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.”
COMMODORE 64
CP/M® OPERATING
SYSTEM USER'S
GUIDE

Published by
Commodore Business Machines, Inc.
and
Howard W. Sams & Co., Inc.
# TABLE OF CONTENTS

## 1. INTRODUCTION TO CP/M ON YOUR COMMODORE 64

- 1.1 Overview of CP/M on Your Commodore 64 ...................... 7
- 1.2 How To Use This Manual ........................................ 9
- 1.3 Digital Research License Information ......................... 12
  - 1.3.1 Digital Research License Agreement ...................... 12
- 1.4 Registration Information ....................................... 15
- 1.5 Warranty and Service Information ........................... 15
- 1.6 Get More out of Your Commodore Computer .................. 16
  - 1.6.1 Power/Play: The Home Computer Magazine ................ 16
  - 1.6.2 Commodore: The Microcomputer Magazine ................ 16
  - 1.6.3 Commodore Information Network: The Paperless User Magazine ................ 17

## 2. SETTING UP YOUR COMMODORE 64

- 2.1 Unpacking and Connecting the Z80 Cartridge .................. 19
- 2.2 Installing the Z80 Cartridge ................................... 20
  - 2.2.1 Using the Z80 Cartridge with VIC Peripherals ............ 22

TABLE OF CONTENTS 3
2.2.2 Using the Z80 Cartridge with
CBM Series Peripherals .................. 22
• 2.3 Connecting Disk Drives ............... 24
  2.3.1 Connecting VIC 1541
  Disk Drives .......................... 24
  2.3.2 Connecting CBM Series
  Disk Drives .......................... 24

3. USING YOUR COMMODORE
64 PERIPHERALS FROM
CP/M .................................... 25
  • 3.1 Printer Interface ..................... 26
  • 3.2 The Commodore 64 Serial Interface ...... 27
  • 3.3 The IEEE Interface Cartridge ............ 27
  • 3.4 Daisy Chaining Peripherals ............. 28
  • 3.5 The Commodore 64 User Port .......... 29

4. GETTING STARTED ............... 31
  • 4.1 Bringing CP/M onto Your Commodore 64 32
    4.1.1 Starting CP/M ....................... 33
    4.1.2 Making Copies of Your CP/M
    System Disk .......................... 34
  • 4.2 The COPY Utility .......................... 35
    4.2.1 Formatting a Disk with the
    COPY Utility .......................... 35
    4.2.2 Creating a Disk Backup with the
    COPY Utility .......................... 37
    4.2.3 Copying the System Tracks
    with the COPY Utility .................. 38
  • 4.3 The CONFIG Utility .................... 39
    4.3.1 Using CONFIG to Change the
    Number of Disk Drives ................. 40
    4.3.2 Using CONFIG to Change the
    Printer Type .......................... 41
    4.3.3 Using CONFIG to Change the
    Initial Caps Mode ..................... 41
    4.3.4 Using CONFIG to Change the
    Function Key Assignments ............. 42
    4.3.5 Using CONFIG to Change the
    Key Codes ............................. 44
4.3.6 Using CONFIG to Save the New I/O Setup .................................. 45

- 4.4 Generating a New CP/M System with SYSGEN ..................................... 45
  4.4.1 Relocating CP/M ................................................................. 46
  4.4.2 Saving the New System ...................................................... 47
  4.4.3 Using SYSGEN ................................................................. 48

- 4.5 The Commodore 64 Keyboard and Screen with CP/M .......................... 49

5. CP/M OPERATION .................................... 51

- 5.1 How to Use This Chapter .................................................. 52
- 5.2 CP/M File Naming Conventions ........................................ 52
- 5.3 Input/Output Hardware Conventions .................................. 55
  5.3.1 Loading Programs from Disk:
    Single Drive ................................................................. 56
  5.3.2 Loading Programs from Disk:
    Dual Drive ................................................................. 57
- 5.4 CP/M Command Structure .................................................. 57
- 5.5 CP/M Commands ............................................................ 61
  5.5.1 pgm-name (Load and Run a CP/M Program) ................................ 61
  5.5.2 x: (Change the Currently Logged Disk) .................................. 63
  5.5.3 ASM ................................................................. 64
  5.5.4 DDT ................................................................. 66
  5.5.5 DIR ................................................................. 71
  5.5.6 DUMP ................................................................. 73
  5.5.7 ED ................................................................. 73
  5.5.8 ERA ................................................................. 82
  5.5.9 LOAD ................................................................. 83
  5.5.10 MOVCPM .............................................................. 83
  5.5.11 PIP ................................................................. 85
  5.5.12 REN ................................................................. 91
  5.5.13 SAVE ................................................................. 92
  5.5.14 STAT ................................................................. 93
  5.5.15 SUBMIT ............................................................... 97
  5.5.16 SYSGEN .............................................................. 100
  5.5.17 TYPE ............................................................... 102
  5.5.18 USER ............................................................... 103
  5.5.19 XSUB ............................................................... 104
6. CP/M ON THE COMMODORE 64 .................. 107
   • 6.1 The Structure of CP/M ................. 108
      6.1.1 How CP/M Works on Your
            Commodore 64 ..................... 109
      6.1.2 6510 Memory Use .................... 111
      6.1.3 Addresses under CP/M .............. 113
      6.1.4 Z80 Memory Use ..................... 114
   • 6.2 The BOOT Programs .................... 116
   • 6.3 The BIOS Programs .................... 117
   • 6.4 CP/M Disk Organization ............... 120
   • 6.5 The CP/M BDOS ....................... 121
      6.5.1 Sample BDOS Function Call .......... 123
   • 6.6 Calling a Z80 Program from the 6510 .. 134
      6.6.1 Some Examples ..................... 135
   • 6.7 Calling a 6510 Program from the Z80 .. 136
      6.7.1 Switching on the 6510 .............. 137
   • 6.8 Program Execution under CP/M .......... 139

7. APPENDICES ............................... 141
   • A. Commodore 64 Memory Map .............. 142
   • B. Bibliography .......................... 144
   • C. CP/M Command List .................... 148
   • D. ASCII, CHR$, and Hexadecimal
      Character Codes ........................ 151
   • E. BIOS and BOOT Listings (both 6510
      and Z80) .............................. 155

8. HARDWARE SCHEMATICS ................... 239
   • Z80 Schematic
   • Commodore 64 Schematic
CHAPTER 1

INTRODUCTION TO CP/M ON YOUR COMMODORE 64

- Overview of CP/M on Your Commodore 64
- How To Use This Manual
- Digital Research License Information
- Registration Information
- Warranty and Service Information
- Get More out of Your Commodore Computer
Your purchase of the Commodore Z80 add-on microprocessor cartridge puts you in the elite group of owners of a dual processor home microcomputer. No one but Commodore—the originator of the home microcomputer—could design and manufacture an inexpensive home or personal computer that accommodates the two most common microprocessors in the microcomputer industry:

- the Commodore MOS 6510 (6502 type) microprocessor
- the Z80A microprocessor

The 6510 microprocessor is the main processor on your Commodore 64. The 6510 is a specially designed variation of the widely distributed 6502 microprocessor found in many popular home and office computers. The 6510 runs the same instruction set as the 6502 but includes some special features that make it work more efficiently in your Commodore 64.

It is the 6510 main processor that is active when your Commodore 64 is running in native mode. In native mode, your Commodore 64 is controlled by its Commodore 64 Kernal operating system, Screen Editor, and the BASIC V2.2 interpreter. Native mode gives you access to a vast library of Commodore 64 applications packages from Commodore or from one of the many independent Commodore 64 software developers around the world.

When you add your Z80 cartridge to the system and start Digital Research's CP/M® operating system, you open the door to more than 15,000 CP/M-based application programs. CP/M is the most popular 8-bit operating system and is used for business applications throughout the world.

If you have a special application need, it's very likely that a CP/M package exists to meet it. CP/M applications are available in such areas as:

- financial reporting
- financial analysis
- investment planning
- word processing
- law
- real estate
farm management
restaurant management
data base
exotic language compilers (PL/I, PASCAL, C)
and many, many more

1.1 OVERVIEW OF CP/M ON YOUR COMMODORE 64

CP/M on your Commodore 64 can run in a maximum of 48K (1K = 1024 characters) of memory. The rest of memory is occupied by the Commodore 64 Kernal routines that provide input/output support for CP/M.

While you are running CP/M under the Z80 processor, the 6510 main processor acts as an input/output processor. When the 6510 is active, your Commodore 64 is executing in native mode. When it's running in native mode, your Commodore 64 "knows" how to handle its keyboard, screen, and peripherals (disks and printer). Rather than duplicate this facility to run under the Z80 processor, CP/M simply calls on the 6510 main processor to perform these tasks.

In addition to CP/M, you get a set of custom utilities that make it easy for you to run CP/M on your Commodore 64. You get:

- The COPY utility that formats diskettes in the CP/M format; easily produces backups of CP/M diskettes, even on single-drive systems; and copies the important CP/M system tracks.
- The CONFIG utility that makes it easy for you to inform CP/M of changes to your system peripherals, load the Commodore 64 function keys for use under CP/M, and re-define keyboard characters to yield any code you want.
- The MOVCPM utility that allows you to create a different sized version of CP/M without the need to learn Z80 Assembler language. MOVCPM relocates all of CP/M, including the BOOT and BIOS programs.
You can load anything you like into the eight Commodore 64 Function Keys. When CP/M is started, the eight function keys are loaded with the following CP/M commands (\texttt{<CR>} stands for \texttt{RETURN}):

\begin{itemize}
  \item F1 \texttt{Z DIRXCRZ}
  \item F2 \texttt{Z DIR B:XCRZ}
  \item F3 \texttt{Z STAT *.XCRZ}
  \item F4 \texttt{Z STAT B:*.XCRZ}
  \item F5 \texttt{Z COPYXCRZ}
  \item F6 \texttt{Z CONFIGXCRZ}
  \item F7 \texttt{Z DDTXCRZ}
  \item F8 \texttt{Z DDT}
\end{itemize}

CP/M on your Commodore 64 supports upper and lower case characters. You can toggle between upper case only and upper/lower case using the Commodore (\texttt{C}) key. For special applications, you can redefine the codes returned to your CP/M programs from the keyboard or sent to the screen from your programs.

\section*{1.2 HOW TO USE THIS MANUAL}

The very first thing to do is to read the \textit{Digital Research License Agreement} in Section 1.3. Next, \textit{fill in and mail} the Digital Research CP/M Registration Card at the end of this manual as soon as possible.

With those tasks accomplished, it's time to start running CP/M on your Commodore 64. \textbf{Chapter 2 tells you how to use your 820 cartridge.} Read this chapter \textit{before} you try to plug it in.

The distribution version of Commodore 64 CP/M assumes that you have a \textit{VIC 1515/1525 printer and a single VIC 1541 disk drive}. If your Commodore 64 is equipped with some other combination, consult \textbf{Chapter 3 for information on using your peripherals.}

\textbf{Chapter 4} is where things really get started. Read this chapter to learn \textbf{how to bring up CP/M on your system.} This chapter also tells you about the \textit{Commodore 64 specific CP/M utilities} that you'll need and talks about using the \textit{Commodore 64 keyboard} with CP/M.
The distribution version of CP/M (the one that you get on
the distribution diskette) is for a 44K CP/M system. You
should use this version if you have the IEEE interface car-
tridge. If you don't, look in Chapter 4 to learn how to con-
struct a 48K version that can take advantage of the addi-
tional 4K of RAM available on your system.

Chapter 5 is a reference section which includes de-
scriptions of all of the CP/M commands and utility pro-
grams that you need to function in the CP/M environment.
Chapter 5 shows you how to execute programs under CP/M
and talks about CP/M files and file naming conventions.

Chapter 6 is for those of you who want to get involved in
the technical workings of CP/M on your Commodore 64.
You DO NOT have to know any of this material to use CP/M.
If interested, you can look into the first few sections of
Chapter 6 to get an idea of how CP/M is implemented on
the Commodore 64 and how CP/M itself is structured.

The balance of Chapter 6 is for the technically sophisti-
cated user. You can learn about the BOOT and BIOS pro-
grams written to support CP/M on the Commodore 64 and
you can learn how to cross-call routines between the two
processors. To understand these sections fully, you should
have a strong working knowledge of both 6510 (6502) and
Z80 Assembler language.

Chapter 7 provides you with the engineering details of
your Z80 cartridge and your Commodore 64. If you
understand computer hardware, you can look here to see
how they did it.

This manual is intended to get you started in CP/M. If
you want to explore the depths of the CP/M operating sys-
tem, look in your local bookstore for one (or more) of the
many CP/M books published in the last few years. We’ve listed some of them in the Bibliography, Appendix B. Skim the books to see which one you like best.

Likewise, this manual does not provide a tutorial in the use of the Z80 microprocessor. If you’re interested in programming the Z80 in Assembler, you’ll need detailed references. The Bibliography contains a list of some of the Z80 books you can find in your bookstore.

1.3 DIGITAL RESEARCH LICENSE INFORMATION

IMPORTANT: Commodore’s license with Digital Research requires that each purchaser of the Commodore 64 CP/M system register with Commodore so that accurate records can be maintained of all CP/M users.

Because Digital Research requires this information, we have provided a post card for you to fill out and send in. The serial number of your CP/M system disk is stamped on the labels of the disks you receive with your Z80 cartridge and CP/M information. Please fill out the card and send it to us.

READ THE LICENSE AGREEMENT CAREFULLY.

1.3.1 Digital Research License Agreement

DIGITAL RESEARCH
Box 579, Pacific Grove, California 93950
SOFTWARE LICENSE AGREEMENT

IMPORTANT:
All Digital Research programs are sold only on the condition that the purchaser agrees to the following license. READ THIS LICENSE CAREFULLY. If you do not agree to the terms contained in this license, return the packaged diskette UNOPENED to your dealer and your purchase price will be refunded. If you agree to the terms contained in this license, fill out the REGISTRATION information and RETURN by mail to Commodore.

