familiarize you with the new device, indicate how to use it most efficiently, and show what information is available and where it can be found. Specifically, the manual provides instruction on the proper installation, setup, use, operation, maintenance, and troubleshooting of the device.

The documentation first covers unpacking and installation instructions, which should be followed carefully to minimize potential problems. Then the manual describes operation, the various functions and special features, and gives examples of applications and capabilities. Procedures for various options and settings are provided as necessary. The manual also covers the integration of the device with other system components and software. Suggestions are provided on existing hardware and software setup in order for the new device to work properly and efficiently.

If applicable, the documentation describes how to install accompanying software and how to program the device. The manual usually 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. This will familiarize you with typical problems for which the diag-

nostic can test. 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 exact troubleshooting requirements. It would be impossible for any troubleshooting guide to consider all these factors, simply because there are an infinite number of possibilities and combinations.

Because of this, it is clear that you can gain a greater benefit from developing a customized troubleshooting diagnostic for the device as it works with your particular system.

Work with the Device. After reading the documentation, unpack and install the device. In order to develop a new diagnostic module for a new peripheral, learn everything possible about how the device functions.

Find out specifically what the device is intended for and 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. This will demonstrate its true functionality and limitations as you fully use the device in an actual application.

Determine Potential Device Problems. Once you know the functions of the device, take the inverse of these functions to consider the device's potential problems. In other words, if a device is supposed to perform a particular function, a potential problem is simply that it does not perform this function. Further research in the device's troubleshooting guide will provide you with more detailed information about potential problems. This will probably be your best resource.

System Design

The system design phase uses the results of your research and problem identification that took place in the needs analysis phase. System design addresses the problems and tasks, and determines how the program will solve the problems and complete the tasks.

For the system design phase of developing the device diagnostic module, use the potential device problems identified in the needs analysis phase to determine what types of tests should be included in the new device module. To do this, create a simple table that will help keep track of your thoughts. In the first column, jot down the potential device problem. In the second column, indicate the working name of the test you would like to develop to check for that problem. In the third column, briefly describe the functions the test will perform, and what it might look like (Table 10-1). This table will help organize your thoughts and assist with the design of the diagnostic test routine.

Note that one test might be able to check for two or three different problems. Be careful not to create too many redundant tests, when one test would satisfacto-

Table 10-1. System Design Table.

Potential Problem	Test for Problem	Test Description

rily do the job. Keep the module as simple and efficient as possible, at the same time being effective and thorough.

Be aware that there might be other problems which no software test will be able to diagnose. These might include those types of problems in which something physical must be done with hardware, and cannot be diagnosed through software.

Program Analysis (Flowcharting)

The program analysis phase lets you design the actual program, taking into consideration the overall logic and flow of the program. Program analysis is done with a graphic aid called a flowchart. The program flowchart is a detailed picture that represents the process, or flow, of steps to be performed within the computer to produce the desired results. The flowchart provides the road map that will be a guide when writing the program code.

Write a separate flowchart for each test identified in the system design table. Each test is considered a sub-module of the device diagnostic program, which in turn will eventually become a module of the entire Apple II System Diagnostic Program.

Program analysis actually consists of creating two different flowcharts: the system flowchart in a plain English organizational overview, and the program flowchart being the bridge between plain English and BASIC, indicating the specific program flow and structure.

Flowcharting Symbols. Although there are numerous flowcharting symbols in existence, only five are needed for the purposes of charting a device diagnostic program. As shown in Fig. 10-1, these symbols are:

- terminal
- input/output
- processing
- decision
- connector

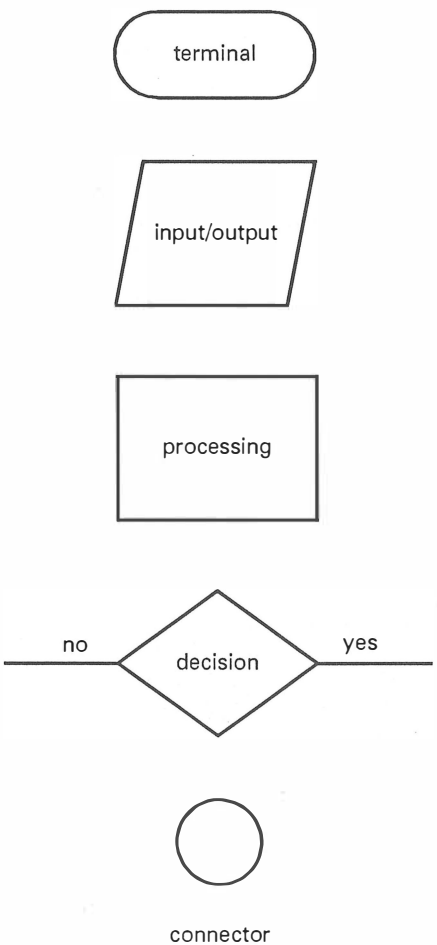


Fig. 10-1. Flowcharting symbols.

The oblong terminal symbol represents the beginning and end of a program.

The parallelogram input/output symbol represents any I/O function. This can include input data to be read into the CPU, and output information to be displayed on the screen or sent to the printer.

The rectangular processing symbol represents computer processing operations. Arithmetic and data movement instructions are typical of the types of steps included in the processing symbols.

The diamond-shaped decision symbol represents a place within the program at which the computer must make a decision based on logic/comparison operations, also performed by the computer. Exit paths from the decision symbol are usually determined by a yes or no answer to a conditional statement written in the form of a question. If the computer results indicate a yes, the program branches in one direction. If the results indicate a no, the program branches in another direction.

The round connector symbol is used when additional flow lines might cause confusion and reduce rather than enhance understanding of the program flow. It is usually identified with a number or letter, and connects with a matching connector symbol by the same number or letter indicated elsewhere in the flowchart.

Arrows connect the various flowcharting symbols together to indicate the direction (or any change of direction) in which a program or segment of a program will flow (Fig. 10-2).

System Flowchart. Creating the system flowchart is the first stage of program analysis and flowcharting. The system flowchart expresses the fundamental ideas of the program.

The general order in which the program will flow to attain the desired results is abstracted in this flowchart. The system flowchart also lists the intermediate functions to be carried out. This technique lets you think out the program as a whole without being concerned about more specific details, such as where loops and branches need to occur.

Figure 10-3 is an example of a system flowchart for a simple program segment asking users to enter their name, and then greeting them by name.

Program Flowchart. The program flowchart is based on the system flowchart, but is far more detailed. Following the order and task listing detailed by the system flowchart, the program flowchart actually charts the inputs, branches, decisions, loops, and other structural details of the program. It is the implementation of the system flowchart. While the system flowchart lists the necessary tasks to achieve a desired goal, the program flowchart shows specifically how these tasks will be completed in the program. Figure 10-4 shows the program flowchart based on the system flowchart shown in Fig. 10-3.

If your design of the device diagnostic module includes more than one test, you will probably have several system flowcharts, representing each of the sub-mod-

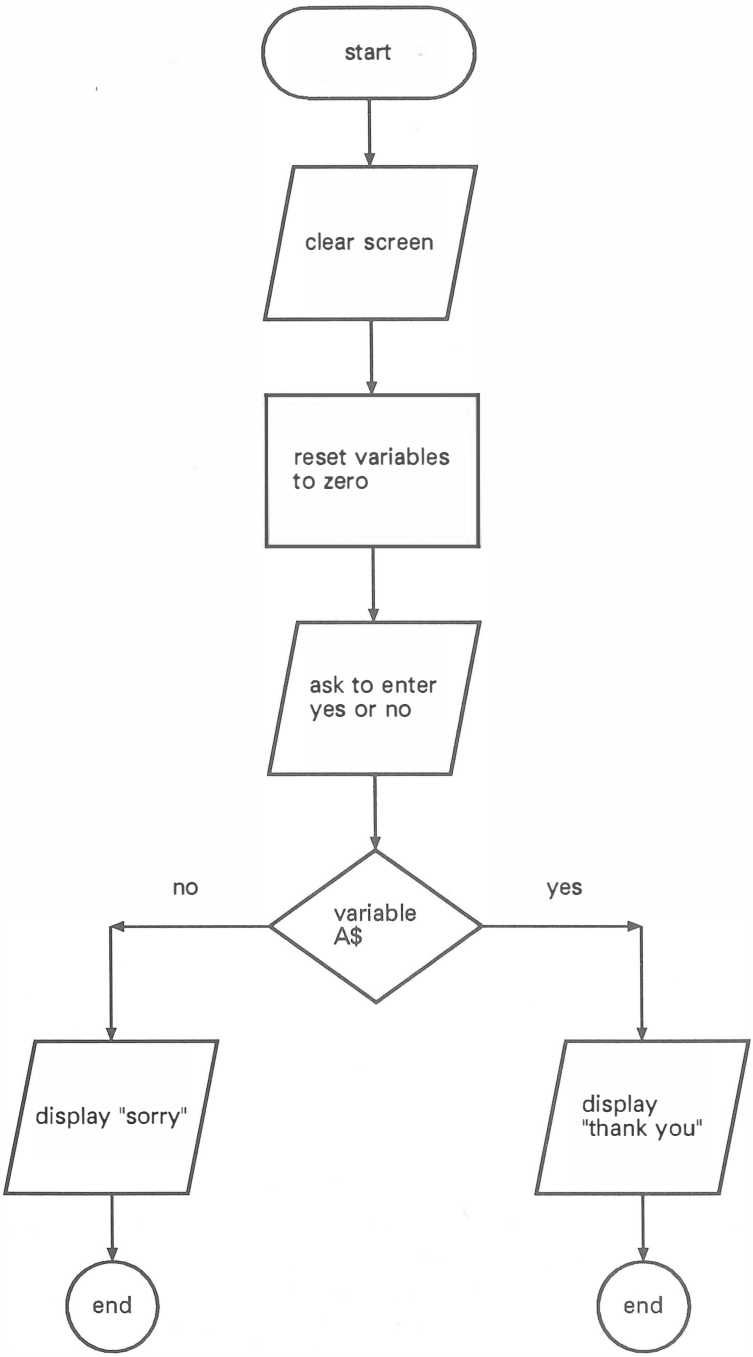


Fig. 10-2. Flowchart.

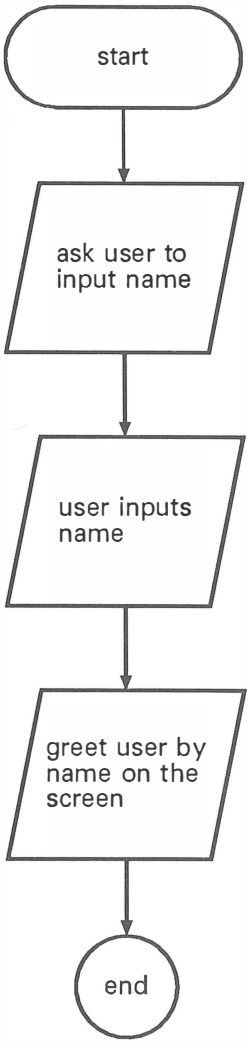


Fig. 10-3. System flowchart.

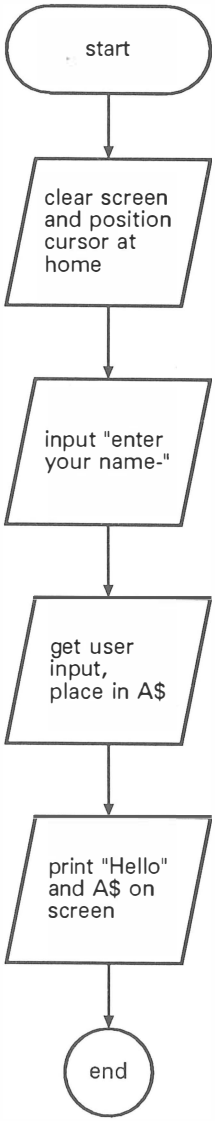


Fig. 10-4. Program flowchart.

ule tests. Create a separate program flowchart for each of the separate system flowcharts. When these are all completed, connect all the flowcharts together in the order required to create one flowchart for the entire device diagnostic module.

If the system flowchart includes a step which you do not know how to translate into the program flowchart, some additional research will be needed. Refer to a BASIC manual, the Applesoft BASIC reference guide, and the device manual for

further information on how to program for a particular task. Include details of your research results in the flowchart, so it will not be necessary to look it up again when it is time for the coding stage.

Although the program flowchart does not generally use the BASIC commands, the name of the BASIC command might incidentally be included in a given flowchart box. If desired, include the pertinent BASIC command in the flowchart box to which it applies. This additional level of detail can help later when coding.

Make the program flowchart as detailed as possible. 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.

Program Preparation (Coding)

After the program flowchart is diagrammed, begin the program preparation phase—the actual coding of the new device diagnostic program.

When developing a new test, first make sure that all aspects of the system are functioning properly. A good way to do this is to run the System Exerciser for several cycles and see whether any errors are encountered. 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 be misleading, and the test code itself could become obsolete.

Write a standalone version of the new device diagnostic module. Translate all steps of the program flowchart from English into BASIC. Use line numbers in increments of ten so that there is space to insert lines later if something was forgotten, a bug needs to be fixed with another line, or if additional functionality necessitates additional lines of code.

The following is the completed BASIC code based on the program flowchart shown in Figure 10-4.

```
10 HOME
20 INPUT "Enter your name-";A$
30 PRINT "Hello ";A$
40 END
```

Begin the line numbers around 6000, or wherever the System Diagnostic Program ended, so that subsequent integration into the System Diagnostic Program will be easier. This will eliminate the need for renumbering the program upon integration, which is a very tedious process, and could cause more problems.

Refer to the code of existing modules as listed in the individual chapters or in the Appendices, to familiarize yourself with the techniques used in coding those modules.

It might be necessary to stop several times throughout the coding process to do further research. Find out which BASIC commands perform a particular function, what the syntax of that command is, as well as what kinds of parameters the command includes. The Applesoft BASIC manual and device documentation are the best resources for this information. They also provide examples. Try these examples to get a feel for how the command works. When convinced that this is the proper command to use for this application, include it with the necessary parameters and variables in your device diagnostic program.

While writing the code, save the program to disk periodically, perhaps once every five or ten minutes. This will avoid the loss of valuable work in the event of a power or system failure.

