commodore
COMPUTER

Commodore Business Machines, Inc. — Computer Systems Division,
1200 Wilson Drive, West Chester, PA 19380

COMMODORE C128-40, COMMODORE B128-80, COMMODORE B256-80
USER'S GUIDE

commodore
COMPUTER

User's Manual Statement

"This equipment generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

— reorient the receiving antenna
— relocate the computer with respect to the receiver
— move the computer away from the receiver
— plug the computer into a different outlet so that computer and receiver are on different branch circuits.

"If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful: 'How to Identify and Resolve Radio-TV Interference Problems.' This booklet is available from the U.S. Government Printing Office, Washington, D.C. 20402, Stock No. 004-000-00345-4."

First Edition—1983
First Printing—1983

Printed in the United States of America

TABLE OF CONTENTS

PREFACE

This *User's Guide* introduces you to the C128-40, B128-80 and B256-80 personal computers. We've included instructions for Set-up, Optional Equipment and Configurations, Keyboard Operation, Loading and Saving Programs, and information about Available Software. The Appendices contain the Extended BASIC 4.0 definitions and abbreviations, screen and character codes, memory and register maps, mathematical functions, pinouts, BASIC conversions, error messages, and a bibliography.

Although the C128-40, B128-80 and B256-80 computers have a great deal in common, each has certain options and configurations which distinguish it from the other two. All of these differences are clearly labeled.

Once your system is set up and functioning properly, the chapters on Keyboard Operation and Loading and Saving Programs sections and the various Appendices will help you use your new system.

ORGANIZATION

This *User's Guide* is organized into six chapters.

Chapter 1

INTRODUCTION presents the highlights of each computer. In addition, it explains advantages of joining a User's Club and subscribing to the Commodore magazines and the Commodore Information Network.

Chapter 2

SETUP contains the instructions you need to unpack, connect, and install each of the computers. The C128-40, B128-80 and B256-80 are described in separate sections that help you get your machine up and running. The chapter also describes the wide variety of configurations and optional equipment (peripherals) available for each of the computers. Finally, this chapter presents a few "trouble shooting" and diagnostic procedures that can help you make adjustments to solve minor problems that may show up after you've installed a computer system.

Chapter 3

THE KEYBOARD AND ITS FEATURES takes you on a tour of the keyboard on the C128-40, B128-80 and the B256-80 computers.

Chapter 4

LOADING AND SAVING PROGRAMS tells you how to load and save both prepackaged software and your own custom designed programs. The chapter explains how this is done on both the Commodore DATASSETTE™ and the Commodore Disk Drives. For further details on the operating systems, we suggest you consult the manuals that come with your DATASSETTE™, 5¼ inch Floppy Disk Drive(s), or the fast and powerful Commodore Hard Disk Drive(s).

Chapter 5

AVAILABLE SOFTWARE briefly describes some of the great business, scientific, and educational software newly available for the C128-40, B128-80, and B256-80 computers. For a complete listing of all available software for the entire line of Commodore products, watch for the *Commodore Resource Encyclopedia*

coming soon to your local Commodore dealer and the computer departments of fine bookstores everywhere.

Chapter 6

APPENDICES are designed to be an expanded "quick reference" section of the major features that programmers and many end users want most. For an in-depth presentation of the information, we suggest that you check out the highly detailed *C128-40, B128-80 and B256-80 Programmer's Reference Guide.*

HOW TO USE THIS GUIDE

Let's take a look at the easy way to use this manual.

1. As you look at the edge of each page you will notice that there is what we call an "inset tab." The "inset tab" shows you exactly where the six chapters are located. Note that the beginning of each chapter is a solid blue page. Both of these features make it easy for you to get to the information you need quickly.

2. To help you unpack, hook up, set up, and begin operating your computer, Chapter 2 ("Setup") contains many detailed illustrations that can make the installation of your equipment, with all its options, a quick and easy task. In addition, for all C128-40 owners, there is a step-by-step explanation at the end of Chapter 2 on making the proper color adjustments to your TV set or monitor.

3. When we discuss a specific key, or want you to hit a particular key, we show you a visual cue (*Example:* RETURN).

4. Since this manual covers *three* separate computers, any section that does not pertain to all three systems will be clearly labeled.

5. **Please note that this manual is not designed to teach the computer language BASIC** (the primary language used in all Commodore computers). If you want to learn BASIC language programming techniques, or any of the other languages available for use with your computer(s), we suggest that you consult the "Bibliography" (Appendix R) for books that teach programming.

CHAPTER 1

INTRODUCTION

- Features Overview
- User's Clubs, Magazines, and the Commodore Information Network

Your membership in the ever-growing family of high-power computer users begins today, now that you own one of the *new generation* of personal computers from Commodore.

Each of these computers gives you the ability to design the ideal computer system for *your* business, scientific, or educational needs. Furthermore, as your needs change, your C128-40, B128-80, or B256-80 can be expanded to include the options you need to meet the demands of tomorrow's world.

All *new generation* computers have certain standard built-in features which work the same way on each machine. However, the special differences between the C128-40, the B128-80 and the B256-80 give you a choice of styles, memory capacities, and advanced color graphics and sound capabilities. The "Features Overview" that follows is a brief summary of the similarities and differences of the C128-40, B128-80 and B256-80 computers. You will also find a few examples of the wide variety of configurations you can create for your own computer. We're sure that you'll find hours of enjoyment ahead of you whether you're an experienced programmer or someone interested for now in using prepackaged software only. So take a few minutes to familiarize yourself with the many functions and applications you can perform with your new computer.

Finally we will look at the User's Clubs and why Commodore encourages their development. In addition, we explain the Commodore family of magazines, *Commodore* and *Power/Play*, and our paperless magazine, the *Commodore Information Network*.

FEATURES OVERVIEW

COMMODORE C128-40 Advanced Personal Computer

Memory

- 128K RAM expands to 256K internally, 704K externally, totaling 960K

Features

- Extended BASIC 4.0
- IEEE-488 bus
- RS-232C interface
- Dual 8 bit user ports
- 40 col. x 25 line screen display
- Works with user-supplied color monitor or TV with external modulator
- 16 colors
- Bit mapped graphics (320 x 200 dots (pixels))
- Two control ports for joysticks or paddles
- 48K ROM (24K internal and 24K external via cartridge)
- 6509 microprocessor
- Direct audio output
- 10 programmable function keys
- Separate calculator keypad for quick computations
- Optional future processors include Z-80 or 8088 for CP/M® and/or CP/M®-86
- Future languages will include U.C.S.D. Pascal

Commodore B128-80 Advanced Business Computer

Memory

- 128K RAM expandes to 256K internally, 704K externally, totaling 960K.

Features

- BASIC 4.0
- IEEE-488 bus
- RS-232C interface
- Dual 8 bit user ports
- 80 col. x 25 line screen display
- Integral display
- Easy read green phosphor screen
- Swivel and tilt built-in monitor
- Includes integral disk drive options
- Detachable keyboard

- 6509 microprocessor
- Direct Audio Output
- 10 programmable function keys
- (Version also to be available excluding integral disk and with external Video Display Unit option)
- Separate calculator keypad for quick computations
- Optional Z-80 and 8088 processors
- Optional CP/M®, CP/M®-86, and U.C.S.D. Pascal compatibility

Commodore B256-80 Advanced 16-bit Professional Computer

Memory

- 256K RAM, expandable externally to 960K (almost a megabyte online RAM memory!)

Features

- BASIC 4.0
- IEEE-488 bus
- RS-232C interface
- 8 bit user port
- 80 col. x 25 line screen display
- Integral display
- Easy read green phosphor screen
- Swivel and tilt screen
- Includes integral disk drive options
- Detachable keyboard
- 10 programmable function keys
- Built-in dual processors: standard 6509 processor, and 16-bit 8088 processor
- Direct Audio Output
- Separate calculator keypad for quick computations
- Optional processor includes Z-80 for CP/M®
- Optional 8087 math coprocessor
- Other future languages will include U.C.S.D. Pascal

USER'S CLUBS, MAGAZINES, AND THE COMMODORE INFORMATION NETWORK

Commodore wants you to know that our support for users only begins with your purchase of a Commodore computer. That's why we've created two publications with Commodore information from around the world, and a "two-way" computer information network full of valuable input by and for Commodore computer users in the U.S. and Canada from coast to coast.

In addition, we wholeheartedly encourage and support the growth of *Commodore User's Clubs* all over the globe. They are an excellent source of information for every Commodore computer user, from the beginner to the most experienced. The magazines and network, which are described below, have the most up-to-date information on how to get involved with the User's Club in your area.

Furthermore, your local Commodore dealer is an excellent source of Commodore support and information. Your dealer can always provide literature and hardware support to fill your changing computing needs.

Power/Play: The Home Computer Magazine

When it comes to entertainment, learning at home, and practical home applications, *Power/Play* is the prime source of information for Commodore computer owners. It directs you to the User's Club nearest you and tells you about its activities. It describes software, games, programming techniques, telecommunications, and new products. *Power/Play* is your personal connection to other Commodore users, outside software and hardware developers, and to Commodore itself. Published quarterly, it's only $10.00 for a whole year of home computing excitement.

Commodore: The Microcomputer Magazine

Widely read by educators, businesspeople, and students, as well as home computerists, *Commodore* is our main vehicle for sharing exclusive information on the more technical uses of Commodore systems. Regular departments cover the business, science, and education fields, programming tips, technical tips, and many other features of interest to anyone who uses, or is thinking about purchasing, Commodore equipment. *Commodore* is the ideal complement to *Power/Play*. It is published bimonthly, and a subscription costs only $15.00 per year.

Commodore Information Network

The magazine of the future is here today. To supplement your subscriptions to *Power/Play* and *Commodore* magazines, the *Commodore Information Network*—our "paperless magazine"—is available now. All you need is a Commodore computer, a telecommunications device called a modem, and your home or business telephone.

Join our computer club, get help with a computing problem, "talk" to other Commodore friends, or get up-to-the-minute information on new products, software, and educational resources. Soon you will even be able to save yourself the trouble of typing in the program listings you find in *Power/Play* and *Commodore* by "down loading" directly from the *Information Network*. The best part of the network is that most of the answers to your questions are there before you even ask them. How's that for service?

To "call" our electronic magazine you only need a modem and a subscription to CompuServe™, one of the nation's largest telecommunications networks. To make it easy for you to subscribe, Commodore will give you a *free* year's subscription to CompuServe™ inside every Vicmodem package.

Just dial your local number for the CompuServe™ data bank nearest you and then connect your phone to the modem. When the CompuServe™ video text appears on your screen, type "G CBM" on your keyboard. When the *Commodore Information Network's* table of contents, or "menu," appears on the screen, it's your turn to choose from one of our 16 departments. So make yourself comfortable, and enjoy the "paperless magazine" that all the other magazines are writting about.

For more information about the *Commodore Information Network* or about CompuServe™, visit your local Commodore dealer or contact CompuServe™ customer service at 1-800-848-8990 (in Ohio, 614-457-8600).

COMMODORE INFORMATION NETWORK

Main Menu Description
Direct Access Codes
Special Commands
User Questions
Public Bulletin Board
Magazines and Newsletters
New Product Announcements
Commodore New Direct
Commodore Dealers
Educational Resources
User Groups
Descriptions
Questions and Answers
Software Tips
Technical Tips
Directory Descriptions

CHAPTER 2

SETUP

- C128-40 Unpacking/Packing
- C128-40 Installation
- C128-40 Hookup and Configurations Available
- B128-80 and B256-80 Unpacking/Packing
- B128-80 and B256-80 Installation
- B128-80 and B256-80 Hookup and Configurations Available
- Expanding Your System (Peripherals)
- Color Adjustment for the C128-40
- Trouble Shooting

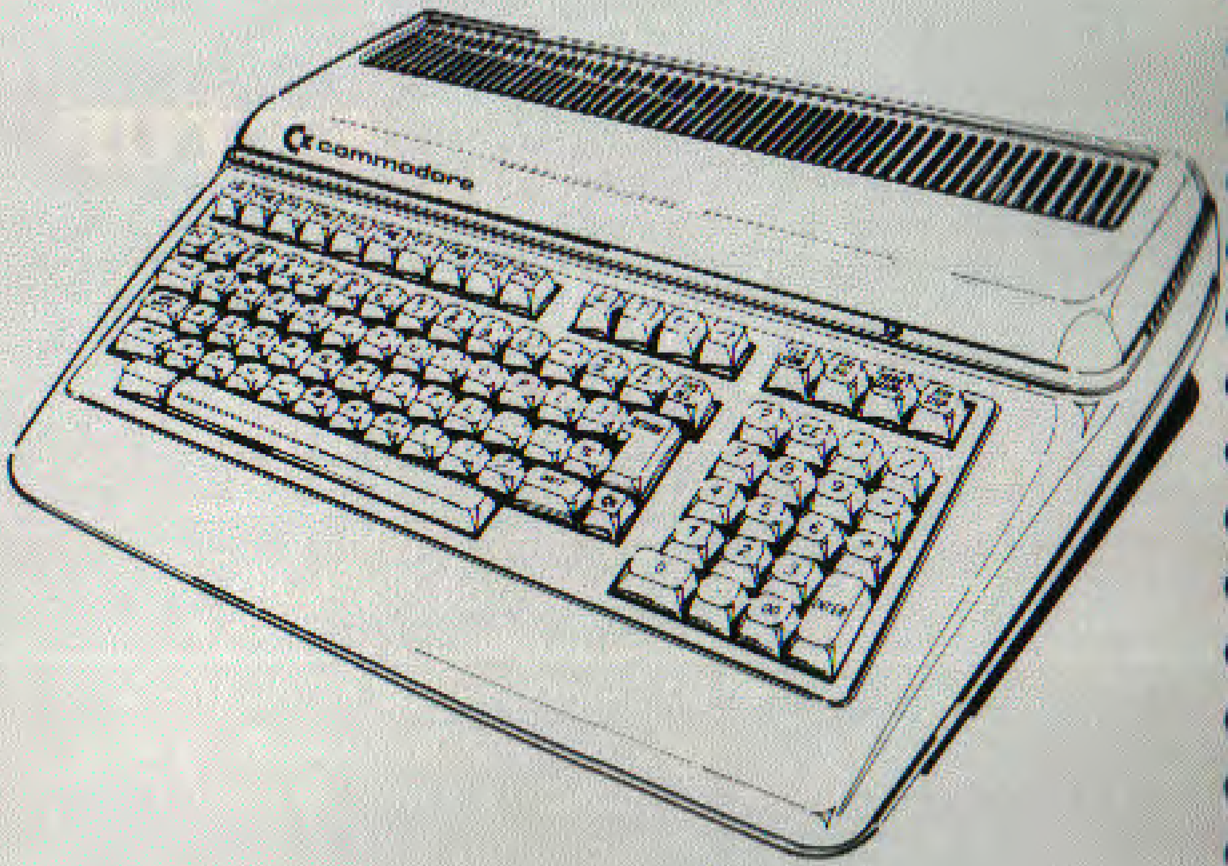


Fig. 1—Commodore 128 (Front view)

COMMODORE C128-40

Unpacking/Packing

The following step-by-step instructions show you how to connect your new computer to your monitor, television set, or sound system, and how to make sure everything is working properly.

Before attaching anything to the computer, check the contents of the container. Besides this guide, you should find the following items:

1. Computer
2. AC Power cord

If any items are missing or damaged, check back with your dealer immediately for repair or replacement. Be sure to save the packing materials for future storage or transport.

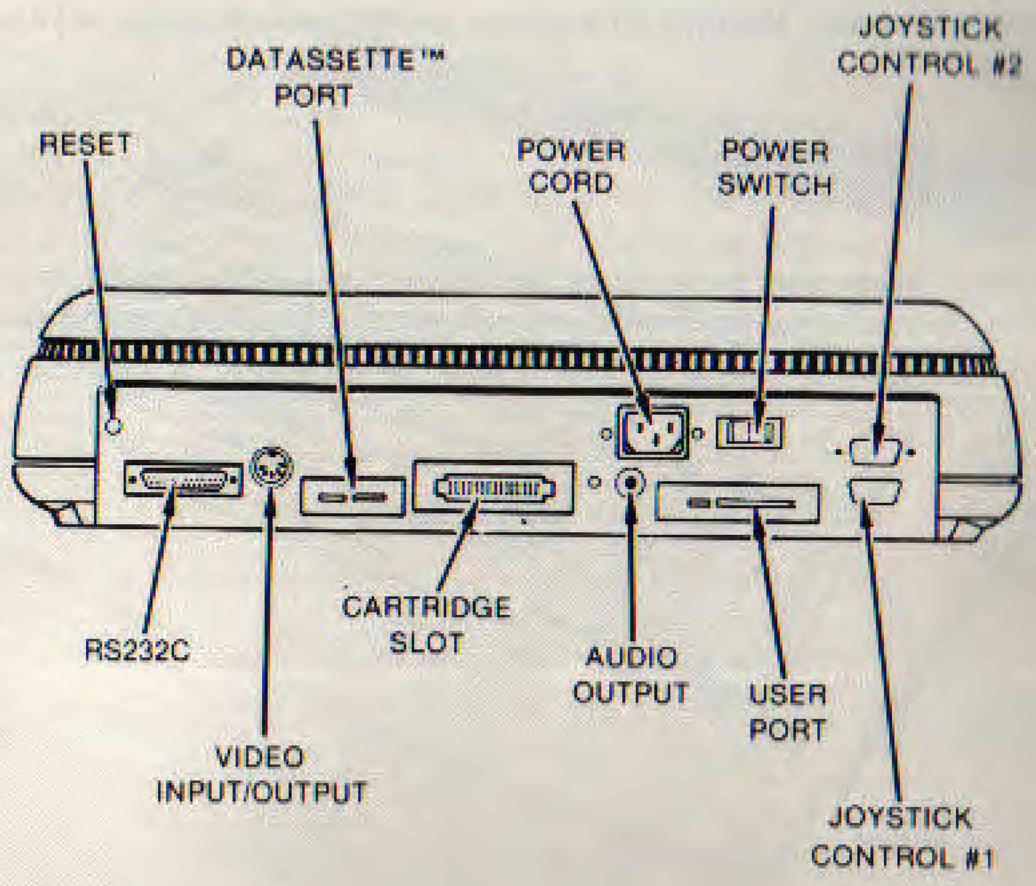


Fig. 2—Commodore 128 (Rear view)

Installation Commodore C128-40

Connect the computer to your TV as described below. See figure 2 to locate each input and output on the back of your C128-40.

1. Make sure that your computer and your video display monitor are turned **OFF** before starting your installation. C128-40 computers have their power switch located in the **back** of the machine on the **left** hand side.
2. Attach an RF modulator (not included) at the connector labeled Audio/Video (5–pin DIN). Line up the pins with the corresponding holes and push the connector in. The cable will only go in one way.
3. Attach one end of the TV cable to the RCA "phono"-type jack on the RF Modulator. Push in to connect the other end of the cable to the antenna switchbox.
4. If you have a VHF antenna, disconnect it from your TV set.
5. Connect your VHF antenna cable to the screen terminals labeled "Antenna Input" on the switchbox. If your antenna cable is the round 75-ohm coaxial type, use a 75-ohm to 300-ohm adapter (*not supplied*) to attach your antenna cable to the switchbox.
6. Connect the twin lead output cable of the antenna switchbox to the VHF antenna terminals of your TV set. If your set is one of the newer types with a round 75-ohm VHF connector, you will need a 300-ohm to 75-ohm converter (*not supplied*) to connect the switchbox to the 75-ohm VFH antenna input on the set.
7. Set the TV's VHF tuner to the channel number indicated on the computer's Channel Selector Switch (channel 3 or 4). If a strong local TV signal is present on one of these channels, select the other channel to avoid possible interference.
8. Plug the 3-prong AC power cord into a 120 volt, 60 Hz AC outlet.

Your C128-40 should now be connected properly. No additional connections are required to use the computer with your TV. The antenna switchbox will connect the computer to the TV when the slide switch is in the "Computer" position. When the switch is in the "TV" position your set will operate as a normal television. If you want to connect your C128-40 to a monitor, see installation instructions for B128-80 Low Profile on page 27.