DIGITAL RESEARCH agrees to grant and the Customer agrees to accept, on the following terms and conditions, nontransferable and
nonexclusive licenses to use the software program(s) (Licensed Programs) herein delivered with this agreement.

TERM:

This agreement is effective from the date of receipt of the above referenced program(s) and shall remain in force until terminated by the Customer upon one month's prior written notice, or by Digital Research as provided below.

Any license under this Agreement may be discontinued by the Customer at any time upon one month's prior written notice. Digital Research may discontinue any license or terminate this Agreement if the Customer fails to comply with any of the terms and conditions of this Agreement.

LICENSE:

Each program license granted under this Agreement authorizes the Customer to use the Licensed Program(s) in any machine-readable form on any single computer system (referred to as System). A separate license is required for each System on which the Licensed Program(s) will be used.

This Agreement and any of the licenses, programs, or materials to which it applies may not be assigned, sublicensed, or otherwise transferred by the Customer without prior written consent from Digital Research. No right to print or copy, in whole or in part, the Licensed Program(s) is granted except as hereinafter expressly provided.

PERMISSION TO COPY OR MODIFY LICENSED PROGRAMS:

The Customer shall not copy, in whole or in part, any Licensed Programs which are provided by Digital Research in printed form under this Agreement. Additional copies of printed materials may be acquired from Digital Research.

Any Licensed Program which is provided by Digital Research in machine-readable form may be copied, in whole or in part, in printed or machine-readable form in sufficient number for use by the Customer with the designated System, to understand the contents of such machine-readable material, to modify the Licensed Program as provided below, for backup purposes, or for archive purposes, provided, however, that no more than five (5) printed copies will be in existence under any license at any one time without prior written consent from Digital Research. The Customer agrees to maintain appropriate records of the number and location of all such copies of Licensed Programs. The original, and any copies of the Licensed Programs, in whole or in part, which are made by the Customer shall be the property of Digital Research. This does not imply, of course, that
Digital Research owns the media on which the Licensed Programs are recorded. The Customer may modify any machine-readable form of a Licensed Program for his or her own use and merge it into other program material to form an updated work, provided that, upon discontinuance of the license for such Licensed Program, the Licensed Program supplied by Digital Research will be completely removed from the updated work. Any portion of the Licensed Program included in an updated work shall be used only if on the designated System and shall remain subject to all other terms of this Agreement.

The Customer agrees to reproduce and include the copyright notice of Digital Research on all copies, in whole or in part, in any form, including partial copies of modifications, of Licensed Programs made hereunder.

PROTECTION AND SECURITY:

The Customer agrees not to provide or otherwise make available any Licensed Program including but not limited to program listings, object code, and source code, in any form, to any person other than the Customer or Digital Research employees, without prior written consent from Digital Research, except with the Customer's permission for purposes specifically related to the Customer's use of the Licensed Program.

DISCONTINUANCE:

Within one month after the discontinuance of any license under this Agreement, the Customer will furnish to Digital Research a certificate certifying that through his or her best effort, and to the best of his or her knowledge, the original and all copies, in whole or in part, in any form, including partial copies in modifications, of the Licensed Program(s) received from Digital Research or made in connection with such license have been destroyed, except that, upon prior written authorization from Digital Research, the Customer may retain a copy for archive purposes.

DISCLAIMER OF WARRANTY:

Digital Research makes no warranties with respect to the Licensed Programs. The sole obligation of Digital Research shall be to make available all published modifications or updates made by Digital Research to Licensed Programs which are published within one (1) year from date of purchase, provided the Customer has returned the Registration Card delivered with the Licensed Program.

LIMITATION OF LIABILITY:

THE FOREGOING WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE. IN NO EVENT WILL DIGITAL RESEARCH BE
LIABLE FOR CONSEQUENTIAL DAMAGES EVEN IF DIGITAL RESEARCH
HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

GENERAL:
If any of the provisions, or portions thereof, of the Agreement are
invalid under any applicable statute or rule of law, they are to that
extent to be deemed omitted.

1.4 REGISTRATION
INFORMATION

Please fill out the CP/M Registration Card that is enclosed
with your Z80 cartridge and CP/M system. Mail the com-
pleted card to:

DIGITAL RESEARCH
P.O. Box 579
Pacific Grove, CA 93950

We need the information on the card to provide informa-
tion on system updates and to inform you of related new
products. The serial number of your CP/M system is the
number stamped on the label of the CP/M disks.

1.5 WARRANTY

If your unit is defective when you buy it, return it im-
mediately to the original place of purchase. Your dealer will
be able to give you the fastest service if you have problems.
You can also send your unit directly to Commodore for re-
placement. The warranty card enclosed in your unit's pack-
age lists addresses for service. Be sure to enclose your re-
cceipt and a note explaining the problem. See your warranty
card for more information.
1.6 GET MORE OUT OF YOUR COMMODORE COMPUTER

Commodore wants you to know that our support for users only starts 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 with valuable input for 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 around the world. They are an excellent source of information for every Commodore computer owner, from the beginner to the most advanced. The magazines and network, which are more fully described below, have the most up-to-date information about how to get involved with the User's Club in your area.

Finally, your local Commodore dealer is a useful source of Commodore support and information.

1.6.1 POWER/PLAY: The Home Computer Magazine

For entertainment, learning at home and practical home applications, POWER/PLAY is the prime source of information for Commodore home users. From it you will learn where your nearest user clubs are and what they're doing. You'll also learn about 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 sells for $10.00 a year.

1.6.2 COMMODORE: The Microcomputer Magazine

 Widely read by educators, businessmen, and students as well as by home computerists, COMMODORE Magazine is our main vehicle for sharing information on the more technical use of Commodore systems. Regular departments cover business, science and education, programming tips, and "excerpts from a technical notebook." There are many other features of interest to anyone who uses or is thinking
about purchasing Commodore equipment for business, scientific, or educational applications. **COMMODORE** is the ideal complement to **POWER/PLAY**. It is published bi-monthly, and subscriptions are $15.00 a year.

**1.6.3 COMMODORE INFORMATION NETWORK: The Paperless User Magazine**

This is the magazine of the future. To supplement and enhance your subscriptions to **POWER/PLAY** and **COMMODORE** magazines, the **COMMODORE INFORMATION NETWORK**—our "paperless magazine"—is available now over the telephone using your Commodore computer and modem.

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** or **COMMODORE** by downloading direct from the Information Network (a new user service planned for early 1983). The best part is that most of the answers are there even before you ask the questions.

To call our electronic magazine, you need only a modem and a subscription to CompuServe™, one of the nation's largest telecommunications networks. (To make it easy for you, Commodore includes a FREE one year subscription to CompuServe™ in each VICMODEM package.)

Just dial your local number for the CompuServe™ data bank and connect your phone to the modem. When the CompuServe™ video text appears on your screen, type G CBM on your computer keyboard. When the **COMMODORE INFORMATION NETWORK** table of contents, or "menu," appears on your screen, choose from one of our sixteen departments, make yourself comfortable, and enjoy the paperless magazine that other magazines are writing about.

For more information, visit your Commodore dealer or contact CompuServe™ customer service at 800-848-8990 (in Ohio, 614-457-8600).
<table>
<thead>
<tr>
<th>Commodore Information Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Menu Description</td>
</tr>
<tr>
<td>Direct Access Codes</td>
</tr>
<tr>
<td>Special Commands</td>
</tr>
<tr>
<td>User Questions</td>
</tr>
<tr>
<td>Public Bulletin Board</td>
</tr>
<tr>
<td>Magazines and Newsletters</td>
</tr>
<tr>
<td>Products Announced</td>
</tr>
<tr>
<td>Commodore News Direct</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Commodore Dealers</td>
</tr>
<tr>
<td>Educational Resources</td>
</tr>
<tr>
<td>User Groups</td>
</tr>
<tr>
<td>Descriptions</td>
</tr>
<tr>
<td>Questions and Answers</td>
</tr>
<tr>
<td>Software Tips</td>
</tr>
<tr>
<td>Technical Tips</td>
</tr>
<tr>
<td>Directory Descriptions</td>
</tr>
</tbody>
</table>
CHAPTER 2

SETTING UP YOUR COMMODORE 64

- Unpacking and Connecting the Z80 Cartridge
- Installing the Z80 Cartridge
- Connecting Disk Drives
It's very easy to set up your Commodore 64 to run CP/M. You turn off your computer, plug in the Z80 cartridge, turn on your disks and computer and get started. Follow the directions in this chapter carefully.

REMEMBER: YOU MUST TURN OFF YOUR COMMODORE 64 BEFORE YOU INSERT THE Z80 CARTRIDGE. IF YOU INSERT THE CARTRIDGE WITH THE POWER ON, YOU WILL DESTROY THE CARTRIDGE!!

2.1 UNPACKING AND CONNECTING THE Z80 CARTRIDGE

Before using CP/M on your Commodore 64, you must correctly connect your Commodore 64 to your TV and peripherals. For instructions on connecting your Commodore 64 to your TV, disk, and printer, read the manual that comes with your computer.

When you purchase CP/M for your Commodore 64, you get these items:

1. Z80 cartridge.
2. CP/M system disk.
3. Other disk.

Before you can connect your Z80 cartridge, you must know where to connect it. Figure 2.1 shows a diagram of the side and back panel connections for your computer.

Your Commodore 64 has these side panel connections:

1. **Power socket.** The free end of the cable from the power supply is attached here to supply power to your Commodore 64.
2. **Power switch.** This turns the power to your Commodore 64 on and off.
3. **Game ports.** These accept a joystick, one or more game controllers, or lightpen equipment. The lightpen plugs into port 1 only.
Your Commodore 64 has these back panel connections:

4. **Cartridge slot.** The rectangular slot to the left accepts program or game cartridges. *This is the connection for your Z80 cartridge.*

5. **Channel selector.** Use this switch to select the TV channel that will display your computer's picture.

6. **TV connector.** This connector supplies the picture and sound to your TV.

7. **Audio & video output.** This connector supplies direct audio (which you connect to your stereo system) and "composite" video (which you connect to a monitor).

8. **Serial port.** This is the connection for your VIC peripherals (1541 drives and 1515/1525 printer). You must connect your VIC disk drive to this port and your VIC printer to your VIC disk drive.

9. **Cassette interface.** This is the connection for your DATASSETTE™ recorder.
10. **User port.** This is a port for various interface cartridges such as the VICMODEM or RS-232 communications cartridge.

### 2.2 INSTALLING THE Z80 CARTRIDGE

Now that you know where your Commodore 64 connections are, you’re ready to install your Z80 cartridge. You connect the Z80 cartridge directly to your Commodore 64 if you are using the VIC 1541 disk drive. You connect the Z80 cartridge to an IEEE interface cartridge if you’re using the CBM 4040 disk drives or the CBM 4022 printer.

#### 2.2.1 Using the Z80 Cartridge with VIC Peripherals

If you’re using VIC *peripherals* like the VIC 1541 disk drives and the VIC 1525 printer, follow these easy steps:

1. **TURN OFF THE POWER TO YOUR COMPUTER!**
2. Install the Z80 cartridge in the cartridge slot marked 4 in the diagram in Figure 2.1.
3. Turn on your computer and you’re ready to start using CP/M on your Commodore 64.

---

**REMEMBER! IF YOU INSERT THE Z80 CARTRIDGE WITH THE POWER TO THE COMPUTER TURNED ON, YOU WILL DAMAGE THE CARTRIDGE!**

---

#### 2.2.2 Using the Z80 Cartridge with CBM Series Peripherals

If you’re using CBM *series peripherals* like a CBM 4040 disk drive or a CBM 4022 printer, you follow a slightly different procedure for connecting the Z80 cartridge. Remember, you need to use the IEEE interface cartridge if you’re using a CBM peripheral.

The IEEE interface cartridge has a connector for other
cartridges (like the Z80 cartridge) and also has a connector for the CBM peripherals. Figure 2.2 shows a diagram of the IEEE cartridge connections.

Follow these easy steps to connect your Z80 cartridge to your Commodore 64 when you're using the IEEE Interface cartridge and CBM series peripherals:

1. TURN OFF THE POWER TO YOUR COMPUTER!
2. Install the IEEE interface cartridge in the cartridge slot marked 4 in the diagram in Figure 2.1.
3. Install the Z80 cartridge into the IEEE cartridge slot as shown in the diagram in Figure 2.2.
4. Connect your CBM peripherals to the connector on the IEEE cartridge.
5. Turn on your computer and you're ready to start using CP/M on your Commodore 64.

REMEMBER: IF YOU INSERT THE Z80 CARTRIDGE WITH THE POWER TO THE COMPUTER TURNED ON, YOU WILL DAMAGE THE CARTRIDGE!

Figure 2.2 IEEE Interface Cartridge Diagram
2.3 CONNECTING DISK DRIVES

The method you use to connect your disk drives depends on the types of drives you use. You can use either a VIC series disk drive (like the 1541) or a CBM series single or dual disk drive (like the 4040) with your Commodore 64.

You don’t have to write any special code to use your disk drives under CP/M. The system accesses your disk drives as Drive A and Drive B, regardless of which type of drive you’re actually using.

If you use a single disk drive, CP/M uses Drive A and uses a virtual drive for Drive B (CP/M will prompt you to change the physical disk in the drive when you ask for Drive B). If you’re using a CBM series dual drive, CP/M uses Drive A and Drive B.

2.3.1 Connecting VIC 1541 Disk Drives

You can use one VIC 1541 disk drive. Like all Commodore peripherals, the VIC 1541 disk drive can be “daisy chained.” That is, you can connect your VIC disk drive to a VIC printer.

Connect the single VIC disk drive to the serial port (marked B in the diagram in Figure 2.1). For full details on connecting a VIC 1541 disk drive to your Commodore 64, see the manual that comes with the drives.

If you’re also using a VIC 1525 printer, connect the printer to the connector in the back of your VIC 1541 disk drive.