After you finish with a segment of the program in which a particular task is begun, processed, and completed, save the program to disk and try it out. It's easier to pinpoint simple problems this way, rather than waiting until the entire program is written to try it out. It is also encouraging and gratifying to find out instantly that the program is working as expected.

When it seems like the program code is complete, and functions are working as expected, save the program and create a backup. Print out a program listing and continue to the module testing phase.

Module Testing

There is no such thing as a perfect, bug-free program, no matter how experienced and talented the programmer is. Just as there might have been problems while programming the module, there will probably be problems after the program is completely written. These bugs might manifest themselves by the program simply not working, the program crashing when trying to perform a certain function, the program doing something other than what is expected, or it might be something as simple as a display not looking quite right on the screen.

Because of these bugs, the module testing phase is extremely important and necessary. Test the new device diagnostic module thoroughly as a standalone program before integrating it with the System Diagnostic Program. Try out each and every function possible. Wherever different values are to be input by the user, try every different value and combination imaginable. Particularly use extreme values, such as very high values and very low values, and make sure the program can accommodate these inputs without breaking down. Also find out how the program handles the input of incorrect values and conditions.

When a bug is found, see if it is repeatable. Then try out other variations of this action to see if it causes the same result. This will provide you with more informa-

tion on how to fix the bug. Make notes on the program listing, such as the lines where you think the bug might exist, and possible solutions.

After you have sufficient information about the bug, list the code and edit the appropriate lines. It may be necessary to do additional research to find out why the problem is happening in the first place, and to get ideas on possible solutions.

When you believe the bug is fixed, try the program out again. Go through the same sequence of events that lead up to the bug occurring, and see whether it happens again. Try all the different values and combinations imaginable. When finally satisfied that the bug is fixed in this particular place, try all the other functions with different values and combinations as well. Often fixing a bug will create another bug.

When you feel that the program is thoroughly tested and edited, and that it can withstand any condition without a bug cropping up, hand the program over to someone else for testing. It is a general rule that the programmer cannot thoroughly test his own program. In fact, professional software development companies often have a separate testing or quality assurance group, whose sole function is to test software after coding is complete. This is because as the programmer, you are too familiar with your own work, and as a result, you can become blind to certain aspects of the program. An outside, impartial user is the truest test of the new program.

Have the tester follow the same general testing techniques described above. They will invariably try different functions, different values, different limits, and different combinations than you might have ever considered. As a result, the tester might find additional bugs to be fixed. This is the best way to create a good program as close to perfection as possible.

After all testing cycles are completed, you can finally feel confident about integrating the diagnostic program as a new module in the System Diagnostic Program.

Integration

The integration phase consists of appending the program code to an existing program, in this case the System Diagnostic Program, and making sure that the two programs work together harmoniously without adversely interfering with one another. Integration also consists of editing lines of code so that the two programs are intermeshed, using each other's resources and subroutines, and referring appropriate parts of one program to parts of the other so that they work together as one cohesive program.

When you insert the new module into the System Diagnostic Program, modify lines 50 through 190 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 6000, or wherever the last diagnostic module stopped.

Integrated Program Testing

When the new diagnostic program is fully integrated into the System Diagnostic Program, a final integrated program testing phase is necessary. Carry out the same testing procedures on the new module as before. Some other bugs might crop up as a result of being a part of the larger program. The two programs might interact with one another in an adverse way, and these kinds of bugs must be discovered before using the revised program for actual diagnostics.

For the same reason, test all the other modules in the System Diagnostic Program. Find out whether the new program is interfering with the normal operations and integrity of the existing module. Fix any bugs found in the new diagnostic, and cycle the testing again, until no more bugs are discovered.

At this point, all program development phases are finally completed. Save the newly revised program, and back it up on another diskette. The new program is now ready for actual use in a diagnostic situation.

DEVELOPING THE GAME PADDLE DIAGNOSTIC MODULE

Up to this point, development of a new device diagnostic module has been treated in generalized terms. Provided in this section is a concrete example to make the development process clearer. This example uses the game paddle as the new peripheral for which a diagnostic module will be developed. The step-by-step process includes researching the functions and potential problems of the game paddle, designing the flowcharts for the Game Paddle Diagnostic, developing and testing the new code, and finally integrating and testing the new program as a new module in the System Diagnostic Program.

Between the general principles outlined in the previous section and the concrete examples described in this section, you should be able to gain a good understanding of how a diagnostic module can be developed for any peripheral added to your Apple II system.

Functions of the Game Paddle

The game paddle is a pointing and firing device (Fig. 10-5) used to either position an object on the screen, or to fire a graphic object, like a missile, at

another graphic object, as part of a game. A jet airplane can be flown, a graphic robot can be moved, or a laser cannon can be controlled with the game paddle.

The game paddle is a mechanical device which consists of a tracking wheel controlled by a potentiometer, and a fire button controlled by a switch. A cable connects it to the motherboard in one of the two game paddle sockets (Fig. 10-6). Each game paddle socket can accommodate two game paddles. This means the system can allow up to four game paddles at the same time.

Tracking a Graphic Object. Depending on the application software using the game paddle, a certain object on the screen is programmed to move in accordance with the movement of the game paddle control wheel. When the wheel is turned toward the right, or clockwise, the object on the screen tracks to the right. When the game paddle is turned toward the left, or counterclockwise, the object tracks to the left. The game paddle control causes the particular object on the screen to follow its movements.

A potentiometer varies the signal received by the game port according to the direction, speed, and extent to which the user is turning the game paddle wheel. This operation is very similar to that of a light dimmer switch. The varying signals cause different results in the game, usually positioning an object at a certain place on the screen.

Pressing the Fire Button. When you press the fire button, you generally cause a particular graphic to move quickly, as if shot, from the object being controlled by the wheel to wherever the object is pointing. The graphic usually represents something like a bullet, missile, or cannonball being shot from one location to another in a game.

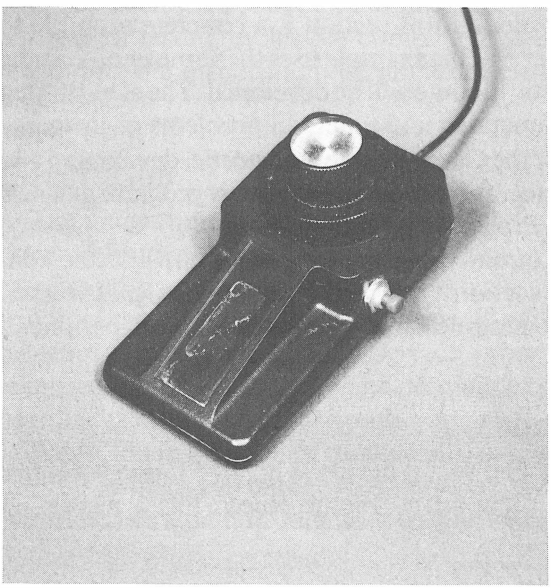


Fig. 10-5. The game paddle.

Information about the Game Paddle

There are several sources that can provide the information necessary to write a game paddle program, particularly a diagnostic program. The Applesoft BASIC manual has a great deal of valuable information for programming in Applesoft BASIC. The game paddle user manual might provide more specific information. The amount of information included in the game paddle manual depends on the manufacturer. The manual will often include information on installation and maintenance, uses and applications for the game paddle, and perhaps sample programs and a troubleshooting guide.

Discover new uses for the game paddle. 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.

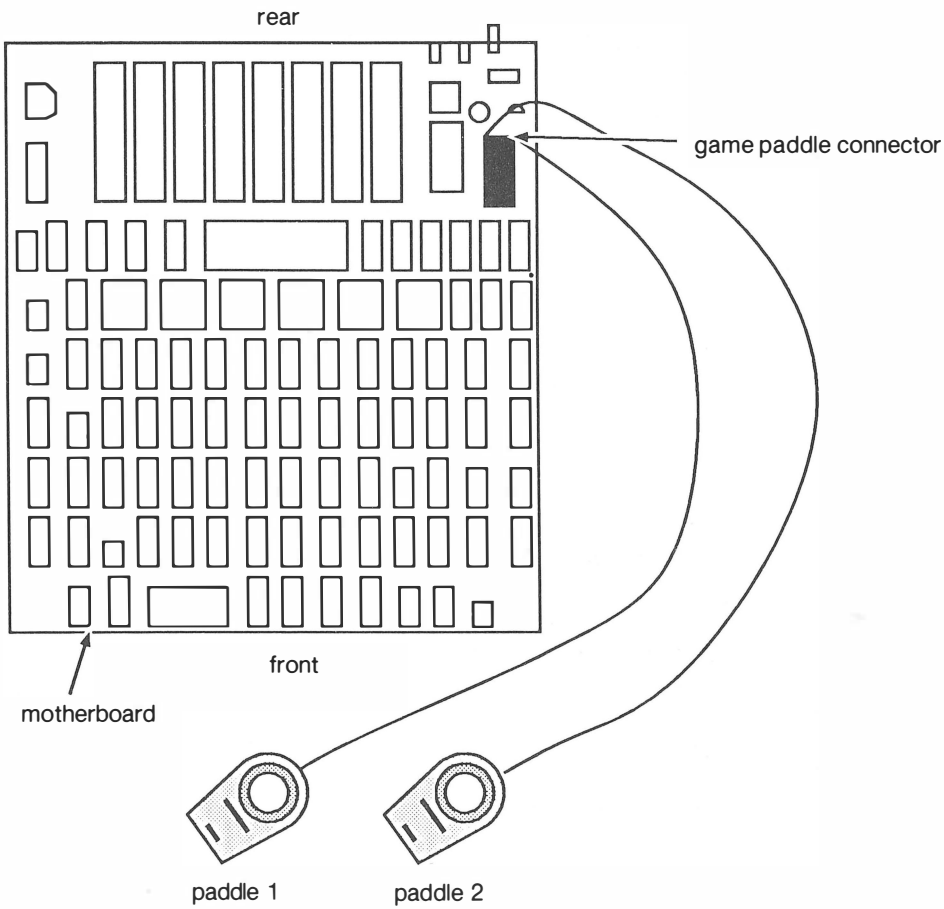


Fig. 10-6. Game paddle interface connections.

Game Paddle Preventive Maintenance

Keep the game paddle clean and free of dust. Open it and spray contact cleaner on the potentiometer wheel and fire button switch every three to six months in order to keep it working freely.

Game Paddle Troubleshooting and Repair Guidelines

The game paddle user manual usually provides 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: Game Paddle Does Not Work at All

If the game paddle does not work at all, it could be a problem with the interface connection, the game paddle, or the system unit. To isolate the problem, try the following solutions in the order given.

Solutions:

- Make sure the game paddle is firmly seated in its game paddle port. Unplug the game paddle from its socket port on the motherboard, and carefully plug it back in.
- If the game paddle still does not work, unplug the game paddle cable from its game paddle socket port, and plug it into the second port.

  If the game paddle works in the other port, then there is a problem in the original port in the system unit. Refer to Chapter 3 for further instructions on troubleshooting the system unit.
- If the game paddle does not work in any of the four game paddle ports, try operating the game paddle on another system. Also run the Game Paddle Diagnostic on the other system. If the game paddle works on the other system, then there is a problem with the system unit. Refer to Chapter 3 for system unit troubleshooting procedures.
- If the game paddle does not work on the other system, then you have a faulty game paddle. Spray the potentiometer wheel with contact cleaner (Fig. 10-7). If necessary, follow the instructions in the game paddle user guide to open it up, and then spray the underside of the potentiometer wheel and the fire button switch with contact cleaner.

  If the game paddle still does not work, refer to the game paddle documentation for further troubleshooting and repair instructions.

Problem: Game Paddle Does Not Move the Object Properly

The game paddle should be able to move its object all the way to the left and to the right of the screen in a given application. If the game paddle is unable to move the object in this manner, or if the object shakes or moves with difficulty, then you have a problem with the control wheel potentiometer.

Solutions:

- Check the cable connection and make sure the game paddle connection is properly plugged into its port.
- Following instructions found in the game paddle user guide, carefully disassemble the game paddle. Spray contact cleaner onto the potentiometer.
- Swap another game paddle onto the system, if available. If the other game paddle works on the system, then your game paddle is defective, and should be replaced. If the other game paddle does not work, then you have a problem with the system unit. Refer to Chapter 3 for further solutions.

Problem: Fire Button Does Not Work

The switch that controls the fire button can also get stuck. This would be indicated if the fire button is pressed, and either nothing happens or it keeps on firing, even after the button is released.

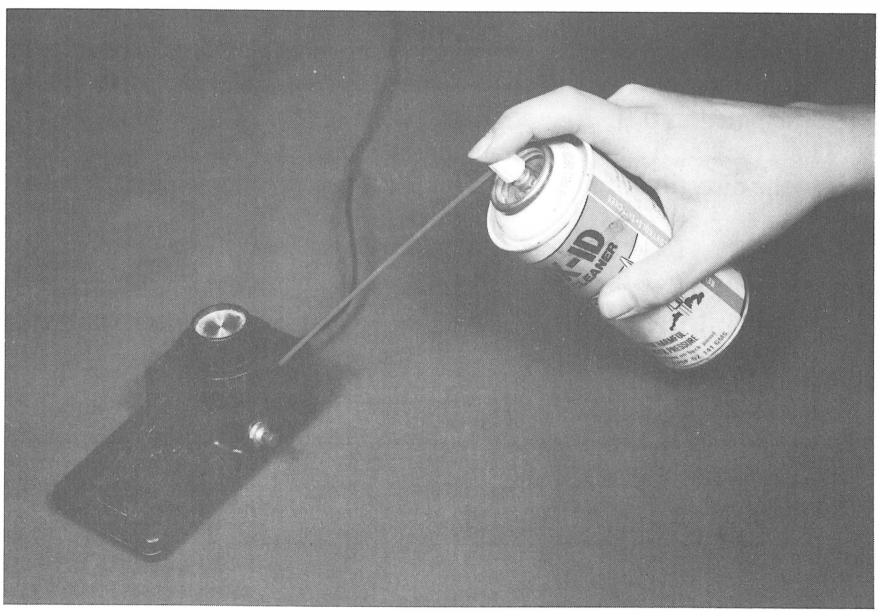


Fig. 10-7. Cleaning the game paddle.

Solutions:

- Check the cable connection and make sure the game paddle connection is properly plugged into the port.
- Following instructions found in the game paddle user guide, carefully disassemble the game paddle. Spray contact cleaner onto the fire button switch. This should unstick the switch.
- Swap another game paddle onto the system, if available. If the other game paddle works on the system, then your game paddle is defective, and should be replaced. If the other game paddle does not work on the system, then you have a problem with your system unit. Refer to Chapter 3 for further solutions.