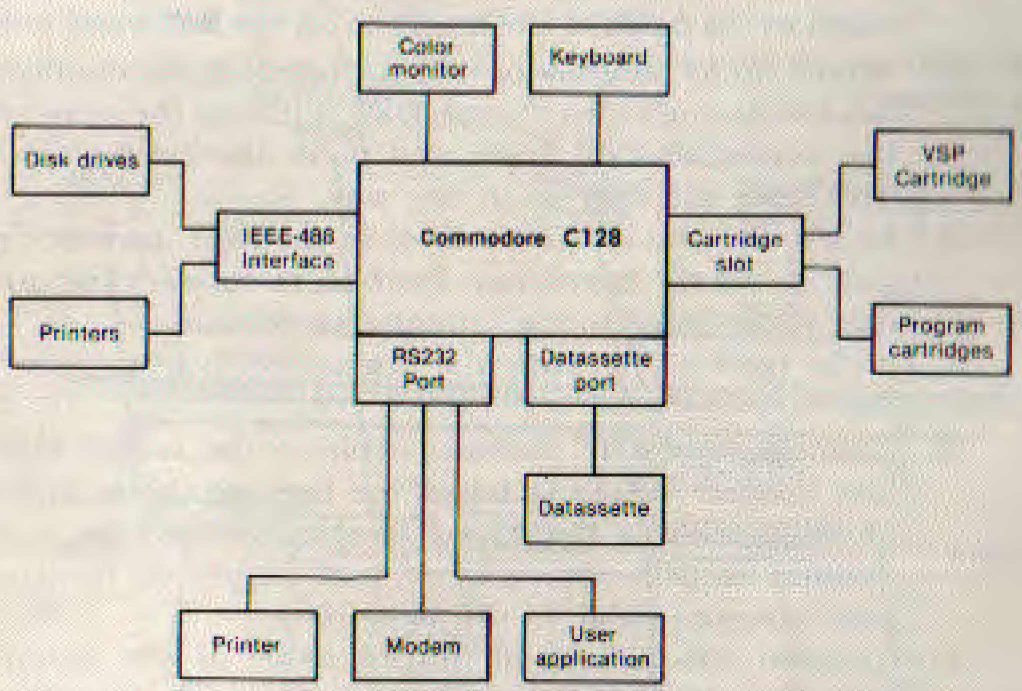


Fig. 3—Hook-up and Configurations available

COMMODORE B128-80 and B256-80

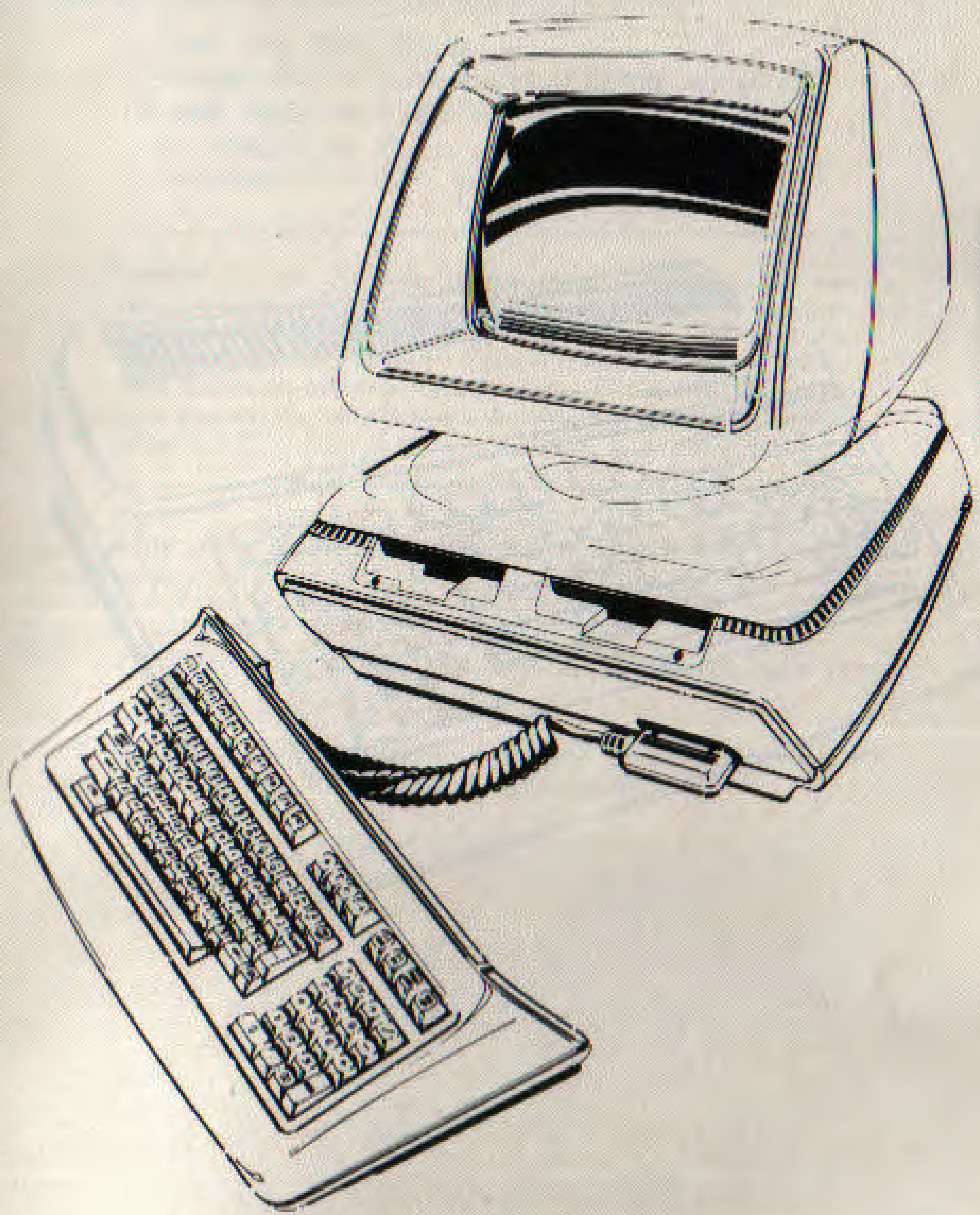


Fig. 4—B Series High Profile (Front view)

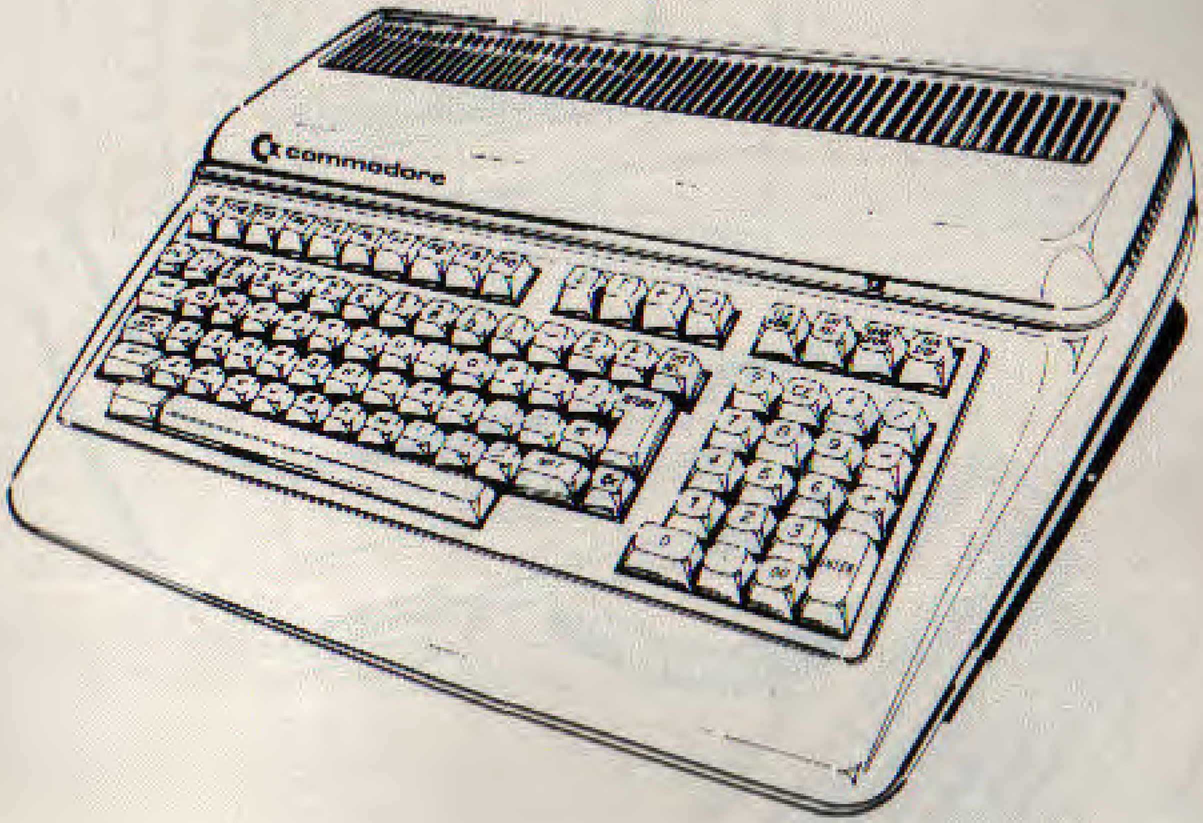


Fig. 5—B Series Low Profile (Front view)

Unpacking/Packing

Most B series computers are shipped in two parts:

1. Base and Video Display Screen (one unit) (known as the 'B' Series High Profile model) or

1a. Base alone (no Video Display Screen) (known as the 'B' Series Low Profile model)

2. Keyboard with cable attached

CAUTION: The base and display (Number 1 above) are connected to each other. The keyboard and base are to be connected with the "telephone-type" cable enclosed. If your computer has an attached Video Display Screen, it is **very important** that you **do not try** to remove or disconnect the video display screen from the computer base. If you must remove the screen from the base for any reason, take the entire unit back to your dealer for the appropriate modification.

In addition to the above items and this guide, you should find the following items:

3. AC Power Cord (120 volts)
4. Video Cable (8-pin DIN to RCA phono-type, Low Profile only)

If any items are missing or damaged, check back with your dealer immediately for repair or replacement.

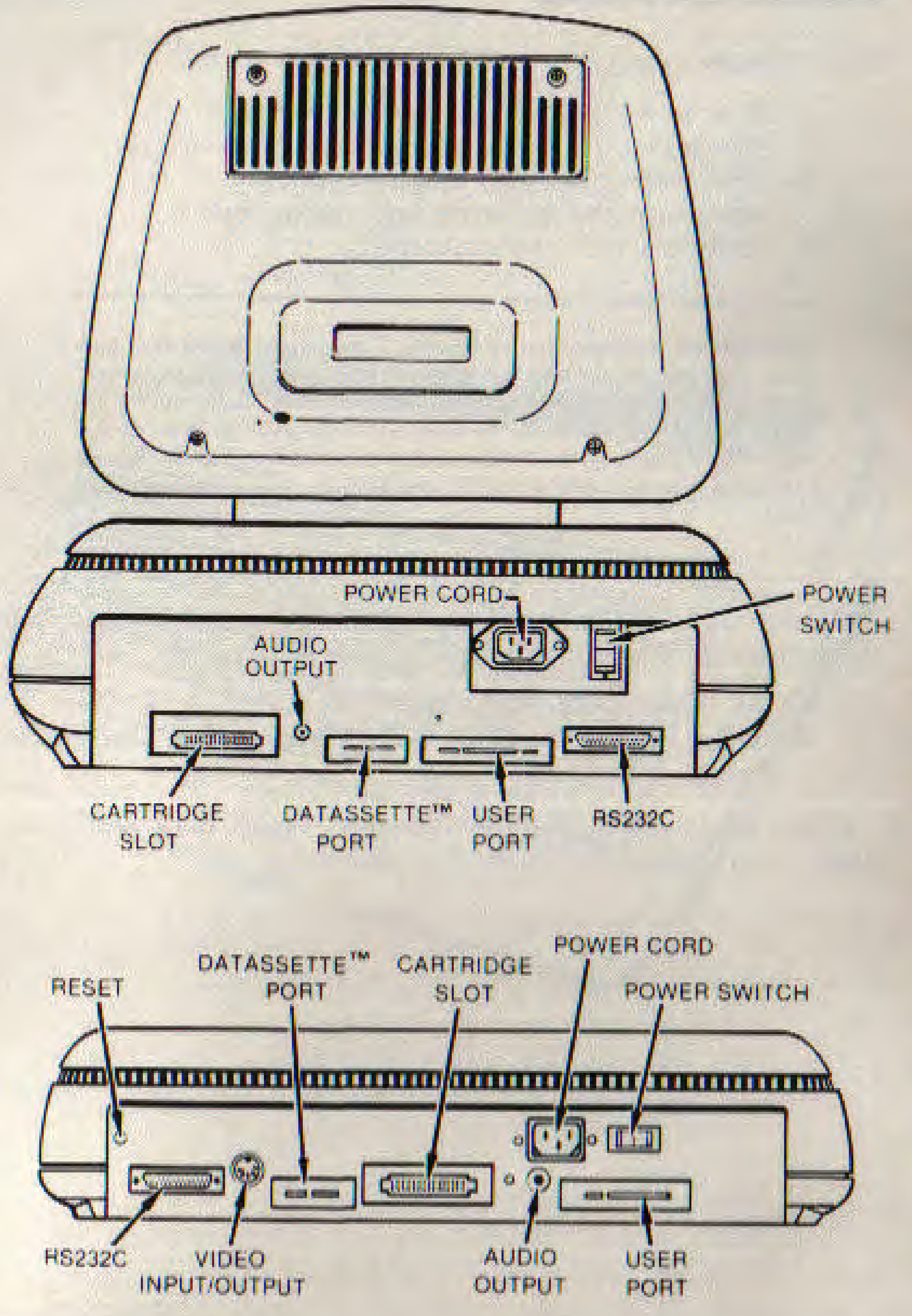


Fig. 6–B Series (Rear view)

Installation (B Series)

High Profile Model

1. Make sure that your computer is turned *off* before starting installation. The B Series High Profile computers have their power switch located in the **back** of the machine on the **left** hand side.
2. Plug the 25 PIN CABLE attached to the keyboard into the connector on the **lower right** hand side of the base/video display unit. *Make sure that the Commodore Logo is facing up.*
3. Plug the 3-prong AC power cord into the power cord jack located in the **back** of the base/video display unit, on the **left** hand side. The power cord fits only one way.
4. Plug the 3-prong AC power cord into a standard wall outlet.

Low Profile Model

Connect the computer to your monitor as described below. See figure 6 to locate each input and output on the back of your B Series computer.

1. Make sure that your computer and your monitor are turned *off* before starting installation. The 'B' series Low Profile computers have their power switch located in the **back** of the machine on the **left** hand side.
2. Attach the video cable to the computer at the connector labeled audio/video (5 pin DIN). Line up the pins with the corresponding holes and push the connector in. The cable will only go in one way.
3. Attach the two RCA phono-type jacks to your video monitor inputs. See the monitor's manual for instructions.
4. Plug the computer AC power cord into a 120 volt, 60 Hz AC outlet.

Your B Series Low Profile should now be connected properly. No additional connections are required to use the computer with your monitor.

NOTE: Save the packing materials that your computer came in. Then, to pack your equipment back in the box for storage or transit, reverse the procedures described above.

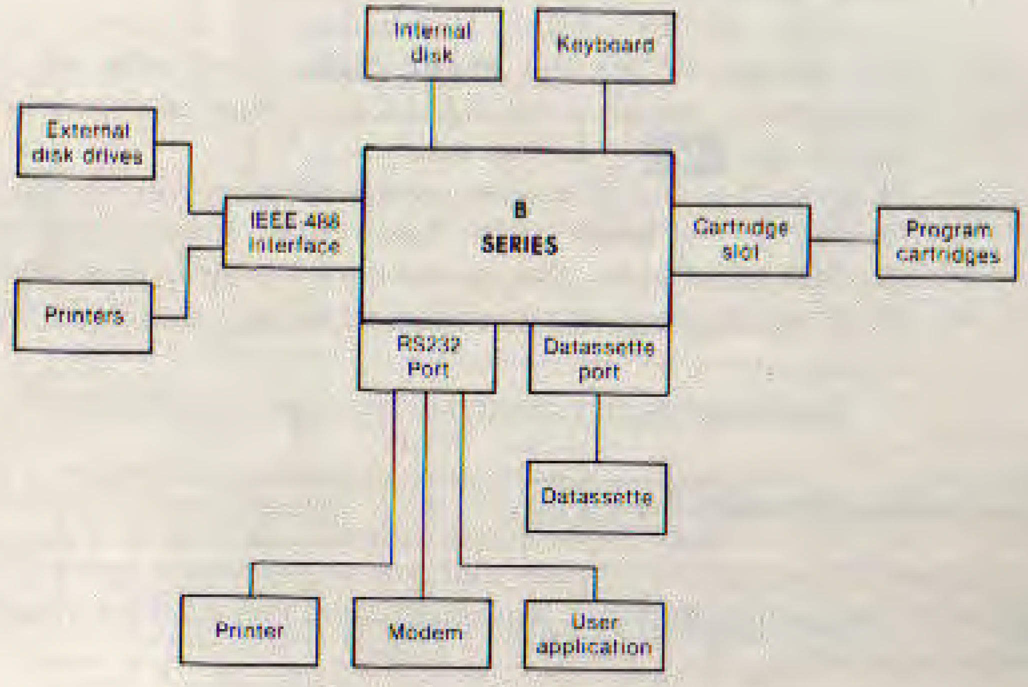


Fig. 7 Hook-up and Configurations available

Expanding Your System (Peripherals)

Printers

A full range of printers is available, designed to match any need. Low cost, high speed dot matrix units are ideal for most applications. Where letter quality printing is required, the Commodore "daisy-wheel" printer produces the best results.

Internal Disk Drive Units (B series only)

Commodore offers you the option of purchasing a B128-80 or B256-80 computer with a pair of internal disk drives. These drives are built into the units just above the keyboard. The two internal disk drives together are capable of storing ½ million bytes of information. This is in addition to the memory capacity of your external disk drives. Internal drives are convenient space savers. Unlike external drives, these integral devices rely on the main Central Processing Unit (CPU) for operation.

External Disk Drive Units

Single or dual floppy disk units, with storage capacity from 170,000 characters to over 2 million characters, can be easily attached to store programs and data. Hard disk units with capacities of 5 and 7.5 million characters can also be used with equal ease.

Commodore's Datassette™ Recorder

This low cost tape unit lets you store programs and data on cassette tape and retrieve them later. The Datassette™ may also be used to load pre-written programs.

RS-232C Port

Your computer comes equipped with an industry-standard RS-232C serial interface. This interface provides you with access to a wide variety of peripherals, such as printers, terminals, modems, and data collection equipment.

The RS-232C interface is implemented using the fully programmable 6551 Asynchronous Communications Interface Adapter. With the 6551, you can program your RS-232C interface to match exactly the requirements of the device you're connecting to it.

The Extended BASIC 4.0 interpreter includes file level software support for the RS-232C interface channel. Open the RS-232C channel as you would any other file and access it with standard BASIC input/output statements. The RS-232C Port is device # 2. See Appendix Q.

CBM IEEE Port

Your advanced computer supports the full range of Commodore CBM peripherals via the built-in IEEE-488 interface. Most disk units are "intelligent," meaning they have their own microprocessor and memory. You can connect up to five disk drives at one time to your computer by "daisy chaining" them together through the IEEE-488 connector port.

IMPORTANT: The device numbers that are used with the IEEE port must be within the range of 4 to 31 inclusive.

Commodore CP/M®

Your B256-80 computer is equipped with an 8088 processor that allows you to access the existing library of CP/M-86 programs.

Commodore CP/M® is not a "computer dependent" operating system. Instead it uses some of the memory space normally available for programming to run its own operating system. Access to CP/M® makes even more prepackaged software available to you, in addition to the wide variety of Commodore software.

Color Adjustment for the C128-40

There is a simple way to get a pattern of colors on the TV so you can easily adjust the set. Even though you may not be familiar with the operation of the computer right now, just follow along, and you'll see how easy it is to use the C128-40.

First, look on the left side of the keyboard and locate the CTRL key. This stands for ConTRol and is used, in conjunction with other keys, to instruct the computer to do a specific task.