2.3.2 Connecting CBM Series Disk Drives

When using CBM series peripherals (like the CBM 4040 disk drive or the CBM 4022 printer), you need to connect your peripherals to the IEEE interface cartridge. Figure 2.2 shows a diagram of the IEEE interface cartridge.

You can daisy chain your CBM printer to your CBM disk drive. For more details on connecting your CBM disk drive, see the manual that comes with your IEEE interface cartridge.
CHAPTER 3

USING YOUR COMMODORE 64 PERIPHERALS FROM CP/M

- Printer Interface
- The Commodore 64 Serial Interface
- The IEEE Interface Cartridge
- Daisy Chaining Peripherals
- The Commodore 64 User Port
CP/M, as implemented on your Commodore 64, can access any standard Commodore 64 peripheral (except the RS-232 port and the modem) using standard CP/M device access protocols. This involves calls to the appropriate CP/M BDOS functions. (You can also call the BIOS directly, although this is not recommended.)

The actual peripheral interface drivers reside in the CP/M BIOS. This special BIOS, unique to your Commodore 64, is in two parts. One part executes under the Z80 add-on processor and the other under the 6510 main processor.

Peripheral device access is set up through a series of parameters by the Z80 part of the BIOS. The actual device access is carried out by the 6510 part of the BIOS operating in Commodore 64 native mode.

You must configure CP/M—using the CONFIG utility—so that it knows what kind of printer you have and how many disk drives you have. If you change the type of printer or the number of disk drives on the system, you must use the CONFIG utility to inform CP/M of the change.

### 3.1 PRINTER INTERFACE

CP/M must know what type of printer you have. Generally you will have a VIC 1515, VIC 1525, or CBM 4022 printer. For purposes of the CONFIG utility, the 1515 and 1525 are the same, and the 4022 represents any CBM series printer.

The VIC 1515 and 1525 printers use the standard Commodore 64 serial bus. The 4022 printer (or any other CBM series printer) requires the optional IEEE interface cartridge.

Once you have properly attached the printer to your Commodore 64 and have run the CONFIG utility under CP/M, you can print using programs that run under CP/M or using standard CP/M BDOS calls from Z80 Assembler language programs.
3.2 THE COMMODORE 64 SERIAL INTERFACE

Your Commodore 64 comes standard with a bit serial interface through which you communicate with the Commodore 64 disk drives and printers. Access to the Commodore 64 serial interface is handled automatically under CP/M.

If you attach a nonstandard device to the Commodore 64 bit serial interface, you must prepare code to handle that device. The actual device handling code must execute in Commodore 64 native mode (under the 6510 main processor). Of course, you also need device handling code to run under the Z80, controlling execution of the native mode device-handling routine.

3.3 THE IEEE INTERFACE CARTRIDGE

If you want to connect your Commodore 64 to IEEE bus compatible devices, you can do that using the IEEE interface cartridge.

The IEEE interface cartridge plugs into the cartridge slot on the rear of your Commodore 64. The interface cartridge includes a slot for plugging in your Z80 cartridge. (See the instructions that come with your IEEE interface cartridge.)

The interface cartridge allows you to attach Commodore's own IEEE-compatible peripherals. These more capable, more expensive peripherals are usually available only for Commodore's business computers. The IEEE interface cartridge also provides a link to a multitude of IEEE–bus-based products. For example, many industrial and scientific instruments and devices are controlled using the IEEE bus protocols. With the IEEE interface cartridge, your Commodore 64 can control and collect data from these devices.
NOTE: If you do acquire the IEEE interface cartridge, you will have 44K—NOT 48K—available for CP/M. Be sure to generate a 44K version of CP/M before you install the IEEE interface cartridge.

If you are also installing IEEE bus peripherals, especially disk drives, remember to run the CONFIG utility on your 44K CP/M, informing it of your new peripherals.

### 3.4 DAISY CHAINING PERIPHERALS

The advanced architecture of the standard Commodore 64 serial bus and of the Commodore IEEE serial bus permits peripherals to be linked to one another in a "daisy chain."

Daisy chaining of peripherals means that you need not buy another interface card or connector every time you add a peripheral to your Commodore 64. The peripherals simply connect to each other to be accessed through a single port on your Commodore 64.

You can daisy chain VIC peripherals on the standard Commodore 64 serial bus or CBM series peripherals through the IEEE interface cartridge, as shown in Figure 3.1.

---

**VIC PERIPHERALS SYSTEM**

(Uses Standard Commodore 64 Serial Port)

Computer → VIC Disk Drive → VIC Printer

**CBM PERIPHERALS SYSTEM**

(Requires IEEE Interface Cartridge)

Computer → CBM Dual Disk Drive → CBM Printer

or

Computer → CBM Printer → CBM Dual Disk Drive

---

Figure 3.1 Daisy Chaining Peripherals.
3.5 THE COMMODORE 64 USER PORT

Your Commodore 64 user port can accommodate some useful optional devices. Most interesting from CP/M are the VICMODEM and the RS-232 communications cartridge.

If you acquire one of these cartridges and you want to access it from CP/M, you must write the processing code for execution in native mode under the 6510 main processor. This is necessary because these cartridges generate non-maskable interrupts which must be handled by the 6510 processor.

You can gain access to special code for handling these cartridges through BIOS65 function codes 7, 8, or 9. (See the discussion of the CP/M BIOS in Chapter 6 for details on using these function codes.)

In designing this code, you should consider receiving a certain number of characters—say 128 or 256—into a shared buffer. When you have received these characters, inform the device you are communicating with that you are not ready to receive data. You can then safely switch control from the 6510 main processor to the Z80, which can do whatever is required with those characters.

For detailed information on programming for the RS-232 port, see the Commodore 64 Programmer's Reference Manual.
CHAPTER 4

GETTING STARTED

- Bringing CP/M onto Your Commodore 64
- The COPY Utility
- The CONFIG Utility
- Generating a New CP/M System with SYSGEN
This chapter tells you how to start using CP/M on your Commodore 64. Read it carefully. It's very easy to bring CP/M onto your computer, but you should be sure that you understand the information in this chapter before you start CP/M or run any programs under it.

In this chapter you will learn:

- how to load and run your CP/M system
- how to format new disks and make backup copies of your system
- how to use the special Commodore 64 CP/M utilities
- how to generate a new version of CP/M
- how to use the special Commodore 64 keyboard under CP/M

The distribution 44K version of CP/M assumes that you are using the IEEE interface cartridge. If you don’t have the IEEE interface cartridge, you can generate a 48K version of CP/M by following the instructions in Section 4.4.

4.1 BRINGING CP/M ONTO YOUR COMMODORE 64

It is easy to bring CP/M onto your Commodore 64. Before you load CP/M, be sure that you’ve correctly installed your Z80 cartridge and your disk drive(s) and printer. If you haven’t done this, read Chapter 2 for installation instructions.

After installing your Z80 cartridge and peripherals, follow the instructions in Section 4.1.1 to load your CP/M system. Once you’ve loaded CP/M and made copies of the system disks for backup, you’re ready to try any of the commands in Chapter 5.

NOTE: Remember to make copies of your CP/M disks before you do any other processing. You need a backup copy of the disks that you purchased.
4.1.1 Starting CP/M

To bring CP/M onto your Commodore 64 system, you start the computer and load the CP/M system. Just follow these easy steps and make a backup copy of your system disks right after you get CP/M to start for the first time:

1. Turn on your equipment (peripherals and computer). Your Commodore 64 will print its usual "sign on" message:

   **** COMMODORE 64 BASIC V2 ****
   64K RAM SYSTEM 38911 BASIC BYTES FREE
   READY.

2. Put the disk marked Commodore CP/M®*V.64 into your disk drive. This disk contains your CP/M system.

3. Your Commodore 64 is in native mode. Type the following:

   LOAD "*",8 <CR>
   or
   LOAD "CPM", 8

4. Your Commodore 64 reads the disk and answers:

   SEARCHING FOR * (or CPM instead of *)
   LOADING
   READY.

5. The Commodore 64 segment of CP/M is now loaded into your computer. To load the Z80 segment and begin executing CP/M, type:

   RUN <CR>

6. Your Commodore 64 now reads the disk again to load the CP/M system into your Z80. While it is loading CP/M, your computer will print a row of 27 asterisks (*) across the top of the screen. When CP/M is loaded, your Commodore 64 will print:
7. Your CP/M system is now loaded and ready to run. Enter the following CP/M command to get a list of the files on your CP/M disk:

DIR <CR>

CAUTION! BEFORE PROCEEDING, MAKE A BACKUP COPY OF YOUR CP/M DISKS!

4.1.2 Making Copies of Your CP/M System Disk

Now that you've started CP/M, you must make backup copies of your system disks. It is bad practice to use the disks that you purchased as your standard operating disks. You could accidentally destroy the disk and then you would not be able to run your CP/M system.

So, make a backup copy and use the copy as your CP/M system disk. After you make the backup copy, store your original disk in a cool, dry place, away from magnetic fields.

To make your backup copy:

1. Use the COPY utility on your CP/M disk to format a new disk. The COPY utility is discussed in detail in Section 4.2.
2. Then use the COPY utility to copy your CP/M disk to the backup disk. The COPY utility prompts you along the way, depending on the number of drives you're using. Just follow its instructions.
3. Store your original disks in a safe place, somewhere cool, dry, and away from magnetic fields.
4.2 THE COPY UTILITY

The COPY utility is a special Commodore 64 CP/M utility that allows you to:

- FORMAT a diskette for use with CP/M.
- Make a BACKUP of a CP/M diskette.
- Copy the CP/M SYSTEM TRACKS from one diskette to another.

You should use this utility to make a backup copy of your CP/M system disks as soon as you get CP/M up and running. Each COPY utility function is described in a separate section below.

To load the COPY utility, enter:

COPY<CR>

CP/M loads the COPY.COM file and writes:

COMMODORE 64 COPY UTILITY 1.0
1. FORMAT DISK
2. BACKUP DISK
3. COPY SYSTEM TRACKS ONLY
4. EXIT
PLEASE CHOOSE FUNCTION (1-4)

You then choose which COPY utility function you want to use and answer the questions that COPY asks.

4.2.1 Formatting a Disk with the COPY Utility

You must format a diskette before you can write any information on it. You must format disks that you'll use under CP/M with the COPY utility.

You format disks when:

- You get new disks and you want to prepare them to be used with CP/M.
- You want to erase all of the information currently on a disk.
To use the COPY function to format disks, you enter 1 as follows:

...COPY utility messages...
PLEASE CHOOSE FUNCTION (1-4) 1
FORMAT DISK UTILITY
INITIALIZES DISK FOR CP/M
CAUTION! FORMAT ERASES ALL DATA
PLACE DISK TO BE FORMATTED IN
DRIVE 0 AND PRESS ENTER
OR
PRESS SPACEBAR TO RETURN TO MENU

Now, remove your system disk from the drive and place the new disk (the one that you want to format) into the drive.

CAUTION! REMEMBER THAT YOU MUST REMOVE YOUR SYSTEM DISK OR ELSE YOU WILL ERASE YOUR SYSTEM DISK!!

COPY now writes formatting information to your disk. Any information on the disk will be erased and all of the tracks are made available for data. No files remain on the disk after you run COPY's FORMAT. COPY writes these messages during the formatting:

FORMATTING DISK, PLEASE WAIT...
FORMAT COMPLETE
PRESS ANY KEY TO CONTINUE

You can now format another disk, copy information to your newly formatted disk, or exit back to CP/M, depending on your answer. If you want to format another disk, you need to insert the disk to be formatted into the drive. If you want to copy information, follow the instructions from COPY. If you're exiting back to CP/M, you should put your CP/M system disk into the drive.

NOTE: Remember that COPY erases all information from the disk when you use the COPY FORMAT option.
4.2.2 Creating a Disk Backup with the COPY Utility

You can also use the COPY utility to make backup copies of an entire diskette. While making a backup copy, COPY uses a master disk and a slave disk. The master disk is the disk that you want to make a copy of (the original disk); the slave disk is a formatted disk that will be written to (the copy).

If you are using a single-drive system, the COPY utility will prompt you to insert the master or slave disk into the drive. Be careful when making copies of a disk. Keep track of your master disk so that you don’t accidentally copy garbage over your information (and erase your master disk in the process).

To use COPY’s BACKUP function, enter a 2 in response to the “choose function” message and follow the instructions from COPY:

```
PLEASE CHOOSE FUNCTION (1-4) 2
DISK BACKUP UTILITY
THE ENTIRE MASTER DISK IS
COPIED TO THE SLAVE DISK
INSERT MASTER DISK IN DRIVE 0
PRESS RETURN (OR SPACEBAR FOR MENU)
```

Now insert the disk that you want to copy from into the disk drive. If you decide that you don’t really want to copy your disk, simply press the space bar and COPY returns to its original menu.

Once the master disk is ready, press the return key. COPY then reads a number of sectors from the disk into memory and writes:

```
INSERT SLAVE DISK IN DRIVE 0
PRESS RETURN
```

Put the disk you want to copy to into the drive and press the carriage return. Be careful to keep the master and slave disks in order.

COPY now writes the information from memory onto the
slave disk and then asks that the master disk be replaced in the drive. This alternating master/slave disk placement will continue until the entire master disk is copied onto the slave disk. At that time, COPY returns to its main menu.

4.2.3 Copying the System Tracks with the COPY Utility

You can copy the CP/M system tracks to another disk through the COPY system track copy function. This function copies only the system tracks, not any other information, from a master disk to a slave disk.

You need the CP/M system tracks on any disk from which you intend to “warm start” CP/M (start CP/M without having to reinsert the system disk). You may want to copy the system tracks to a disk containing a program that you will run often. That way, when you hit a \texttt{CTRL-C} to warm start CP/M, you don’t have to replace the disk with your system disk.