Game Paddle Tests

Once you have a fair understanding of the game paddle's functions, as well as its potential problems, you can determine the kinds of tests that would be most helpful in writing a diagnostic module.

Based on the information under the previous "Functions of the Game Paddle" and "Troubleshooting and Repair Guidelines" sections, formulate the system design table. This table should include potential problems, tests for those problems, and a short description of each test. Your system design table might look similar to the one shown in Table 10-2.

The Game Port Option. First of all, because the Apple II includes four game paddle ports, the diagnostic user instructions should include a prompt for which port is to be tested. This will allow discrete testing of all four ports. The four ports are designated as 0, 1, 2, and 3 (Fig. 10-8).

The Tracking Test. The game paddle includes a potentiometer which determines the direction of movement for the object it is controlling. Because of this,

Table 10-2. Game Paddle Diagnostic System Design Table.

Potential Problem	Test for Problem	Test Description
Game paddle does not work at all	Hardware Only	N/A
Game paddle does not move the object properly	Tracking Test	Tracks movement of wheel across the screen, from far left to far right
Fire button does not work	Fire Button Test	Provides on-screen response to indicate that the fire button has been pressed.

one particularly useful test would be an on-screen display tracking the direction to which the game paddle is pointing. This would let you diagnose whether or not the potentiometer is tracking properly.

Such a test would display a tracking gauge with a block indicating where the game paddle wheel is positioned along its rotational range. When you rotate the game paddle toward the left, the block pointer tracks along the gauge to indicate the wheel's movement and direction (Fig. 10-9). Likewise, when you rotate the game paddle toward the right, the block tracks toward the right along the gauge. The far right and far left positions on the gauge match up to the game paddle's far right and left extremes.

A value can be read by a BASIC program that can be related to a position on the screen. In order to develop a tracking test like this, determine how information is sent from the game paddle to the system unit, and how this information is accessed. This is a relatively simple process. The position of the wheel can be determined by the PDL BASIC instruction, which returns a value from 0 (extreme left) to 255 (extreme right). This value can be used to arithmetically derive a value between 1 and 40, representing the number of columns on a low-resolution graphics screen display. The derived value can then be used for positioning the indicator block along the diagnostic gauge. A continuous loop in the program can

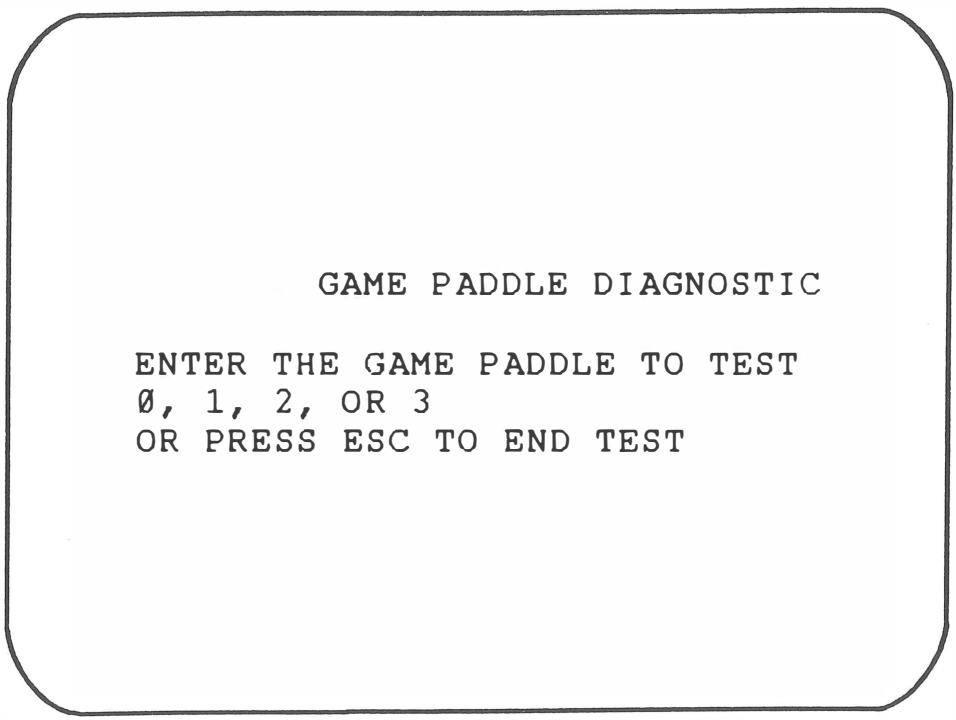


Fig. 10-8. Game paddle diagnostic test instructions.

let the diagnostic continuously check whether this value has been changed, meaning that the game paddle has been moved.

If the pointer fails to track along the gauge, or if the tracking is incorrect or inconsistent, refer to "Game Paddle Troubleshooting and Repair Guidelines" above for solutions to this problem.

The Fire Test. A second test necessary for game paddle diagnostics is to test the fire button. When the fire button is pressed, the routine PEEKs into the appropriate game paddle memory location to check for a value greater than 127. The game paddle memory location for port 0, as labeled on the motherboard, is −16287. For port 1, the location is −16286. For port 2, the location is −16285. (Due to inherent Apple II restrictions, the fire button for port 3 cannot be tested.) A value greater than 127 residing in the game paddle memory location indicates that the fire button has been pressed.

The Fire Test can serve two purposes: to test the fire button, and to quit from the current Game Paddle port test. After you press the fire button, the Game Paddle Diagnostic menu will be displayed. At this point, you can test the game paddle on another port.

If the fire button does not cause the Game Paddle Diagnostic menu to display, then the fire button is probably defective. Pressing CTRL C kills the program and exits to the Applesoft BASIC environment. Refer to "Problem: The Fire Button

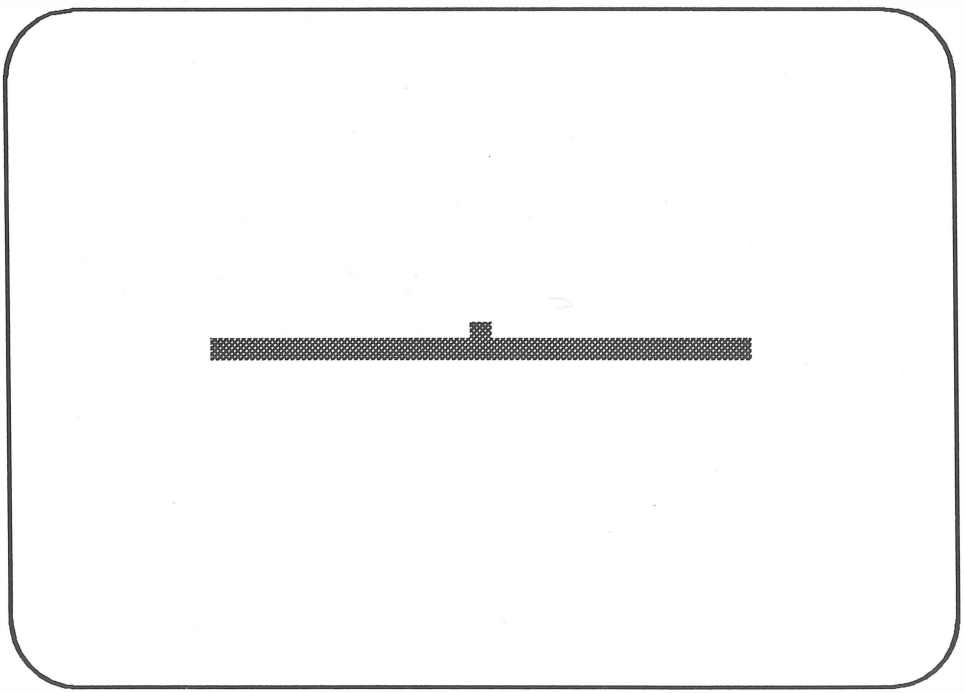


Fig. 10-9. Game paddle diagnostic test gauge.

Does Not Work" above to troubleshoot and repair the fire button. Then run the test again.

The Game Paddle Module Flowchart

Once you have determined the functions to be tested as part of the Game Paddle 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 flowchart: first the system flowchart, and then the program flowchart.

For the system flowchart, use the appropriate flowcharting symbols to list the general task overview, organization, and order of events to take place within the program. Create a separate system flowchart for the Game Port Option, the Tracking Test, and the Fire Button Test. Your flowcharts might look similar to the ones shown in Fig. 10-10.

Based on the system flowchart, detail the program flowchart. Include all loops, inputs, branches, and any other processes needed for the program structure. Try to include the actual BASIC commands within the descriptions to help you with later coding.

Have your Applesoft BASIC programming manual nearby for reference. This will help you research any specialized instructions used to access commands for the game paddle. The documentation that came with the game paddle might also contain additional useful information.

Make the program flowchart as detailed as possible, to provide the most information. When finished, put the three segments of the flowchart together to form one flowchart representing the Game Paddle Diagnostic Program.

Writing the Game Paddle Diagnostic Module

After the flowchart has been thoroughly developed and diagrammed, the actual diagnostic code can be written. Again, refer to the BASIC programming reference books. Refer to the Appendix to see examples of how the other diagnostic modules have performed certain activities.

Research and experiment with any special commands, instructions, formats, or other idiosyncracies about programming the game paddle on the Apple II. Save and try out each program segment while you work, to make sure everything is proceeding smoothly.

The Game Paddle Diagnostic Module Listing. Based on the flowchart, the program code might look similar (although it could have many variations) to the listing shown in Fig. 10-11.

How the Game Paddle Diagnostic Module Works. Just as it was done with the other modules within the System Diagnostic Program, the first order of business in a new module is to provide an identifying REMARK statement. This takes place for the Game Paddle Diagnostic module at line 5000.

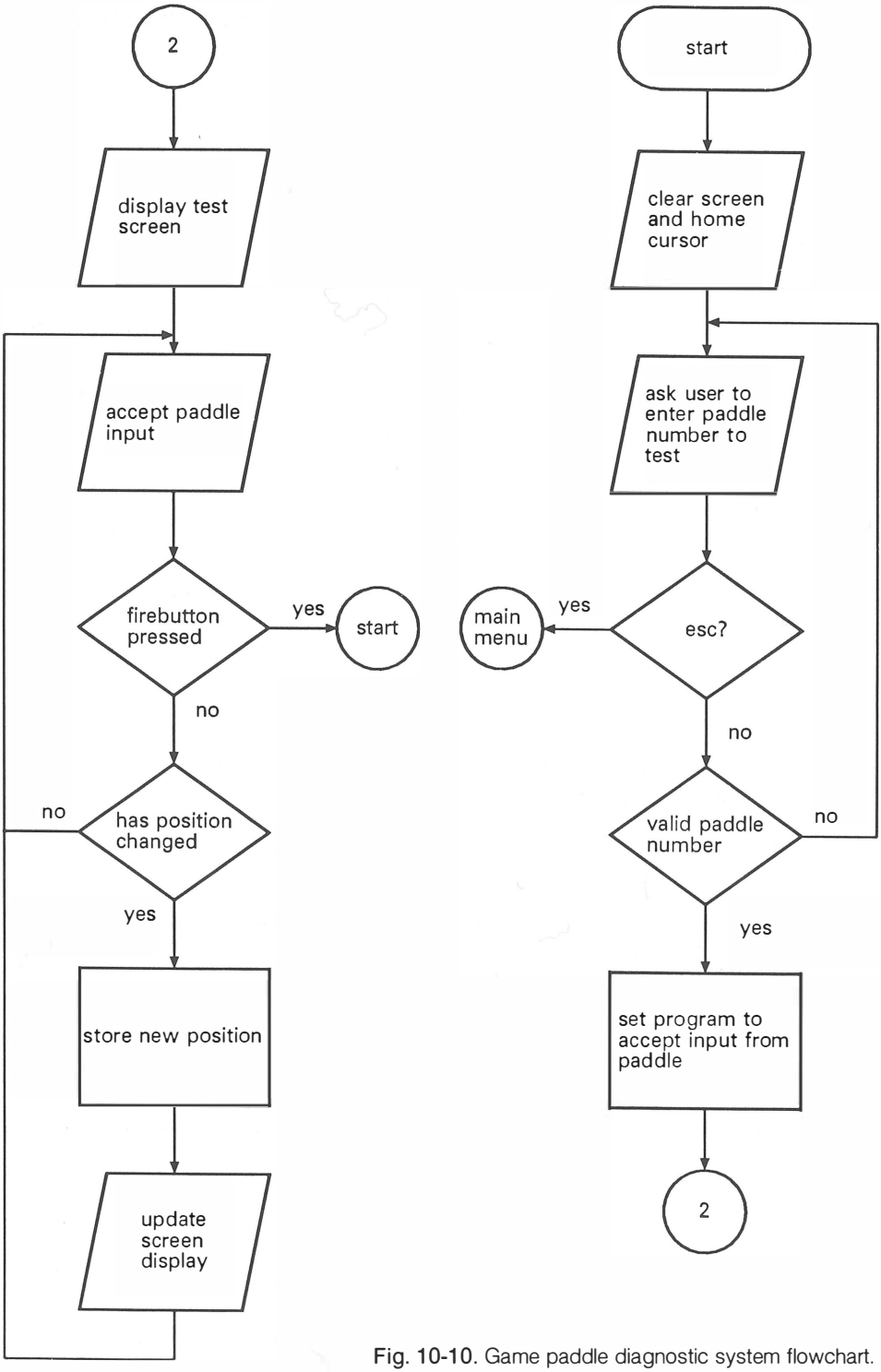


Fig. 10-10. Game paddle diagnostic system flowchart.

```
5000  REM  GAME PADDLE DIAGNOSTIC MODULE
5005  HOME : PRINT  TAB( 12)"GAME PADDLE DIAGNOSTIC":
      PRINT
5010  PRINT "ENTER THE GAME PADDLE TO TEST":
      PRINT "0, 1, 2, OR 3":
      PRINT "OR PRESS ESC TO END TEST"
5020  GOSUB 8000
5030  IF  ASC (A$) = 27 THEN  GOTO 50
5040  A =  VAL (A$)
5050  IF A > 3 THEN  PRINT  CHR$ (7): GOTO 5020
5055  PRINT : PRINT "PRESS FIRE BUTTON TO END TEST"
5060  GR : COLOR= 13: HLIN 10,29 AT 19
5070  B =  PEEK (-16287 + A)
5080  IF B > 127 THEN  GOTO 5140
5090  LET X =  PDL (A):X = X * 20 / 256 + 10
5100  IF X = Z THEN  GOTO 5070
5110  COLOR= 0: PLOT Z,20
5120  COLOR= 13: PLOT X,20
5130  LET Z = X: GOTO 5070
5140  TEXT : GOTO 5000
```

Fig. 10-11. Game paddle diagnostic module listing.