To use a control function, hold down the CTRL key while depressing a second key.

1. Hit the OFF/RVS key. This will put the characters in reverse video. Normally, a character is colored against the background. In reverse video, this is reversed, like the negative of a photograph.
2. Hold down the SPACE bar and a blue bar will move across the screen.
3. To try all 16 colors, hold down the CTRL key and *in order* press the calculator keypad keys 0 through 9 ? , / , − , CE , * , + while using the SPACE bar to create color bars across the screen.

0	Black	5	Green
1	White	6	Blue
2	Red	7	Yellow
3	Cyan	8	Orange
4	Purple	9	Brown
?	Light red	CE	Light green
/	Gray 1	*	Light blue
−	Gray 2	+	Gray 3

4. To return to the normal display mode: (a) press the SHIFT and OFF/RVS keys together, and (b) press the CTRL key and the calculator pad key 6 .
5. Hit SHIFT RETURN and you will be back to the usual character mode.
6. Adjust the color and tint controls on your TV so that the display matches the colors you've selected.
7. At this point everything should be adjusted and working properly. If things still don't look quite right, consult the "Trouble Shooting" section which follows.

Trouble Shooting

This section presents some of the minor problems you may encounter after you've connected the parts of your computer. If you come across a problem not listed here, your first step should be to check with your local Commodore dealer to help correct your difficulty.

C128-40

PROBLEM	CAUSE	SOLUTION
1. Power indicator light not on	Computer not on	Make sure power switch is in the on position
	Power Cable not plugged in	Check power socket for loose or disconnected power cable
	Power Supply not plugged in	Check connection with wall outlet
	Bad fuse in the computer	Take system to authorized dealer for replacement of fuse

C128-40 (Continued)

PROBLEM	CAUSE	SOLUTION
2. No picture on video screen	TV on wrong channel	Check other channel for picture (VHF TV channel 3 or 4 is used with RF modulator and standard TV. *If you're using a monitor with a 5-pin DIN cable, this solution is not appropriate)*
	Incorrect hookup	Computer connects to VHF antenna terminals on standard TV or to video and audio inputs on video monitor
	Video cable not plugged in Computer set for wrong channel	Check TV output cable connection Set computer for same channel you have selected on your TV
3. Random pattern on TV screen with cartridge in place	Cartridge not inserted properly	*Turn power off* and then reinsert the cartridge
4. Picture without color	Poorly tuned TV	Return TV set

C128-40 (Continued)

PROBLEM	CAUSE	SOLUTION
5. Picture with poor color	Bad color adjustment on your TV	Adjust the color, hue, and brightness controls on your TV
6. Picture with excess audio background noise	TV volume setting is up too high	Lower the TV's audio volume control
7. Picture is OK but you don't have sound	TV volume setting is down too low	Adjust the audio volume control
	Auxiliary output not properly connected	Connect the sound jack to the auxiliary input on your amplifier and then turn the amp. selector switch to the "Aux." position
8. Picture is too dark or too light	Brightness level is set incorrectly	Adjust the brightness control level on your TV, monitor, or built-in video display screen
9. Characters on the screen are hard to read	Contrast ratio between characters and background is too great or too small	Adjust the contrast control on your TV, monitor, or built-in Video Display Screen
10. It is impossible to write programs on the computer	The SHIFT LOCK key is down	Hit the SHIFT LOCK key so that it returns to the standard "up" position.

B128-80 AND B256-80

PROBLEM	CAUSE	SOLUTION
1. Power indicator light not on	Computer not on	Make sure power switch is in the *On* position
	Power cable not plugged in	Check power socket for loose or disconnected power cable
	Power supply not plugged in Bad fuse in Computer	Check connection with wall outlet Take system to authorized dealer for replacement of fuse
2. No picture on video screen	Incorrect hookup	Computer connects to video and audio inputs on video monitor.
	Video cable not plugged in	Check monitor output cable connection
3. Random pattern on monitor with cartridge in place	Cartridge not inserted properly	*Turn power off* and then reinsert the cartridge
4. Picture with excess audio background noise	Monitor volume setting is up too high	Lower the monitor's audio volume control
5. Picture is OK but you don't have sound	Monitor volume setting is down too low	Adjust the audio volume control

B128-80 AND B256-80 (Continued)

PROBLEM	CAUSE	SOLUTION
	Auxiliary output not properly connected	Connect the sound jack to the auxiliary input on your amplifier and then turn the amp. selector switch to the "Aux." position
6. The picture is too dark or too light	Brightness level is set incorrectly	Adjust the brightness control level on your monitor or built-in video display screen
7. Characters on the screen are hard to read	Contrast ratio between characters and background is too great or too small	Adjust the contrast control on your monitor, or built-in Video Display Screen
8. A buzzing noise is coming from inside the computer (fan is for units with built-in disk drives only)	Internal fan is loose or malfunctioning	Return the unit to your local dealer for the repair
9. It is impossible to write programs on the computer	The SHIFT LOCK key is down	Hit the SHIFT LOCK key so that it returns to the standard "Up" position.

CHAPTER 3

THE KEYBOARD AND ITS FEATURES

- Format Keys
- Editing Keys
- Function Keys
- Calculator Pad Keys

The keyboard is your most important means of communication with the computer. Now that you've got everything set up and adjusted, please take a few moments to familiarize yourself with the keyboard.

The keyboard is similar to a standard typewriter keyboard. However, a number of new keys control specialized functions. What follows is a brief description of the various keys and how they function.

Format Keys

RETURN and **ENTER**

These keys signal the computer to look at the information you typed and to enter that information into memory.

SHIFT

This key works the same way it does on a typewriter. Many keys are capable of displaying three letters or symbols. In the "normal" mode the standard alphabet of lowercase and uppercase characters are displayed. Use the SHIFT key to get the uppercase characters.

In the "graphics" mode, the uppercase and graphics characters are displayed, and the SHIFT key gives you the graphics character shown on the front of the key.

OFF/RVS

This key gives you access to the reversed image of all the available characters on your computer. Pressing the OFF/RVS key will give you the reversed characters. Holding down the SHIFT key and pressing the OFF/RVS key will put you back into unreversed video.

NORM GRAPH

This key gives you access to the *graphics* mode. Pressing the NORM/GRAPH key will give you uppercase characters and the shifted graphics characters. Press the NORM/GRAPH key a second time and your computer will return to the standard character set of uppercase and lowercase letters.

Editing Keys

No one is perfect and Commodore takes that into account. Editing keys let you correct typing mistakes and move information around on the screen.

Cursor Control Keys: ↓ ↑ ← →

The cursor control keys marked with down, up, left, and right arrows are located on the top right of the keyboard. The keys look like this: ↓ , for moving the cursor down, ↑ , for moving the cursor up, and ← , for moving the cursor left, and → , for moving the cursor right. The cursor keys have a special repeat feature that keeps the cursor moving until you release the key.

INS/DEL

If you hit this key, the cursor will move a space to the left, erasing (DEleting) the previous character you typed. If you're in the middle of a line, the character to the left is deleted and the characters to the right automatically move together to close up the space.

Holding down the SHIFT key while pressing the INS/DEL key allows you to insert information on a line. For example, if you noticed a typing mistake in the beginning of a line — perhaps you left out part of a name — you could use the cursor left key to move back to the error and then hit a shifted INS/DEL to make a space. Then just type in the missing letter.

CLR HOME

This positions the cursor at the "home" position of the screen, which is the upper left-hand corner. Holding down the SHIFT key while pressing the CLR/HOME key will clear the screen and place the cursor in the "home" position.

Function Keys

F1 **F2** **F3** **F4** **F5** **F6** **F7** **F8** **F9** **F10**

The ten function keys in the upper left position of the keyboard can be "programmed" to handle a variety of functions. They are numbered from 1 to 10 and each key can be defined in many ways to handle repetitive tasks.

F11 F12 F13 F14 F15 F16 F17 F18 F19 F20

You will not find these function keys labeled separately on your computer. In order to use this second set of function keys, hold down the SHIFT key while pressing one of the ten function keys marked F1 through F10 . In each case the function accessed will be the one whose number is 10 more than on the keyboard.

CTRL

Standing for CONTROL, this key allows you to set colors (on the Commodore 128 only), and perform other specialized functions. You hold the CTRL key down while depressing another designated key to set colors or get a control function.

RUN/STOP

Normally, depressing this key will stop the execution of a BASIC program. It signals the computer to stop doing something. Holding down the SHIFT key while pressing the RUN/STOP key lets you automatically load the first program from a cassette tape.

C=

The C= key performs a specialized function. It is used most often when you are listing a program on the screen. Pressing the C= key will stop a program from continuing to "scroll" down the screen. Press *any* key to restart the scrolling function.

ESC

This key is used in conjunction with any of the 26 alphabet keys A through Z to perform a wide variety of special functions. To perform any function listed below, hit the ESC key and then the appropriate letter.

A	Automatic insert
B	Set bottom
C	Cancel automatic insert
D	Delete line
E	Nonflashing cursor—B series only
F	Flashing cursor—B series only
G	Enable (turn *on*) bell

H	Disable (turn *off*) bell
I	Inset line
J	Move to the start of the line
K	Move to the end of the line
L	Enable scrolling
M	Disable scrolling
N	Normal screen—B series only
O	Cancel insert, quote and reverse
P	Erase to the start of the line
Q	Erase to the end of the line
R	Reverse screen—B series only
S	Solid cursor—B series only
T	Set the top of the page
U	Underscore cursor—B series only
V	Scroll up
W	Scroll down
X	Cancel escape sequence
Y	Normal character set—B series only
Z	Alternate character set—B series only

Calculator Pad Keys

For your calculating convenience, all of the standard calculator functions have been made easily accessible via the calculator keypad on the right of the keyboard.

?

This key is the standard Commodore abbreviation for the PRINT statement in BASIC. Therefore, by preceding a calculation with a ? , the computer can act as a calculator. For example:

```
?23.45*8
 187.6
```

The ? key, when pressed while CTRL is held down, also functions as the color control key for the color of light red on the C128-40 computer only. (Located at the bottom right section of the main keyboard, it will also work as the ? key in the SHIFT ed, *normal* mode or unshifted, *graphics* mode.)

00 0 1 2 3 4 5 6 7 8 9

The number keys are arranged like a regular calculator. It doesn't matter if you're in *normal* mode, *graphics* mode, SHIFT ed or unshifted. Notice that we have included a double zero 00 for your calculating convenience. (In addition, these keys function as the first nine color control keys when pressed in conjunction with the CTRL key on the C128-40 computer only.) All numbers located at the top of the main keyboard section can be used in calculations when your computer is operating in the unshifted, *normal* mode.

.

This serves as a decimal point for your noninteger computations. (The period, located at the bottom right section of the main keyboard, will also work as a decimal point in the unshifted, *normal* mode.)

/

The / key operates as a symbol for the arithmetic operation of division. (It also functions, when pressed in conjunction with the CTRL key, as the color control symbol for the color of gray 1 in the C128-40 computer only.) The / key located at the bottom right section of the main keyboard will also function as a division symbol in the unshifted, *normal* mode.

−

The − key operates as a symbol for the arithmetic operation of subtraction. It also operates as the "unary minus" symbol, which is the minus sign preceding negative numbers. (In addition the − functions, when pressed in conjunction with the CTRL key, as the color control symbol for the color of gray 2 in the C128-40 computer only.) The − key located at the top right section of the main keyboard will also function as the symbol for subtraction and unary minus in the unshifted, *normal* mode.

CE

This key is somewhat similar to the "Clear Entry" key found on most calculators. Use this key when you want to eliminate the last number entered. It will clear the last number on a computation line, back to the last arithmetic operation. (The CE key also functions, when pressed in conjunction with the CTRL key, as the color control symbol for light green on the C128-40 computer only.)

*

This operates as a symbol for the arithmetic operation of multiplication. (It also functions, when pressed in conjunction with the CTRL key, as the color control symbol for the color of light blue in the C128-40 computer only.) You can also use the * key located at the top right section of the main keyboard when operating in the shifted *normal* mode or the unshifted *graphics* mode.

+

This operates as a symbol for the arithmetic operation of addition. It also operates as the "unary plus" symbol to represent positive numbers. However, "unary plus" is automatically assumed by the computer and is therefore not essential. (The + key also functions, when pressed in conjunction with the CTRL key, as the color control symbol for the color of gray 3 on the C128-40 computer only.) You can also use the + key located center right in the main section of the keyboard when operating in shifted *normal* mode or unshifted *graphics* mode.

CHAPTER 4

LOADING AND SAVING PROGRAMS

- LOADing Prepackaged Programs From Diskette
- LOADing Prepackaged Programs From DATASSETTE™
- LOADing and SAVEing Your Programs on Diskette
- LOADing and SAVEing Your Programs on DATASSETTE™

One of the most important features of your Commodore computer is the ability to *save* and *load* programs to and from cassette tape or disk.

This capability allows you to save for use at a later time the programs you write or to purchase prepackaged programs for use with your C128-40 or B series computer.

Make sure that either the disk drive or DATASSETTE™ unit is attached according to its operating instructions.

For those interested in using prepackaged programs available on cassette or disk, here's what you have to do.

Loading Prepackaged Programs from Diskette

1. Start by carefully inserting the preprogrammed disk into drive zero (0).

NOTE: The computer will always assume that you're putting your disk into drive zero (0) and that you're using disk drive unit number eight (8). These are known as "default values." If you want to use another drive or unit number you must use the optional codes shown in Appendix A in square brackets [+ 1].

2. Make sure that the label on the disk is facing up and is closest to you.

3. Look for a little notch on the disk (it might be covered with a small strip of tape). If you're inserting the disk properly the notch will be on the **left** side.

4. Once the disk is inside, close the protective gate as described in your disk drive manual.

5. Now type:

```
DLOAD"PROGRAM NAME"
```

6. Now hit the RETURN key.
   The disk will make noise and your screen will say:

```
SEARCHING FOR 0: FILE NAME
LOADING
READY.
```

7. When the "ready" comes on and the cursor appears, type:

```
RUN
```

8. Hit the RETURN key and your prepackaged software is ready to use.

Loading Prepackaged Programs from Datassette™

1. Use your DATASSETTE™ recorder and the ordinary audio cassette tapes containing your prepackaged program.

2. Make sure the tape is completely rewound to the beginning of the first side.

3. Now type:

```
LOAD
```

4. Hit the RETURN key.
   The computer will answer with:

```
PRESS PLAY ON TAPE
```

5. You respond by pressing PLAY on your DATASSETTE™ machine.

   At this point the following message will appear:

```
SEARCHING
LOADING
READY.
```

6. When the "ready" comes on and the cursor appears type:

```
RUN
```

7. Hit the RETURN key, and your prepackaged software is ready to use.

Loading and Saving Your Programs on Diskette

Loading a program from disk is simple.

1. Type:

```
DLOAD PROGRAM NAME
```

2. Now hit the RETURN key
   The disk will start whirring and display:

```
SEARCHING FOR 0: PROGRAM NAME
LOADING
READY.
```

IMPORTANT: When you load a new program into the computer's memory, any instructions that were in the computer before you typed "load" will be erased. Make sure you save the program you're working on before loading a new one. Once a program has been loaded, you can run it, list it, or make changes and save the new version.

Saving a program on disk is easy.

1. Type:

```
DSAVE "PROGRAM NAME"
```

2. Now hit the RETURN key.

The disk will start to churn and the computer will respond with:

```
SAVING 0: PROGRAM NAME
OK
READY.
```

Before using a brand new diskette for the first time, you must format it with the HEADER command.

1. Place the new diskette you wish to format in drive 0.
2. Type:

```
HEADER "DISKNAME", D0
```

   where "diskname" is a name for the disk.

3. Now hit the RETURN key.
4. The computer will type:

```
ARE YOU SURE?
```

5. Respond by typing:

```
Y
```

6. Hit RETURN

The disk drive will make noise and the new diskette will be headered. (This process takes a couple minutes).

NOTE: Using the HEADER command erases any previously stored information on the disk. Be cautious when using this command.

Loading and Saving Your Programs on Datassette™

Loading a program from tape is simple.

1. Rewind the tape back to the beginning of the cassette.
2. Type:

```
LOAD "PROGRAM NAME"
```

   If you don't remember the program name, load the next program on the tape by typing:

```
LOAD
```

3. Now hit the RETURN key. The computer will respond with:

```
PRESS PLAY ON TAPE
```

4. Now depress the PLAY key on your DATASSETTE™ recorder.

The following message will appear on your screen:

```
SEARCHING FOR "PROGRAM NAME"
LOADING
READY.
```

After entering a program, you may want to save it on cassette tape. Here's all you do:

1. Type:

   ```
   SAVE PROGRAM NAME
   ```

   The program name can be up to 16 characters.

2. Now hit the RETURN key.

   The computer will respond with:

   ```
   PRESS PLAY AND RECORD ON TAPE
   ```

3. Press both the RECORD and PLAY keys on the DATASSETTE™.

After the program is saved on tape, the "ready" prompt will reappear, indicating that you can start working on another program or just turn the computer off and take a break.

CHAPTER 5

AVAILABLE SOFTWARE

Much of the wide variety of software available for the Commodore 64 will be made available for the C128-40.

Also, much of the exciting and varied software available for the CBM 8000 series will be converted to the B series computers.

For your programming convenience, assembler software will be available for both the C and B series computers. An assembler is virtually essential to your machine language level programming. Also, a BASIC compiler will be available for the machines, allowing compilation of your BASIC programs into highly efficient machine language.

The graphics capabilities of your C128-40 computer are placed right at your fingertips by using the popular LOGO computer language. LOGO allows graphics definitions and movement, text display, and multi-level graphic character display. The instructions are easy to use and the Commodore keyboard is used for easy screen editing.

Many of Commodore's *Easy* group of software will be made available for the C128-40 or the B series.

EasyCalc provides one of the largest electronic worksheets available. It allows selective row reporting and printing and instant "what if" calculations. Integration of spreadsheets, from disk to memory, allows summation and consolidation of different sizes and type matrices.

EasyFile is a database product that allows you to define the data you wish to keep track of and how it looks on the screen. It also gives you the ability to enter and retrieve your data with editing and selective criteria.

EasyScript is a word processor which provides you with all of the options of much larger products and even more. Variable text width, right margin justification or alignment, complete cursor movement, function keys for frequently used options, and vertical and horizontal tabs allow easy text formatting. Range selection for text movement options, search and replace, scrolling in all directions, and user defined characters make text editing simple.

EasySchedule, soon to be available for the C128-40, allows input of data in easy logical steps. Day, week, month, and year at a glance lets you see the scheduled items by time slots. The "Zoom" option lets you zero in on smaller and smaller time slots. Certain key areas of the program can be tailored to your specific needs.

EasyPlot, soon to be available for the B series computers, provides you with full page printing of all charts and graphs. Integrates with *EasyCalc* for data, or allows you to input up to 80 data points directly. It will plot up to four data series at a time, depending on the type of plot required. *EasyPlot* produces bar charts, scatter diagrams, pie charts, and line graphs.

EasySpell, also to be available for the B series, is an ingenious disk-based program compatible with the *EasyScript* word processor. This computerized spelling checker automatically corrects your spelling errors and counts the words in your manuscript or contract.

Software for the B series will also include terminal software, *Calc Result*, and *BPI Accounting*. For the C128-40, *Calc Result*, and an entry level word processor, the *Word Machine* and *Name Machine* programs, will be available.

The *Programmer's Reference Guide* for the C128-40 and B series of computers will provide you with much more detailed information on the technical aspects of your computer than can be adequately described in this short book. Look for it at your local dealer or book store.

CHAPTER 6

APPENDICES

APPENDIX A

BASIC 4.0 COMMAND, STATEMENT, AND FUNCTION LIST

The BASIC commands, statements, and functions below are explained in more detail in the *Programmer's Reference Guide* for the C128-40, B128-80, and B256-80. This list is designed as a "quick reference" for someone already familiar with the BASIC programming language. For those who are just beginning to learn about BASIC, see Appendix R for a list of tutorial-type books on learning BASIC.