To copy the system tracks using COPY, enter 3 for your selection from COPY’s main menu. Then follow the instructions:

\begin{verbatim}
PLEASE CHOOSE FUNCTION (1-4)    3
SYSTEM TRACK COPY UTILITY
COPIES SYSTEM TRACKS FROM MASTER DISK
TO SLAVE DISK
INSERT MASTER DISK IN DRIVE 0
PRESS RETURN (OR SPACEBAR FOR MENU)
\end{verbatim}

The disk with the COPY utility contains the CP/M system tracks (otherwise, you wouldn’t have been able to start your system). Simply press the \texttt{RETURN} key or, if you really don’t want to make a copy, press the \texttt{SPACE} bar.

When you press the \texttt{RETURN} key, COPY reads the system tracks into memory and then writes:

\begin{verbatim}
INSERT SLAVE DISK IN DRIVE 0
PRESS RETURN
\end{verbatim}
Remove the master disk from the drive and insert the disk on which you want the system tracks copied into the drive. When you press the \textit{RETURN} key, COPY will write the CP/M system tracks (tracks 0 and 1) to the disk in the drive. After the system tracks are written, COPY returns to its main menu.

\section*{4.3 THE \texttt{CONFIG} UTILITY}

You use the Commodore CP/M \texttt{CONFIG} utility to \textit{change} the current I/O configuration for your CP/M system. Commodore provides the \texttt{CONFIG} utility so that you can add peripherals to your CP/M system quickly and easily.

CP/M needs to know what peripherals you're using. For example, if you're using only a single disk drive, CP/M will prompt you to change the diskette in the drive when you log to another disk. If you're using two drives, a properly configured CP/M will simply use the second physical drive.

\begin{quote}
\textbf{NOTE:} You CANNOT mix VIC (serial) peripherals and CBM (IEEE interface) peripherals on the same system.
\end{quote}

Each of the \texttt{CONFIG} changes is described in a separate section below. To use the \texttt{CONFIG} utility, you enter:

\begin{verbatim}
CONFIG <CR>
\end{verbatim}

CP/M then loads the file called \texttt{CONFIG.COM} and writes:

\begin{verbatim}
COMMODORE 64 I/O CONFIGURATION UTILITY
THE CURRENT I/O ASSIGNMENTS ARE:
  NUMBER OF DRIVES: 1
  PRINTER TYPE: 1515
  INITIAL CAPS MODE: ON
DO YOU WISH TO:
  1. CHANGE NUMBER OF DISK DRIVES
  2. CHANGE PRINTER TYPE
  3. CHANGE INITIAL CAPS MODE
\end{verbatim}
4. CHANGE FUNCTION KEY ASSIGNMENTS  
5. CHANGE KEY CODES  
6. SAVE CURRENT I/O SETUP ON DISK  
7. RETURN TO CP/M  

PLEASE ENTER SELECTION (1-7)  

You simply select the type of change that you want to make and answer the questions that CONFIG asks. CONFIG makes all the necessary changes to your CP/M system, for both the Commodore 64 native code and the Z80 code. Adding or changing peripherals to your Commodore 64 CP/M system is as easy as running CONFIG and answering the questions.

4.3.1 Using CONFIG to Change the Number of Disk Drives

The CP/M system that you receive assumes that you are using a single disk drive. You may actually have the CBM 4040 dual disk drives. CONFIG toggles back and forth between one and two disk drives.  

To change the number of drives, you run CONFIG like this:

CONFIG<CR>

when the CONFIG Messages are printed, choose selection 1.

CONFIG then processes your answer and changes the number of drives available to CP/M. If you originally had one disk drive, CONFIG prints:

COMMODORE 64 I/O CONFIGURATION UTILITY  
THE CURRENT I/O ASSIGNMENTS ARE:  
NUMBER OF DRIVES: 2  
PRINTER TYPE: 1515  
INITIAL CAPS MODE: ON  
DO YOU WISH TO:  

rest of CONFIG messages...

PLEASE ENTER SELECTION (1-7)
If you had two disk drives when you started CONFIG, you will see this for the number of drives:

**NUMBER OF DRIVES: 1**

### 4.3.2 Using CONFIG to Change the Printer Type

Your original CP/M system assumes that you will be using a VIC 1515 or (1525) printer. You may want to add a CBM 4022 (or other CBM) printer. CONFIG toggles back and forth between 1515 and 4022 printer types.

To change the printer type, you run CONFIG like this:

```
CONFIG<CR>
```

when the CONFIG Messages are printed, choose selection 2.

CONFIG then processes your answer and changes the printer type. If you originally had a VIC 1515 printer, CONFIG prints:

```
COMMODORE 64 I/O CONFIGURATION UTILITY
THE CURRENT I/O ASSIGNMENTS ARE:
  NUMBER OF DRIVES: 1
  PRINTER TYPE: 4022
  INITIAL CAPS MODE: ON
DO YOU WISH TO:
rest of CONFIG messages...

PLEASE ENTER SELECTION (1-7)
```

If you had a CBM 4022 printer when you started CONFIG, you get this for the printer type:

```
PRINTER TYPE: 1515
```

### 4.3.3 Using CONFIG to Change the Initial Caps Mode

Your original CP/M system assumes that you will be using the all caps mode (all upper case letters when you press the
keys). CONFIG toggles back and forth between initial caps ON and OFF.

With initial caps ON, you get only upper case letters. With initial caps OFF, you get upper and lower case letters. Remember that you can also toggle between caps ON and OFF at any time by pressing the $C$ key.

To change the initial caps mode, you run CONFIG like this:

```
CONFIG<CR>
```

when the CONFIG Messages are printed, choose selection 3.

CONFIG then processes your answer and changes the printer type. If you originally had initial caps ON, CONFIG prints:

```
COMMODORE 64 I/O CONFIGURATION UTILITY
THE CURRENT I/O ASSIGNMENTS ARE:
    NUMBER OF DRIVES: 1
    PRINTER TYPE: 1515
    INITIAL CAPS MODE: OFF
DO YOU WISH TO:
```

rest of CONFIG messages...

```
PLEASE ENTER SELECTION (1-7)
```

If you had initial caps OFF when you started CONFIG, you will see this:

```
INITIAL CAPS MODE: OFF
```

### 4.3.4 Using CONFIG to Change the Function Key Assignments

Your CP/M system loads initial values into the eight Commodore 64 function keys. You can change any of these function key values through CONFIG.

If you save the new I/O configuration to disk, the new values will be loaded into the function keys when you next start CP/M. If you don’t save the new configuration to disk,
the function keys are loaded with the new values but are reset to the original values when you next start CP/M.

To change the function key values, you run CONFIG like this:

```
CONFIG<CR>
```

when the CONFIG Messages are printed, choose selection 4.

CONFIG then prints:

```
F1: "DIR"<CR>
F2: "DIR B:"<CR>
F3: "STAT *.**"<CR>
F4: "STAT B:*.*"<CR>
F5: "COPY"<CR>
F6: "CONFIG"<CR>
F7: "DDT"<CR>
F8: "DDT"
ENTER FUNCTION KEY NUMBER (1-8)
    TO CHANGE PRESET VALUES.
ENTER 9 TO LEAVE FUNCTION
    KEY UTILITY.
```

To change function key 8 to "PIP<CR>", use CONFIG like this:

```
ENTER FUNCTION KEY NUMBER (1-8)  8

TYPE IN TEXT USING "RETURN"
OR "CTRL-Z" AS TERMINATOR

F8: "PIP<RETURN KEY>"
ENTER FUNCTION KEY NUMBER (1-8)  9
```

This changes the value in function key 8 to PIP<CR> while you are using CP/M.

If you end your new key entry with a `CTRL-Z`, instead of a `RETURN` the function key is loaded without a terminating carriage return.
If you want to save this value as the initial value for function key 8 for the next time you start CP/M, you must also choose CONFIG selection 6 to save the new I/O configuration to disk. Otherwise, the next time you boot CP/M, your function keys will contain the same initial values as they did this time; any changes you made through CONFIG will be lost.

### 4.3.5 Using CONFIG to Change the Key Codes

Your CP/M system loads a table containing the hexadecimal values for each of the Commodore 64 keyboard keys. You can change any of these function key values through CONFIG. Appendix D contains a table of ASCII characters, hexadecimal values, and the Commodore 64 keyboard characters.

---

**NOTE:** Be careful if you change the alphabetic characters. You may not be able to recover if you change characters that you need to run CP/M programs or commands. If you SAVE the character changes on disk (through CONFIG selection 6), you may have trouble recovering at all.

---

To change the keyboard key values, you run CONFIG like this:

```
CONFIG<CR>
```

when the CONFIG Messages are printed, choose selection 5.

CONFIG then prints:

```
PRESS KEY TO EXAMINE KEY CODE
TO CHANGE KEY CODE, ENTER DATA IN HEXADECIMAL AFTER "CHANGE TO"
TO EXIT KEY CODE MODE, TYPE "RETURN"
TWICE AFTER "PRESS KEY"
TO KEEP CURRENT KEY CODE, TYPE
"RETURN" AFTER "CHANGE TO"
```
You just changed the capital Q (hexadecimal value 51) to a lower case q (hexadecimal value 71). You won't be able to enter a capital Q unless you use CONFIG to change it back again. If you don't want to make any more changes, just press the RETURN key twice to return to the CONFIG main menu.

4.3.6 Using CONFIG to Save the New I/O Setup

Once you've made changes to your I/O assignments through CONFIG, you may or may not want to save the new assignments. You will probably want to save the new information if you've changed the disk drive or printer data. You may not want to save the I/O information if you've changed the function key assignments for a special run and don't want the new values to be used the next time you start CP/M.

To save your new I/O assignments to disk, select 6 from the CONFIG menu. CONFIG then writes information to your CP/M system data and the next time you start CP/M, the new information will be used.

Remember, you can make changes that only affect the current CP/M version (the one in memory when you make the changes) if you want some special-purpose alterations. If you don't select CONFIG choice 6, the alterations will not be in effect the next time you load CP/M.

4.4. Generating a New CP/M System with SYSGEN

You can generate CP/M on your Commodore 64 to run in any memory size from 20K to 48K. If you are using the standard Commodore 64 serial bus to attach your peripherals—disk and printer—you should use a 48K version of CP/M. If you acquire the IEEE interface cartridge, you must
use a 44K version of CP/M. You may also want to generate a smaller version of CP/M if you need space to load a 6510 routine that you are invoking from a CP/M program.

NOTE: If you don't intend to save the new CP/M on an existing CP/M disk, the first step in generating a new version of CP/M is to format a disk. Disk formatting is discussed in detail in Chapter 4 under the COPY utility.

Once you have the disk formatted for CP/M, you must use the COPY utility to copy the System tracks from one of your existing CP/M disks to the new disk. This operation places the 6510 loader into its proper place.

Once you have properly initialized your disk, you use a series of CP/M utility programs to generate the new version of CP/M and save it on your disk. These utilities are:

- MOVCPM
- SAVE
- SYSGEN

These utilities have a number of options on their use. In the following discussions, we consider only the most frequently used options. A more detailed exploration of all the utility options is found in Chapter 5.

In general, you will be generating either a 44K or a 48K version of CP/M on your Commodore 64. We'll use generating a 48K version as an example. Other versions are generated in exactly the same way but with a different memory size specified.

4.4.1 Relocating CP/M

MOVCPM is a system utility that relocates the CP/M operating system to execute in any memory size you specify.

To generate a 48K version of CP/M, you enter:

```
MOVCP.48 *
```

where:

48 is the memory size

* instructs MOVCPM to leave the relocated CP/M image in memory.
MOVCPM responds with:

CONSTRUCTING 48K CP/M vers 2.2
READY FOR "SYSGEN" OR
"SAVE 37 CPM48.COM"

This is the end of MOVCPM execution. You follow this by running either the SYSGEN or the SAVE utility. Normally, you use the SYSGEN utility. Use the SAVE utility if you want to "patch" the operating system.

NOTE: Your Commodore 64 version of MOVCPM properly adjusts all of the CP/M code, including the BOOT80 and BIOS80 programs. You do NOT have to reassemble these programs and use DDT to patch them into the new version of the operating system as you do on less capable CP/M systems.

Execution of MOVCPM as shown above leaves a copy of the relocated CP/M operating system, including BOOT80, CCP, BDOS, and BIOS80, in the Transient Program Area (TPA) ready to be saved as a file on your disk or written directly to the system tracks. (To learn more about CP/M structure, read Chapter 6.)

If you choose to save a copy, you can SYSGEN it later.

4.4.2 Saving the New System

The SAVE built-in command writes the content of the TPA (in this case, a copy of your newly relocated CP/M) to the specified disk file. The MOVCPM command tells you how many 256-byte pages to save. MOVCPM on your Commodore 64 always tells you to save 37 pages.

To save your relocated version of CP/M, enter:

SAVE 37 CPM48.COM

This command will write the relocated CP/M to a file named "CPM48.COM". This is a full copy of a 48K version of the CP/M operating system. You can use the saved copy of CP/M in subsequent SYSGEN commands or for direct alteration under DDT.
4.4.3 Using SYSGEN

A version of CP/M that you have saved in a disk file cannot be directly executed. You must first SYSGEN it to the system tracks of a CP/M disk.

SYSGEN writes the specified version of the CP/M operating system to the proper locations on the system tracks of a CP/M disk. SYSGEN can read a version of the operating system from one of two places:

- The system tracks of diskette.
- A memory image of CP/M loaded into the TPA by the MOVCPM or DDT programs.

If you are using a file containing a SAVED version of CP/M, you must first bring it into memory with the DDT program. In our example, you enter:

```
DDT CPM48.COM
```

then exit from DDT with a G0 command.

If your source for the new version of CP/M is the system tracks of your disk or a memory resident image, you simply enter:

```
SYSGEN
```

and SYSGEN responds with:

```
SOURCE DRIVE NAME
(OR RETURN TO SKIP)
```

At this point you can specify the drive (A or B) whose system tracks you want read. If you simply hit the key, SYSGEN assumes that a copy of CP/M is already loaded into the TPA.

Whatever way you get the CP/M version loaded into memory, SYSGEN will ask you:

```
DESTINATION DRIVE NAME
(OR RETURN TO REBOOT)
```

If you respond with a destination drive name (A or B), SYSGEN will write CP/M to the system tracks of that drive.
If you simply hit the RETURN key, SYSGEN will reboot from whatever disk is currently in Drive A.