5000 REM GAME PADDLE DIAGNOSTIC MODULE

Line 5005 clears the screen and positions the cursor with the HOME instruction. The test title is also displayed on the screen.

5005 HOME : PRINT TAB(12)"GAME PADDLE DIAGNOSTIC": PRINT

Line 5010 asks you to enter the game paddle number to be tested. This can be 0, 1, 2, or 3. This line also informs you that the ESC key must be pressed to end the test.

5010 PRINT "ENTER THE GAME PADDLE TO TEST" PRINT " 0, 1, 2, OR 3":
PRINT "OR PRESS ESC TO END TEST"

Line 5020 branches to the Get-a-key subroutine at line 8000 to get your response.

5020 GOSUB 8000

Line 5030 checks if you have pressed the ESC key, as indicated by a value of 27 in A$. If this is the case, the Game Paddle Diagnostic ends, and the program branches back to line 50 to display the System Diagnostic Main Menu.

```
5030 IF ASC (A$) = 27 THEN GOTO 50
```

Line 5040 sets variable A equal to the value of A$, which was the game paddle number you selected in line 5020. This statement converts that string value into a true number as recognized by the program in variable A.

```
5040 A = VAL (A$)
```

If the value in variable A is greater than 3, it will be an invalid entry. Line 5050 checks for this, and if it is the case, a beep is sounded and the program returns to line 5020 to let you try again.

```
5050 IF A > 3 THEN PRINT CHR$ (7): GOTO 5020
```

Line 5055 prompts you to press the game paddle's fire button to end the test.

```
5055 PRINT : PRINT "PRESS FIRE BUTTON TO END TEST"
```

Line 5060 sets up the screen to prepare for the actual test. The GR instruction switches the screen to the low-resolution graphics mode. The color is set to 13, the numeric value for white, and this will be used to draw a white line. A horizontal line is then drawn at row 19 from column 10 through 29.

```
5060 GR : COLOR= 13: HLIN 10,29 AT 19
```

In line 5070, a PEEK is done into the memory location that holds the status of whether or not the fire button is pressed. This value is placed into variable B.

```
5070 B = PEEK (-16287 + A)
```

If the value in variable B is greater than 127, this indicates that the fire button has been pressed. Line 5080 checks for this, and if this is the case, the program branches to line 5140 which ends the Game Paddle Diagnostic and returns the program back to the beginning of the diagnostic, at which point you can either test another game paddle or return to the System Diagnostic Main Menu.

```
5080 IF B > 127 THEN GOTO 5140
```

If the fire button has not been pressed, the program proceeds to line 5090. This line locates the pointer on the gauge which indicates where the game paddle wheel is currently set. PDL is an Applesoft BASIC command which examines the position of the game paddle wheel and produces a value between 0 and 255 to

represent this position. The value of A can be 0, 1, 2, or 3, depending upon which game paddle is being tested. X is set equal to the value being returned for the game paddle being tested. At this point, the PDL value must be related to one of the low-resolution screen locations in order to paint its position on the gauge. To do this, the formula X * 20 / 256 + 10 is used, and the pointer is placed at the correct location on the gauge, depending on where the game paddle wheel is set.

```
5090 LET X = PDL (A):X = X * 20 / 256 + 10
```

Line 5100 checks whether the game paddle has been moved. If it has not been moved, X is set equal to Z, which causes the program to branch to line 5070, checking whether the fire button has been pressed. This continues in a loop until the game paddle is moved or the fire button is pressed.

```
5100 IF X = Z THEN GOTO 5070
```

If X is not equal to Z, then the game paddle has been moved, and the program proceeds to line 5110. Because the game paddle has been moved, the gauge pointer must be moved. The first thing that needs to be done to move the pointer is to erase the current pointer. To do this, the color is set equal to 0, the numeric value for black. The PLOT command indicates where the black is to be painted. Z is the column location, and 20 is the row location. Because black is being drawn over the pointer on a black background, the pointer is erased.

```
5110 COLOR= 0: PLOT Z,20
```

Line 5120 resets the color to 13, the value for white, in order to begin redrawing the pointer at the new location. The PLOT command indicates the location of the new pointer, and draws a block for the pointer, with X being the column location, and 20 being the row location as before. The value of the game paddle wheel location is stored in X, and was determined in line 5090.

```
5120 COLOR= 13: PLOT X,20
```

Line 5130 sets Z equal to X. This tells the program that the pointer has been moved. If the values of Z and X are identical, then the game paddle has not been moved. The program branches back to line 5070 to check again whether the fire button has been pressed or the paddle wheel has been moved. This loop continues until the fire button is pressed.

```
5130 LET Z = X: GOTO 5070
```

When the fire button is pressed, the program branches from line 5080 to line 5140. This line closes out the present game paddle test by switching from the low-resolution graphics mode back to the text mode with the TEXT instruction. The game paddle wheel gauge is erased, and the program branches back to the beginning of the Game Paddle Diagnostic. At this point, you can either test another game paddle or quit from the Game Paddle Diagnostic.

```
5140 TEXT : GOTO 5000
```

Testing the Game Paddle Diagnostic Program

Follow the testing procedures described under the "Module Testing" section earlier in this chapter. Thoroughly test the Game Paddle Diagnostic, and fix any bugs you find. Have someone else test the diagnostic as well.

Integrating with the System Diagnostic Program

Once the code for the Game Paddle Diagnostic is written and thoroughly tested on its own, 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. Examine lines 50 through 90 of the program where the Main Menu module resides. Add the Game Paddle Diagnostic identifier to the PRINT statement that lists the menu options on the screen. Also, insert the appropriate IF statement for the Game Paddle Diagnostic identifier in lines 110-180, so that the program branches to the beginning of the Game Paddle Diagnostic module when the appropriate key, perhaps a G, is pressed.

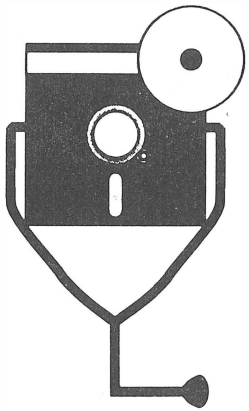
Once this is done, add the code to the program. A good place to append the new code is starting at line 6000, as was done in this chapter's example. This begins the new module where the last diagnostic module stopped.

Testing the Game Paddle Diagnostic Module

The last thing to do is test the Game Paddle Diagnostic as it now resides as an integrated module of the System Diagnostic Program. Although it might have worked perfectly on its own, there are now many other factors playing into the program as a whole. Run the tests thoroughly, trying to break the program. If a problem is found, locate and fix the code where the problem occurred.

Also test all the other modules, and see whether the newly-added Game Paddle Diagnostic has interfered with the operation of existing modules.

Once you have completed the research, design, flowcharting, coding, integration, and testing procedures, the Game Paddle Diagnostic module is ready for real work whenever troubleshooting and diagnostics are needed for game paddle functions.