The following conventions are used to describe the BASIC commands, statements, and functions in this appendix:

- *Keywords* appear in uppercase letters. Keywords are the words that your computer knows. These words cannot be used as part of your filenames or other variable names unless they are surrounded by quotation marks (" "). We do not recommend the use of any keywords for variable names. **You must enter these keywords exactly as they appear.**
- *Arguments* appear in lowercase letters. Another word for argument is parameter. You select those arguments needed for your statement. Arguments include: variables, expressions, linenumbers, etc.
- A *vertical bar* (|) separates arguments in cases where you can select any one of a list of arguments.
- *Square brackets* ([]) are used to show optional arguments. You select any, or none, of the arguments listed, depending on your needs.

- *Angle brackets* (<>) mean that you **must** choose one of the arguments.
- An *Ellipsis*, a sequence of three dots (...), means that an option or argument can be repeated more than once, if necessary.
- *Quotation marks* (" ") are used to enclose character strings, filenames and other items.
- *Variable* means any valid BASIC variable name.
- *Expression* means any valid BASIC expression, such as A+B+2 or .5+X.

BASIC COMMANDS

BACKUP

FORMATS AND COPIES

BACKUP Ds TO Dd [ON Uz]

All the diskette files on drive s are duplicated to drive d. Here "s" is the number of your source drive, "d" is the number of your destination drive, and "z" is the drive unit number if you have more than one disk drive unit.

CATALOG

CATALOG [D$] [,ON Uy]

Displays the names of the files on your diskette.

COLLECT

COLLECT [Ds] [,ON Uy]

The available space on drive s is collected. Note improperly closed files are deleted. Here "s" is the drive number and "y" is the drive unit number.

CONCAT

CONCAT [Ds,] "sourcefile" TO [Dd,] "destfile" [,ON Uz]

Concatenates (merges) two data files. Here "s" represents the drive number of the disk drive containing the "sourcefile", "d" is the drive number of the "destfile", and "z" is the drive unit number if more than one drive unit is present.

CONT

CONT

Continues or re-starts the execution of a program which was interrupted because you hit the STOP key or the program executed a STOP statement or an END statement.

COPY

COPY [Ds,] "sourcefile" TO [Dd,] "destfile" [,ON Uz]

The file in drive s is copied onto the file in drive d of unit z. Here "s" is the drive number of the file named "source file," "d" is the drive number of the file named "destfile," and "z" is the drive unit number.

DELETE

DELETEs [from linenumber]—[to linenumber]

DELETE from memory a group of lines from the BASIC program currently in memory. When using DELETE be aware that:

1. DELETE	(erases the entire program)
2. DELETE linenumber—	(erases from linenumber to end)
3. DELETE—linenumber—	(erases from start to linenumber)
4. DELETE linenumber—linenumber	(erases from linenumber to linenumber)

DIRECTORY

DIRECTORY [Dnumber] [,"filename"] [,ON Uy]

Displays the names of the files on your diskette. Only those files that match the specified filename are displayed. Here "number" is the number of the disk drive that contains the diskette.

DLOAD

DLOAD "filename" [,Ddrive-number] [,Uunit]

Loads (bring into memory) a BASIC program that is stored on a disk. Here "drive-number" is the drive number of the disk and "unit" is the drive unit number.

DSAVE

DSAVE "filename" [,Ddrive-number] [,Uunit]

Stores a BASIC program on disk. Here "drive-number" is the drive number of the disk and "unit" is the drive unit number.

HEADER

HEADER "diskname", Dd [,Ivv] [,ON Uz]

Formats a diskette, assigning it a disk name and identification number. Here "d" is the disk drive number, "vv" can be any alphabetic characters you want to use as identification numbers, and "z" is the drive unit number if you have more than one drive unit.

LIST

LIST [[from linenumber] [—] [to linenumber]]

Displays one or more lines of the BASIC program currently in memory. When using LIST be aware that:

1. LIST	(lists the entire program)
2. LIST linenumber—	(lists all lines from that number to the end)
3. LIST—linenumber	(lists all lines up to that number)
4. LIST linenumber—linenumber	(lists all lines from linenumber to linenumber)

LOAD

LOAD ["filename"[,device]]

LOADs (bring into memory) a program that is stored on an external device, such as a tape cassette or disk. Here "device" is the device number of the storage device you are using. If no device number is used, the computer will assume that you want to use the DATASSETTE™, which is device number 1.

NEW

NEW

Erases the current BASIC program and data from memory so that a NEW program can be loaded or entered.

RENAME

RENAME [,Ddr] "oldname" TO "newname" [,ON Uz]

Changes the name of a file on a diskette without altering the file. Here "dr" represents the number of the disk drive in which the files are located and "z" represents the drive unit number if you have more than one disk drive unit.

RUN

RUN [linenumber]

Begins executing the BASIC program currently in memory. When using RUN be aware that:

1. RUN	(starts a program from the beginning)
2. RUN linenumber	(starts a program from that linenumber)

NOTE: If the line number does NOT exist you will get an error message.

SAVE

SAVE ["filename"[,device]]

Stores BASIC programs and data on an external storage medium, such as a tape cassette or diskette. Here "device" is the device number of the storage device you are using. If no device number is used, the computer will assume that you want to use the DATASSETTE™, which is device number 1.

SCRATCH

SCRATCH "filename" [,Dd] [,ON Uz]

Erases a single file from a diskette. Here "d" represents the disk drive number where the file can be found, "z" represents the unit number of the disk drive if you are using more than one drive unit.

VERIFY

VERIFY ["filename"] [,device]

Checks a program on tape or disk against the program currently in memory. Here "device" is the device number of the storage device you are using. If no device number is used, the computer will assume that you want to use the DATASSETTE™, which is device number 1. When using VERIFY be aware that:

1. VERIFY	(checks the next program on tape)
2. VERIFY "filename"	(searches for filename on tape and then checks against memory)
3. VERIFY "filename" ,8	(searches for filename on disk and then checks against memory)

BASIC STATEMENTS

APPEND

[linenumber] APPEND #lf, "filename" [,Dd] [,ON Uz]

Opens a sequential data file called "filename" and positions the file pointers beyond the current end of the file. Here "lf" represents the logical file number, "d" is the disk drive number and "z" is the unit number of the disk drive if more than one unit is present.

BANK

[linenumber] BANK expression

Sets the indirection bank number for use with some BASIC commands such as PEEK, POKE, BLOAD, and BSAVE. Remember that you have 16 BANKs to choose from.

BLOAD

[linenumber] BLOAD [fileopts] [[ON]Bbanknumber] [,Plowoffset]

Loads a binary file into any memory location. Here "fileopts" is your file options, "banknumber" lets you select one of the 16 banks, and "lowoffset" is location in the bank to start loading.

BSAVE

BSAVE fileopts [☑[ON]Bbanknumber] [,Plowoffset] [TO Phighoffset]

Saves a binary file from any specified memory location. Here "fileopts" is your file options, and "banknumber" lets you select one of the 16 banks. The "lowoffset" is the location in the bank beginning the information to be SAVEd. The "highoffset" is the location in the bank ending the information to be SAVEd.

CLOSE

[linenumber] CLOSE filenumber

CLOSEs a file which was opened with a previous OPEN statement.

NOTE: The filenumber must be the same number used in the OPEN statement. See your Datassette™, Disk Drive, or Printer manual(s) for more details.

CLR

[linenumber] CLR

CLeaRs all BASIC variables in memory, but leave the program itself intact. This statement is automatically executed when a RUN command is given.

CMD

[linenumber] CMD filenumber [,printlist]

Redirects output. For example, you should use the CMD statement when you want to send output normally directed to the screen to a file or the printer. Here "printlist" is a list of character strings, numeric variables, or expressions which is written to the device when the CMD statement is executed.

NOTE: The CMD statement must be used in conjunction with a corresponding OPEN statement and both the OPEN statement and the CMD statement must have the same filenumber.

DATA

[linenumber] DATA value [,value,. . . ,value]

Uses the DATA statement in conjunction with the READ statement(s) to enter numeric and string constants into a program.

DCLOSE

[linenumber] DCLOSE [#lf] [,ON Uz]

Closes a single file or all the files currently open on a disk unit. Here "lf" represents the logical file number of the file to be closed on the disk and "z" is the number of the disk drive unit if more than one unit is present. When using DCLOSE be aware that:

1. DCLOSE	(closes all files currently open on default device)
2. DCLOSE #lf	(closes the file associated with the logical file number "lf")
3. DCLOSE ON Uz	(closes all files currently open on unit "z")

DEF FN

[linenumber] DEF FNna (argument)=formula

DEF FN allows you to define a complex calculation as a single function with a short name. In the case of a long formula that is used many times in your program, this statement can save time and space. Here "na" represents any legal variable name. When appended to the letters FN, this becomes the function name. The "(argument)" can be any numeric variable, and it must be enclosed in parentheses.

NOTE: You must define your function with this DEF FN statement before you can "call" it using the function name.

DIM

[linenumber] DIM variable(subscript1 [,. . . ,subscriptn]), [variable(subscript1[,. . . ,subscriptn]). . .]

Allocates storage for an array and specifies the maximum values for the array variable subscripts. Here "subscript" represents the size of the DIMensions of the array you are creating.

NOTE: You must use the DIM statement to DIMension arrays containing more than 10 elements, and "subscripts" must be enclosed in parentheses. In addition, you should be aware of the fact that the number of elements in one dimension of an array equals the value of the "subscript" plus 1.

```
EXAMPLE:   10 DIM A$(40),      B7(15),      CC%(4,4,4)
                  ↑               ↑              ↑
             41 elements     16 elements   125 elements
```

DISPOSE

[linenumber] DISPOSE <FOR | GOSUB>

Used in error trap processing to eliminate unwanted NEXT or RETURN addresses from the stack.

DOPEN

DOPEN #lf, "filename" [,Ly] [,Dd] [,ON Uz] [,W]

Opens a data file for a read and/or write access. Here "lf" represents the logical filenumber of the file to be opened, "y" is the record length for a nonsequential file, "d" is the disk drive number in which the disk containing the file is located, and "z" is the drive unit number if more than one drive unit is present. A sequential file is opened for write access if "W" *is not* present; *it is opened for read access if "W" is present.*

END

[linenumber] END

Finishes a program that is running and returns control to the screen.

FOR...TO...STEP

[linenumber] FOR variable=expression1 TO expression2 [STEP expression3]

Always associated with a NEXT statement. FOR...TO...STEP controls repetitive execution of a group of statements called a loop. Here "variable" is the counter for the loop, "expression1" represents the start value of the counter, "expression2" is the end value of the counter, and "expression3" is the increment value, added to the current value of the counter. When using FOR...TO...STEP be aware that:

1) FOR variable=expression1 TO expression2 (increments a STEP value of 1)

GET

linenumber GET variable

Scans the keyboard buffer and reads a single character from the buffer. If no character is typed, then a null (empty) character is assigned.

NOTE: GET is usually followed by a string variable. If a numeric variable is used and a NONnumeric key is pressed, your program will halt and you will get an error message. In addition, the GET statement may be placed in a loop to check for a null string (" ") result. This loop will continue until a key is pressed.

GET#

[linenumber] GET# filenumber, variable

Reads a single character from logical file "filename".

NOTE: GET# can only be used with previously OPENed device or file.

GOSUB

[linenumber] GOSUB linenumber2

Always used with the RETURN statement. GOSUB is used to branch to the subroutine which starts with the BASIC statement labeled "linenumber2".

NOTE: GOSUBs without RETURNS will result in a buildup of return addresses, eventually causing an out of stack error.

GOTO or GO TO

[linenumber] GOTO linenumber2

Unconditionally branches to the BASIC statement with the line number "linenumber2".

IF...GOTO

[linenumber] IF expression GOTO linenumber2

Evaluate the conditions in "expression" and, if these conditions are met, branch to the statement located at "linenumber2".

IF...THEN...ELSE

[linenumber] If expression THEN then-clause [: ELSE else-clause] *Evaluates the conditions in "expression", then, depending on the results, performs the action required in the "then-clause" or the "else-clause".*

NOTE: In most cases the "expression" will involve one or more of the following relational operators:

= < <= > >= <> AND OR NOT

INPUT

[linenumber] INPUT [promptstring :] variable list

The INPUT statement allows the program to get information from the person using the program by assigning that data to a variable. Most of the time, the information needed by the program is in the form of a question. Therefore, after the INPUT "prompt string" is printed, the program stops and a question mark (?) is PRINTed to the screen. The computer then waits for the person using the program to type in a response and hit the <RETURN> key. Here the "promptstring" represents the prompt, which is usually in the form of a question, and "variablelist" is the list of variable names to associate with the data typed in.

INPUT#

linenumber INPUT# filenumber, variablelist

INPUT# is similar to the INPUT statement, except it takes data from a previously OPENed file or device. The "filenumber" represents the number the file has been OPENed. The "variablelist" represents the name(s) to associate with the input data.

LET

[linenumber] [LET] variable = expression

The LET keyword is hardly ever used in BASIC programs because it is an optional keyword. However, the fact that it is understood places it at the heart of all BASIC programs. Here LET sets "variable" to the value defined by "expression".

NEXT

[linenumber] NEXT [variable, . . . , variable]

Always used in conjunction with a FOR statement. NEXT is used to mark the end of a FOR . . . NEXT loop. When the program reaches a NEXT statement, it checks its corresponding FOR statement to see if the limit of the loop has been reached. If the limit has not been reached then the loop "variable" in the FOR statement is increased by the STEP value. If the loop is finished then execution continues with the statement(s) in the line following the NEXT statement.

NOTE: NEXT may or may not be followed by a "variable" name. If no names are listed, the last loop started is finished. If one "variable" name is given, it must correspond to an appropriately positioned FOR statement. If more than one "variable" follows a single NEXT, each loop is finished, in order, from left to right and each "variable" must have a corresponding FOR with the same "variable" name. See FOR . . . TO . . . STEP for more information.

ON...GOSUB

[linenumber] ON expression GOSUB linenumber list

ON . . . GOSUB is similar to an IF . . . THEN statement in that it evaluates the "expression" found after ON and then, based on the result, moves the program execution to the corresponding subroutine line number following the GOSUB. Here "expression" represents the formula and "linenumber list" contains the list of the line numbers for each subroutine. The appropriate subroutine will be chosen depending upon the value of "expression".

ON...GOTO

[linenumber] ON expression GOTO linenumber list

ON . . . GOTO is similar to an IF . . . THEN statement in that it evaluates the "expression" found after ON and then, based on the result, moves the program execution to the corresponding line number following the GOTO. Here "expression" represents the formula and "linenumber list" contains the list of the line numbers to GOTO. The appropriate statement will be chosen depending upon the value of "expression".

OPEN

[linenumber] OPEN filenumber [,devicenumber [,command [,openstring]]]

The OPEN statement establishes an Input/Output (I/O) channel to the screen or an external device, such as a tape cassette, a disk drive, a printer, or to the IEEE serial bus. Here "filenumber" represents a number between 0 and 255 that is used to specify in statements like CLOSE, CMD, GET#, INPUT#, and PRINT# the specific file intended. Only statements with a corresponding "filenumber" will be able to access the file. The label "devicenumber" refers to the appropriate external device as mentioned above. You should be aware that:

1.	OPEN 1,3	(OPENs the screen as a device)
2.	10 OPEN 2,1,0,"D"	(OPENs the Datassette™ for READing and then searches for file "D")
3.	20 OPEN 3,4	(OPENs a channel to the printer)
4.	OPEN 4,8,15	(OPENs the command channel on the disk)

NOTE: As is indicated, the OPEN statement may be executed in both *direct* and *program* modes. Linenumbers are always necessary in program mode. For more information see the *Programmer's Reference Guide* for your computer or any of the manuals that come with your external devices such as disk drive, printer, or Datassette™.

POKE

[linenumber] POKE location, value

The POKE statement is always followed by two numbers or formulas. Here "location" represents a memory location from 0 through 65535 and "value" is a decimal value from 0 through 255 which is placed in "location" replacing any previously stored value. A previously executed BANK command selects the bank poked to.

PRINT

[linenumber] PRINT [printlist]

PRINT writes the information specified in "printlist" to the screen. Here "printlist" can include any of the following:

1. Text which must always be enclosed in quotation marks (" ")
2. Variable names
3. Functions
4. Punctuation marks (used for formatting data)

NOTE: For more details see your *Programmer's Reference Guide.*

PRINT#

[linenumber] PRINT# filenumber, printlist

The PRINT# statement writes the values specified in "printlist" to the file associated with "filenumber".

NOTE: The "filenumber" used in the PRINT# statement must refer to the corresponding file number used with a device or file that has already been OPENed.

PRINT USING or PRINT# USING

[linenumber] PRINT [#filenumber,] USING formatlist ; printlist [;]

Allows you to define a "formatlist" controlling the appearance of the string and numeric output specified in "printlist". See the *Programmer's Reference Guide* for details.

PUDEF

[linenumber] PUDEF controlstring

Redefines four symbols which are printed with the PRINT USING statement's formatting options. See the *Programmer's Reference Guide* for details. In this example, "controlstring" represents the format that you've designed for your data.

READ

[linenumber] READ variablelist

READ processes one or more DATA statements and assigns the data item values, on a one-to-one basis, to the variables in "variablelist".

If the data in a DATA statement is not the same type as the *variable in the "variablelist", a syntax error which refers to the line number of the DATA statement results.*

RECORD

[linenumber] RECORD# lf, rno [,bno]

Adjusts a relative file pointer to select any byte (character) of any record in the relative file. Consult the Programmer's Reference Guide for more details.

REM

[linenumber] REM [text]

A nonexecutable statement that is LISTed in your BASIC program.

RESTORE

[linenumber] RESTORE [linenumber2]

Resets the pointer for the current DATA statement to be processed by READ statements to linenumber2.

RESUME

[linenumber] RESUME [NEXT | linenumber2]

Used in error handling to resume execution after an error is trapped and processed by your error handling routine.

RETURN

[linenumber] RETURN

Completes subroutine processing and branches back to the statement following the GOSUB which started the subroutine.

STOP

[linenumber] STOP

Terminates program execution and returns to command level.

SYS

[linenumber] SYS jumpaddress

Calls the assembly language subroutine at location "jumpaddress" (which is a decimal, not hexadecimal, address) and executes the machine language program starting at that address.

NOTE: SYS will only jump into Segment 15 of the computer Random Access Memory (RAM) locations. This means that even if you were to select any other Bank or memory segment the program will still ONLY jump into Segment 15.

NOTE: All machine language programs must end with an RTS (ReTurn from Subroutine) statement.

TRAP

[linenumber] TRAP [linenumber2]

Stops from functioning BASIC's normal error handling and lets your program perform its own error handling.

WAIT

[linenumber] WAIT location,, mask1 [,mask2]

Continually tests the data at "location" and re-executes the WAIT statement or proceeds to the next executable statement, depending on the values of selected bits at location.

BASIC FUNCTIONS

ABS

ABS (expression)

The absolute value of "expression".

ASC

ASC (expression)

The numeric value that represents the ASCII code of the first character of "expression".

ATN

ATN (expression)

The arctangent of the "expression" in radians.

CHR$

CHR$ (expression)

A string containing a single character whose value is the character with the ASCII code represented by "expression".

COS

COS (expression)

The cosine of "expression" in radians.

ERR$

ERR$ (expression)

A character string which contains the text of the error message represented by "expression".

EXP

EXP (expression)

The value of e (approx. 2.71828183) raised to the power represented by "expression".

FRE

FRE (expression)

The number of free bytes in a memory segment or bank indicated by "expression".

INSTR

INSTR (expression1,expression2[,expression3])

"Expression1" is searched, beginning at the optional character position represented by "expression3", for the occurrence of the string represented by "expression2".

INT

INT (expression)

The largest integer which is less than or equal to the value specified in "expression".

LEFT$

LEFT$ (expression1, expression2)

The leftmost "expression2" characters in the string represented by "expression1".

LEN

LEN (expression)

The number of characters in "expression".

LOG

LOG (expression)

The natural logarithm of "expression".

MID$

MID$ (expression1,expression2[,expression3])

A string containing "expression3" characters of the string "expression1", beginning at the character position "expression2".

PEEK

PEEK (expression)

The byte read from memory location "expression" in the bank selected by a previously executed BANK instruction.

POS

POS (expression)

The column where the next character will be written on the line currently containing the cursor.