---

NOTE: IF you SYSGEN a CP/M system that is different in size from the one you ran the SYSGEN under, DO NOT try to reboot from a disk containing the new system. This will cause the operating system to crash. Re-insert the disk from which you loaded SYSGEN before you tell it to reboot.

---

To test a newly SYSGENed version of CP/M, you'll have to start it from native mode on your Commodore 64.

### 4.5 THE COMMODORE 64 KEYBOARD AND SCREEN WITH CP/M

The Commodore 64 has a full typewriter-style keyboard that behaves as such when you are running CP/M. All of the CP/M **CTRL** shifted control codes operate as they are supposed to. In addition, the **STOP/RUN** key on your Commodore 64 keyboard acts like a **CTRL** -C to produce a warm boot of the CP/M operating system.

In the Commodore 64 version of CP/M, you have the option of using only upper case or both upper and lower case. You toggle between them using the Commodore **C** key on the keyboard. You can use the CONFIG utility to tell CP/M to start with upper only or with upper/lower case enabled.

Table 5.3 contains a complete list of the **special CP/M control keys**. These are identical to those defined for CP/M, with a few additional functions taken from your Commodore 64 keyboard.

The Commodore 64 graphics characters and screen color control are not generally available to CP/M. But there is no reason that you can't store values into your Commodore 64 6567 Video Interface Chip's control registers just as you do when running in native mode. To arrive at the proper addresses for the control registers, examine Section 6.1.3, which explains the address mapping between the Z80 and 6510 processors.
The control values that you insert into the registers are the same as those you use in native mode. As an example, suppose you want to use your Commodore 64 graphics character set. Running in native mode, you simply touch the graphics key to switch on the graphics character set. From a CP/M program running under the Z80, you have to control it directly through a store into the appropriate 6567 control register.

The character set selection control register is at

6510 address 53,272 decimal or $D018 hexadecimal

which converts to the Z80 address base:

Z80 address 49,176 decimal or $C018 hexadecimal

The character set control register normally contains a $17. To invoke the graphics character set, you must store a $15 in the register:

MVI A, 15H ;LOAD THE CONTROL VALUE IN A
STA 0C018H ;STORE $15 IN THE 6567 CONTROL REGISTER

Once you’ve executed this code, the graphics character set is available to you. This operation does not change the character codes reaching your CP/M programs from the keyboard—only the display is changed.

You can use the same technique to alter colors, activate Sprites, or even play music through your Commodore 64 6581 Sound Interface Device. If you want to store characters directly into the screen matrix, remember to store Commodore 64 screen codes, not ASCII codes.

To use the dynamic features of your Commodore 64 from CP/M, all you have to do is remember that the 6510 addresses for the control registers must be reduced by $1000 (4096) in your CP/M programs.
CHAPTER 5

CP/M OPERATION

• How to Use This Chapter
• CP/M File Naming Conventions
• Input/Output Hardware Conventions
• CP/M Command Structure
• CP/M Commands
This chapter tells you how to use CP/M on your Commodore 64. It is not a detailed lesson on CP/M and its internal workings. It is an introduction to CP/M’s conventions and notations, and an introduction to the commands that you can use under CP/M.

If you want detailed information on the internal workings of CP/M, get one of the many fine books listed in Appendix B, the Bibliography. That level of detail is far beyond the scope of this book.

5.1 HOW TO USE THIS CHAPTER

Section 5.2 describes the CP/M file naming conventions. You should follow some reasonable conventions for naming your own files so that you can easily identify their contents.

Section 5.3 discusses the CP/M disk identification conventions. CP/M uses disk A and disk B; your Commodore 64 identifies these disks as disk 0 and disk 1. Section 5.3 also tells you how CP/M differs when you use the VIC 1541 or the CBM 4040 drive.

Section 5.4 describes the CP/M command structure and gives a table of all the CP/M commands that you get with your Commodore 64 CP/M system.

Section 5.5 provides brief descriptions of the CP/M commands. If you need more detail, see one or more of the CP/M books listed in Appendix B. Some books are more technical than others, so find the one with the amount of detail you are most comfortable with.

5.2 CP/M FILE NAMING CONVENTIONS

When you are using CP/M on your Commodore 64, you should follow the CP/M file naming conventions. CP/M files have the general format:

[DISK-ID:] FILENAME [.TYPE]
where:

*DISK-ID* is an optional disk drive identifier (such as A or B) that is needed when you want to use a file not on the currently logged disk.

*FILENAME* is a one- to eight-character name used to identify your file to CP/M.

*TYPE* is an optional one- to three-character name used to further identify your file.

Some examples of CP/M filenames are:

A:SAMPLE.BAS  A BASIC sample program stored on the disk on Drive A.
MY.TXT         A text file.
PROGRAM.COM    A program that is executable.
10/25/82.DRY    A diary entry.

CP/M lets you use any alphabetic or numeric character in your file names, as well as some special characters. CP/M reserves a few of the special characters for its own use. You *cannot* use the following characters in a CP/M file name:

< > . , ; : = ? * [ ]

With some software packages, files must be named with specific types, such as SUB for a SUBMIT file or ASM for an Assembly Language source file. Read the information with your software packages to see if you need to follow any naming conventions for that package's files.

Even if you don't have to follow any specific rules in naming your files, you should try to use reasonable naming conventions. In this way, when you get a directory listing (a list of all the files on a disk), you will have some idea of what's in the files.

A file named MORTGAGE.BAS is easier to recognize as the set of source statements for a BASIC program that calculates mortgage rates than a file named X127GY9.123. In other words, it makes sense to name your data files in ways that represent their contents. For example, a file named
01/15/83.DTA could contain the data you collected on January 15, 1983.

Since there are so many CP/M users (over 500,000 to date), certain standard filename types have been adopted. The most commonly used types are shown in Table 5.1.

<table>
<thead>
<tr>
<th>TYPE</th>
<th>FUNCTION OR CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>*.ASM</td>
<td>Assembly language source file</td>
</tr>
<tr>
<td>.BAK</td>
<td>Backup file</td>
</tr>
<tr>
<td>.BAS</td>
<td>BASIC program source file (for some BASIC interpreters like CBASIC)</td>
</tr>
<tr>
<td>*.COM</td>
<td>Directly executable transient program</td>
</tr>
<tr>
<td>.DAT</td>
<td>Data file</td>
</tr>
<tr>
<td>.DOC</td>
<td>Document or text file (required by some word processing packages)</td>
</tr>
<tr>
<td>*.HEX</td>
<td>File containing data in hexadecimal format; an Intel HEX format object code file</td>
</tr>
<tr>
<td>.INT</td>
<td>Output file from some compilers (CBASIC, JRT PASCAL) that contains intermediate code</td>
</tr>
<tr>
<td>*.LIB</td>
<td>Library file</td>
</tr>
<tr>
<td>.LST</td>
<td>Program listing (usually output from a language processor like a compiler, interpreter, or assembler)</td>
</tr>
<tr>
<td>.PRN</td>
<td>Print file (usually output from an assembler or compiler)</td>
</tr>
<tr>
<td>.PRT</td>
<td>Print file (usually output from an interpreter or compiler)</td>
</tr>
</tbody>
</table>
Table 5.1 Commonly Used CP/M File Types

<table>
<thead>
<tr>
<th>TYPE</th>
<th>FUNCTION OR CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>.SRC</td>
<td>Source file from the CP/M User's Group</td>
</tr>
<tr>
<td>*.SUB</td>
<td>Command file for a SUBMIT run</td>
</tr>
<tr>
<td>.SYM</td>
<td>Symbol table file (generated by some compilers, assemblers, and interpreters)</td>
</tr>
<tr>
<td>.TEX</td>
<td>Text file (required by some word processors)</td>
</tr>
<tr>
<td>.TXT</td>
<td>Text file (required by some word processors)</td>
</tr>
<tr>
<td>*.###</td>
<td>Either a temporary file or an improperly saved (and unusable) file</td>
</tr>
</tbody>
</table>

NOTE: Those filename types marked with an asterisk (*) must be adopted if you want to use associated software packages or system functions. That is, all CP/M directly executable programs must be named "filename.COM."

5.3 INPUT/OUTPUT HARDWARE CONVENTIONS

CP/M has certain conventions that must be followed when you are reading files from a disk or writing files to a disk.

The first disk drive physically attached to the system is called drive A. The next is drive B. When you are using a single 1541 disk drive, your Commodore 64 CP/M uses a slightly different way of telling which disk is in the drive (this is described in some detail below).

When you begin CP/M, you will be "logged" to drive A and you will see the prompt "A>" on your screen. This means that if you specify a filename in a command and you don't
specify a disk-id before the filename, the disk on drive A will be searched for the file.

You can log to drive B by entering the command:

B:

After entering the B: command, any filename that you specify without a disk-id preceding the filename will be read from or written to drive B.

You can change back and forth between drive A and drive B by simply entering the above command. You can tell which drive you're currently accessing by looking at the prompt: it will be A> when you're using drive A or B> when you're using drive B.

Your Commodore 64 CP/M can use either the VIC 1541 single disk drive or the CBM 4040 dual disk drive. Read the sections below that cover the type of disk drive you have attached to your Commodore 64.

5.3.1 Loading Programs from Disk: Single Drive

It is easy to load and run a CP/M program. You first place the program disk into your disk drive and then enter the filename followed by a carriage return, for example:

MYPROG <CR>

CP/M then goes to the currently logged disk and looks for the file called MYPROG.COM. If CP/M finds this file, the data in the file are read into the computer’s memory and CP/M begins executing those instructions.

If the file is not found on the disk, then CP/M prints the filename followed by a question mark:

MYPROG?

In such cases, check to see if you have the correct disk in the drive, log to the correct disk, or correct the program name.

For a single-drive system, if you are logged to drive A and your program is on drive B, then remove disk A from the drive, insert disk B, and enter:
CP/M will first ask that the appropriate disk be placed in the drive by writing:

```
INSERT DISK B INTO DRIVE 0, PRESS RETURN
```

You should put the appropriate disk into the drive and press the RETURN key. CP/M will then search the disk for the file called OTHERPGM.COM, load the file, and run it.

### 5.3.2 Loading Programs from Disk: Dual Drive

When using the CBM 4040 dual disk drive, you don’t have to physically change the disk in the drive when you want to log to another disk. Since there are two drives, you can insert two disks into the drive: disk A and disk B.

When you enter the B> command to log to disk B, CP/M will not ask you to insert a disk into the drive. Instead, CP/M will use the disk already in drive B.

If you want to change which disk is in a drive, you should change the disk and then tell CP/M that a different disk is in the drive by entering a CTRL-C command. This makes CP/M read the directory from the disk and keeps you from writing over information that you want to keep.

You must have the Commodore 64 IEEE interface cartridge when you use the CBM 4040 dual disk drive. You cannot plug the dual disk drive into the Commodore 64 without the interface cartridge.

### 5.4 CP/M COMMAND STRUCTURE

Your Commodore 64 CP/M system includes a Console Command Processor (CCP) through which you interact with CP/M. The CCP reads and interprets the commands you enter at the keyboard.

The CP/M commands are listed in Table 5.2 and described in some detail later in this chapter.
In general, the CP/M commands are of two types:

- **Built-in commands** which are a part of the CCP itself. Being part of the CP/M operating system, built-in commands are included whenever you load CP/M.
- **Transient commands** which are loaded into the Transient Program Area (TPA) from a disk and then executed. Transient commands reside on the disk as COM files.

<table>
<thead>
<tr>
<th>COMMAND NAME</th>
<th>BUILT-IN (B)</th>
<th>COMMAND FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pgm-name</code></td>
<td>T</td>
<td>Load and execute the program stored on the disk as file <code>pgm-name.COM</code>.</td>
</tr>
<tr>
<td><code>x:</code></td>
<td>B</td>
<td>Change the currently logged disk to disk <code>x</code>.</td>
</tr>
<tr>
<td><code>ASM</code></td>
<td>T</td>
<td>Load the CP/M assembler and assemble the specified program from the disk.</td>
</tr>
<tr>
<td><code>DDT</code></td>
<td>T</td>
<td>Load the CP/M debugger (DDT) and begin executing the debugger.</td>
</tr>
<tr>
<td><code>DIR</code></td>
<td>B</td>
<td>List the filenames in the disk directory.</td>
</tr>
<tr>
<td><code>DUMP</code></td>
<td>T</td>
<td>Dump the contents of the specified file to the screen in hexadecimal format.</td>
</tr>
<tr>
<td><code>ED</code></td>
<td>T</td>
<td>Load and execute the CP/M text editor program.</td>
</tr>
<tr>
<td><code>ERA</code></td>
<td>B</td>
<td>Erase the specified file(s) from the disk.</td>
</tr>
<tr>
<td>COMMAND NAME</td>
<td>BUILT-IN (B) or TRANSIENT (T)</td>
<td>COMMAND FUNCTION</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>LOAD</td>
<td>T</td>
<td>Produce an executable (COM) file from an assembled (HEX) file.</td>
</tr>
<tr>
<td>MOVCPM</td>
<td>T</td>
<td>Recreate the CP/M system for the specified memory size.</td>
</tr>
<tr>
<td>PIP</td>
<td>T</td>
<td>Copy specified file(s).</td>
</tr>
<tr>
<td>REN</td>
<td>B</td>
<td>Rename the specified file.</td>
</tr>
<tr>
<td>SAVE</td>
<td>B</td>
<td>Save the contents of memory as the specified file on the disk.</td>
</tr>
<tr>
<td>STAT</td>
<td>T</td>
<td>Provide status information about specified files, no file, or all files, and list the number of available bytes remaining on the disk.</td>
</tr>
<tr>
<td>SUBMIT</td>
<td>T</td>
<td>Read the specified file and execute the commands in a batch processing mode.</td>
</tr>
<tr>
<td>SYSGEN</td>
<td>T</td>
<td>Create a new CP/M system diskette.</td>
</tr>
<tr>
<td>TYPE</td>
<td>B</td>
<td>Type the contents of the specified file onto the screen.</td>
</tr>
<tr>
<td>USER</td>
<td>B</td>
<td>Change the currently logged user number to the specified value.</td>
</tr>
<tr>
<td>XSUB</td>
<td>T</td>
<td>Allow the entering of data as well as CP/M commands in a SUBMIT file.</td>
</tr>
</tbody>
</table>

In addition to the commands listed in Table 5.2, your CP/M system includes a number of built-in line editing...
commands. The CP/M line editing commands, shown in Table 5.3, have the general form:

```
CTRL -x
```

where:

CTRL means hold down the CONTROL key on your Commodore 64.