Appendix A

Apple II Series System Diagnostic Program Listing

```
10 REM APPLE ][ SERIES
20 REM SYSTEM DIAGNOSTIC PROGRAM
30 REM COPYRIGHT 1988 TAB BOOKS INC.
50 REM MAIN MENU MODULE
60 HOME : PRINT TAB( 12)"SYSTEM DIAGNOSTIC": PRINT
70 PRINT TAB( 3)"<K>EYBOARD TEST <S>ERIAL COMM TEST":
   PRINT TAB( 3)"<M>ONITOR TEST <D>ISK DRIVE TEST"
80 PRINT TAB( 3)"<P>RINTER TEST <E>XERCISER":
   PRINT TAB( 3)"<G>AME PDL TEST <Q>UIT"
90 PRINT : PRINT TAB( 12)"ENTER SELECTION"
100 GOSUB 8000: REM GET A KEY
110 IF A$ = "K" OR A$ = "k" THEN GOTO 200
120 IF A$ = "M" OR A$ = "m" THEN GOTO 350
130 IF A$ = "P" OR A$ = "p" THEN GOTO 1250
140 IF A$ = "S" OR A$ = "s" THEN GOTO 3500
150 IF A$ = "D" OR A$ = "d" THEN GOTO 2500
160 IF A$ = "E" OR A$ = "e" THEN GOTO 4000
170 IF A$ = "G" OR A$ = "g" THEN GOTO 5000
180 IF A$ = "Q" OR A$ = "q" THEN END
190 PRINT CHR$ (7): GOTO 100
```

```
200 REM KEYBOARD DIAGNOSTIC MODULE
205 HOME :S = 0: GOTO 210
207 GOSUB 8000
210 HOME : PRINT TAB( 7)"KEYBOARD DIAGNOSTIC": PRINT
215 PRINT TAB( 9)"PRESS ESC TO END TEST": PRINT
220 PRINT TAB( 13)"ASCII": PRINT TAB( 3) "CHARACTER VALUE KEY COMBINATION":
    PRINT TAB(3)"--------- ----- ---------- -----"
230 IF S = 1 THEN GOTO 250
235 S = 1
240 GOSUB 8000
250 IF ASC (A$) = 27 THEN GOTO 50
260 IF ASC (A$) < 32 THEN GOTO 305
300 PRINT TAB( 6)A$;: PRINT TAB( 15) ASC (A$);: GOTO 207
305 K$ = CHR$ (ASC (A$) + 64)
310 PRINT TAB( 6)A$; TAB( 15) ASC (A$); TAB( 23) "CTRL ";K$: GOTO 207

350 REM MONITOR DIAGNOSTIC MODULE
360 HOME : PRINT TAB( 6)"MONITOR DIAGNOSTIC": PRINT : PRINT TAB( 5)"1...DISPLAY TEST":
    PRINT TAB( 5)"2...ALIGNMENT TEST"
370 PRINT TAB( 5)"3...CHARACTER SET TEST": PRINT TAB( 5)"4...SCROLL TEST":
    PRINT TAB( 5) "5...LOW RES COLOR TEST"
380 PRINT : PRINT TAB( 5)"PRESS ESC TO END TEST"
390 GOSUB 8000: IF ASC (A$) = 27 THEN GOTO 50
400 IF A$ = "1" THEN GOTO 470
410 IF A$ = "2" THEN GOTO 540
420 IF A$ = "3" THEN GOTO 660
430 IF A$ = "4" THEN GOTO 730
440 IF A$ = "5" THEN GOTO 860
465 PRINT CHR$ (7): GOTO 390

470 REM DISPLAY TEST
475 HOME : PRINT TAB( 14)"DISPLAY TEST": PRINT
480 PRINT "ENTER ANY CHARACTER TO FILL SCREEN": PRINT : PRINT "PRESS ESC TO END TEST"
490 GOSUB 8000
495 HOME
500 IF ASC (A$) = 27 THEN GOTO 350
510 FOR I = 1 TO 959
520 PRINT A$;: NEXT I
525 IF EX = 1 THEN GOTO 680
530 GOTO 490

540 REM ALIGNMENT TEST
550 GR : POKE - 16302,0: CALL - 1998: COLOR= 15
560 FOR I = 1 TO 45 STEP 8
```

```
570 HLIN 0,39 AT I - 1
575 HLIN 0,39 AT 47
580 NEXT I
590 FOR I = 1 TO 40 STEP 8
600 VLIN 0,47 AT I - 1
610 NEXT I
620 VLIN 0,47 AT 39
640 GOSUB 8000
650 TEXT : GOTO 350

660 REM CHARACTER SET TEST
670 HOME : PRINT TAB( 11)"CHARACTER SET TEST":
    PRINT : PRINT "PRESS ESC TO END TEST": PRINT
680 FOR I = 0 TO 128
690 PRINT CHR$ (I) + " ": NEXT I
695 IF EX = 1 THEN GOTO 750
700 GOSUB 8000
710 GOTO 350

720 REM SCROLL TEST
730 HOME : PRINT TAB( 15)"SCROLL TEST": PRINT
740 INPUT "ENTER NUMBER OF REPETITIONS-";A
750 IF A < 35 THEN A = 35
760 N = 31
770 FOR I = 1 TO A
780 N = N + 1: IF N > 71 THEN N = 32
790 FOR X = 1 TO 40
800 PRINT CHR$ (N);:N = N + 1
810 IF N > 71 THEN N = 32
820 NEXT X
830 NEXT I
840 IF EX = 1 THEN RETURN
850 PRINT : PRINT "PRESS ESC TO END TEST": GOSUB 8000: GOTO 350
860 REM LOW RES COLOR GRAPHICS TEST
870 HOME : PRINT TAB( 10)"LOW RES COLOR GRAPHICS TEST": PRINT
872 PRINT "VALID COLORS ARE:": PRINT : PRINT "BLACK";
    TAB(13)"MAGENTA"; TAB( 24)"DARK BLUE": PRINT
    "VIOLET"; TAB( 13)"DARK GREEN"; TAB( 24)"GRAY 1"
    4 PRINT "MEDIUM BLUE"; TAB( 13)"LIGHT BLUE";
    TAB( 24)"BROWN": PRINT "ORANGE"; TAB( 13)"GRAY 2";
    TAB( 24)"PINK": PRINT "GREEN"; TAB( 13)"YELLOW";
    TAB( 24)"AQUA": PRINT "WHITE"
875 PRINT
```

```
876 PRINT "PRESS ANY KEY TO CONTINUE": PRINT "OR PRESS
    ESC TO END TEST": GOSUB 8000: IF ASC (A$) = 27 THEN GOTO 350
878 PRINT
880 INPUT "ENTER COLOR-";C$
890 IF C$ = "BLACK" THEN C = 0: GOTO 1060
900 IF C$ = "MAGENTA" THEN C = 1: GOTO 1060
910 IF C$ = "DARK BLUE" THEN C = 2: GOTO 1060
920 IF C$ = "VIOLET" THEN C = 3: GOTO 1060
930 IF C$ = "DARK GREEN" THEN C = 4: GOTO 1060
940 IF C$ = "GRAY 1" THEN C = 5: GOTO 1060
950 IF C$ = "MEDIUM BLUE" THEN C = 6: GOTO 1060
960 IF C$ = "LIGHT BLUE" THEN C = 7: GOTO 1060
970 IF C$ = "BROWN" THEN C = 8: GOTO 1060
980 IF C$ = "ORANGE" THEN C = 9: GOTO 1060
990 IF C$ = "GRAY 2" THEN C = 10: GOTO 1060
1000 IF C$ = "PINK" THEN C = 11: GOTO 1060
1010 IF C$ = "GREEN" THEN C = 12: GOTO 1060
1020 IF C$ = "YELLOW" THEN C = 13: GOTO 1060
1030 IF C$ = "AQUA" THEN C = 14: GOTO 1060
1040 IF C$ = "WHITE" THEN C = 15: GOTO 1060
1050 PRINT : PRINT CHR$ (7): PRINT "INVALID COLOR! - RE-ENTER": PRINT : GOTO 880
1060 GR : POKE - 16302,0: CALL - 1998
1070 COLOR= C/1080 FOR Y = 0 TO 47
1090 FOR X = 0 TO 39
1100 PLOT X,Y: NEXT X
1110 NEXT Y
1120 GOSUB 8000: TEXT : GOTO 870

1130 REM PRINTER DIAGNOSTIC MODULE
1140 HOME : PRINT TAB( 11)"PRINTER DIAGNOSTIC":
     PRINT : PRINT TAB( 3)"1...SLIDING ALPHA TEST"
1150 PRINT TAB( 3)"2...CHARACTER SET PRINT TEST":
     PRINT TAB( 3)"3...ECHO CHARACTER PRINT TEST"
1160 PRINT TAB( 3)"4...HORIZONTAL TAB TEST": PRINT TAB( 3)"5...LINE FEED TEST": PRINT
1170 PRINT TAB( 10)"PRESS ESC TO END TEST"
1180 GOSUB 8000: IF ASC (A$) = 27 THEN 50
1190 IF A$ = "1" THEN GOTO 1280
1200 IF A$ = "2" THEN GOTO 1430
1210 IF A$ = "3" THEN GOTO 1520
1230 IF A$ = "4" THEN GOTO 1610
1240 IF A$ = "5" THEN GOTO 1680
1245 PRINT CHR$ (7): GOTO 1180
1250 HOME : PRINT TAB( 13)"PRINTER SETUP"
```

```
1260 PRINT : PRINT "ENTER PRINTER CARRIAGE LENGTH:": INPUT "80 OR 132-";W
1265 IF W <> 132 THEN W = 80
1270 PRINT "ENTER THE PRINTER CONTROLLER": INPUT "SLOT NUMBER (0-7)-";SL
1275 GOTO 1140

1280 REM SLIDING ALPHA TEST
1290 HOME : PRINT TAB( 11)"SLIDING ALPHA TEST": PRINT
1300 INPUT "ENTER THE NUMBER OF REPETITIONS-";X
1310 IF X < 35 THEN X = 35
1320 PR# SL
1325 PRINT "SLIDING ALPHA TEST"
1330 N = 31
1340 FOR I = 1 TO X
1350 L$ = "":N = N + 1: IF N > 111 THEN N = 32
1360 FOR A = 1 TO W
1370 L$ = L$ + CHR$ (N):N = N + 1
1380 IF N > 111 THEN N = 32
1390 NEXT A
1400 PRINT L$: NEXT I
1410 IF EX = 1 THEN PR# 0: GOTO 1440
1420 PR# 0: GOTO 1130

1430 REM CHARACTER SET PRINT TEST
1440 HOME : PRINT TAB( 6)"CHARACTER SET PRINT TEST": L$ = ""
1445 PR# SL: REM ACTIVATE PRINTER
1447 PRINT "CHARACTER SET PRINT TEST"
1450 FOR I = 32 TO 128
1460 IF N = W THEN PRINT L$:L$ = "":N = 1
1470 L$ = L$ + CHR$ (I) + " ":N = N + 2
1480 NEXT I
1490 IF LEN (L$) > 0 THEN PRINT L$
1500 IF EX = 1 THEN PR# 0: GOTO 1555
1515 PR# 0: GOTO 1130

1520 REM ECHO CHARACTER PRINT TEST
1530 HOME : PRINT TAB( 8)"ECHO CHARACTER PRINT TEST"
1540 PRINT : PRINT TAB( 8)"ENTER CHARACTER TO ECHO":
     PRINT : PRINT TAB( 8)"PRESS ESC TO END TEST"
1550 GOSUB 8000: IF ASC (A$) = 27 THEN GOTO 1130
1555 L$ = ""
1560 FOR I = 1 TO W
1570 L$ = L$ + A$: NEXT I
1575 PR# SL: PRINT "ECHO CHARACTER PRINT TEST"
1580 PRINT L$: PR# 0
```

```
1590 IF EX = 1 THEN GOTO 1620
1600 GOTO 1530

1610 REM HORIZONTAL TAB TEST
1620 HOME : PRINT TAB( 10)"HORIZONTAL TAB TEST"
1625 PR# SL
1627 PRINT "HORIZONTAL TAB TEST"
1630 FOR I = 1 TO W
1640 PRINT TAB( I)"*": NEXT I
1650 PR# 0
1660 IF EX = 1 THEN GOTO 1690
1670 GOTO 1130

1680 REM LINE FEED TEST
1690 HOME : PRINT TAB( 13)"LINE FEED TEST"
1695 PR# SL: PRINT "LINE FEED TEST"
1700 PRINT "THIS IS THE FIRST LINE....."
1710 FOR I = 2 TO 5
1720 FOR A = 1 TO I
1730 PRINT CHR$ (10);: NEXT A
1740 N$ = STR$ (I)
1750 PRINT "THERE HAVE BEEN ";N$;" "LINE FEEDS";: NEXT I
1760 PRINT CHR$ (12);: PRINT "THIS IS A NEW PAGE....."
1770 PR# 0
1780 IF EX = 1 THEN RETURN
1790 GOTO 1130

2500 REM DISK DRIVE DIAGNOSTIC MODULE
2510 HOME :D$ = CHR$ (4):H$ = "":L$ = ""
2515 PRINT TAB( 10)"DISK DRIVE DIAGNOSTIC": PRINT
2520 FOR I = 1 TO 239
2530 H$ = H$ + CHR$ (64):L$ = L$ + CHR$ (63): NEXT I
2540 IF EX = 1 THEN GOTO 2590
2550 PRINT "ENTER DISK DRIVE CONTROLLER": INPUT "SLOT NUMBER (0-7)-";S
2560 IF S < 0 OR S > 7 THEN S = 6
2570 PRINT : INPUT "ENTER DRIVE NUMBER (1 OR 2)-";D: PRINT
2580 IF D < 1 OR D > 2 THEN D = 1
2590 B$ = H$:N$ = "HIGH VALUES"
2600 ONERR GOTO 3000
2605 FOR I = 1 TO 2
2610 PRINT : PRINT "WRITING ";N$;: TO DISK"
2620 PRINT D$;"OPEN TEST,S";S;",D";D
2630 PRINT D$;"WRITE TEST"
2635 FOR L = 1 TO 430
```

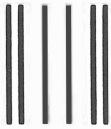
```
2640 PRINT B$: NEXT L
2650 PRINT D$;"CLOSE TEST"
2670 PRINT : PRINT D$: PRINT "READING ";N$;" FROM DISK": PRINT
2675 PRINT D$;"OPEN TEST,S";S;",D";D
2680 PRINT D$;"READ TEST"
2685 FOR L = 1 TO 430
2690 INPUT B1$
2700 IF B1$ < > B$ THEN PRINT D$: PRINT "ERROR READING ";N$:E = E + 1
2710 NEXT L
2720 PRINT D$;"CLOSE TEST"
2730 PRINT D$;"DELETE TEST"
2740 B$ = L$:N$ = :LOW VALUES": NEXT I
2750 PRINT D$: PRINT : PRINT "THERE WERE ";E;" ERRORS
     ENCOUNTERED": PRINT "DURING TEST"
2760 IF EX = 1 THEN POKE 216,0 : RETURN
2770 PRINT : PRINT "PRESS ESC TO END TEST":
     GOSUB 8000: POKE 216,0: GOTO 50
3000 ER = PEEK (222)
3002 IF ER = 5 THEN GOTO 2720
3005 PRINT D$;"CLOSE TEST": PR# 0: IN# 0
3010 PRINT : PRINT TAB( 5)"A FATAL ERROR WAS ENCOUNTERED!" : PRINT
3020 PRINT "ERROR CODE:";ER: PRINT
3030 IF ER = 4 THEN PRINT "THE DISKETTE IS WRITE-PROTECTED": GOTO 3090
3040 IF ER = 6 THEN PRINT "FILE NOT FOUND": GOTO 3090
3050 IF ER = 7 THEN PRINT "VOLUMN MISMATCH": GOTO 3090
3060 IF ER = 8 THEN PRINT "I/O ERROR - TRY A NEW DISKETTE"
3070 IF ER = 9 THEN PRINT "OUT OF SPACE ON DISK - TRY AN EMPTY ONE": GOTO 3090
3080 IF ER = 11 THEN PRINT "NO BUFFERS AVAILABLE -
     REBOOT SYSTEM": PRINT "AND RUN DIAGNOSTIC AGAIN"
3090 PRINT : PRINT "PRESS ESC TO CONTINUE": GOSUB 8000
3100 GOTO 2750

3500 REM SERIAL COMMUNICATION DIAGNOSTIC MODULE
3510 HOME : PRINT TAB( 7)"SERIAL COMMUNICATION DIAGNOSTIC" : PRINT
3512 PRINT "ENTER SERIAL COMM INTERFACE": INPUT "SLOT
     NUMBER (0-7)-";SL: IF SL < 0 OR SL > 7
     THEN SL = 2
3513 PRINT : PRINT "WIRE PINS 2 AND 3 TOGETHER":
     PRINT "OR INSTALL A LOOPBACK CONNECTOR":
     PRINT "BEFORE PROCEEDING WITH TEST": PRINT :
     PRINT "PRESS ANY KEY TO CONTINUE": GOSUB 8000
3515 S = 0: GOTO 3532
3520 B$ = ""
```

```
3530 GET A$: IF ASC (A$) = 27 THEN GOTO 50
3532 HOME : PRINT TAB( 7)"SERIAL COMMUNICATION
     DIAGNOSTIC": PRINT: PRINT TAB( 9)"PRESS ESC TO
     END TEST": PRINT: PRINT TAB( 7)"TRANSMITTED
     RECEIVED ASCII"
3534 PRINT TAB( 8)"CHARACTER CHARACTER VALUE": PRINT
3536 IF S = 1 THEN GOTO 3540
3538 S = 1
3540 PR# SL
3550 PRINT TAB( 12)A$;
3560 IN# SL
3570 GET B$
3580 IF B$ = "" THEN GOTO 3520
3590 PR# 0
3600 PRINT TAB( 24)B$; TAB( 32) ASC (B$): GOTO 3520

4000 REM SYSTEM EXERCISER
4010 HOME :PS = 0:EX = 1: PRINT TAB( 10)"SYSTEM EXERCISER": PRINT
4020 INPUT "TEST MONITOR? (Y OR N)-";M$: PRINT
4030 INPUT "TEST PRINTER? (Y OR N)-";P$
4040 IF P$ = "Y" OR P$ = "y" THEN GOTO 4050
4045 GOTO 4060
4050 PRINT " ENTER PRINTER CARRIAGE LENGTH":
     INPUT " (80 OR 132)-";W: IF W <> 132 THEN W = 80
4055 INPUT " ENTER PRINTER CONTROLLER SLOT NUMBER-":
     INPUT " (0-7)-";SL: IF SL < 0 OR SL > 7 THEN SL = 1
4060 PRINT : INPUT "TEST DISK DRIVE(S)? (Y OR N)-";
     DD$(0)
4070 IF DD$(0) = "N" OR DD$(0) = "n" THEN GOTO 4120
4080 INPUT " HOW MANY DRIVES TO TEST-";DD(0)
4090 FOR I = 1 TO DD(0)
4100 INPUT " ENTER DRIVE CONTROLLER SLOT NUMBER":
     INPUT " (0-7)-";SS(I): IF SS(I) < 0 OR SS(I) > 7 THEN SS(I) = 6
4105 INPUT " ENTER DRIVE NUMBER (1 OR 2)-";DD(I):
     IF DD(I) < 1 OR DD(I) > 2 THEN DD(I) = 1
4110 NEXT I
4120 IF M$ = "N" OR M$ = "n" THEN GOTO 4140
4130 A$ = "X":A = 50: GOSUB 510
4140 IF P$ = "N" OR P$ = "n" THEN GOTO 4160
4150 A$ = "E":X = 35: GOSUB 1320
4160 IF DD$(0) = "N" OR DD$(0) = "n" THEN GOTO 4200
4170 FOR Q = 1 TO DD(0)
4180 S = SS(Q):D = DD(Q): GOSUB 2500
```

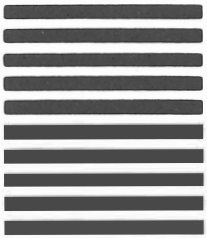
```
4190 HOME : NEXT Q
4200 PS = PS + 1: GOTO 4120
4300 HOME : PRINT D$;"CLOSE TEST": PRINT D$: PRINT
     TAB( 10)"SYSTEM EXERCISER": PRINT
4310 PRINT "TOTAL PASSES THROUGH EXERCISER-";PS
4320 PRINT "PRESS ESC TO END TEST": GOSUB 8000:
     EX = 0: POKE 216,0: GOTO 50

5000 REM GAME PADDLE DIAGNOSTIC MODULE
5005 HOME : PRINT TAB( 12)"GAME PADDLE DIAGNOSTIC": PRINT
5010 PRINT "ENTER THE GAME PADDLE TO TEST": PRINT "0, 1, 2, OR 3":
     PRINT "OR PRESS ESC TO END TEST"
5020 GOSUB 8000
5030 IF ASC (A$) = 27 THEN GOTO 50
5040 A = VAL (A$)
5050 IF A > 3 THEN PRINT CHR$ (7): GOTO 5020
5055 PRINT : PRINT "PRESS FIRE BUTTON TO END TEST"
5060 GR : COLOR= 13: HLIN 10,29 AT 19
5070 B = PEEK (-16287 + A)
5080 IF B > 127 THEN GOTO 5140
5090 LET X = PDL (A):X = X * 20 / 256 + 10
5100 IF X = Z THEN GOTO 5070
5110 COLOR= 0: PLOT Z,20
5120 COLOR= 13: PLOT X,20
5130 LET Z = X: GOTO 5070
5140 TEXT : GOTO 5000

8000 GET A$: RETURN
```