NOTE: *All machine language programs must end with an RTS (ReTurn from Subroutine) statement.*

NOTE: "expression" is necessary, but ignored.

RIGHT$

RIGHT$ (expression1, expression2)

A string consisting of the rightmost "expression2" characters in the string "expression1".

RND

RND (expression)

A random number between 0 and 1. "Expression" is used as the seed value. See the *Programmer's Reference Guide* for details.

SGN

SGN (expression)

A value indicating whether the value of "expression" is positive (gives +1), negative (gives −1), or zero (gives 0).

SIN

SIN (expression)

The sine of "expression" in radians.

SPC

SPC (expression)

Prints the number of blank spaces on the screen indicated by the number in "expression".

SQR

SQR (expression)

The square root of "expression".

STR$

STR$ (expression)

A character string containing a string representation of the value represented by "expression".

TAB

TAB (expression)

Positions the cursor in the column represented by "expression".

TAN

TAN (expression)

The tangent of "expression" in radians.

USR

USR (expression)

Calls the user-written assembly language subroutine which has starting address stored in locations 3 and 4 of bank 15. The argument ("expression") is stored in the floating point accumulator prior to entering the subroutine. See the *Programmer's Reference Guide* for more details.

VAL

VAL (expression)

The numeric value of the string "expression".

RESERVED SYSTEM VARIABLES

AND Logical operator.

DS$ Disk Status reserved word.

EL Line number last error occurred.

ER Error# of last error occurrence.

OR Logical operator.

NOT Logical operator.

STatus The system status for the last Input/Output operation.

TI$me The character string representation of the current time-of-day registers.

RESERVED SYSTEM SYMBOLS

+ Plus sign:	arithmetic addition or string concatenation
− Minus sign:	arithmetic subtraction and unary minus
* Asterisk:	arithmetic multiplication
/ Slash:	arithmetic division

(**blank**) Blank:	separates keywords and variable names
= Equal sign:	value assignment and relationship testing
< Less than:	used in relationship testing
> Greater than:	used in relationship testing
↑ Up arrow:	arithmetic exponentiation
, Comma:	used in variable lists to format output also separates command parameters
. Period:	decimal point in floating point constants
; Semicolon:	used in variable lists to format output
: Colon:	separates multiple BASIC statements on a program line
" Quotation mark:	encloses string constants
? Question mark:	abbreviation for the keyword PRINT
(Left parenthesis:	expression evaluation and functions
) Right parenthesis:	expression evaluation and functions
% Percent:	declares a variable name as an integer
# Number:	comes before the logical file number in input/output statements
$ Dollar sign:	declares a variable name as a string
π Pi:	the numeric constant 3.141592654

APPENDIX B

BASIC 4.0 ABBREVIATIONS

KEYWORD	ABBREVIATION	TYPE
ABS	a SHIFT B	function—numeric
APPEND	a SHIFT P	statement
ASC	a SHIFT S	function—numeric
ATN	a SHIFT T	function—numeric
BACKUP	b SHIFT A	command
BANK	ba SHIFT N	statement
BLOAD	b SHIFT L	command
BSAVE	b SHIFT S	command
CHR$	c SHIFT H	function—string
CATALOG	c SHIFT A	command
CLOSE	cl SHIFT O	statement
CLR	c SHIFT L	statement
CMD	c SHIFT M	statement
COLLECT	co SHIFT L	command
CONCAT	con SHIFT C	statement
CONT	c SHIFT O	command
COPY	co SHIFT P	command
COS	none	function—numeric
DATA	d SHIFT A	statement
DCLOSE	d SHIFT C	statement
DEF FN	d SHIFT E	statement
DELETE	de SHIFT L	command
DIM	d SHIFT I	statement
DIRECTORY	di SHIFT R	command
DISPOSE	di SHIFT S	statement
DLOAD	d SHIFT L	command
DOPEN	d SHIFT O	statement

KEYWORD	ABBREVIATION	TYPE
DSAVE	d SHIFT S	command
END	e SHIFT N	statement
ERR$	none	function—string
EXP	e SHIFT X	function—numeric
FOR	f SHIFT O	statement
FRE	f SHIFT R	function—numeric
GET	g SHIFT E	statement
GET#	none	statement
GOSUB	go SHIFT S	statement
GOTO	g SHIFT O	statement
HEADER	h SHIFT E	command
IF...GOTO	none	statement
IF..THEN..ELSE	none	statement
INPUT	none	statement
INPUT#	i SHIFT N	statement
INSTR	in SHIFT S	function—numeric
INT	none	function—numeric
LEFT$	le SHIFT F	function—string
LEN	none	function—numeric
LET	l SHIFT E	statement
LIST	l SHIFT I	command
LOAD	l SHIFT O	command
LOG	none	function—numeric
MID$	m SHIFT I	function—string
NEW	none	command
NEXT	n SHIFT E	statement
ON...GOSUB	none	statement
ON...GOTO	none	statement
OPEN	o SHIFT P	statement
PEEK	p SHIFT E	function—numeric
POKE	p SHIFT O	statement
POS	none	function—numeric
PRINT	?	statement
PRINT#	p SHIFT R	statement
PRINT USING	?us SHIFT I	statement
PUDEF	none	statement
READ	r SHIFT E	statement
RECORD	re SHIFT C	statement
REM	none	statement

KEYWORD	ABBREVIATION	TYPE
RENAME	re SHIFT N	command
RESTORE	re SHIFT S	statement
RESUME	res SHIFT U	statement
RETURN	re SHIFT T	statement
RIGHT$	r SHIFT I	function—string
RND	r SHIFT N	function—numeric
RUN	r SHIFT U	command
SAVE	s SHIFT A	command
SCRATCH	s SHIFT C	command
SGN	s SHIFT G	function—numeric
SIN	s SHIFT I	function—numeric
SPC	s SHIFT P	function—special
SQR	s SHIFT Q	function—numeric
STATUS	st	function—numeric
STOP	s SHIFT T	statement
STR$	st SHIFT R	function—string
SYS	s SHIFT Y	statement
TAB	t SHIFT A	function—special
TAN	none	function—numeric
TI$	none	function—string
TRAP	t SHIFT R	statement
USR	u SHIFT S	function—special
VAL	none	function—numeric
VERIFY	u SHIFT E	command
WAIT	w SHIFT A	statement

APPENDIX C

SCREEN DISPLAY CODES

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
@		0	U	u	21	*		42
A	a	1	V	v	22	+		43
B	b	2	W	w	23	,		44
C	c	3	X	x	24	–		45
D	d	4	Y	y	25	.		46
E	e	5	Z	z	26	/		47
F	f	6	[		27	0		48
G	g	7	£		28	1		49
H	h	8	]		29	2		50
I	i	9	↑		30	3		51
J	j	10	←		31	4		52
K	k	11	SPACE		32	5		53
L	l	12	!		33	6		54
M	m	13	"		34	7		55
N	n	14	#		35	8		56
O	o	15	$		36	9		57
P	p	16	%		37	:		58
Q	q	17	&		38	;		59
R	r	18	'		39	<		60
S	s	19	(		40	=		61
T	t	20	)		41	>		62

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
?		63	╭	U	85	├		107
─		64	╳	V	86	▗		108
♠	A	65	○	W	87	└		109
│	B	66	♣	X	88	┐		110
─	C	67	▕	Y	89	▂		111
─	D	68	♦	Z	90	┌		112
─	E	69	┼		91	┴		113
─	F	70	▒		92	┬		114
│	G	71	│		93	┤		115
│	H	72	π	▒	94	▎		116
╮	I	73	◥	▨	95	▍		117
╰	J	74	SPACE		96	▐		118
╯	K	75	▌		97	▔		119
└	L	76	▄		98	▀		120
╲	M	77	▔		99	▃		121
╱	N	78	▁		100	┘	✓	122
┌	O	79	▏		101	▖		123
┐	P	80	▒		102	▝		124
●	Q	81	▕		103	┘		125
─	R	82	▒		104	▘		126
♥	S	83	◤	▨	105	▚		127
▏	T	84	▕		106			

Codes from 128-255 are reversed images of codes 0-127.

CHR$ CODES

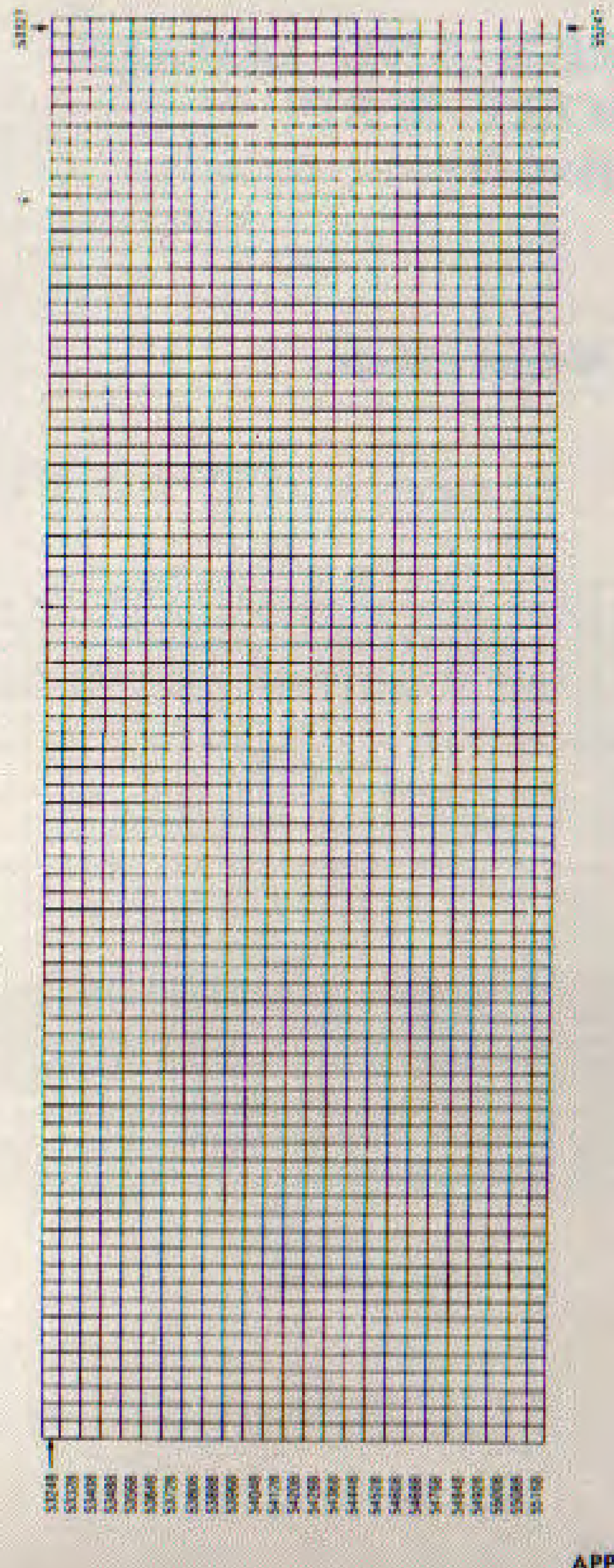
PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
	0		23	.	46	E	69
	1		24	/	47	F	70
	2		25	0	48	G	71
	3		26	1	49	H	72
	4		27	2	50	I	73
WHT	5	RED	28	3	51	J	74
	6	CRSR →	29	4	52	K	75
	7	GRN	30	5	53	L	76
DISABLES SHIFT C=	8	BLU	31	6	54	M	77
ENABLES SHIFT C=	9	SPACE	32	7	55	N	78
	10	!	33	8	56	O	79
	11	"	34	9	57	P	80
	12	#	35	:	58	Q	81
RETURN	13	$	36	;	59	R	82
SWITCH TO LOWER CASE	14	%	37	<	60	S	83
	15	&	38	=	61	T	84
	16	'	39	>	62	U	85
CRSR ↓	17	(	40	?	63	V	86
RVS ON	18	)	41	@	64	W	87
CLR HOME	19	*	42	A	65	X	88
INST DEL	20	+	43	B	66	Y	89
	21	,	44	C	67	Z	90
	22	−	45	D	68	[	91

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
£	92	╭	117	SWITCH TO UPPER CASE	142	▕	167
]	93	╳	118		143	▒	168
↑	94	○	119	BLK	144	◤	169
←	95	♣	120	CRSR ↑	145	▕	170
─	96	│	121	RVS OFF	146	├	171
♠	97	♦	122	CLR HOME	147	▗	172
│	98	┼	123	INST DEL	148	└	173
─	99	▒	124	╭	149	┐	174
─	100	│	125	╳	150	▂	175
─	101	π	126	○	151	┌	176
─	102	◥	127	♣	152	┴	177
│	103		128	│	153	┬	178
│	104	♠	129	♦	154	┤	179
╮	105		130	┼	155	▎	180
╰	106		131	PUR	156	▍	181
╯	107		132	CRSR ←	157	▐	182
└	108	f1	133	YEL	158	▔	183
╲	109	f3	134	CYN	159	▀	184
╱	110	f5	135	SPACE	160	▃	185
┌	111	f7	136	▌	161	┘	186
┐	112	f2	137	▄	162	▖	187
●	113	f4	138	▔	163	▝	188
─	114	f6	139	▁	164	┘	189
♥	115	f8	140	▏	165	▘	190
│	116	SHIFT RETURN	141	▒	166	▚	191

CODES	192-223	SAME AS	96-127
CODES	224-254	SAME AS	160-190
CODE	255	SAME AS	126

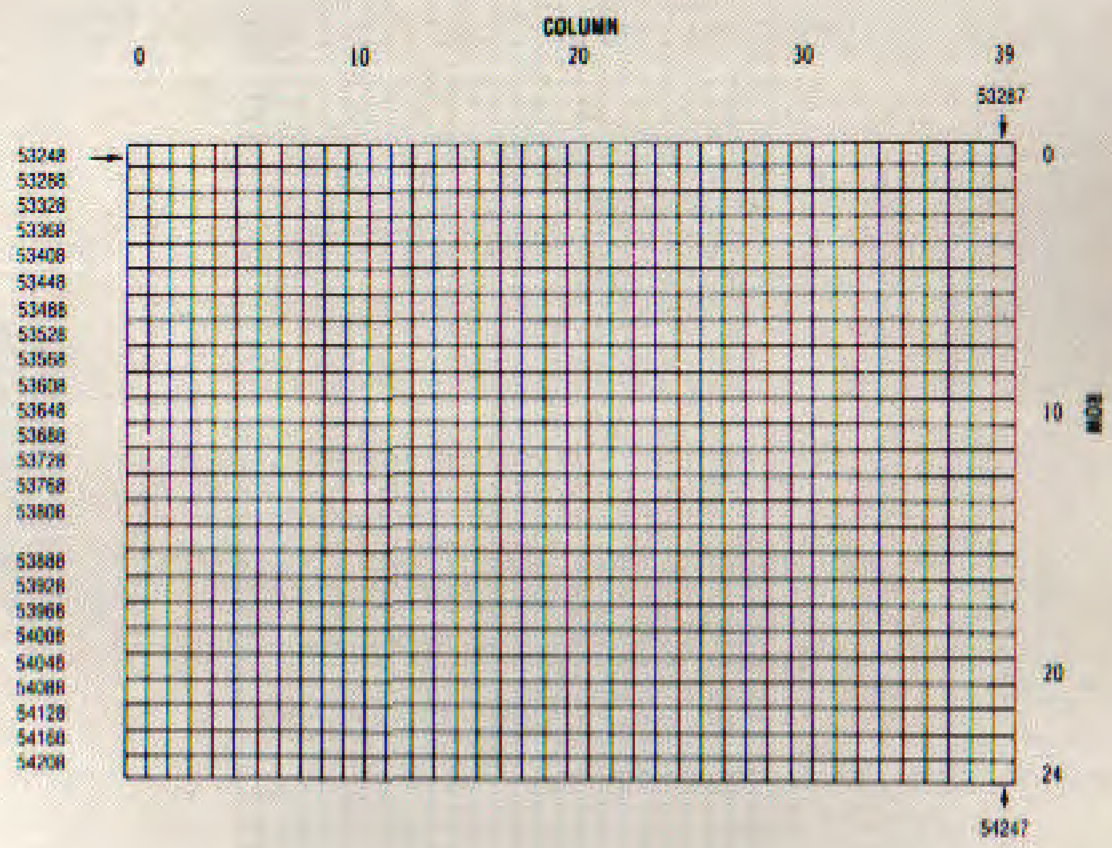
APPENDIX E

SCREEN MEMORY MAP (B Series)

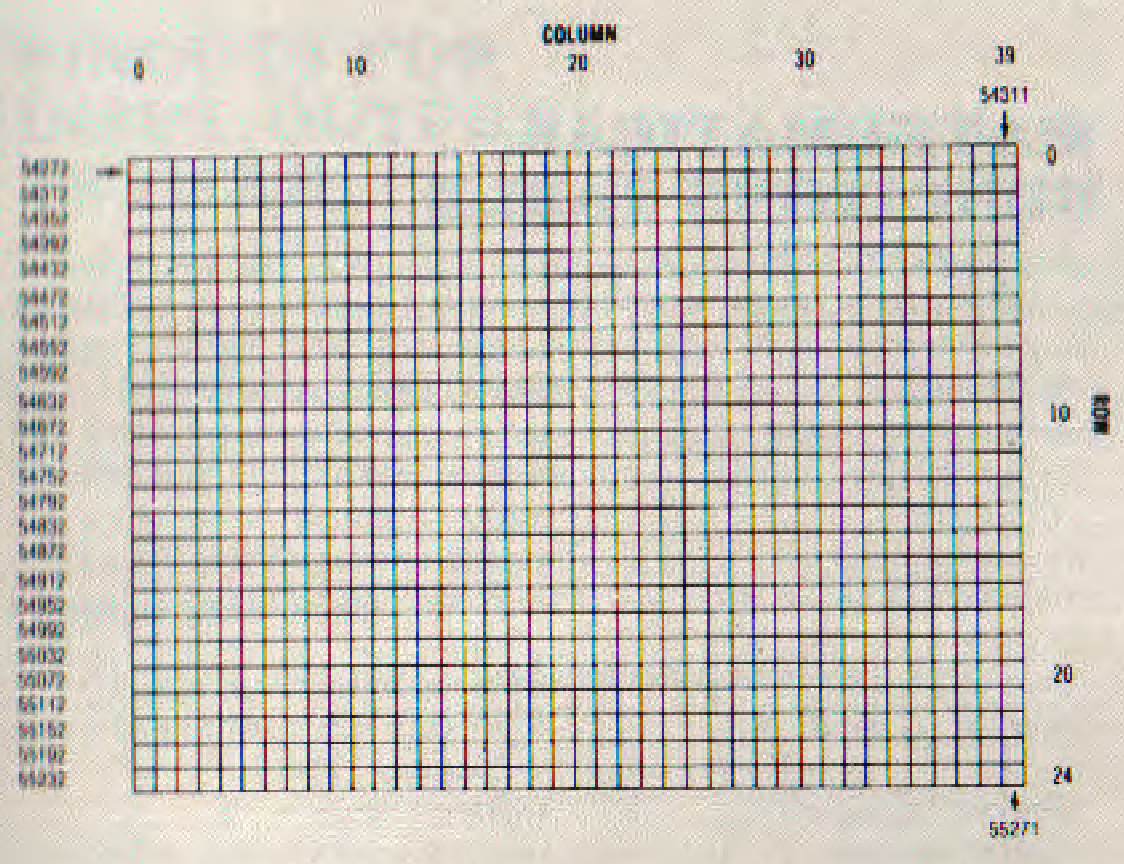


SCREEN AND COLOR MEMORY MAP (C128-40)

Screen Memory Map



Color Memory Map



APPENDIX G

MATHEMATICAL FUNCTIONS TABLE

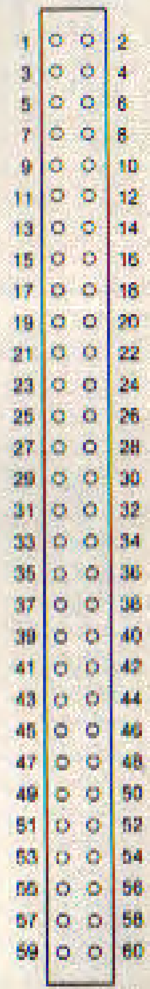
FUNCTION	BASIC EQUIVALENT
secant	sec(x)=1/cos(x)
cosecant	csc(x)=1/sin(x)
cotangent	cot(x)=1/tan(x)
inverse sine	arcsin(x)=atn(x/sqr(-x*x+1))
inverse cosine	arccos(x)=-atn(x/sqr(-x*x+1)) +π/2
inverse secant	arcsec(x)=atn(x/sqr(x*x-1))
inverse cosecant	arccsc(x)=atn(x/sqr(x*x-1)) +(sgn(x)-1*π/2
inverse cotangent	arcot(x)=atn(x)+"/2
hyperbolic sine	sinh(x)=(exp(x)-exp(-x))/2
hyperbolic cosine	cosh(x)=(exp(x)+exp(-x))/2
hyperbolic tangent	tanh(x)=exp(-x)/ (exp(x)+exp(-x))*2+1
hyperbolic secant	sech(x)=2/(exp(x)+exp(-x))
hyperbolic cosecant	csch(x)=2/(exp(x)-exp(-x))
hyperbolic cotangent	coth(x)=exp(-x)/ (exp(x)-exp(-x))*2+1
inverse hyperbolic sine	arcsinh(x)=log(x+sqr(x*x+1))
inverse hyperbolic cosine	arccosh(x)=log(x+sqr(x*x-1))
inverse hyperbolic tangent	arctanh(x)=log((1+x)/(1-x))/2
inverse hyperbolic secant	arcsech(x)=log((sqr(-x*x+1) +1/x)
inverse hyperbolic cosecant	arccsch(x)=log((sgn(x)* sqr(x*x=1/x)
inverse hyperbolic cotangent	arccoth(x)=log((x+1)/(x-1))/2

APPENDIX H

PINOUTS FOR INPUT/OUTPUT DEVICES

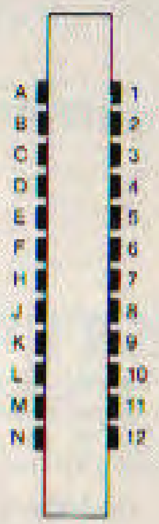
Your computer is equipped with several specialized chips all in BANK 15. The 6526 Complex Interface Adapter is located at 56320 ($DC00). The 6551 Asynchronous Communication Interface Adapter is located at 56576 ($DD00). Your computer has two 6525 Tri-port Interface chips located at 56832 ($DE00) and 57088 ($DF00). For more information, consult your *Programmer's Reference Guide*.