\( x \) is one of the keys on your Commodore 64 keyboard.

**Table 5.3 CP/M Built-in Line Editing Commands**

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRL -C</td>
<td>Perform a CP/M warm-start.</td>
</tr>
<tr>
<td>or RUN/STOP</td>
<td></td>
</tr>
<tr>
<td>CTRL -E</td>
<td>Move to the beginning of the next line.</td>
</tr>
<tr>
<td>CTRL -H</td>
<td>Delete one character and erase it from the screen.</td>
</tr>
<tr>
<td>or DEL</td>
<td></td>
</tr>
<tr>
<td>CTRL -J</td>
<td>Perform a carriage return and line feed.</td>
</tr>
<tr>
<td>CTRL -M</td>
<td>Perform a carriage return.</td>
</tr>
<tr>
<td>or RETURN</td>
<td></td>
</tr>
<tr>
<td>CTRL -P</td>
<td>Toggle printer/console output. On first use, send all screen messages to the printer; on next use, send all screen messages to the screen.</td>
</tr>
<tr>
<td>CTRL -R</td>
<td>Repeat the current command line.</td>
</tr>
<tr>
<td>CTRL -S</td>
<td>Temporarily halt listing of data on the screen. Press any key to continue listing.</td>
</tr>
<tr>
<td>CTRL -U</td>
<td>Cancel current command line.</td>
</tr>
<tr>
<td>or CTRL -X</td>
<td></td>
</tr>
<tr>
<td>⌘</td>
<td>Toggle between all upper case and upper/lower case letters. ⌘ is the Commodore key.</td>
</tr>
</tbody>
</table>
5.5 CP/M COMMANDS

This section gives you a brief description of the Commodore 64 CP/M commands. It is not intended to be a detailed description of how CP/M commands operate, nor does it attempt to describe every possible way you can use the CP/M commands.

If you need to learn how CP/M works or if you need more detail on how the commands work, you should purchase one or more of the excellent CP/M teaching texts on the market. Skim these books and pick those that present the information in a way that you can easily understand.

The following notation is used in describing the CP/M commands:

- **Underlined** words show arguments (parameters) which you replace with your own values.
- **BOLDFACE** keywords must be entered exactly as shown.
- A **vertical bar (|)** separates arguments where you may select any one of the list of arguments.
- **Square brackets ([ ])** are used to show optional arguments. You select any or none of the arguments listed, depending on your needs.
- **Braces ({ })** show that you must choose one of the arguments.

5.5.1 pgm-name (Load and Run a CP/M Program)

Format: \[disk-id:]filename<CR>\]

where:

*disk-id* is an optional disk identifier.

*filename* is the name of the file containing the program to be loaded and run. Programs must be stored in files named *filename*.COM.

Description:

CP/M programs are stored in files named *filename*.COM. When you type the name of one of
these program files and hit the carriage return key, CP/M does the following:

1. Searches the currently logged disk or the disk specified by disk-id for the program file filename.COM.
2. Loads the program file into memory.
3. Begins executing the instructions in the program.

If the file is not found on the disk, CP/M prints a message like this:

FILENAME?

When you get this message, make sure you have the correct disk in the disk drive, that you've spelled the program filename correctly, and that the program is stored in a COM file.

Example 1:

To load and execute your program which is stored in the file MYPROG.COM, enter:

MYPROG <CR>

CP/M searches the currently logged disk for the file MYPROG.COM, loads the file, and begins executing the instructions. If the file is not on the disk, you will see the error message:

MYPROG?

Example 2:

You have a single drive system and are currently logged to disk A. You want to load and run the program XYZ from disk B. Enter the CP/M command:

B:XYZ <CR>

CP/M then responds with:

PLACE DISK B INTO THE DISK DRIVE AND HIT RETURN
Put the appropriate disk into the disk drive and press the \texttt{RETURN} key. Then, CP/M searches for the file named \texttt{XYZ.COM}, loads the file, and begins executing its instructions.

\section*{5.5.2 \texttt{x}: (Change the Currently Logged Disk)}

\textbf{Format:} \texttt{disk-id:}

where:

\texttt{disk-id} is the disk identifier

\textbf{Description:} Under CP/M, you are always "logged" to a disk. You can tell which disk CP/M is using by looking at the prompt message. If it's "A>", you're logged to disk A; if it's "B>", you're logged to disk B.

You can change the logged disk by entering:

\texttt{DISK-ID:}

CP/M then asks you to insert the appropriate disk into the disk drive and hit the carriage return. CP/M remembers which disk you're currently logged to and will request another disk if you ask for a file or program and use the \texttt{disk-id} qualifier.

\textbf{Example:} You have a single drive system and are currently logged to disk A. You want to log to disk B. To do this, you would enter:

\begin{verbatim}
B: <CR>
\end{verbatim}

CP/M then writes:

\texttt{INSERT DISK B INTO DRIVE 0, PRESS RETURN}

When you insert the disk into the drive and hit the carriage return, CP/M is logged to that disk. The CP/M prompt will now be:

\begin{verbatim}
B>
\end{verbatim}
5.5.3 ASM

Format: \texttt{ASM filename[.parms]}

where:

\texttt{filename} is the name of the file containing the program to be assembled. The file must be named \texttt{filename.ASM}.

\texttt{parms} contains up to three characters specifying the drive(s) for the source file, HEX file, and PRN file.

Description:

The ASM command loads and executes the CP/M Assembler which processes 8080 instructions. The CP/M Assembler:

1. Assembles the assembly language statements contained in the file \texttt{filename.ASM}.
2. Generates an object file in hexadecimal format and places the object file in \texttt{filename.HEX}.
3. Produces a print file in \texttt{filename.PRN}.

The \texttt{parms} string is an optional character string which tells the assembler where to read and write its files. You can specify up to three characters in \texttt{parms}. Each character position has a special meaning:

- Position 1: The source drive for the file containing the assembly language statements.
- Position 2: The destination drive for the object (HEX) file.
- Position 3: The destination drive for the print (PRN) file.

If you specify a "Z" for positions 2 and/or 3, the assembler will not generate a HEX (position 2) or PRN (position 3) file. If you specify an "X" for position 3, the listing will appear on your screen instead of in a file. Table 5.4 lists the ASM error messages.
NOTE: CP/M was written for the Intel 8080 microprocessor. The Z80 processor in your Commodore 64 is compatible with the 8080 processor but offers a much larger instruction set, more internal registers, and other advantages.
If you want to use the full Z80 instruction set, you'll have to get an assembler that recognizes the Z80 instructions.

<table>
<thead>
<tr>
<th>ERROR CODE</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Data error. The data element cannot be placed into the specified data area. For example, you cannot put the value 500 in a one-byte area.</td>
</tr>
<tr>
<td>E</td>
<td>Expression error. The assembler could not evaluate the expression.</td>
</tr>
<tr>
<td>L</td>
<td>Label error. The label is used out of context. This could be a duplicate label.</td>
</tr>
<tr>
<td>N</td>
<td>Not implemented. You tried to use a feature that is not implemented, such as using macros.</td>
</tr>
<tr>
<td>O</td>
<td>Overflow. The expression is too complicated to evaluate.</td>
</tr>
<tr>
<td>P</td>
<td>Phase error. A label's value changed between passes of the assembler.</td>
</tr>
<tr>
<td>R</td>
<td>Register error. The value specified as a register does not match the value needed by the op code.</td>
</tr>
<tr>
<td>S</td>
<td>Syntax error. The statement contains a syntax error and could not be evaluated.</td>
</tr>
<tr>
<td>U</td>
<td>Undefined label. You used a label which does not exist in the program.</td>
</tr>
<tr>
<td>V</td>
<td>Value error. There is an improperly formed operand in the expression.</td>
</tr>
</tbody>
</table>
Examples:

ASM APROG.BBB  Assemble the assembly language program contained in the file B:APROG.ASM and put the object file in B:APROG.HEX and the print file in B:APROG.PRN.

ASM PGM2.BZZ  Assemble the assembly language program contained in the file B:PGM2.ASM. Do not generate either the object (HEX) file or the print (PRN) file.

ASM PGMFOR.AAX  Assemble the assembly language program contained in the file A:PGMFOR.ASM. Put the object file (PGMFOR.HEX) onto Disk A. Print the listing on the screen.

5.5.4 DDT

Format:  

\[
\text{DDT} \ [ \ [\text{disk-id:}] \ \text{filename} [.\text{type}] \ ]
\]

where:

disk-id  is an optional disk identifier.

filename.type  is a valid CP/M filename for the file containing the information to be loaded and processed by DDT.

Description:

DDT is the CP/M Dynamic Debugging Tool which you can use to interactively test and debug programs. You can load any file into memory using DDT. If you load an executable file, you can directly control its execution from your console.

---

**NOTE:** You can also use DDT to look at a file in both ASCII and hexadecimal format.
DDT loads the file into the TPA (Transient Program Area) in memory. You can then use the commands shown in Table 5.5 to operate on the information in the TPA.

You must know 8080 assembly language instructions to use DDT. If you don’t know the assembly language instructions, don’t try to use DDT. Appendix B gives a list of some of the currently available Z80 assembly language books.

NOTE: DDT recognizes only the subset of Z80 instructions that is identical to the Intel 8080 microprocessor instruction set.

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>As</td>
<td>Assemble. Begin entering assembly language instructions at address s.</td>
</tr>
<tr>
<td>D[s,f]</td>
<td>Display. Display the contents of memory in both hexadecimal and ASCII formats. Begin at address s and end at address f. If you don’t specify f, 16 display lines are shown. If you don’t specify s, the starting address is the current display address.</td>
</tr>
<tr>
<td>Fs,f,c</td>
<td>Fill memory. Fill memory with the hexadecimal byte c. Begin storing the byte c at location s and end at location f. You use the F command to fill a block of memory with one value, for example, all zeros or blanks.</td>
</tr>
<tr>
<td>G[s,b1,b2]</td>
<td>Go. Begin executing the instructions at location s with optional breakpoints at locations b1 and</td>
</tr>
<tr>
<td>COMMAND</td>
<td>MEANING</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>b2</td>
<td>If you don't specify location s, execution begins at the current address.</td>
</tr>
<tr>
<td>Hc1,c2</td>
<td>Hexadecimal sum/difference. Add (or subtract, depending on the signs) the hexadecimal constants c1 and c2.</td>
</tr>
<tr>
<td>Ifilename[type]</td>
<td>Input. Insert the filename filename.type into the default file control block for the TPA. You must use an R command to actually read the file.</td>
</tr>
<tr>
<td>L[s[f]]</td>
<td>List. List the assembly language mnemonics beginning at address s and ending at address f. If you don't specify a value for s, the listing begins at the current address. If you don't specify a value for f, 12 lines are listed.</td>
</tr>
<tr>
<td>Ms,f,d</td>
<td>Move a block of information. Move the contents of a block of memory. Begin moving data from address s and end at address f. Move the information to address d.</td>
</tr>
<tr>
<td>R[o]</td>
<td>Read a disk file. Read the file whose filename and type are in the file control block into the program area beginning at offset o. You use an I command to set the file information in the file control block. If you don't specify an offset value, the file is read into memory beginning at address 100H.</td>
</tr>
<tr>
<td>COMMAND</td>
<td>MEANING</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>Ss</td>
<td>Examine and modify memory values. DDT begins processing at location s. All addresses and their contents are listed. If you hit a carriage return, the contents are not changed. If you want to change the value, enter a new value before you hit the carriage return. To stop the listing, hit a period (.)</td>
</tr>
<tr>
<td>T[n]</td>
<td>Trace program execution. DDT traces execution and displays registers and flags for n steps. n may be 1 through 65535. If you don't specify a value for n, DDT executes and traces one statement.</td>
</tr>
<tr>
<td>U[n]</td>
<td>Untrace. This performs the same processing as the T command except that the registers and flags are not displayed for each step.</td>
</tr>
<tr>
<td>X[r]</td>
<td>Examine and modify CPU registers. The examine command lets you examine and optionally modify the contents of the CPU registers shown in Table 5.6. If you don't specify a value for r, all of the CPU registers are displayed in the format shown in Table 5.7.</td>
</tr>
</tbody>
</table>

---

### Table 5.6 DDT CPU Registers/Status Flags

<table>
<thead>
<tr>
<th>NAME</th>
<th>MEANING</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS FLAGS:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Carry flag</td>
<td>0/1</td>
</tr>
<tr>
<td>Z</td>
<td>Zero flag</td>
<td>0/1</td>
</tr>
<tr>
<td>M</td>
<td>Minus flag</td>
<td>0/1</td>
</tr>
</tbody>
</table>

CP/m OPERATION  69
Table 5.6 (Continued)

<table>
<thead>
<tr>
<th>NAME</th>
<th>MEANING</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS FLAGS:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Even parity flag</td>
<td>0/1</td>
</tr>
<tr>
<td>I</td>
<td>Interdigit carry</td>
<td>0/1</td>
</tr>
<tr>
<td>REGISTERS:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Accumulator</td>
<td>0-FF</td>
</tr>
<tr>
<td>B</td>
<td>BC register pair</td>
<td>0-FFFF</td>
</tr>
<tr>
<td>D</td>
<td>DE register pair</td>
<td>0-FFFF</td>
</tr>
<tr>
<td>H</td>
<td>HL register pair</td>
<td>0-FFFF</td>
</tr>
<tr>
<td>S</td>
<td>Stack pointer</td>
<td>0-FFFF</td>
</tr>
<tr>
<td>P</td>
<td>Program counter</td>
<td>0-FFFF</td>
</tr>
</tbody>
</table>

Examples:

DDT Loads DDT and waits for you to enter commands.