See page 259 for a Special Companion Diskette Offer.


```
170 PRINT : PRINT CHR$ (7);: PRINT "INVALID COLOR! - RE-ENTER": GOTO 80
180 HGR : POKE - 16302,0: HCOLOR= C
```



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

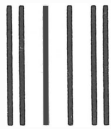


BUSINESS REPLY MAIL

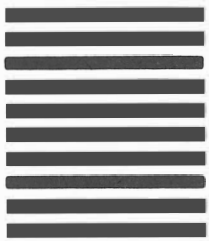
FIRST CLASS PERMIT NO. 9 BLUE RIDGE SUMMIT, PA 17214

POSTAGE WILL BE PAID BY ADDRESSEE

TAB BOOKS Inc.
Blue Ridge Summit, PA 17214-9988



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

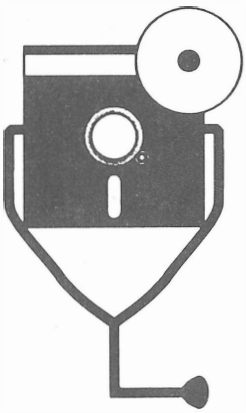


BUSINESS REPLY MAIL

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

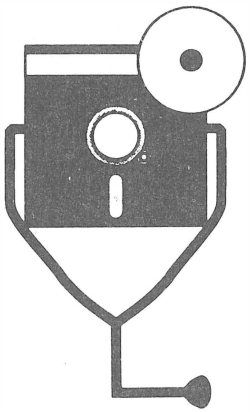


Appendix B

High-Resolution Color Graphics Test Program Listing

```
1 REM APPLE ][ SERIES
5 REM HIGH-RES COLOR GRAPHICS TEST
8 REM COPYRIGHT 1988 TAB BOOKS INC.
10 HOME : PRINT TAB( 5)"HIGH RES COLOR GRAPHICS TEST": PRINT
20 PRINT "VALID COLORS ARE:": PRINT
30 PRINT "BLACK 1"; TAB(9)"GREEN"; TAB(17) "VIOLET"
40 PRINT "WHITE 1"; TAB(9)"BLACK 2"; TAB(17) "ORANGE"
50 PRINT "BLUE"; TAB(9)"WHITE 2"
60 PRINT : PRINT "PRESS ANY KEY TO CONTINUE": PRINT "OR PRESS ESC TO END TEST": PRINT
70 GET A$: IF ASC (A$) = 27 THEN END
80 INPUT "ENTER COLOR-";C$
90 IF C$ = "BLACK 1" THEN C = 0: GOTO 180
100 IF C$ = "GREEN" THEN C = 1: GOTO 180
110 IF C$ = "VIOLET" THEN C = 2: GOTO 180
120 IF C$ = "WHITE 1" THEN C = 3: GOTO 180
130 IF C$ = "BLACK 2" THEN C = 4: GOTO 180
140 IF C$ = "ORANGE" THEN C = 5: GOTO 180
150 IF C$ = "BLUE" THEN C = 6: GOTO 180
160 IF C$ = "WHITE 2" THEN C = 7: GOTO 180
170 PRINT : PRINT CHR$ (7);: PRINT "INVALID COLOR! - RE-ENTER": GOTO 80
180 HGR : POKE - 16302,0: HCOLOR= C
```

```
230 FOR Y = 0 TO 191
240 FOR X = 0 TO 279
250 HPLOT X,Y: NEXT X
260 NEXT Y
270 GET A$: TEXT : GOTO 10
```



Glossary

A

AC connector A three-pronged cable that runs from the computer power supply to the AC electrical outlet.

AC electronics Alternating current. An electric current that periodically reverses its direction. The type of current supplied at the AC electrical wall outlet.

AC strip See power strip.

access arm The mechanical disk drive component that follows instructions from the disk drive controller to position the drive head over the proper track location on the diskette, allowing it to read and write data.

accumulator A temporary register within the microprocessor that stores the result of an arithmetic or logic operation.

adder The primary arithmetic circuit of the CPU's arithmetic-logic unit. The adder outputs the sum of two or more numbers presented as inputs. The adder also works with the accumulator and other storage registers to perform addition, subtraction, multiplication, and division functions.

alignment The visual placement of columns and rows on the visual display monitor.

analog signal A continuous, varying physical signal, such as sound waves, light readings, or voltage. Compare with digital signal.

antistatic mat A piece of insulating material on which computer devices are placed to prevent electrostatic discharge.

Applesoft BASIC An extended version of the BASIC high-level programming language, permanently stored in ROM on the Apple II series personal computer.

application program Software designed for a specific purpose such as word processing, inventory control, or accounts payable.

arithmetic-logic unit Sometimes referred to as the ALU. It is the circuitry within the microprocessor that handles all addition, subtraction, multiplication, division, and comparison functions. Registers, accumulators, adders, and comparers comprise the arithmetic-logic section of the CPU.

ASC An Applesoft BASIC instruction that returns the ASCII value of the first character in a string.

ASCII value American Standard Code for Information Interchange. A code from 0 to 127 is assigned to text characters, and is used to exchange information among data processing and communication systems. One byte of an ASCII value represents an alphabetic, numeric, special, or control character.

B

backup A copy of a program or data file on a separate diskette, which you make to protect yourself against accidental loss of programs or data.

bad sector A portion of a diskette that has become defective. See sector.

BASIC An acronym for Beginners All-Purpose Symbolic Instruction Code. An easy-to-learn, easy-to-use, high-level, programming language most often used with personal computers.

baud rate The speed at which a communicating device transmits information, often, but not always, the number of bits that can be transmitted in one second. Typical baud rates for the Apple II systems are 300 and 1200.

bit Abbreviation for binary digit: either of the numbers 1 or 0. 1 indicates an ON condition, 0 indicates an OFF condition. The bit is the smallest unit of data or storage in a computer. See byte and digital.

bit map The series of ON and OFF pixel representations used to paint characters and graphics on a high-resolution screen. See also pixel.

branch A term used to denote the change of program control from one pathway in the code to another pathway that is not sequential. The program control can branch forwards or backwards, and is often caused by a GOTO or GOSUB statement.

bridge An inadvertent connection on a circuit board between two conductive paths.

buffer A temporary storage area used to hold data until it is needed at a later point.

bug A fault in a computer program which causes undesirable results such as the

program crashing or hanging up, the program doing something other than the expected function, or the program screen not displaying correctly.

burn-in A continuous testing mode that exercises all system functions, particularly useful because most computer problems occur when it is first set up, or when the computer is moved.

byte A unit of data consisted of a fixed number of bits. The Apple II uses a byte size of eight bits, and can hold any value from 0 to 255. The byte is smallest unit of data addressable by the computer.

C

cable The wire with a connector on both ends that runs from one electrical assembly to another.

cable connectors The plug on either end of a cable, used to connect into a receptacle on an electrical assembly such as the computer system unit or printer, to connect the two units together. Connectors are usually characterized as either male or female, with the male having connector pins, and female having connector sockets.

cable pins The metal pins in a male cable connector.

capacitors An electrical device consisting of conductive plates separated by an insulating material. Used for storing an electrical charge. Also known as a condenser.

characters per second (cps) The rate at which a printer places characters on paper.

carriage The mechanism which carries the ribbon, print element, and print head assembly across the length of the platen while printing a line.

carriage width The extent of the printer carriage, measured in maximum columns available. Printers are either 80 or 132 columns wide.

carrier The tone received in a communication system before data can be sent. When the receiving modem is called, the carrier is sounded just before the communication connection is established.

cathode ray tube (CRT) The video display screen on which visual output of computer programs and data can be viewed. The CRT consists of a large vacuum tube with an electron gun and viewing face. Images are produced on the screen coated with phosphors that emit light when struck by a focused electron beam.

central processing unit (CPU) The computer circuitry which controls the interpretation and execution of instructions. Calculations are made, processes are run, input/output functions are coordinated, data is processed and transferred through the arithmetic-logic unit and general control circuits in the microprocessor, and through primary storage sections in the system memory RAM and ROM chips.

character generator A ROM which defines the character set that can be drawn on the CRT. It produces the bit map that creates each character as accessed through the keyboard or by a program.

character length The number of bits, usually seven or eight, making up a standard character or word to be used in communication between two computers. In the Apple II, the character length is eight bits. Same as word length.

character set The full set of characters available for display on the CRT. The character set of the Apple II systems consist of 128 characters.

character string code value (CHR$) A numeric representation used in BASIC programs to access a symbol or function not directly accessible from the keyboard. Directly corresponding to ASCII values. See also ASCII value.

chip See integrated circuit chip.

CHR$ value See character string code value.

clockwise The direction of a round object to be turned. It indicates that the item should be turned toward the right, in the same direction as the hands turning on the face of a clock.

code The lines of instruction to the computer that comprise a computer program.

COLOR The Applesoft BASIC command that uses a numeric representation to set the color for the low-resolution graphics mode.

command An instruction that causes the computer or peripheral device to perform a particular action. A command can be typed from the keyboard or entered within a program.

communication The exchange of data and information between two computers or between a computer and a peripheral device such as a printer.

comparer A special arithmetic circuit within the microprocessor which makes comparisons of one number to another for logic functions.

configuration The manner in which a computer system is set up, including the subsystems employed and the parameters used within each of the subsystems.

connector See cable connector.

connector symbol The round flowchart symbol used to transfer flow among pages of a lengthy chart, usually identified with a number or letter, and connected with a matching connector symbol by the same designation elsewhere in the flowchart.

contacts The exposed wires, or legs, of the IC chip plugged into a printed circuit board. These wires receive electrical current from the power supply and send them into the chip's circuitry.

control section The portion of the CPU that selects, interprets, and oversees the execution of computer program instructions.

controller A separate circuit on the motherboard, or a separate printed circuit board plugged into an expansion slot, which directs the activities of a peripheral device such as the keyboard, disk drive, or printer.

counterclockwise The direction of a round object to be turned. Indicates that the items should be turned toward the left, opposite the direction that the hands on the face of a clock turn.

CPU See central processing unit.

crash See head crash and system crash.

CRT See cathode ray tube.

D

daisy wheel The round print element used in letter-quality printing. On the end of each petal is a character that is part of the character set provided by the entire daisy wheel. The printer motor spins the daisy wheel to position the desired character, and then the print hammer strikes the character against the ink ribbon to form the character on the paper.

daisy-wheel printer See letter-quality printer.

data A collection of computer instructions or records. A representation of facts, concepts, numbers, letters, symbols, or instructions. The raw material, or source, of information. When compiled, processed, or refined by the user in a logical format, the data can create information in a human-usable form. See information.

data size See character length.

DC electronics Direct current. Electric current which flows in one direction. Used by the power circuits in computer systems.

decision symbol The diamond-shaped flowchart graphic representing a place within a computer program at which the computer must make a decision based on logic/comparison operations also performed by the computer. Exit paths from the decision symbol are then determined based on the outcome of the decision.

denatured isopropyl alcohol See isopropyl alcohol.

device driver See printer driver.

diagnose To determine the cause of a computer malfunction.

digital signal A format of discrete (as opposed to continuous) electrical impulses of ON (+5 volts) and OFF (0 volts). When the signal is ON, is it interpreted as a 1 bit. When the signal is OFF, it is interpreted as a 0 bit. The computer operates with digital signals, meaning that it interprets all computer programs and data as a series of 1s and 0s. Contrast with analog.

digitizer tablet A tablet used in computer drafting and graphics to create line images on a high-resolution CRT screen. It converts an analog quantity (the graphic lines) into a digital format for use in the computer.

DIP-switch Acronym for dual inline package. A type of IC package that contains a number of microswitches. Each microswitch can be set ON or OFF. The collection of microswitches can each control a different setting, mode, or

configuration; or the entire switch can have a specified ON/OFF sequence of the microswitches that indicate a particular setting. See switch setting.

directionality A communication parameter indicating whether the device with which the computer is exchanging data can send as well as receive data. Printers are unidirectional communication devices, because they can receive data from the computer, but not send data back. Computers connected to modems are bidirectional communication devices, because they can send as well as receive data.

disk See diskette.

disk drive The computer subassembly which provides a means for high-speed storage and retrieval for programs and data, recording data in electronic form on floppy diskettes.

diskette The recording media used in microcomputer disk drives consisting of thin, flexible plastic coated with magnetic iron oxide emulsion, and encased in a protective plastic covering. The Apple II diskette has a capacity of 143 kilobytes of memory.

disk drive controller The circuit in the Apple II system unit to which the disk drive connects and by which it is operated. The drive controller instructs the disk drive when and how to read and write information, as directed by the CPU and software commands.

disk drive mechanism The mechanical assembly of motor, head, and access arm which allows the disk drive to read data from the diskette and to write data onto the diskette.

display matrix The column/row configuration of the CRT screen. The Apple II has five available display matrices. The text mode works within a matrix of 40 columns by 24 rows, as well as 80 columns by 24 rows. The low-resolution graphics mode works within a matrix of 40 columns by 48 rows. The high-resolution monochrome graphics mode works within a display matrix of 280 by 192 pixels. The high-resolution color graphics mode works within a display graphics mode of 140 x 192 pixels.

downtime The amount of time during which a computer system is not operable due to a malfunction.

dot matrix The method of creating a character using a matrix of usually seven or nine dots. This dot matrix is used both within a system's character generator to draw characters on the screen, as well as in a dot-matrix printer to print characters on paper.

dot matrix printer A printer using a dot matrix as an array of small wires in a single print head to create characters for printing on the paper. When a character is sent to the printer, the shape of the character is punched through the wires and through the ink ribbon to create its impression on the paper.

drive See disk drive.

drive motor The mechanical motor that spins the diskette at a constant speed of 300 to 400 revolutions per minute.

driver See printer driver.

Dvorak keyboard A specialized keyboard layout often preferred for high-speed typing.