Commodore 128



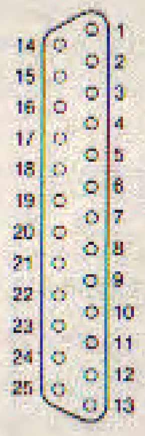
System Bus Connectors

Pin	Type	Pin	Type
1	+5V	31	BDB4
2	+5V	32	BDB2
3	+5V	33	BDB5
4	+5V	34	BDB1
5	GND	35	BDB6
6	GND	36	BDB0
7	GND	37	BAB13
8	GND	38	BDB7
9	GND	39	BAB14
10	GND	40	BAB15
11	$\overline{\text{BRAS}}$	41	BAB1
12	$\overline{\text{IRQ3}}$	42	BAB0
13	-12V	43	BAB2
14	$\overline{\text{EXTRES}}$	44	BAB11
15	+12V	45	BAB3
16	$\overline{\text{S.O}}$	46	BAB10
17	$\overline{\text{RES}}$	47	BAB12
18	LPEN	48	BAB4
19	SR/$\overline{\text{W}}$	49	BAB9
20	$\overline{\text{EXTBUFCS}}$	50	BAB5
21	TOOCLK	51	BAB8
22	$\overline{\text{DISKROMCS}}$	52	BAB6
23	BOOTCLK	53	BP0
24	BA	54	BAB7
25	Sϕ2	55	BP1
26	$\overline{\text{SCAS}}$	56	BP2
27	Sϕ1	57	$\overline{\text{NMI}}$
28	CS1	58	BP3
29	BOB3	59	RDY
30	$\overline{\text{EXTPRTCS}}$	60	$\overline{\text{IRQ}}$



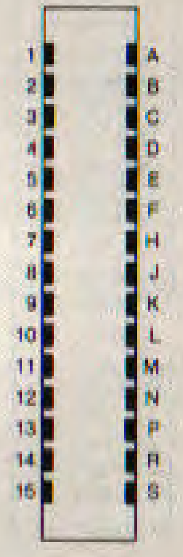
IEEE Connector

Pin	Type	Pin	Type
1	D1	A	D5
2	D2	B	D6
3	D3	C	D7
4	D4	D	D8
5	FOI	E	REN
6	DAV	F	GND
7	NRFD	H	GND
8	NDAC	J	GND
9	IFC	K	GND
10	SRQ	L	GND
11	ATN	M	GND
12	SHIELD	N	GND



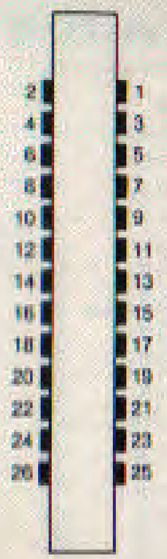
RS232—C Connector

Pin	Type	Pin	Type
1	SHIELD	14	NC
2	T × D	15	NC
3	R × D	16	NC
4	RTS	17	NC
5	CTS	18	- 12V
6	DSR	19	NC
7	GND	20	DTR
8	DCD	21	NC
9	NC	22	NC
10	NC	23	NC
11	+ 5V	24	XMIT CLK
12	NC	25	NC
13	NC		



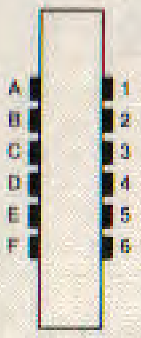
Cartridge Connector

Pin	Type	Pin	Type
1	AB0	A	BDB0
2	AB1	B	BDB1
3	AB2	C	BDB2
4	AB3	D	BDB3
5	AB4	E	BDB4
6	AB5	F	BDB5
7	AB6	H	BDB6
8	AB7	J	BDB7
9	AB8	K	GND
10	AB9	L	GND
11	AB10	M	SR/W
12	AB11	N	SΦ2
13	AB12	P	CSBANK1
14	+ 5V	R	CSBANK2
15	+ 5V	S	CSBANK3



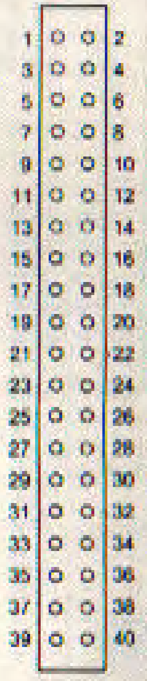
User Connector

Pin	Type	Pin	Type
1	GND	14	2D0
2	PB2	15	1D7
3	GND	16	1D6
4	PB3	17	1D5
5	PC	18	1D4
6	FLAG	19	1D3
7	2D7	20	1D2
8	2D6	21	1D1
9	2D5	22	1D0
10	2D4	23	CNT
11	2D3	24	+ 5V
12	2D2	25	IRQ
13	2D1	26	SP



Cassette Connector

Pin	Type	Pin	Type
1/A	GND	4/D	CAS READ
2/B	+ 5V	5/E	CAS WRITE
3/C	CAS MOTOR	6/F	CAS SWITCH



Co—Processor Dram Bus Connector

Pin	Type	Pin	Type
1	EXPMAD3	21	GND
2	DRAMO0	22	BUSY1
3	EXPMAD2	23	GND
4	DRAMD1	24	P2REFREQ
5	EXPMAD7	25	GND
6	DRAMO2	26	P2REFGNT
7	EXPMAD6	27	GND
8	DRAMD3	28	BP0
9	EXPMAD5	29	GND
10	DRAMD4	30	BP1
11	EXPMAD4	31	GND
12	DRAMD5	32	BP2
13	EXPMAD1	33	SPARE
14	DRAMD6	34	BP3
15	EXPMAD0	35	PROCRES
16	DRAMD7	36	BUSY2
17	GND	37	EXTBUFR/W
18	GND	38	ERAS
19	GND	39	DRAMR/W
20	GND	40	ECAS

Audio/Video 5—Pin Din Connector

Pin	Type
1	+ 12V(LK6)/ + 5V(LK7)
2	GND
3	Audio Out
4	Comp. Video
5	Audio In

Game Port 1

Pin	Type
1	Game 0
2	Game 1
3	Game 2
4	Game 3
5	Pot Y
6	Trigger(Light Pen)
7	+ 5V
8	GND
9	Pot X

Game Port 2

Pin	Type
1	GAME 0
2	GAME 1
3	GAME 2
4	GAME 3
5	Pot Y
6	Trigger
7	+ 5V
8	GND
9	Pot X

Game Ports are 9 Pin D-Subminiature Male Plug Connectors

Commodore 128 System Power Supplies

Voltage	Current
+ 5 V	5/5 A
+ 12 V	0.4 A
– 12 V	0.3 A

System Power Consumption

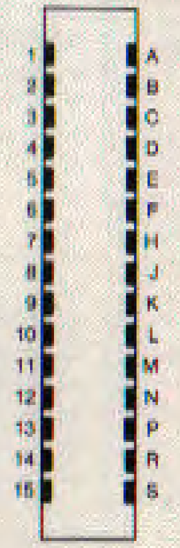
Item	Voltage	Current (Max)
Main PC Board	+ 5 V + 12 V – 12 V	4.1 A 0.150 A 0.050 A
Co-Processor	+ 5 V	0.8 A
Low Cost Disk	+ 5 V	0.225 A
Cartridge	+ 5 V	0.500 A 0.050 A
RS232-C	+ 5 V – 12 V	0.075 A
Cassette	+ 5 V + 12 V	0.050 A 0.150 A

Notes

1. Maximum Power Output of Power Supply is 32 Watts. Therefore Maximum Current for all outputs can not be provided simultaneously.

2. Maximum System with all features drawing Maximum Power will exceed power rating of Supply.

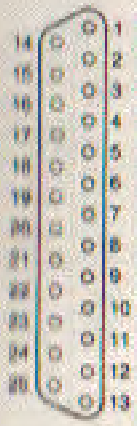
B Series



Connector Pin—Outs

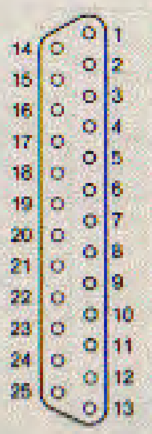
Pin	Type	Pin	Type
1	RO	A	BD0
2	A1	B	BD1
3	A2	C	BD2
4	A3	D	BD3
5	A4	E	BD4
6	A5	F	BD5
7	A6	H	BD6
8	A7	J	BD7
9	A8	K	GND
10	A9	L	GND
11	A10	M	SR/W
12	A11	N	SO2
13	A12	P	NOT CSBANK 1
14	+ 5 VDC	R	NOT CSBANK 2
15	+ 5 VDC	S	NOT CSBANK 3

Keyboard Connector



Pin	Type	Pin	Type
1	PA0	2	PA2
3	PA4	4	PA6
5	PB0	6	PB1
7	PB2	8	PB3
9	PB4	10	PB5
11	PB6	12	PB7
13	PC5	14	PA1
15	PA3	16	PA5
17	PA7	18	PC0
19	PC1	20	PC2
21	PC3	22	GND
23	GND	24	GND
25	PC4		

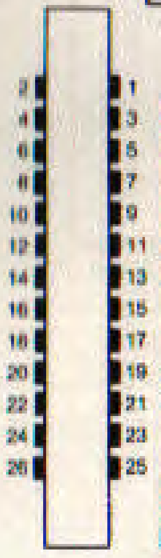
RS 232C Connector



Pin	Type
1	SHIELD
2	T × D
3	R × D
4	RTS
5	CTS
6	DSR
7	GND
8	DCD
11	+ 5 VDC
18	– 12 VDC
20	DTR
24	R × C

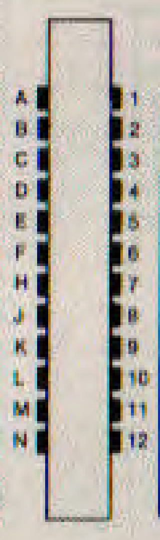
All others N.C.

User Connector

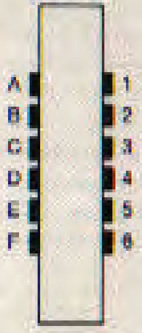


Pin Type	Pin Type
1 - GND	2 - PB2
3 - GND	4 - PB3
5 - NOT PC	6 - NOT FLAG
7 - 2D7	8 - 2D6
9 - 2D5	10 - 2D4
11 - 2D3	12 - 2D2
13 - 2D1	14 - 2D0
15 - 1D7	16 - 1D6
17 - 1D5	18 - 1D4
19 - 1D3	20 - 1D2
21 - 1D1	22 - 1D0
23 - NOT CNT	24 - + 5 VDC
25 - NOT IRQ	26 - SP

IEEE Connector

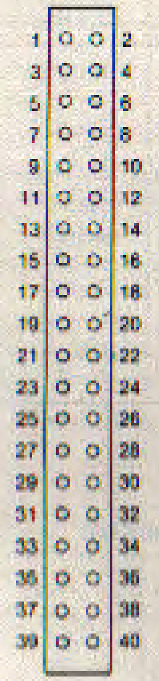


Pin	Type	Pin	Type
1	D1	A	D5
2	D2	B	D6
3	D3	C	D7
4	D4	D	D8
5	EOI	E	REN
6	DAV	F	GND
7	NRFD	H	GND
8	NDAC	J	GND
9	IFC	K	GND
10	SRQ	L	GND
11	ATN	M	GND
12	SHIELD	N	GND



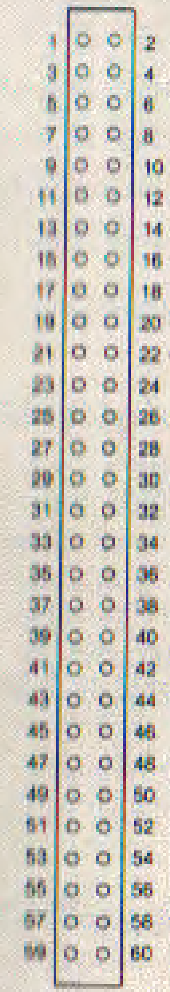
Cassette Connector

Pin	Type
1.A	GND
2.B	+5 VDC (CASSETTE ONLY)
3.C	CASMTR
4.D	CASREAD
5.E	CASWR
6.F	CASSW



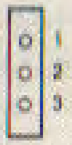
Co-Processor Connector

Pin	Type	Pin	Type
1	EXTMA3	2	DRAMO0
3	EXTMA2	4	DRAMO1
5	EXTMA7	6	DRAMO2
7	EXTMA6	8	DRAMO3
9	EXTMA5	10	DRAMO4
11	EXTMA4	12	DRAMO5
13	EXTMA1	14	DRAMO6
15	EXTMA0	16	DRAMO7
17	GND	18	GND
19	GND	20	GND
21	GND	22	NOT BUSY1
23	GND	24	NOT P2REFREQ
25	GND	26	NOT P2REFGRNT
27	GND	28	BP0
29	GND	30	BP1
31	GND	32	BP2
33	N.C	34	BP3
35	NOT·PROGRES	36	NOT BUSY2
37	EXTBUFR/W	38	NOT ERAS
39	DRAM R/W	40	NOT ECAS



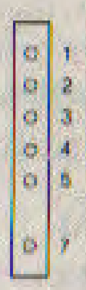
Expansion Connector

Pin	Type	Pin	Type
1	+5 VDC	2	+5 VDC
3	+5 VDC	4	+5 VDC
5	GND	6	GND
7	GND	8	GND
9	GND	10	GND
11	NOT BRAS	12	IRQ3
13	−12 VDC	14	NOT EXTRES
15	+12 VDC	16	NOT S.O
17	NOT RES	18	LPEN
19	SR/W	20	NOT EXTBUFCS
21	TODCLK	22	NOT DISKROMCS
23	BOOTCLK	24	N.C
25	S02	26	NOT BCAS
27	S01	28	NOT CS1
29	BD3	30	NOT EXTPRTCS
31	BD4	32	BD2
33	BD5	34	BD1
35	DB7	36	BD0
37	BA13	38	BD7
39	BA14	40	BA15
41	BA1	42	BA0
43	BA2	44	BA11
45	BA3	46	BA10
47	BA12	48	BA4
49	BA9	50	BA5
51	BA8	52	BA6
53	BP0	54	BA7
55	BP1	56	BP2
57	NOT NMI	58	BP3
59	RDY	60	NOT IRQ



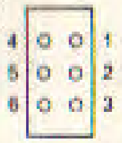
Audio Jack

Pin	Type
1	TO SPEAKER
2	N.C
3	TO SPEAKER



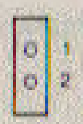
Video Connector

Pin	Type
1	VIDEO
2	GND
3	VERTICAL SYNC
4	GND
5	HORIZONTAL SYNC
6	KEY
7	GND



Power Connector

Pin	Type
1	50/60 Hz
2	−12 VDC
3	+12 VDC
4	GND
5	GND
6	+5 VDC



Reset Connector

Pin	Type
1	TO RESET SWITCH
2	TO RESET SWITCH

APPENDIX I

CONVERTING FROM STANDARD BASIC TO EXTENDED BASIC 4.0

If you have programs written in a BASIC other than Commodore BASIC, some minor adjustments may be necessary before running them with Commodore BASIC. Here are some specific things to look for when converting BASIC programs.

String Dimensions

Delete all statements that are used to declare the length of strings. A statement such as DIM A$(I,J), which dimensions a string array for J elements of length I, should be converted to the Commodore BASIC statement DIM A$(J).

Some BASICs use a comma or ampersand for string concatenation. Each of these must be changed to a plus sign, which is the operator for Commodore BASIC string concatenation.

In Commodore BASIC, the MID$, RIGHT$, and LEFT$ functions are used to take substrings of strings. Forms such as A$(I) to access the "Ith" character in A$, or A$(I,J) to take a substring of A$ from position I to position J, must be changed as follows:

Other BASIC	Commodore BASIC
A$(I)=X$	A$=LEFT$(A$, I−1)+X$+MID$(A$, 160z1)
A$(I,J)=X$	A$=LEFT$(A$,I−1)+X$+MID$(A$,J+1)

Multiple Assignments

Some BASICs allow statements of the form:

```
10 LET B=C=0
```

to set B and C equal to zero. Commodore BASIC would interpret the second equal sign as a logical operator and set B equal to -1 if C equaled 0. Instead, convert this statement to two assignment statements:

```
10 C=0:B=0
```

Multiple Statements

Some BASICs use a backslash to separate multiple statements on u line. With Commodore BASIC, be sure all statements on a line are separated by a colon.

MAT Functions

Programs using the MAT functions available in some BASICs must be rewritten using FOR . . . NEXT loops to execute properly.

Differences From Older Commodore BASIC

TI references must be changed. The current smallest unit of time is 1/10 sec. rather than 1/60 sec. Therefore, VAL (TI$)*6 equals previous version's TI value.

ER is now a reserved variable. All references must be changed to use a new variable name. ER returns the error number (127 is no error).

EL is now a reserved variable. All references must be changed to use a new variable name. EL returns the line number of the last error (65535 is no error).