DDT PROG.COM Loads DDT and reads the file PROG.COM into the TPA (address 100H). DDT then waits for you to enter commands.

Table 5.7 DDT CPU Register/Flag Display Format

CfZfMfEflf A=bb B=ddddd D=ddddd H=ddddd S=ddddd P=ddddd inst

where:

C, Z, M, E, and I are processor status flags shown in Table 5.6

A, B, D, H, S, and P are the registers shown in Table 5.6

f is a 0 or 1 flag value

bb is a byte value (0 through 255)

ddddd is a double byte value

inst is the disassembled 8080 instruction at the location addressed by program counter (P)
5.5.5 DIR

Format:  \texttt{DIR [disk-id:] [filename.type]}

where:

\textit{disk-id} is an optional disk identifier.

\textit{filename} is an optional valid one- to eight-character CP/M filename.

\textit{type} is a valid one- to three-character CP/M file type. You need to specify a \textit{type} if you use the \textit{filename} parameter.

Description:

You use a \texttt{DIR} command to display the directory of files on a certain disk \textit{disk-id}. If you don't supply a \textit{disk-id} parameter, \texttt{DIR} lists the directory of the disk in the drive currently logged to the system.

You can use the CP/M wildcard (* and ?) characters in your \textit{filename} and \textit{type} parameters. These characters are acted upon as follows:

• \textbf{question mark (?)}

Use a question mark (?) to represent a \textit{single} character in a filename or type. \texttt{DIR} will use the ? to match on \textit{any} character that occupies that position in the filename or type. For example,

\texttt{DIR PGM?.COM}

will display all files that have the first three characters PGM, any fourth character and the type COM. This format will match only files with names PGMx.COM. It will not match PGMxxx.COM.

• \textbf{asterisk (*)}

Use an asterisk (*) to represent an \textit{entire} filename or type or the \textit{remainder} of a filename or type. \texttt{DIR} will match on \textit{any} characters in the positions indicated by the *. For example,

\texttt{DIR PGM*.COM}
will display all files that have the first three characters PGM, regardless of the length of the filename, and the type COM.

If you use a disk-id value, DIR will display only those files on the indicated disk. If you omit the disk-id value, DIR displays the files on the currently logged disk.

Examples:

**DIR**
Display the directory of the currently logged disk. The names of all files on the disk are shown.

**DIR B:**
Display the directory of Disk B.

**DIR B:TEST.COM**
Display the directory information for file TEST.COM on Disk B. You can use this form of the DIR command to check whether the file you want is on that disk.

**DIR *.BAK**
Display the information from the currently logged disk for all files which are of the type BAK.

**DIR TEST*.BAK**
Display the information from the currently logged disk for all files that are of the type BAK and whose filenames contain the first four characters TEST. This will display the files TEST.BAK, TEST1.BAK, TESTXXX.BAK, TEST1234.BAK, or any other file with the first four characters TEST and type BAK.

**DIR TEST???.BAK**
Display the information from the currently logged disk for all files that are of type BAK and have a four- to six-character filename beginning with the letters TEST. This will display the files TEST.BAK, TEST1.BAK, or TESTXX.BAK but will not display the file TEST1234.BAK.
5.5.6 DUMP

Format: DUMP [disk-id:]filename.type

where:

disk-id is an optional disk identifier.

filename is valid CP/M filename of the file whose contents are to be displayed.

type is a valid one- to three-character CP/M file type.

Description:
You use a DUMP command to display the contents of a file in hexadecimal format. The file information is shown on the screen.

Examples:

DUMP A:DATA.TST  Dump the contents of the DATA.TST file on Drive A to the screen. The file information is shown in hexadecimal format.

DUMP MY.DTA  Dump the contents of the MY.DTA file, which is on the currently logged disk, to the screen.

5.5.7 ED

Format: ED [disk-id:]filename.[type] [disk-id2:]filename2.[type2]]

where:

disk-id is an optional disk identifier.

filename is the name of the file containing the data to be edited.

type is a valid CP/M file type for the file containing the data to be edited.
disk-id2 is an optional disk identifier needed when you want the edited file to be written to a disk other than the disk being edited.

filename2 is the name of the output file when you want the edited filename to differ from the original filename.

type2 is the type for the output file when you want the edited file to have a different type than the original file.

Description:
You use the ED command to run the CP/M context editor to create or change CP/M source language, data, and text files. ED works on the data in its buffer, using a character pointer to keep track of its current position. Be sure that you understand how to use ED; you could lose your edited file if you're not careful!

If the file exists when you enter the ED command, CP/M opens it and prepares to operate on it. If the file does not exist, CP/M creates a new file with the specified name. CP/M names its temporary file filename.*** while you are editing the information.

When you are finished editing the file, CP/M changes the name of the original file to filename.BAK and writes the edited information to the file named filename.type when you tell ED to write the data. If you don't tell ED to write the edited information to the file, you will lose the edited data. You must tell ED everything!

If you want to write the edited file to a disk other than the one containing the original file, specify a disk-id2 parameter.

If the file that you are editing is too large to fit in memory, you must tell CP/M's ED processor when to swap information to its work files. The amount of data that can be processed without swapping depends on the size of your CP/M system. The standard Commodore 64 CP/M system is a 44K version.

You use the control characters shown in table 5.8 and the commands shown in table 5.9 when you are editing a file using ED.
Remember that the CP/M ED editor is not a very complex editor. It works in its buffers, and you must tell it *everything*. After you enter the command that tells ED what file to edit, you must tell ED to read in a specified number of lines from the file. In the same way, after you have finished editing, you must be sure to close the processing with an E command to save your edited data.

**NOTE:** Some ED commands (F, I, N, and S) when entered in upper case, automatically translate all subsequent lower case entries to upper case. If you enter these commands in lower case (f, i, n, s), the automatic translation to upper case is not done, and data can be entered in both upper and lower case.

### Table 5.8 CP/M ED Control Characters

<table>
<thead>
<tr>
<th>CHARACTER</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRL -L</td>
<td>Used as a logical carriage return/line feed within a string.</td>
</tr>
<tr>
<td>CTRL -X</td>
<td>Line delete.</td>
</tr>
<tr>
<td>CTRL -Z</td>
<td>String terminator/separator.</td>
</tr>
<tr>
<td>DELET</td>
<td>Delete the previous character.</td>
</tr>
</tbody>
</table>

### Table 5.9 CP/M ED Commands*

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>n:</td>
<td>Move the character pointer to the beginning of line n.</td>
</tr>
<tr>
<td>[+/-]n</td>
<td>Move the character pointer up (-) or down (+) n lines and type the line.</td>
</tr>
<tr>
<td>nA</td>
<td>Append n lines from the original file <em>filename</em> to the buffer in memory.</td>
</tr>
<tr>
<td>COMMAND</td>
<td>FUNCTION</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>0A</td>
<td>Append enough lines from the file to half fill the buffer.</td>
</tr>
<tr>
<td>#A</td>
<td>Append enough lines from the file to fill the buffer or reach the end of file.</td>
</tr>
<tr>
<td>[+/-]B</td>
<td>Move to the top (B) or bottom (-) of the buffer.</td>
</tr>
<tr>
<td>[+/-]nC</td>
<td>Move the buffer character pointer forward (+) or backward (-) (n) characters in the buffer.</td>
</tr>
<tr>
<td>[+/-]nD</td>
<td>Delete (n) characters from the buffer. Delete the characters before ((-1)) or after ((+)) the character pointer.</td>
</tr>
<tr>
<td>E</td>
<td>End the ED session. Rename the original file to (filename.BAK). Close the files and save the new file.</td>
</tr>
<tr>
<td>nFstring[^Z]</td>
<td>Find the character string (string) (n) times. If you don't supply a value for (n), the (string) is found only once. You use the (CTRL) -Z (^Z) to end the (string) when you want to enter another ED command on the same line as the F command. This command performs an automatic translation to upper case. To find a character string that includes lower case letters, use the f form of this command.</td>
</tr>
<tr>
<td>H</td>
<td>Save the new (edited) file. Rename the original file to (filename.BAK).</td>
</tr>
</tbody>
</table>
Table 5.9 (Continued)

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Re-edit the file using the new file as the original file. This is the same as entering an E (end edit) command and then running the ED editor again on the newly saved file.</td>
<td></td>
</tr>
<tr>
<td>I&lt;CR&gt;</td>
<td>Enter insert mode. You must enter a \texttt{CTRL} -Z (^Z) to end insert mode. When you use an I command, you can enter only upper-case characters. The character pointer is moved to the end of the inserted text when you enter the \texttt{CTRL} -Z. To enter both upper-case and lower-case information, use the I command described below.</td>
</tr>
<tr>
<td>Istring(^Z)</td>
<td>Insert the character string \textit{string} at the position in the buffer pointed to by the character pointer. The \texttt{CTRL} -Z marks the end of the string to be inserted. The character pointer is moved to the end of the inserted string. You can enter only upper-case characters with the I command. To insert both upper-case and lower-case information, use the \textit{i}string command described below.</td>
</tr>
<tr>
<td>i&lt;CR&gt;</td>
<td>Enter insert mode. You must enter a \texttt{CTRL} -Z (^Z) to end insert mode. When you use an i command, you can enter both upper-case and lower-case characters. The character pointer is moved to</td>
</tr>
</tbody>
</table>
Table 5.9 (Continued)

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>i string [ ^Z]</td>
<td>Insert the character string string at the position in the buffer pointed to by the character pointer. The [CTRL] -Z marks the end of the string to be inserted. The character pointer is moved to the end of the inserted string. You can enter both upper- and lower-case characters with the i command.</td>
</tr>
<tr>
<td>nJ string [^Z] string2 [^Z] string3 [^Z]</td>
<td>Juxtapose strings. Find string1. Add string2 to the end of string1 and delete all characters from the end of string2 up to but not including the first character of string3. You use the optional final [CTRL] -Z (^^Z) when you want to enter another ED command on the same line.</td>
</tr>
<tr>
<td>[+/-]nK</td>
<td>Delete the following (+) or previous (-) n lines.</td>
</tr>
<tr>
<td>[+/-]nL</td>
<td>Move the character pointer up (-) or down (+) n lines. If n is zero (0), move the character pointer to the beginning of the current line.</td>
</tr>
<tr>
<td>n M commands [^Z]</td>
<td>Execute the ED commands n times. If n is zero (0) or one (1), repeat the ED commands until an error occurs. You use the terminating [CTRL] -Z (^Z) to enter an-</td>
</tr>
<tr>
<td>COMMAND</td>
<td>FUNCTION</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>other ED command on the same line. Any ED commands after the ^Z are executed only once and are not treated as part of the M command.</td>
</tr>
<tr>
<td>nNstring [ ^Z]</td>
<td>Find the nth occurrence of the character string string. You use the optional terminating CTRL-Z (^Z) when you want to enter another ED command on the same line. The N command performs an automatic translation from lower case to upper case. If you want to find a string containing lower-case letters, use the n form of this command.</td>
</tr>
<tr>
<td>O</td>
<td>End the ED session and keep the original file. Do not apply any of the changes made during the session.</td>
</tr>
<tr>
<td>[+/-]nP</td>
<td>Display n pages. Each page is 24 lines. Display the n pages before (-) or after (+) the current position of the character pointer. If you supply a zero (0) for n, the current line and the next 23 lines are listed.</td>
</tr>
<tr>
<td>G</td>
<td>Abandon the editing session. Do not save the new (edited) file. Return to CP/M.</td>
</tr>
<tr>
<td>R[filename]</td>
<td>Read the file and insert the text into the buffer. Move the character pointer to the end of the inserted</td>
</tr>
</tbody>
</table>
Table 5.9 (Continued)

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>text. If you supply a <em>filename</em>, ED reads the file <em>filename</em>.LIB. If you don't supply a value for <em>filename</em>, ED reads the file X$$$$$$$.LIB.</td>
<td></td>
</tr>
</tbody>
</table>

*nString1 ^Zstring2 [*^Z]*

Find *string1* and replace it with *string2*. Repeat this substitution *n* times. If you do not supply a value for *n*, the substitution is performed once. You use the terminating `CTRL-Z` (*^Z*) when you want to enter another ED command on the same line. The *S* command performs an automatic translation from lower case to upper case. If you want to use lower-case letters in your strings, use the *s* form of this command.

`[+-/]nT`

Display the previous (-) or following (+) *n* lines. If *n* is zero (0), or if *n* is not supplied, display the current line. *B#T* displays the entire buffer.

`[+-/]U`

Translate all characters in the buffer to upper case. Plus (+) turns on the translation. Minus (-) turns off the translation.

`[+-/]0V`

Turn on (+) or off (-) the line number display. The 0 displays the amount of free buffer space in bytes and the total buffer size.

`[n]W`

Write the following *n* lines to the *temporary output file*.
<table>
<thead>
<tr>
<th>COMMAND</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename.X</td>
<td>If you do not specify a value for n, only the current line is written to the file.</td>
</tr>
<tr>
<td>[n]X</td>
<td>Write the following n lines to the temporary file X$\ldots$.LIB. You can retrieve these lines with an R command (this is an easy way to move a block of lines). If n is zero (0), ED will DELETE the X$\ldots$.LIB file.</td>
</tr>
<tr>
<td>nZ</td>
<td>Wait n seconds before resuming ED processing.</td>
</tr>
</tbody>
</table>

*NOTES: You can use the operand n1:n2 for any n or n operand in the ED commands shown in this table. If you use the n1:n2 form, the ED processor will operate on the lines n1 through n2. If you use this form and omit either n1 or n2, ED assumes the current line for the missing operand.