E

echo To display a character entered at the keyboard, either on the monitor or the printer.

edge connector The conductive metal tab extended from the edge of a printed circuit board that is used to plug the board into an expansion slot in the system unit. The edge connector allows the current flowing from the system unit power supply and through the motherboard and the expansion slots to be routed into the circuitry of its PC board.

electrical outlet The wall socket which provides the AC supply.

electron beam The focused stream of electrons collected and emitted by the CRT electron gun to contact the back of the picture tube to create a small, clearly defined spot of light.

electromechanical A mechanical (physical) device or operation activated or regulated by electric circuitry.

electronic components Devices such as integrated circuits, resistors, capacitors, and diodes, usually found on a PC board.

emulator A communication program which causes the computer to similate the code sequences, keyboard functions, and monitor functions of another computer of a different design. Used in communication to enhance compatibility between different types of computers. The DEC VT100 is a common emulation mode used in communication.

emulsion The thin film of iron-oxide coating on a diskette which provides the magnetic properties necessary for data to be stored on the diskette.

END A BASIC instruction that terminates the program and returns the computer back to the root BASIC environment.

environment An operating mode in a computer, the program currently running on the computer. For example, the operating system environment, the BASIC environment, the System Diagnostic Program environment.

etch To burn an image permanently into the screen display by leaving one particular static image displayed on the monitor for long periods of time.

expansion slot A connection on the motherboard into which interface controller boards can be plugged for additional functionality and control of peripheral devices.

F

file A named collection of related program or information data operated on as a unit, and usually stored on a diskette.

flag A program variable whose contents (usually 1 or 0) indicate whether a condition is true or whether an event has occurred.

floppy diskette See diskette.

flowchart A graphic programming diagram to help programmers design and map each step of the program process and direction, indicating actions to be performed within the computer to produce the desired results. This flowchart is then used to help write the actual program code.

flowcharting symbols Graphics used within a flowchart to represent global processes taking place in the program. The input/output, process, decision, terminal, and connector flowchart symbols are the most commonly used flowchart symbols.

form feed A software command or printer button that causes the printer to roll the paper forward to the top of the next page of continuous form paper.

format See initialize.

forms tractor A device with paper clamps and guide bars that connects to the printer to feed continuous form paper around the platen.

FOR-NEXT A two-part BASIC instruction that allows a series of instructions to be performed in a loop for a specified number of times before proceeding to the subsequent instruction in the program code. The FOR statement opens the loop and establishes the counter and the number of loops to be run. The lines following the FOR statement comprise the body of the loop, and are executed until the NEXT statement is encountered. The NEXT statement closes the loop and returns program control back to the FOR statement.

G

game paddle An electromechanical input device which consists of a tracking wheel controlled by a potentiometer, and a fire button controlled by a switch. The game paddle is most frequently used in computer game software to track objects and fire projectiles.

GET A BASIC instruction that reads a single keystroke and stores it into a designated variable.

GOSUB-RETURN A two-part BASIC instruction that causes program control to branch to and return from a designated subroutine elsewhere in the program. When the GOSUB instruction is encountered, the program branches to the specified line number. Program execution takes over at that subroutine until the RETURN statement is encountered, at which time program control returns to the location in the code immediately following the GOSUB statement.

GOTO A BASIC instruction that causes the program to branch to the specified line number within the program code.

GR An Applesoft BASIC instruction that switches the monitor display from the current display mode to the low-resolution color graphics mode.

ground A conductor with zero potential voltage, used as the reference point for all potential voltage in the system. When a conducting object has no potential voltage, electrical charges cannot run through it or be attracted to it by a difference in potential.

H

handshaking protocol A series of communication data signals used to establish modem communication. Such signals include Carrier Detect, Request to Send, Data Set Ready, and Clear to Send.

hardcopy Computer output provided on a paper copy via a printer.

hardware Physical computer equipment, such as the system unit, monitor, printer, IC chips. Contrast with software.

hard-wire A direct cable connection between the serial communication ports of two computers, or between any two computer devices within close proximity of one another. A hard-wire connection is made to allow the communication and sharing of data without the need for a modem.

HCOLOR An Applesoft BASIC instruction that uses a numeric representation to set the color for the high-resolution graphics mode.

head The magnetic device on the end of the disk drive mechanical access arm that travels microns over the diskette. The head is used to magnetize the diskette emulsion to write data onto a diskette, and to sense an existing magnetic pattern to read data from a diskette.

head-arm assembly The disk drive component that consists of the drive head and access arm. This assembly follows instructions from the disk drive controller to locate, read, and write data to and from the diskette. See head and access arm.

head crash A catastrophic disk drive malfunction in which the head comes into contact with the diskette, usually caused by the head being out of adjustment, by a warped diskette, or by contamination within the disk drive.

Hertz The unit of frequency of vibration or oscillation, equal to one cycle per second. The current provided the AC outlet alternates at a rate of 60 Hertz, changing polarity 60 times per second.

HGR An Applesoft BASIC command that switches the monitor display matrix from the current mode (usually the text mode) to the high-resolution color graphics mode.

high-resolution graphics mode The display mode that has a display matrix of 140 x 192 color pixel locations, and 280 x 192 monochrome pixel locations.

HLIN The Applesoft BASIC instruction used in either the low-resolution or high-resolution graphics modes to draw a horizontal line from one specified point to another at a specified row location.

HOME The Applesoft BASIC instruction that clears the screen and positions the cursor at the top left corner of the screen.

HPLOT The Applesoft BASIC instruction used in the high- resolution graphics mode to draw a pixel in the current color at the specified coordinates.

I

IC chip See integrated circuit chip.

IF-THEN The two-part BASIC conditional instruction that makes a decision based on the truth or fallacy of a specified statement. When the IF statement is true, the THEN statement is executed. When the IF statement is not true, the THEN statement is not executed, and program execution proceeds to the subsequent instruction.

IN The Applesoft BASIC instruction that switches input from the keyboard to the device whose controller resides in the specified expansion slot number.

information A collection of data organized in such a manner that it produces a complete idea, situation, condition, or other meaning assigned by humans. See also data.

initialize The process of dividing a diskette into the tracks and sectors on which the computer can write and read data. On the Apple II, diskette initialization, (also known as formatting), is done with the INIT command. The initialization program also writes operating system data and small utility programs onto the diskette, and blocks out any defective portions of the diskette. See also track, sector, and bad sector.

input The process of entering data from an external source into the computer for further processing. Typical input devices include the keyboard, mouse, and game paddle. Typical input data can include characters in a word processing program, numbers in a spreadsheet program, or records in a database program.

INPUT The BASIC instruction that prompts and waits for user input from the keyboard during program execution. When the user input is received, the data is placed into an indicated variable.

input/output symbol The parallelogram flowchart graphic representing any function within a computer program that takes data into the computer, from an external device such as a keyboard, mouse, or disk drive, or sends data out from the computer to an external device such as the monitor, printer, or disk drive.

integrated circuit (IC) chip A device that contains a number of integrated electronic circuits on a thin silicon wafer substrate contained within a small

plastic or ceramic package, connecting to and obtaining power from a PC board with wire pins. The integrated circuit chip is the building block of the computer. Often referred to as IC, or chip.

integrated program testing A method for checking the quality, integrity, and functionality of a new software program after it has been integrated with a larger program.

integration The appending of a program module into an existing program, editing lines of code so that they are using one another's resources and subroutines, and so that code line references are correct. Integration also includes checking that the two programs work together as one without adversely interfering with one another's functions.

interface A shared boundary between two systems or devices, for example, the connection of a computer to a peripheral device via a cable. Often there is an interface board to provide circuitry for the connected device.

interface cable The cable used to connect the computer to a peripheral device. The cable runs from a port at the system unit to a receptacle on the peripheral device, providing the pathway from the controller circuits to the device.

I/O (input/output) Pertaining to functions allowing data into the computer and sending data out of the computer for computer-human communication. See input and output.

I/O port A receptacle which allows the connection of input devices (mouse, game paddle) as well as output devices (printer, modem).

iron oxide A substance with magnetic properties covering the surface of the diskette as a thin film emulsion. The iron-oxide provides the magnetic properties necessary for data to be stored on the diskette.

isopropyl alcohol A quickly drying liquid solvent used to clean computer components and assemblies. This is not rubbing alcohol.

J

joystick An electromechanical input device with a directional pointer and a fire button which are controlled by a series of switches. The joystick is most frequently used in computer game software to point out and track objects and to fire projectiles.

K

keyboard The primary computer input device. A typed key generates an ASCII code which is sent to the CPU for interpretation, processing, and display on the monitor screen.

kilobyte 1024 bytes. Computer system memory and diskette capacity are generally measured in kilobytes. See byte.

L

letter-quality printer An impact printer that uses a daisy wheel or similar print element that contains individually formed characters to print characters on the page. The printer motor spins the daisy wheel to position the desired character, then the print hammer strikes the character against the ink ribbon to form the character on the paper. See also daisy wheel and dot-matrix printer.

line feed 1. A software command or printer button that causes the printer to roll the paper forward one line. 2. On the monitor, a keyboard entry that ends the current line and positions the cursor on the next line, also known as a carriage return.

listing See program listing.

load To transfer a program from disk to system memory.

location A position on the monitor screen as determined by the intersection of the column/row matrix. For example, in the text mode which has 40 columns and 24 rows, the intersection of any column and any row constitute one screen memory location. See also display matrix.

loopback connector A 25-pin serial plug with pins 2 and 3 wired together, used for testing the serial port.

loopback A serial communication interface test which causes a signal to be transmitted and immediately returned to the source.

low-resolution graphics mode The display mode that has a display matrix of 40 horizontal and 48 vertical screen locations which can display 16 different colors.

M

maintenance The process of keeping equipment in an operational condition. Refers to the repair of malfunctioning equipment as well as to those procedures performed on an operational system to prevent malfunctions.

mass storage Retention of programs and data on an external device such as a disk drive or cassette tape drive.

memory The retention of programs and data. Synonymous with storage. System memory is provided through the use of volatile random access memory (RAM) chips in system memory. Non-volatile internal storage of certain machine, operating system, and language programs are provided through the use of read-only memory (ROM) chips on the motherboard as part of the CPU. Long-term memory of other programs and user data is provided through the use of a mass storage device such as the disk drive or cassette tape drive. See also RAM, ROM, storage, system memory, and volatile memory.

microcomputer The personal computer, the smallest category of computer, consisting of a microprocessor, storage, and input/output devices.

micro-floppy A 3 ½-inch floppy diskette.

microprocessor The portion of the CPU that performs the arithmetic-logic and control functions for the computer. See also central processing unit, arithmetic-logic unit, and control section.

mini-floppy A 5 ¼-inch floppy diskette; the diskette used in Apple II system disk drives.

modem The communication device whose name is derived from the term MODulator-DEModulator. This device allows the transmission and reception of computer data between two computers across telephone lines. The modem at the transmitting computer converts the computer's digital signals into the analog sound waves which can be sent over the telephone lines (modulation). A second modem connected to the receiving computer reconverts the analog signals back into digital signals in order to feed the data into the computer (demodulation). See also analog and digital.

modular A systematic approach to writing a large software program by breaking it down into smaller sections, or modules, in order to simplify development, design, and coding efforts. The Apple II System Diagnostic Program is written, presented, and accessed in modules.

module testing A method for checking the quality, integrity, and functionality of the new software module before it is integrated with the larger program.

monitor See cathode ray tube.

motherboard The large printed circuit board within the system unit which contains the microprocessor chip, system memory chips, peripheral interface controllers, and expansion slots. All other boards receive control signals or information from the motherboard. Also known as the system board.

motor A machine that converts electrical energy into mechanical energy. This conversion process induces a magnetic field which can prove destructive to diskettes on which data is magnetically stored.

mouse A computer input device with a tracking ball on its underside and one to three buttons on top. The mouse is used to position a cursor, make selections, or create drawings on the monitor screen.

N

needs analysis The computer program development phase which consists of the research and subsequent determination of the particular problems to be solved or tasks to be accomplished by the program. The needs analysis phase defines program requirements before any program code is written.

NEW The BASIC instruction used within the BASIC root environment to indicate that subsequent keystrokes will constitute the start of a new computer program.

null An empty value assigned to a variable within a computer program. Setting a variable equal to null clears the variable.

O

ONERR GOTO The Applesoft BASIC instruction that disables the normal Applesoft error handling scheme in order to enable a different error handling routine within the current program.

online service A computer center of information to which computer users with modems can call and link into the service's mainframe computer. Online services provide a wide range of information for research purposes, bulletin boards to communicate with other users, and banking, shopping, airline reservation, and other consumer services.

operating system A collection of special control and utility programs that controls the overall operation of the computer.

output To send computer-processed data to another device external to the computer. The monitor is the primary output device, on which constant visual feedback of processing results is provided. Other output devices include the printer, for hardcopy output; the disk drive, for storage output written on a diskette; and the modem, for communication of data to another computer.

oxidation The buildup of a dull gray or black film on metal caused by contact with air and/or humidity. This buildup can reduce the conductivity of edge connectors and cable pins and ultimately cause computer malfunctions.

P

parallel interface A type of connection which transfers 7 or 8 bits of data simultaneously, each bit over a separate line or wire. Compare with serial interface.

parity A communication configuration parameter used as an error checking scheme to ensure accurate data transmission. When parity is employed, the last bit in a data word is set so that the total number of bits is either an odd number for odd parity, or an even number, for even parity. Parity is primarily used to detect noise errors in the communication line.

PC board See printed circuit board.

PDL An Applesoft BASIC instruction that returns a value representing the position of the game paddle control wheel. This value can range from 0 (extreme left) to 255 (extreme right).

PEEK A BASIC instruction that reads a byte from the specified computer memory location for further use within the program.

peripheral A computer input/output or storage device external to the computer system unit, for example, the printer, disk drive, or game paddle. Often connected to the computer by a cable, usually by means of a peripheral controller board. Also referred to as an I/O (input/output) device.

phosphor A coating on the CRT picture tube that glows when electrons emitted by the electron gun strike it. This reaction forms a clearly defined light spot. A number of these light spots serve to form letters and images.

pixel Picture element. A memory location on a high-resolution graphic screen that is turned either on or off in order to paint a graphic image or character on the screen.

platen The cylinder in the printer mechanism around which the paper rolls.

PLOT An Applesoft BASIC instruction used in the low-resolution color graphics mode to draw a block in the current color at the specified coordinates.