APPENDIX J

ERROR MESSAGES

#	MESSAGE	EXPLANATION
0.	?stop key detected	Occurs when doing a KERNAL I/O function and the STOP key is pressed. May occur during LOAD or SAVE (or OPEN, CLOSE, GET#, INPUT#, PRINT# when the cassette tape is moving). Cassette tape files should be CLOSEd to save previously written information. Disk files are not damaged.
1.	?too many files	You are trying to OPEN more than 10 files at a time. Decrease the number of OPEN or DOPEN files by CLOSING them.
2.	?file open	An attempt was made to redefine file parameter information by repeating an OPEN command on the same file twice.
3.	?file not open	The operating system must have device number and command information provided by the OPEN statement. If an attempt is made to read or write a file without having done this previously, then this message appears.
4.	?file not found	The named file specified in OPEN or LOAD was not found on the device specified. In the case of tape I/O, an end of tape mark was encountered.
5.	?device not present	No device on the IEEE was present to handshake an attention sequence. May happen on OPEN, CLOSE, CMD, INPUT#, GET#, PRINT#. If filename is not specified with OPEN, this error will not occur.
6.	?not input file	Tape files, once opened for writing, cannot be read without first CLOSEing, rewinding tape, and OPENing for INPUT. This message appears when an attempt is made to read an output file.
7.	?not output file	Tape files cannot be read and updated in place. Device is the keyboard and it cannot be written to.
8.	?missing filename	LOADs and SAVEs from the IEEE port (e.g., the disk) require a filename to be specified. Supply the filename.
9.	?illegal device number	Occurs if you try to access a device in an illegal manner. For example, LOADing or SAVEing on the keyboard, screen, or RS-232.
10.	are you sure ?	This is a prompt for BACKUP, SCRATCH, and HEADER. It is not an error message and

#	MESSAGE	EXPLANATION
		should not occur during BASIC program execution.
11.	?bad disk	Media failure on HEADER command.
14.	break	This occurs when the STOP key is pressed during normal BASIC execution. The CONTinue command can be used to restart the program.
15.	extra ignored	Too many items of data or separators (,) were typed in response to an INPUT statement. Only the first few items were accepted.
16.	redo from start	Is not actually a fatal error printed in the standard format but is a diagnostic which is printed when data in response to INPUT is non-numeric where a numeric quantity is required. The INPUT continues to function until acceptable data has been received.
20.	?next without for	Either a NEXT is improperly nested or the variable in a NEXT statement corresponds to no previously executed FOR statement.
21.	?syntax error	BASIC cannot recognize the statement you have typed. Caused by such things as missing parentheses, illegal characters, incorrect punctuation, misspelled keyword.

#	MESSAGE	EXPLANATION
22.	?return without gosub	A RETURN statement was encountered without a previous GOSUB statement being executed.
23.	?out of data	A READ statement was executed but all of the data statements in the program have been read. The program tried to read too much data, or insufficient data was included in the program. Carriage returning through a line READY on the Commodore 128 or B Series video display sometimes yields this error because the message is interpreted as READ Y.
24.	illegal quantity	Occurs when a function is accessed with a parameter out of range caused by: 1. A matrix subscript out of range (0 < X < 32767) 2. LOG (negative or zero argument) 3. SQR (negative argument) 4. A↑B where A<0 and B not integer. 5. Call of USR before machine language subroutine has been patched in. 6. Use of string functions MID$, LEFT$, RIGHT$, with length parameters out of range (1 < X < 255). 7. Index on . . . GOTO out of range. 8. Addresses specified for PEEK, POKE, WAIT, and SYS out of range (1 < X < 255). 9. Byte parameters of WAIT,

#	MESSAGE	EXPLANATION
		POKE, TAB and SPC out of range (0 < X < 255).
25.	overflow	Numbers resulting from computations or input that are larger than binary 1.70141184E+38 cannot be represented in BASIC's number format. Underflow is not a detectable error but numbers less than binary 2.93873587E−39 are indistinguishable from zero.
26.	?out of memory	May appear while entering or editing a program as the text completely fills memory. At run time, assignment and creation of variables may also fill all variable memory. Array available declarations consume large areas of memory even though a program may be rather short. The maximum number of FOR loops and simultaneous GOSUBs are dependent on each other. This context is stored on the microprocessor hardware stack whose capacity may be exceeded. To determine the type of memory error, examine the results of FRE. If there is a large number of bytes available, it is most likely a FOR-NEXT or GOSUB problem. A subroutine which terminates in GOTO rather than RETURN will eventually cause an out of memory error as stack pointers build up.
27.	?undefined statement	An attempt was made to GOTO, GOSUB, or THEN to a statement which does not exist.
28.	?bad subscript	An attempt was made to reference a matrix element which is outside the dimensions of the matrix. This may happen by specifying the wrong number of dimensions or a subscript larger than specified in the original dimension.
29.	?redim'd array	After an array was dimensioned, another dimension statement for the same array was encountered. For example, an array variable is defined by default when it is first used, and later a DIM statement is encountered.
30.	?division by zero	Zero as a divisor would result in numeric overflow—thus it is not allowed. When this message appears, it is most expedient to list the statement and look for division operators.
31.	?illegal direct	A single 80 column buffer area is used by BASIC to process incoming characters. This same buffer is used to hold a statement that is being interpreted in direct mode. INPUT will not work because incoming characters would overwrite the variable list following INPUT to be processed. DEF cannot be used in direct

#	MESSAGE	EXPLANATION
		mode for a different but similar reason. The name of a function is stored in the BASIC variable area with pointers to the string of characters which define the function. Since the function exists only in the input buffer, it is wiped out the first time a NEW command IS typed in.
32.	?type mismatch	The left-hand side of an assignment statement was a numeric variable and the right-hand side was a string, or vice versa; or a function which expected a string argument was given a numeric one, or vice versa.
33.	?string too long	Attempt by use of the concatenation operator to create a string more than 255 characters long.
34.	?file data	Occurs when an INPUT# statement finds a string while attempting to read a numeric value.
35.	?formula too complex	This indicates that BASIC has run out of string temporary pointers to keep track of substrings in evaluating a string expression. Break the string expression into two smaller parts to cure the problem.
37.	?undefined function	Reference was made to a user defined function which had never been defined.
38.	?load error	Only occurs when loading a program from cassette tape. This means that there were more than 31 errors in the first tape block or that there were errors in exactly the same corresponding positions of both blocks.
39.	?verify error	The contents of memory and a specified file do not compare.
40.	?out of stack	Too many levels of FOR . . . NEXT or GOSUBs have been executed. No recovery possible.
41.	?unable to resume	A fatal error has occurred, such as running out of stack.
42.	?unable to dispose	All of the DISPOSE type items have been disposed of or none exist.
43.	?out of text	If any LOAD or DLOAD exceeds the end of the text bank of (64K) this error will result. This error will not occur when using the BLOAD command.

APPENDIX K

NONERROR MESSAGES

The messages listed below are available through the ERROR MESSAGE code numbers by using the ERR$ calling codes listed next to each message. However, these messages are *not* Error Messages so they *will not appear* on the screen unless you specifically call for them in your programming or call for them as a standard operating procedure.

MESSAGE	EXPLANATION
12. (carriage return) ready (carriage return)	This message lets you know that your system is ready to use.
13. (space) in (space)	This message is similar to ready.
17. your last "evaluated" number	This is the last number that has been evaluated through the numerical output buffer. (e.g., print 10*10: if you use an ER$ code 17, the number on your screen will equal the last evaluation—in this case, 100.)
18. more (carriage return)	
19. power on message	***COMMODORE BASIC 128, B4.0*** ***COMMODORE BASIC 256, B4.0***

APPENDIX L

6581 (SID) CHIP NOTE VALUE TABLE (C128-40 ONLY)

MUSICAL	OSCILLATOR			MUSICAL	OSCILLATOR		
NOTE	FREQUENCY			NOTE	FREQUENCY		
	DECIMAL	HI	LO		DECIMAL	HI	LO
0 C0	268	1	12	32 C2	1072	4	48
1 C#0	284	1	28	33 C#2	1136	4	112
2 D0	301	1	45	34 D2	1204	4	180
3 D#0	318	1	62	35 D#2	1275	4	251
4 E0	337	1	81	36 E2	1351	5	71
5 F0	358	1	102	37 F2	1432	5	152
6 F#0	379	1	123	38 F#2	1517	5	237
7 G0	401	1	145	39 G2	1607	6	71
8 G#0	425	1	169	40 G#2	1703	6	167
9 A0	451	1	195	41 A2	1804	7	12
10 A#0	477	1	221	42 A#2	1911	7	119
11 B0	506	1	250	43 B2	2025	7	233
16 C1	536	2	24	48 C3	2145	8	97
17 C#1	568	2	56	49 C#3	2273	8	225
18 D1	602	2	90	50 D3	2408	9	104
19 D#1	637	2	125	51 D#3	2551	9	247
20 E1	675	2	163	52 E3	2703	10	143
21 F1	716	2	204	53 F3	2864	11	48
22 F#1	758	2	246	54 F#3	3034	11	218
23 G1	803	3	35	55 G3	3215	12	143
24 G#1	851	3	83	56 G#3	3406	13	78
25 A1	902	3	134	57 A3	3608	14	24
26 A#1	955	3	187	58 A#3	3823	14	239
27 B1	1012	3	244	59 B3	4050	15	210

MUSICAL NOTE	OSCILLATOR FREQUENCY DECIMAL	HI	LO	MUSICAL NOTE	OSCILLATOR FREQUENCY DECIMAL	HI	LO
64 C4	4291	16	195	96 C6	17167	67	15
65 C#4	4547	17	195	97 C#6	18188	71	12
66 D4	4817	18	209	98 D6	19269	75	69
67 D#4	5103	19	239	99 D#6	20415	79	191
68 E4	5407	21	31	100 E6	21629	84	125
69 F4	5728	22	96	101 F6	22915	89	131
70 F#4	6069	23	181	102 F#6	24278	94	214
71 G4	6430	25	30	103 G6	25721	100	121
72 G#4	6812	26	156	104 G#6	27251	106	115
73 A4	7217	28	49	105 A6	28871	112	199
74 A#4	7647	29	223	106 A#6	30588	119	124
75 B4	8101	31	165	107 B6	32407	126	151
80 C5	8583	33	135	112 C7	34334	134	30
81 C#5	9094	35	134	113 C#7	36376	142	24
82 D5	9634	37	162	114 D7	38539	150	139
83 D#5	10207	39	223	115 D#7	40830	159	126
84 E5	10814	42	62	116 E7	43258	168	250
85 F5	11457	44	193	117 F7	45830	179	6
86 F#5	12139	47	107	118 F#7	48556	189	172
87 G5	12860	50	60	119 G7	51443	200	243
88 G#5	13625	53	57	120 G#7	54502	212	230
89 A5	14435	56	99	121 A7	57743	225	143
90 A#5	15294	59	190	122 A#7	61176	238	248
91 B5	16203	63	75	123 B7	64814	253	46

APPENDIX M

6581 (SID) CHIP REGISTER MAP

The 6581 Sound Interface Device is located starting at location 55808 ($DA00). Below is a brief register map. For detailed information, consult the Programmer's Reference Guide.

	ADDRESS					REG #	DATA								REG NAME	REG
	A_4	A_3	A_2	A_1	A_0	(HEX)	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Voice 1	TYPE
0	0	0	0	0	0	00	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	FREQ LO	WRITE-ONLY
1	0	0	0	0	1	01	F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	FREQ HI	WRITE-ONLY
2	0	0	0	1	0	02	PW_7	PW_6	PW_5	PW_4	PW_3	PW_2	PW_1	PW_0	PW LO	WRITE-ONLY
3	0	0	0	1	1	03	—	—	—	—	PW_{11}	PW_{10}	PW_9	PW_8	PW HI	WRITE-ONLY
4	0	0	1	0	0	04	NOISE	⊓⊓	◿◿	⋀⋀	TEST	RING MOD	SYNC	GATE	CONTROL REG	WRITE-ONLY
5	0	0	1	0	1	05	ATK_3	ATK_2	ATK_1	ATK_0	DCY_3	DCY_2	DCY_1	DCY_0	ATTACK/DECAY	WRITE-ONLY
6	0	0	1	1	0	06	STN_3	STN_2	STN_1	STN_0	RLS_3	RLS_2	RLS_1	RLS_0	SUSTAIN/RELEASE	WRITE-ONLY
															Voice 2	
7	0	0	1	1	1	07	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	FREQ LO	WRITE-ONLY
8	0	1	0	0	0	08	F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	FREQ HI	WRITE-ONLY
9	0	1	0	0	1	09	PW_7	PW_6	PW_5	PW_4	PW_3	PW_2	PW_1	PW_0	PW LO	WRITE-ONLY
10	0	1	0	1	0	0A	—	—	—	—	PW_{11}	PW_{10}	PW_9	PW_8	PW HI	WRITE-ONLY
11	0	1	0	1	1	0B	NOISE	⊓⊓	◿◿	⋀⋀	TEST	RING MOD	SYNC	GATE	CONTROL REG	WRITE-ONLY
12	0	1	1	0	0	0C	ATK_3	ATK_2	ATK_1	ATK_0	DCY_3	DCY_2	DCY_1	DCY_0	ATTACK/DECAY	WRITE-ONLY
13	0	1	1	0	1	0D	STN_3	STN_2	STN_1	STN_0	RLS_3	RLS_2	RLS_1	RLS_0	SUSTAIN/RELEASE	WRITE-ONLY
															Voice 3	
14	0	1	1	1	0	0E	F_7	F_6	F_5	F_4	F_3	F_2	F_1	F_0	FREQ LO	WRITE-ONLY
15	0	1	1	1	1	0F	F_{15}	F_{14}	F_{13}	F_{12}	F_{11}	F_{10}	F_9	F_8	FREQ HI	WRITE-ONLY
16	1	0	0	0	0	10	PW_7	PW_6	PW_5	PW_4	PW_3	PW_2	PW_1	PW_0	PW LO	WRITE-ONLY
17	1	0	0	0	1	11	—	—	—	—	PW_{11}	PW_{10}	PW_9	PW_8	PW HI	WRITE-ONLY
18	1	0	0	1	0	12	NOISE	⊓⊓	◿◿	⋀⋀	TEST	RING MOD	SYNC	GATE	CONTROL REG	WRITE-ONLY
19	1	0	0	1	1	13	ATK_3	ATK_2	ATK_1	ATK_0	DCY_3	DCY_2	DCY_1	DCY_0	ATTACK/DECAY	WRITE-ONLY
20	1	0	1	0	0	14	STN_3	STN_2	STN_1	STN_0	RLS_3	RLS_2	RLS_1	RLS_0	SUSTAIN/RELEASE	WRITE-ONLY
															Filter	
21	1	0	1	0	1	15	—	—	—	—	—	FC_2	FC_1	FC_0	FC LO	WRITE-ONLY
22	1	0	1	1	0	16	FC_{10}	FC_9	FC_8	FC_7	FC_6	FC_5	FC_4	FC_3	FC HI	WRITE-ONLY
23	1	0	1	1	1	17	RES_3	RES_2	RES_1	RES_0	FILTEX	FILT 3	FILT 2	FILT 1	RES/FILT	WRITE-ONLY
24	1	1	0	0	0	18	3 OFF	HP	BP	LP	VOL_3	VOL_2	VOL_1	VOL_0	MODE/VOL	WRITE-ONLY
															Misc.	
25	1	1	0	0	1	19	PX_7	PX_6	PX_5	PX_4	PX_3	PX_2	PX_1	PX_0	POT X	READ-ONLY
26	1	1	0	1	0	1A	PY_7	PY_6	PY_5	PY_4	PY_3	PY_2	PY_1	PY_0	POT Y	READ-ONLY
27	1	1	0	1	1	1B	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0	OSC_3/RANDOM	READ-ONLY
28	1	1	1	0	0	1C	E_7	E_6	E_5	E_4	E_3	E_2	E_1	E_0	ENV_3	READ-ONLY

APPENDIX N

6567 (VIC) CHIP REGISTER MAP C128-40

The 6567 Video Interface Chip is located starting at location 55296 ($D800). Below is a brief register map. For explanation see your Programmer's Reference Guide.