POKE A BASIC instruction that writes a specified byte into a specified computer memory location.

potentiometer A variable resistor that lets the user make an adjustment or set a position. Apple II pots include the color trim adjustment on the motherboard, and the color and brightness pots on the rear of the monitor. The game paddle wheel is also a potentiometer. See also variable resistor.

power strip An electrical assembly consisting of a row of several AC plugs and one AC connector that plugs into the wall outlet. The AC connectors for several electrical assemblies can then be plugged into the power strip to obtain electricity for each.

power supply The computer system unit component that takes the 110/120 volts AC from the electrical outlet and rectifies it to 5, 12, or 15 volts DC for use by the integrated circuits. The power supply consists of a transformer, rectifier, filter, and cooling fan.

PR The Applesoft BASIC instruction that switches output from the screen to the output device whose controller resides in the specified expansion slot number.

preventive maintenance A collection of systematic procedures that are performed on the computer as a part of periodic general system care. Such procedures help keep the system running properly and prevent breakdowns.

PRINT The BASIC instruction that displays the specified characters on the screen. If there are no characters indicated between the quotes after the PRINT statement, the line feed will be performed.

printed circuit board A flat, rectangular piece of rigid insulating material, usually fiberglass, onto which circuit paths are printed or etched. Electronic devices such as IC chips, resistors, and capacitors are plugged into the board, and the power supply routes current through the board's circuit paths to enable the devices. Also known as PC board or card.

printer The computer output device that produces data processing results on paper.

printer carriage See carriage.

printer driver A special utility program provided with application software to

configure the software for printing on a particular make of printer. This configuration process ensures that the application communicates properly with the printer so that data is received and interpreted correctly, and so that special printing functions such as boldface and italic can be used.

processing symbol The rectangular flowchart graphic representing a place within the computer program at which the CPU must perform a processing operation such as arithmetic or data movement functions.

program A set of sequenced computer instructions written in a computer language such as BASIC in order to cause the computer to accomplish certain tasks.

program analysis The computer program development phase in which the program is designed, using the system flowchart and program flowchart as design and coding tools. The analysis and flowcharting take the overall program logic and flow into consideration, and determine the procedures that must be taken to perform these steps to accomplish program objectives. See also system flowchart and program flowchart.

program code See code.

program flowchart A flowchart based on the system flowchart which draws and details the logic and sequence of specific program operations, such as branches, loops, decisions, inputs, and other structural and processing functions the program will require. See also flowchart and system flowchart.

program listing A printout of the complete program code. See also code.

program preparation The computer program development phase in which the code is actually written based on the system and program flowcharts that have been designed.

protocol A set of communication conventions and procedures that must be followed by system components in order for communication data transfer to successfully take place.

Q

quit A command often used within application programs to end program operation and return to the BASIC environment.

QWERTY The standard keyboard layout found on typical keyboards and typewriters; named for the first six alphabetic characters found on the second row of the typewriter.

R

random access memory (RAM) A storage chip in which programs and data can be directly and immediately retrieved for the user by its address, regardless of the physical location of the data. Contrasted with sequential access memory, in

which data is retrieved in the order it was saved to memory, causing a slower retrieval time. CPU system memory is provided through the use of volatile RAM chips on the motherboard.

RAM See random access memory.

RCA phono connector The round single-pin video connector used with an RF converter to create the monitor interface. One end of the RCA connector plugs into the system unit RF converter, and the other end plugs into the RF converter on the television.

read To retrieve programs and files from a diskette and place them into system memory for subsequent display and use.

register A labeled temporary storage location in the CPU that can retain a specific amount of data, used to access memory and perform arithmetic calculations.

REM A BASIC instruction which indicates a non-executable remark statement, provided as information and program documentation within the code for the benefit of the programmer and other readers of the code. It has no effect on the program execution itself.

reseating The action of disconnecting and then reconnecting a PC board into its slot, or a cable into its receptacle, to ensure a proper connection.

resolution The clarity of characters or images on the screen or printer, usually characterized by the number of dots or pixels used to create the image. The more dots used, the higher the resolution, and the sharper the image.

RETURN A BASIC instruction that is the second part of the GOSUB-RETURN statement. RETURN causes program control to return to the place in the code from which it branched. See also GOSUB-RETURN.

RF converter An electronic radio frequency unit that allows the use of a television as a computer monitor. The RF converter converts the computer's digital signals into the analog signals required by a television.

read-only memory (ROM) A memory device provided as part of the CPU in addition to RAM system memory. ROM chips have programs permanently stored in their memory, and will retain these programs regardless of whether or not the computer system has power. When the system is turned on, the program can be invoked immediately, without having to load it from an external storage device. Data cannot be written to ROM chips; they are pre-programmed, and can only be read. Typical programs stored in ROM include the operating system, input/output control, and BASIC.

ROM See read-only memory.

routine A portion of a program that performs a certain task which is subordinate to the overall task of the program, for example, the error-handling routine within the System Diagnostic Program.

RS-232 A interface cabling standard used to connect a modem with associated data terminal equipment.

run To invoke and execute a program stored in system memory to begin program operation.

S

screen memory A position on the monitor screen as determined by the intersection of the column/row matrix. For example, in the text mode which has 40 columns and 24 rows, the intersection of any column and any row constitute one screen memory location. See also display matrix.

scroll To move a screen completely filled with text so that another line or another page can be displayed.

sector A subdivision of one diskette track created during diskette initialization. Each Apple II diskette sector can store 256 bytes. Read and write functions use the sector as its smallest unit of operation. See initialize and track.

serial communication interface The computer subsystem consisting of the serial communication circuit, the serial port, and the interface cable. This interface provides the means for the Apple II to transfer data with other computers and devices. Through this interface, the computer can connect with a modem, printer, or other serial devices.

serial interface A type of connection which transfers data one bit at a time over a single line or wire. Compare with parallel interface.

short An accidental low-resistance connection between two points in an electric circuit, causing a deflection in the current flow which often damages the circuit.

slot See expansion slot.

socketed chip An IC chip that is plugged into an IC socket rather than directly into the PC board. Such a chip is easy to remove and replace using a chip extractor.

software Programs that operate on a computer system, controlling and determining the computer's activities. Contrast with hardware.

solder A tin lead alloy which is melted by a soldering iron and applied to metal parts and surfaces for joining the two together. IC chip wire legs are soldered to the underside surface of a PC board in order to create the necessary electrical connection.

soldered chip An IC chip that is connected directly into the PC board and held in place with the chip legs being soldered to the underside of the board.

solenoid An iron-core coil of wire through which current flows to create a controllable magnet for opening or closing mechanical contacts.

spike See voltage spike.

start bit A parameter used in communication configuration between two computers or devices to help coordinate a communication transfer. A start bit signals the beginning of a data character, flagging the attention of the receiving device so that it knows to expect seven or eight more bits of data.

static electricity The giving off of electrical charges that are contained in a particular object, or produced by friction, for example. Also known as electrostatic discharge.

STEP A BASIC instruction used in conjunction with a FOR-NEXT loop. If the loop counter is to increment by a number other than 1, the STEP statement specifies the increment for the counter.

stop bit A parameter used in communication configuration between two computers or devices to help coordinate a communication transfer. A stop bit signals the end of a data character, informing the receiving device that the data character is complete.

storage The storage section of the CPU consists of general-purpose system memory as performed by RAM chips, and stores the currently active program and the data being processed. Several RAM chips are provided on the motherboard, and the maximum amount of storage provided by the RAM chips constitute total system memory, for example 48k, 64k, or 128k. See also memory, RAM, and system memory.

subroutine A segment of a computer program that performs a complete function, and can be referenced and branched to from other locations in the code with a GOSUB-RETURN instruction.

surge protector An electronic device which is connected between the voltage source and the computer power supply in order to isolate and filter out voltage surges, thereby providing the system with a constant 110/120-volt supply and preventing the circuit damage that voltage surges can cause.

swapping The troubleshooting technique of exchanging a component or subassembly in one computer system onto another computer system to find out whether that subassembly is defective.

switch setting The current position of microswitches on a DIP-switch. If a switch setting is incorrect, it can cause a malfunction in the associated peripheral, or in the entire system. See DIP-switch.

synchronization hole The small hole located near the center of the diskette used by the drive circuitry for timing and location purposes. This hole lets the disk drive sense when the diskette is spinning at the correct speed for reading and writing data.

syntax The format in which a computer language instruction must be written in the code.

system An organized group of interrelated components, all working together to achieve a unified function and goal.

system crash A computer system failure in which all data currently residing in system memory is cleared and the system is placed into the machine monitor program environment.

system design The computer program development phase in which the problems to be solved and tasks to be performed are addressed in terms of how a computer program can accomplish these goals.

System Diagnostic Program The BASIC computer program provided with this book that includes tests for the various Apple II subsystems to help troubleshoot and locate system malfunctions.

system flowchart A flowchart based on the system design phase of computer program development, providing a plain English organizational overview of the program structure. See also flowchart and program flowchart.

system memory Storage of programs and data currently in use provided via RAM chips on the motherboard. Several RAM chips are provided on the motherboard, and the maximum amount of storage provided by the RAM chips constitute total system memory, for example 48k, 64k, or 128k. See also memory and RAM.

system unit The assembly which houses the motherboard, CPU, peripheral interface circuits, expansion slots, RAM and ROM chips, and power supply. The system unit controls and coordinates all activities taking place within the computer by issuing commands, processing input and output, and directing the activities of peripheral devices.

T

tab The horizontal positioning along the printer carriage.

TAB The BASIC instruction used in conjunction with the PRINT command to horizontally position the item to be printed at the specified column number, either on the monitor screen or along the printer carriage.

television set monitor A television being used as a computer monitor through the use of an external radio frequency (RF) converter and RCA phono connector.

terminal symbol The oblong flowchart graphic representing the beginning or end of a computer program.

testing A systematic method for checking the quality, integrity, and functionality of a new software program. The objective of software testing is to find program faults, or bugs.

TEXT The Applesoft BASIC instruction that switches the monitor display from one of the graphics modes to the 40-column x 24-row text mode.

track A circular subdivision of the diskette created during diskette initialization. An Apple II diskette is divided into 35 concentric circle tracks. See also initialize and sector.

troubleshoot To systematically determine and rectify a computer hardware or software malfunction.

tuning wand A non-conductive, usually plastic, screwdriver. Used instead of a metal screwdriver when working with components that might contain residual electrical currents.

V

VAL A BASIC instruction that translates a string value containing a number and returns the associated numeric value so that it can be used as a number, for example, in an arithmetic operation.

variable resistor An electronic component that controls an adjustable flow of current. As the resistor terminal is turned in one direction, more current is provided, and this causes a certain action to take place. As the resistor terminal is turned the other direction, less current is supplied, therefore causing a different or opposite action to occur. Often used for making adjustments or setting positions. See also potentiometer.

variable An assigned designation, such as K$ or PR, which can hold values that might change throughout the execution of a computer program. A particular value can be stored in a variable to be used elsewhere in program execution.

video controller The circuitry on the motherboard that directs the activities on the monitor. This circuit includes the character generator, display matrix information, color controls, and high-resolution graphic and low-resolution graphic mode controls.

video display See cathode ray tube.

VLIN An Applesoft BASIC instruction that is used in both low-resolution and high-resolution graphics modes to draw a vertical line from one specified point to another at a specified column location.

volatile memory Temporary memory storage within the RAM chips. The data stored in volatile memory remain present only as long as power is supplied to the computer. When the computer is turned off, the data in the volatile RAM chips in system memory are cleared. When the computer is powered on again, the RAM is empty, and the required software and data must once again be loaded into RAM from the disk drive or from the ROM chips.

voltage spike An instantaneous jolt of power from the AC outlet that far exceeds the normal 110/120-volt supply, so much so that the computer power supply cannot effectively suppress it from the damaging or destroying computer circuits.

voltage surge A temporary increase in electrical voltage lasting long enough to be noticed on a meter.

volt The unit of electromotive force, equal to the product of current and resistance.

W

word length See character length.

write-protect notch The small cutout on the side of the diskette which, when exposed, allows data to be written onto the diskette, but when covered, disables the write function. Write-protection is used to protect data from being accidentally overwritten.

write To record, or save, programs and files residing in system memory onto a diskette for storage and later use.

X

XON/XOFF protocol A communication configuration parameter which establishes a set of communication data signals and procedures which suspend (XOFF) and resume (XON) data transfer to accommodate the capacity of communication data buffers and prevent data from dropping off or overwriting.

Index

Index

Apple Care Manual

Diagnosing and Maintaining Your Apple® II+, IIe, and IIc Computers

Chris Morrison and Teresa S. Stover

A money-saving guide to preventive maintenance and repair you can do yourself!

If you're fed up with the expense, inconvenience, and computer down time that occurs every time something goes wrong with your Apple II system . . . but you're really not sure what to do about it or how to do it . . . this is the book you've been waiting for!

With this ingenious maintenance and troubleshooting manual, you will be able to keep your Apple II in tip-top working order, even if you're a complete novice at do-it-yourself electronic repairs. Stressing preventive maintenance, authors Morrison and Stover provide clear, easy-to-follow advice on diagnosing all kinds of computer problems and making a variety of simple repairs.

The authors discuss each part of the typical Apple IIe, IIc, and II+ systems, including the keyboard, monitor, printer, and disk drive. In addition, they've supplied an easy-to-use System Diagnostics Program written in BASIC that automatically checks the system components and reports on their status. You'll even find guidance in writing additional diagnostics for other peripherals not included in this guide.

To make sure your system repair really solved the problem, Morrison and Stover include an Exerciser Module in the System Diagnostics Program. A lot like test-driving a car after repairs, the Exerciser lets you check all system functions, confirm that all is in working order, and avoid recurring problems.

Chris Morrison is customer support manager for a computer-aided publishing manufacturer who has more than 15 years experience with personal computer systems. Teresa Stover is a technical writer and author of manuals for numerous Silcon Valley computer and electronics firms. Morrison and Stover are also the authors of TAB's *PC Care Manual: Diagnosing and Maintaining your MS-DOS, CP/M or Macintosh System.*

0389 $17.95
ISBN 0-8306-3121-6
90000
9 780830 631216

WINDCREST
Blue Ridge Summit, PA 17294-0850

3121

Apple Care Manual

Diagnosing and Maintaining Your Apple® II+, IIe and IIc

Morrison and Stover

WINDCREST