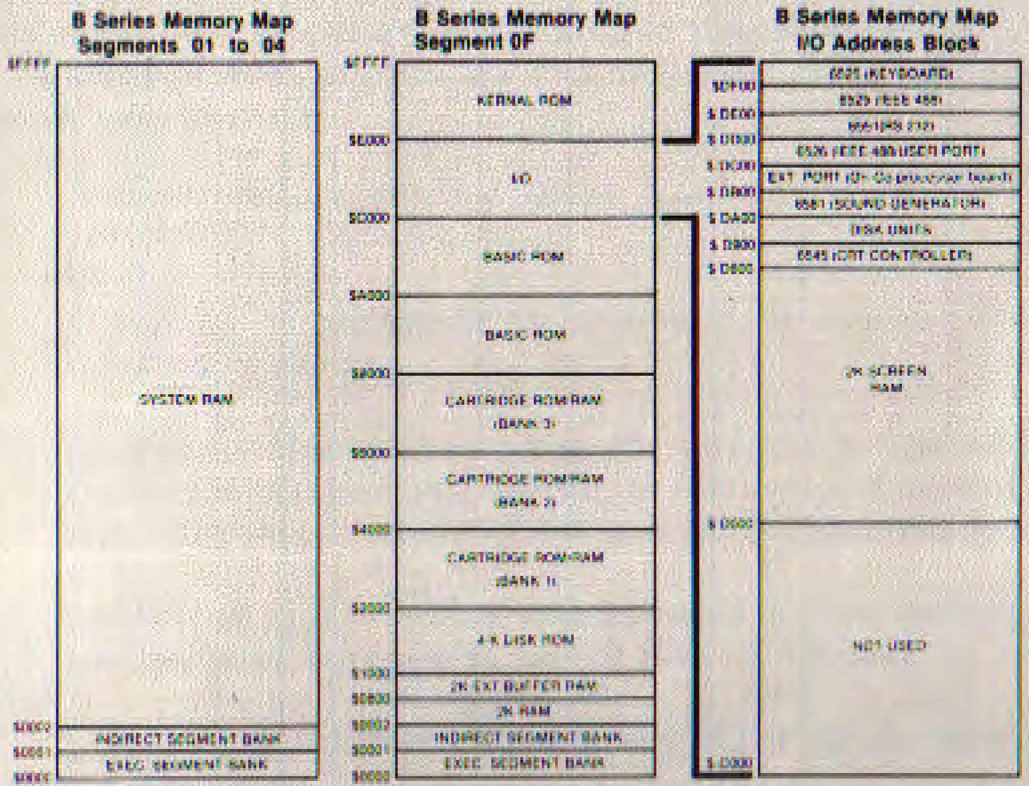
REGISTER MAP

ADDRESS		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DESCRIPTION
00	($00)	M0X7	M0X6	M0X5	M0X4	M0X3	M0X2	M0X1	M0X0	MOB 0 X-position
01	($01)	M0Y7	M0Y6	M0Y5	M0Y4	M0Y3	M0Y2	M0Y1	M0Y0	MOB 0 Y-position
02	($02)	M1X7	M1X6	M1X5	M1X4	M1X3	M1X2	M1X1	M1X0	MOB 1 X-position
03	($03)	M1Y7	M1Y6	M1Y5	M1Y4	M1Y3	M1Y2	M1Y1	M1Y0	MOB 1 Y-position
04	($04)	M2X7	M2X6	M2X5	M2X4	M2X3	M2X2	M2X1	M2X0	MOB 2 X-position
05	($05)	M2Y7	M2Y6	M2Y5	M2Y4	M2Y3	M2Y2	M2Y1	M2Y0	MOB 2 Y-position
06	($06)	M3X7	M3X6	M3X5	M3X4	M3X3	M3X2	M3X1	M3X0	MOB 3 X-position
07	($07)	M3Y7	M3Y6	M3Y5	M3Y4	M3Y3	M3Y2	M3Y1	M3Y0	MOB 3 Y-position
08	($08)	M4X7	M4X6	M4X5	M4X4	M4X3	M4X2	M4X1	M4X0	MOB 4 X-position
09	($09)	M4Y7	M4Y6	M4Y5	M4Y4	M4Y3	M4Y2	M4Y1	M4Y0	MOB 4 Y-position
10	($0A)	M5X7	M5X6	M5X5	M5X4	M5X3	M5X2	M5X1	M5X0	MOB 5 X-position
11	($0B)	M5Y7	M5Y6	M5Y5	M5Y4	M5Y3	M5Y2	M5Y1	M5Y0	MOB 5 Y-position
12	($0C)	M6X7	M6X6	M6X5	M6X4	M6X3	M6X2	M6X1	M6X0	MOB 6 X-position
13	($0D)	M6Y7	M6Y6	M6Y5	M6Y4	M6Y3	M6Y2	M6Y1	M6Y0	MOB 6 Y-position
14	($0E)	M7X7	M7X6	M7X5	M7X4	M7X3	M7X2	M7X1	M7X0	MOB 7 X-position
15	($0F)	M7Y7	M7Y6	M7Y5	M7Y4	M7Y3	M7Y2	M7Y1	M6Y0	MOB 7 Y-position
16	($10)	M7X8	M6X8	M5X8	M4X8	M3X8	M2X8	M1X8	M0X8	MSB of X-position
17	($11)	RC8	ECM	BMM	DEN	RSEL	Y2	Y1	Y0	See text
18	($12)	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	Raster register
19	($13)	LPX8	LPX7	LPX6	LPX5	LPX4	LPX3	LPX2	LPX1	Light Pen X
20	($14)	LPY7	LPY6	LPY5	LPY4	LPY3	LPY2	LPY1	LPY0	Light Pen Y
21	($15)	M7E	M6E	M5E	M4E	M3E	M2E	M1E	M0E	MOB Enable
22	($16)	—	—	RES	MCM	CSEL	X2	X1	X0	See text
23	($17)	M7YE	M6YE	M5YE	M4YE	M3YE	M2YE	M1YE	M0YE	MOB Y-expand
24	($18)	VM13	VM12	VM11	VM10	CB13	CB12	CB11	—	Memory Pointers
25	($19)	IRQ	—	—	—	ILP	IMMC	IMBC	IRST	Interrupt Register
26	($1A)	—	—	—	—	ELP	EMMC	EMBC	ERST	Enable Interrupt
27	($1B)	M7DP	M6DP	M5DP	M4DP	M3DP	M2DP	M1DP	M0DP	MOB-DATA Priority
28	($1C)	M7MC	M6MC	M5MC	M4MC	M3MC	M2MC	M1MC	M0MC	MOB Multicolor Sel
29	($1D)	M7XE	M6XE	M5XE	M4XE	M3XE	M2XE	M1XE	M0XE	MOB X-expand
30	($1E)	M7M	M6M	M5M	M4M	M3M	M2M	M1M	M0M	MOB-MOB Collision
31	($1F)	M7D	M6D	M5D	M4D	M3D	M2D	M1D	M0D	MOB-DATA Collision
32	($20)	—	—	—	—	EC3	EC2	EC1	EC0	Exterior Color
33	($21)	—	—	—	—	B0C3	B0C2	B0C1	B0C0	Bkgd #0 Color
34	($22)	—	—	—	—	B1C3	B1C2	B1C1	B1C0	Bkgd #1 Color
35	($23)	—	—	—	—	B2C3	B2C2	B2C1	B2C0	Bkgd #2 Color
36	($24)	—	—	—	—	B3C3	B3C2	B3C1	B3C0	Bkgd #3 Color
37	($25)	—	—	—	—	MM03	MM02	MM01	MM00	MOB Multicolor #0
38	($26)	—	—	—	—	MM13	MM12	MM11	MM10	MOB Multicolor #1
39	($27)	—	—	—	—	M0C3	M0C2	M0C1	M0C0	MOB 0 Color
40	($28)	—	—	—	—	M1C3	M1C2	M1C1	M1C0	MOB 1 Color
41	($29)	—	—	—	—	M2C3	M2C2	M2C1	M2C0	MOB 2 Color
42	($2A)	—	—	—	—	M3C3	M3C2	M3C1	M3C0	MOB 3 Color
43	($2B)	—	—	—	—	M4C3	M4C2	M4C1	M4C0	MOB 4 Color
44	($2C)	—	—	—	—	M5C3	M5C2	M5C1	M5C0	MOB 5 Color
45	($2D)	—	—	—	—	M6C3	M6C2	M6C1	M6C0	MOB 6 Color
46	($2E)	—	—	—	—	M7C3	M7C2	M7C1	M7C0	MOB 7 Color

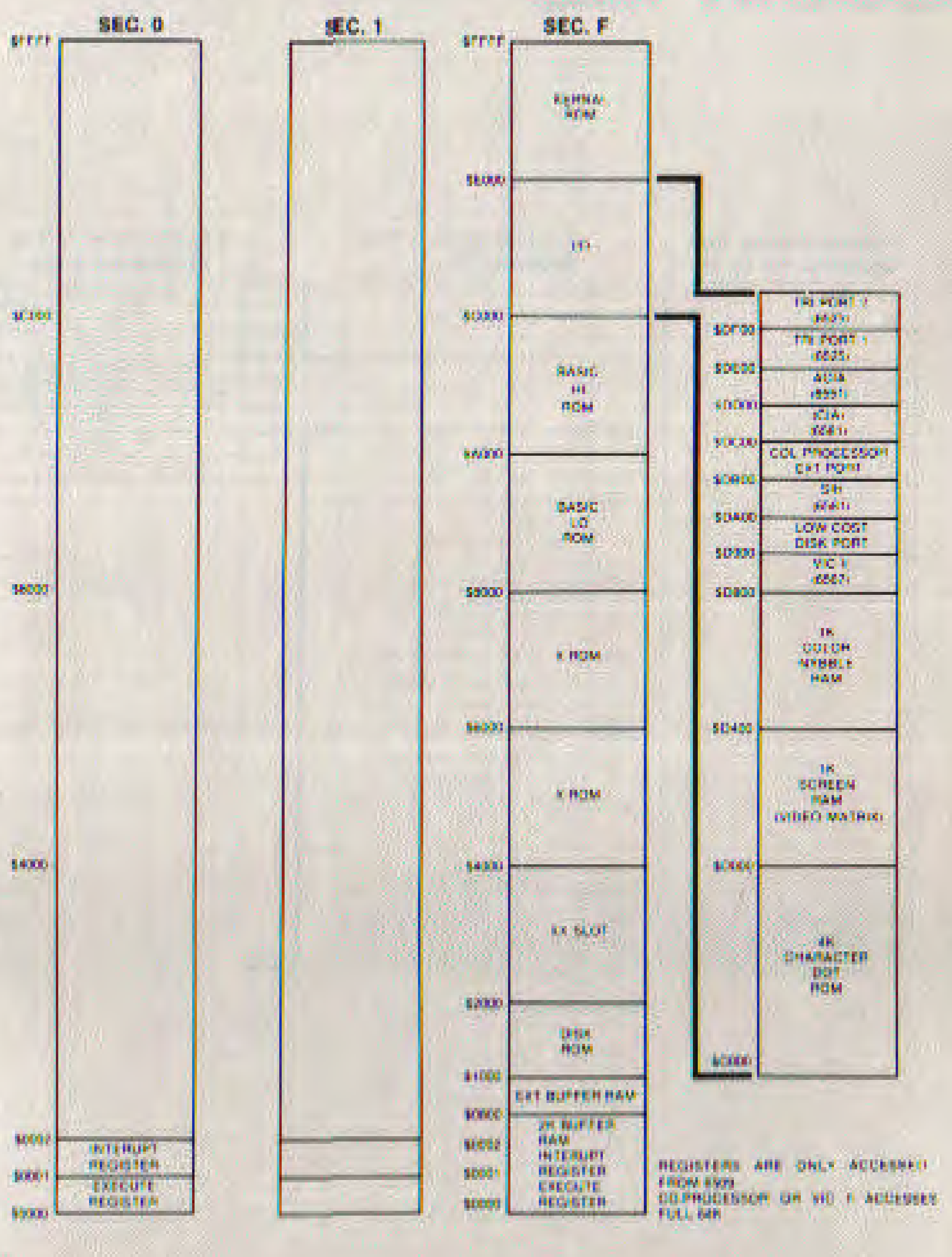
NOTE: A dash indicates a no connect. All no connects are read as a "1."

APPENDIX O

B SERIES MEMORY MAP



C128-40 MEMORY MAP



OPENING THE RS-232 CHANNEL

The OPEN statement for an RS-232 channel has some special arguments that you must understand before you can use it. You must match the operating parameters of the RS-232 interface to those of the device you're connecting to the P-500/B-700.

When you open an RS-232 channel, your OPEN statement must look like this:

```
OPEN filenumber,2,secondary-address,openstring
```

Where:

filenumber is the logical file number to be associated with the RS-232 channel.

secondary-address determines the direction of the RS-232 channel. It can be input, output, or bidirectional and may or may not convert between CBM and ASCII character codes.

openstring is a *four-byte* command stringthat establishes the operating parameters for the RS-232 channel.

The *secondary-address* may take any of the values shown in Table 8.1.

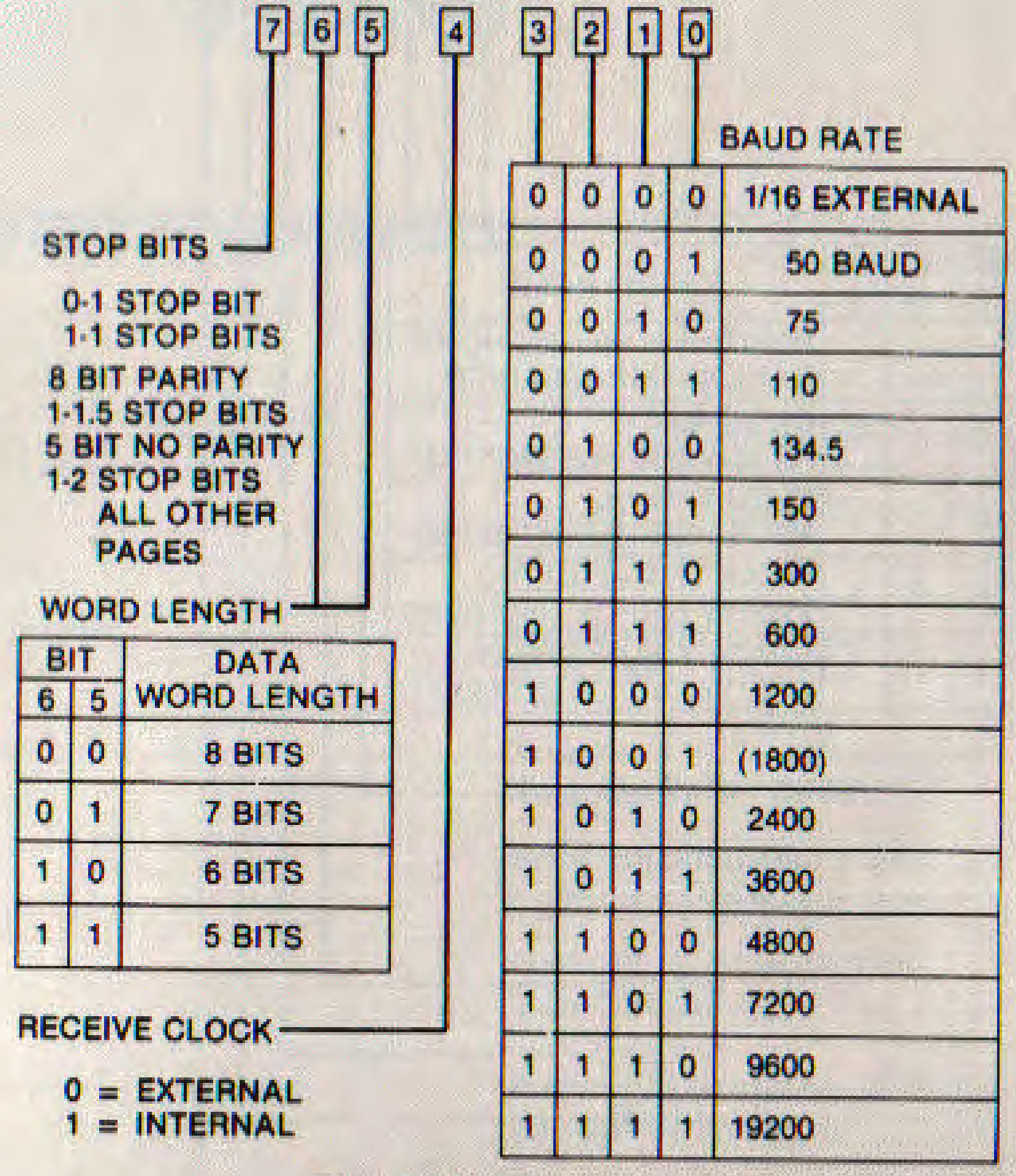
TABLE 8.1 RS-232 DIRECTIONAL SECONDARY ADDRESSES

VALUE	MEANING
1	open an output channel
2	open an input channel
3	open an input/output channel
129	open an output channel and convert CBM to ASCII character codes
130	open an input channel and convert ASCII to CBM character codes
131	open an input/output channel and convert between CBM and ASCII character codes

The *secondary-address* values 1, 2, and 3 do not perform and character conversions. If you're getting ASCII character codes through the RS-232 channel, they are delivered as-is to your program. If you want CBM/ASCII conversion you must select a *secondary-address* value of 129, 130, Or 131.

NOTE: If you are transmitting or receiving non-character data through your RS-232 interface, do NOT request CBM/ASCII character conversion. This will completely scramble your data.

The *openstring* for the RS-232 interface is four bytes long. The first two bytes contain detailed control information. The last two aren't used, but you must include them.



STOP BITS

0-1 STOP BIT
1-1 STOP BITS
8 BIT PARITY
1-1.5 STOP BITS
5 BIT NO PARITY
1-2 STOP BITS
ALL OTHER
PAGES

WORD LENGTH

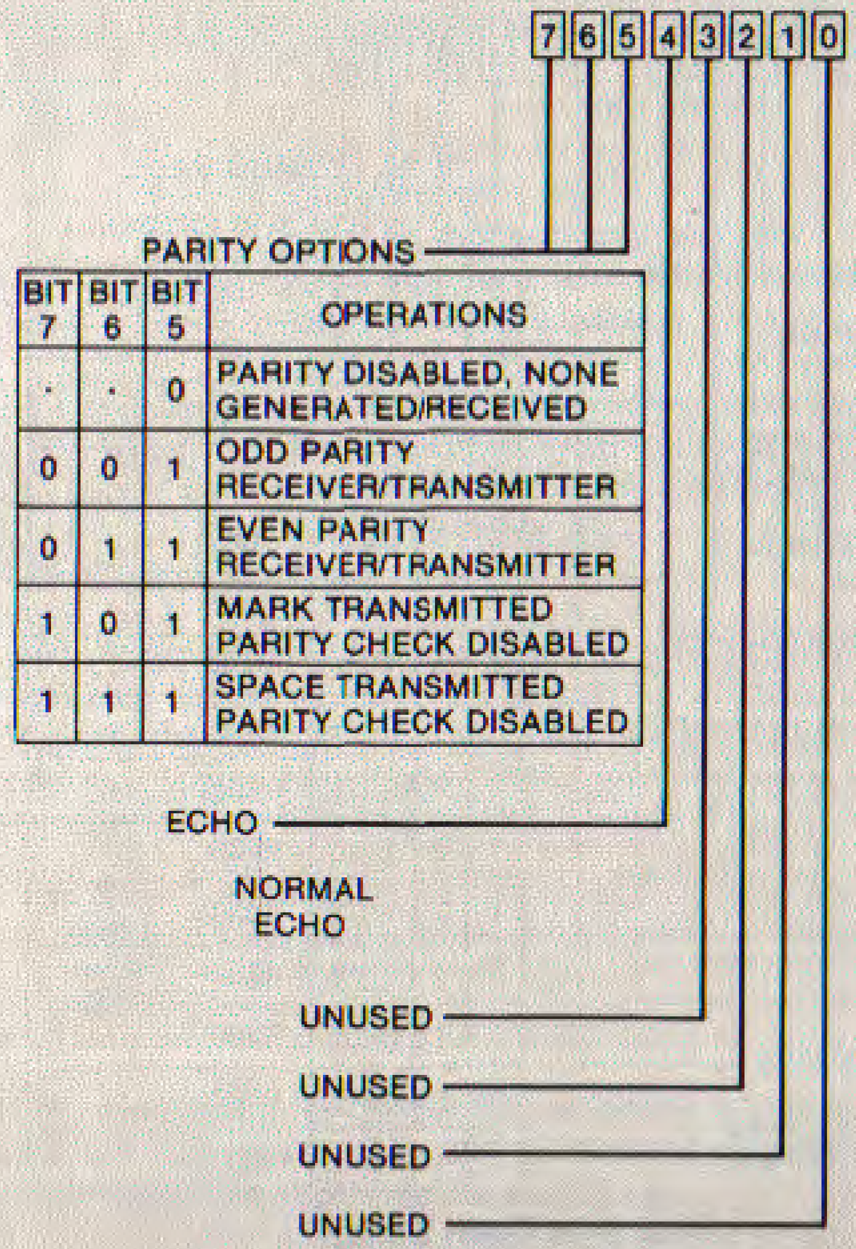
BIT		DATA
6	5	WORD LENGTH
0	0	8 BITS
0	1	7 BITS
1	0	6 BITS
1	1	5 BITS

RECEIVE CLOCK

0 = EXTERNAL
1 = INTERNAL

3	2	1	0	BAUD RATE
0	0	0	0	1/16 EXTERNAL
0	0	0	1	50 BAUD
0	0	1	0	75
0	0	1	1	110
0	1	0	0	134.5
0	1	0	1	150
0	1	1	0	300
0	1	1	1	600
1	0	0	0	1200
1	0	0	1	(1800)
1	0	1	0	2400
1	0	1	1	3600
1	1	0	0	4800
1	1	0	1	7200
1	1	1	0	9600
1	1	1	1	19200

First Byte Open String
RS-232



BIT 7	BIT 6	BIT 5	OPERATIONS
·	·	0	PARITY DISABLED, NONE GENERATED/RECEIVED
0	0	1	ODD PARITY RECEIVER/TRANSMITTER
0	1	1	EVEN PARITY RECEIVER/TRANSMITTER
1	0	1	MARK TRANSMITTED PARITY CHECK DISABLED
1	1	1	SPACE TRANSMITTED PARITY CHECK DISABLED

Second Byte Open String

APPENDIX R

BIBLIOGRAPHY

PUBLISHER	TITLE/AUTHOR
Addison Wesley	*BASIC and the Personal Computer*, Dwyer and Critchfield
Compute	*Compute's First Book of PET/CBM*
Cowbay Computing	*Teacher's PET—Plans, Quizzes and Answers*
	Feed Me, I'm Your PET Computer, Carol Alexander
	Looking Good With Your PET, Carol Alexander
Creative Computing	*Getting Acquainted With Your VIC-20*, T. Hartnell
Dilithium Press	*BASIC Basic-English Dictionary for the Pet*, Larry Noonan
Faulk Baker Associates	*MOS Programming Manual*, MOS Technology
Hayden Book Co.	*BASIC Conversions Handbook: Apple, TRS 80, and PET*, Brain, Oviatt, Paquin, and Stone
	Library of PET Subroutines, Nick Hampshire

PUBLISHER	TITLE/AUTHOR
	PET Graphics, Nick Hampshire
	I Speak BASIC to my PET, Aubrey Jones, Jr.
	BASIC from the Ground Up, David E. Simon
Howard W. Sams	*Mostly BASIC Applications for Your PET*, Howard Berenbon
	PET Interfacing, J. Downey and S. Rogers
	Crash Course in Microcomputers, Louise Frenzol
Little, Brown and Co.	*Computer Games for Businesses, Schools and Homes*, J. Victor Nagigian and William S. Hodges
	The Computer Tutor: Learning Activities for Homes and Schools, Gary W. Orwig and William S. Hodges
McGraw Hill	*Home and Office Use of VisiCalc*, D. Castlewitz and L. Chisauki
	Hands-On BASIC with a PET, Herbert D. Peckman
Osborne/McGraw Hill	*Pet/CBM Personal Computer Guide*, Carroll S. Donahue
	Osborne CP/M User Guide, Thom Hogan
	PET Fun and Games, R. Jeffries and G. Fisher
	PET and the IEEE, A. Osborne and C. Donahue

PUBLISHER	TITLE/AUTHOR
	Some Common Basic Programs, Lon Poole and Mary Borchers
	The 8086 Book, Russell Rector and George Alexy
P.C. Publications	*Beginning Self-Teaching Computer Lessons*
Prentice-Hall, Inc.	*The PET Personal Computer for Beginners*, S. Dunn and V. Morgan
Reston Publishing Co.	*Pet and the IEEE 488 Bus (GPIB)*, Eugene Fisher and C.W. Jensen
	PET BASIC, Richard Huskell
	PET Games and Recreation, Ogelsvy, Lindsey, and Kunkin
	PET BASIC — Training Your PET Computer, Zamora, Carvie, and Albrecht
Total Information Services	*Understanding Your PET/CBM: Vol. 1 BASIC Programming*
	Understanding Your VIC, David Schultz