You can use a # for n in the ED commands. # means to use the largest possible value (65535) for n

Many of the ED commands show a +/− form. You do not need to specify the plus (+) sign. You do need to specify the minus (−) sign if you want to move backward in the file

The F, I, N, and S commands perform an automatic translation to upper case. If you want to enter both upper and lower case data, use the commands f, i, n, and s.

Example:

ED PGMTST.ASM Edit the file PGMTST.ASM. If the file exists, you must remember to read in the data with an A command before attempting to edit it.
Format: \texttt{ERA [disk-id:]filename.type}

where:

\textit{disk-id} is an optional disk identifier.

\textit{filename} is a valid CP/M filename.

\textit{type} is a valid CP/M file type.

Description:

You use an \texttt{ERA} command to erase one or more files from your disk. If you don't specify a \textit{disk-id} parameter, the file is erased from the currently logged disk.

\texttt{ERA} accepts the wildcard (*) notation for the \textit{filename} and \textit{type} parameters. This allows you to erase a group of files with a single command. Be careful that you don't erase files that you want to keep when you use the wildcard notation.

Examples:

\begin{itemize}
\item \texttt{ERA TEST.DTA} \hspace{1cm} Erase the file TEST.DTA from the currently logged disk.
\item \texttt{ERA B:MY.PGM} \hspace{1cm} Erase the file MY.PGM from disk B.
\item \texttt{ERA *.BAK} \hspace{1cm} Erase all files with a type BAK from the currently logged disk.
\item \texttt{ERA A:*.*} \hspace{1cm} \textbf{CAUTION.} Erase all files from disk A. (CP/M asks you whether you really want to erase all files from the disk.)
\item \texttt{ERA TEST.*} \hspace{1cm} Erase all files with the filename TEST from the currently logged disk. This would erase, for example, TEST.DTA, TEST.PGM, TEST.ASM, TEST.BAK, TEST.xxx.
\end{itemize}
5.5.9 LOAD

Format: LOAD [disk-id:]filename

where:

*disk-id* is an optional disk identifier.

*filename* is the name of the file containing output from the assembler.

Description:

You use a LOAD command to process the output from the assembler (see the description of the ASM command) and produce an executable program file. The input file must be named *filename.HEX*. The output file is named *filename.COM*.

You run the output from the LOAD processor by entering the filename and hitting a carriage return (see the description on loading and executing a CP/M program in Section 5.5.1).

Example:

LOAD ASMPGM2   Process the file ASMPGM2.HEX (which was created by the assembler) and produce an executable program in the file ASMPGM2.COM.

5.5.10 MOVCPM

Format: MOVCPM [ { * | size } ] [ * ]

where:

the first * tells CP/M to calculate the amount of memory available for its use.

size is a two-digit number from 20 through 48 which is the maximum amount of memory available for CP/M in your Commodore 64. You use 44 for a 44K version of CP/M.
the second * tells CP/M to leave the new version in memory for later SYSGEN or SAVE command processing.

Description:
You use a MOVCPM command to configure (prepare) a new copy of your CP/M system. Changing CP/M to expect a different memory size is called "moving" the system. The MOVCPM command operates in either of these ways, depending on which parameters you use:

1. "Move" CP/M and immediately execute the new, different sized system. Do not save it on disk.

2. "Move" CP/M and prepare the new system to be saved to disk by a later SYSGEN or SAVE command. The new CP/M system is NOT written to the disk. You must use a SYSGEN or SAVE command to actually write out the new version of the system.

If you do not specify any parameters and use a MOVCPM command like this:

MOVCPM <CR>

CP/M will determine how much memory is available, create a new system, and immediately use the new system.

If you specify the first parameter, you can tell CP/M how much memory it can use by:

- Using the * which tells CP/M to use all available memory.
- Using the size parameter which tells CP/M to use sizeK bytes of memory.

You can use any decimal integer between 20 and 48 for the size value.

If you want to save the new version of CP/M on a disk, you must use the second * parameter and you must supply a first parameter (either size or *). You can use this type of command:

MOVCPM * * <CR>
CAUTION: MOVCPM WILL ONLY CREATE A NEW VERSION OF CP/M. THE NEW VERSION IS NOT SAVED TO A DISK UNTIL YOU USE A SAVE OR SYSGEN COMMAND!

Examples:

**MOVCPM** Create a new version of CP/M, use all available memory, and immediately execute the new version. Do not save this version.

**MOVCPM 40** Create a new version of CP/M using 40K of memory. Do not execute the version but prepare it to be saved to disk through a SAVE or SYSGEN command.

**MOVCPM 28** Create a 28K version of CP/M and execute it. Do not save this version.

**5.5.11 PIP**

Format: **PIP**

or

**PIP destination=source[parameter]**

where:

destination tells where you want to copy the file to.

destination is in the form:

[disk-ld:]filename.type

source tells which file to copy. source has the same format as destination.

parameter is one or more valid PIP parameters separated by zero or more blanks and enclosed in square brackets [ ].

Description:

You use PIP, CP/M's *Peripheral Interchange Program*, to copy files. It doesn't matter what's in the file. PIP
simply copies from the destination file to the source file. The source and destination files can be on the same disk or can be on different disks.

You can specify only the disk-id for the destination when the file is to be copied to a file with the same filename.type on another disk. You can use the wildcard (*) notation for any part of the source filename and/or type.

You use the parameters, or PIP commands, shown in Table 5.10 to have PIP perform some operations on the file during the copy process.

You can use PIP in two different ways:

1. Invoking PIP as a program by entering:

   PIP <CR>

   In this use, PIP is loaded and returns an * on the next line. You can then enter PIP commands, one per line, until you have finished copying all the files you want to copy. You end the PIP session by hitting a carriage return when PIP prints its * prompt message.

2. Invoking PIP with a command string, by entering:

   PIP A:NEW.DTA=B:OLD.DTA <CR>

   In this use, PIP is loaded and copies the file B:OLD.DTA to the new file A:NEW.DTA. After the copying is complete, PIP reboots CP/M and returns control to CP/M.

PIP can also copy from device to device. For this type of operation, you can use any of the devices shown in Table 5.11. PIP also uses some "devices" to perform special operations. These are shown in Table 5.12.

You can use PIP to copy the contents of several files to one file (concatenate several files). You do this by specifying the source filenames, separated by commas. For example, to copy files FILE1.DTA, FILE2.DTA, and FILE3.DTA to the single file ALLDATA.BAK, you use the command:

   PIP ALLDATA.BAK=FILE1.DTA,FILE2.DTA,FILE3.DTA

In the above example, the entire contents of FILE1.DTA are copied to ALLDATA.BAK. Next, PIP copies the entire con-
tents of FILE2.DTA to ALLDATA.BAK, beginning the copy at
the end of the current contents of ALLDATA.BAK (the end
of the copied FILE1.DTA). FILE3.DTA is then copied at the
end of the FILE2.DTA data in ALLDATA.BAK.

NOTE: Be careful when concatenating ASCII files. ASCII files end with a
^Z (CTRL-Z) that PIP copies, along with the data, into your output
file. This produces a file with multiple end-of-file markers embedded in
it. Many programs will stop reading the file at the first ^Z.

Table 5.10 PIP Command Parameters

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dn</td>
<td>Delete all characters after the nth column. Use this when you want to send data to your printer and the data are longer than your printer's carriage. You get only the first n characters.</td>
</tr>
<tr>
<td>E</td>
<td>Echo the characters to the console during the copy operation.</td>
</tr>
<tr>
<td>F</td>
<td>Remove form feed characters during the copy operation. For feed characters are ASCII value 0CH or CTRL-L (^L).</td>
</tr>
<tr>
<td>Gn</td>
<td>Get the file from a different user area. The n can be any decimal integer between 0 and 15.</td>
</tr>
<tr>
<td>H</td>
<td>Check the files for correct Intel Hexadecimal format records.</td>
</tr>
<tr>
<td>I</td>
<td>Ignore any null records when transferring Intel Hexadecimal records. Null records are those that contain only 00H.</td>
</tr>
<tr>
<td>COMMAND</td>
<td>FUNCTION</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>L</td>
<td>Convert all upper-case letters to lower-case letters during the copy operation. Only the letters A-Z are converted to a-z. All other characters are unchanged.</td>
</tr>
<tr>
<td>N</td>
<td>Append a line number to the beginning of each copied line. A line is a record that ends in an ASCII CR/LF (carriage return/line feed), which you usually insert when you press the <strong>RETURN</strong> key. The line numbers begin at one (1) and are incremented by one (1).</td>
</tr>
<tr>
<td>O</td>
<td>Copy object files and non-ASCII files. Treat the <strong>CTRL</strong> -Z (<strong>Z</strong>; end-of-file marker) as any other character.</td>
</tr>
<tr>
<td>Pn</td>
<td>Add a page feed (form feed) every $n$ lines copied. The ASCII form feed character is <strong>CTRL</strong> -L (<strong>L</strong>) or 0CH. You use this when you are copying from a file to your printer.</td>
</tr>
<tr>
<td>Qs ^ Z</td>
<td>Copy only a section of the file. Stop the copy operation when PIP finds the string $s$. The <strong>CTRL</strong> -Z (<strong>Z</strong>) marks the end of the string to be found. The characters in string $s$ are converted to upper case only when you specify the destination and source parameters when you invoke PIP. The conversion to upper case is not done when you load PIP into memory and enter several commands to PIP's prompt of <code>*</code>.</td>
</tr>
<tr>
<td>COMMAND</td>
<td>FUNCTION</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>R</td>
<td>Copy system files. System files have the SYS attribute.</td>
</tr>
<tr>
<td>Ss ^Z</td>
<td>Copy only a section of the file beginning with the first occurrence of the string s. The &lt;CTRL&gt; -Z (^Z) marks the end of the string s. See the description of lower- to upper-case conversion for the s string in the 9 command description.</td>
</tr>
<tr>
<td>Tn</td>
<td>Set tab stops at every n column. This is useful when you are sending output to your printer from a file. The ASCII tab character is 09H or &lt;CTRL&gt; -I (^ I)</td>
</tr>
<tr>
<td>V</td>
<td>Verify the copy operation by comparing the source and destination files after the copy is complete.</td>
</tr>
<tr>
<td>W</td>
<td>Override the read only attribute and copy into a read only (R/O) file.</td>
</tr>
<tr>
<td>Z</td>
<td>Zero the parity bit (8th bit) on ASCII characters.</td>
</tr>
</tbody>
</table>

Examples:

**PIP A:FIRST.DTA=B:TEST.DTA**

Copy the file from disk B called TEST.DTA to the file on disk A called FIRST.DTA.

**PIP B:=A::*.***

Copy all files from disk A to disk B.
PIP CHAPT1.BAK=CHAPT.ONE
Copy the file CHAPT.ONE to the file CHAPT1.BAK. Both files are on the same disk.

PIP CON:=TEST.DTA
Print the file TEST.DTA on the console.

PIP B:BACKUP.PGM=A:PROG234.COM[R]
Copy the system file PROG234.COM on disk A to BACKUP.PGM on disk B.

PIP X.Y=A.B,C.D
Copy the two files A.B and C.D to the file X.Y.

PIP
*B:=A:SYSFILE.XXX[R]
*A:=B:WORDPROG.COM
*B:=A::* .BAK
*<CR>
Copy several files. First, copy the system file SYSFILE.XXX from disk A to disk B. Then copy the program WORDPROG.COM to disk A. Finally, copy all files that have the type BAK from disk A to disk B.

Table 5.11 PIP Logical Devices

<table>
<thead>
<tr>
<th>NAME</th>
<th>DEVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CON:</td>
<td>Console display as PIP output. Keyboard as PIP input.</td>
</tr>
<tr>
<td>LST:</td>
<td>The CP/M list device (printer) for PIP output.</td>
</tr>
<tr>
<td>PRN:</td>
<td>A special form of the CP/M LST device. PRN handles tabs, determines page breaks, and number lines.</td>
</tr>
</tbody>
</table>
Table 5.12 Special PIP Devices

<table>
<thead>
<tr>
<th>NAME</th>
<th>DEVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL:</td>
<td>Send 40 null characters (ASCII value is zero) to the file or device.</td>
</tr>
<tr>
<td>EOF:</td>
<td>Send an end-of-file mark (ASCII value is 1AH) or ^Z (CTRL-Z) to the ASCII (not binary) file or device.</td>
</tr>
</tbody>
</table>

5.5.12 REN

Format:  REN

Format:  REN[dsk-td:]new-file=old-file

where:

dsk-td is an optional disk identifier.

new-file is the new filename. This must be a valid CP/M filename of the form filename[type].

old-file is the current filename. This must be a valid CP/M filename of the form filename[type].

Description:

You use a REN command to change the name of an existing file. The current filename old-file is changed to the new filename new-file. You cannot use the wildcard form of a CP/M filename when you use the REN command. You must specify a valid CP/M filename, but you can specify a blank type.

If you are renaming a file that is on the currently logged disk, you don’t need to specify the dsk-td parameter. You cannot specify two dsk-td parameters. REN changes the name of the file on the same disk on which the file resides; it does not copy the file to another disk. If you want to change the filename and also move the file to another disk, use the PIP command.

Examples:

REN A:PRODPGM.COM=TESTPGM.COM

Change the name of the file
TESTPGM.COM on disk A to PRODPGM.COM.

REN DATA.ARC=DATA.182
Change the name of the file DATA.182 on the currently logged disk to DATA.ARC.

REN B:DATAFILE=TEST.DTA
Change the name of the file TEST.DTA on disk B to DATAFILE.

5.5.13 SAVE

Format: SAVE page-num [disk-id:]filename[.type]

where:

page-num is the number of 256-byte pages from the TPA to save to the specified file.

disk-id is an optional disk identifier.

filename.type is the name of the file to which CP/M will write the page-num*256 bytes.

Description:

You use a SAVE command to save page-num pages (where 1 page = 256K bytes) to the specified file. CP/M copies the information from the TPA which begins at location 100H. You also use the SAVE command when you use the MOVCMPM command to create a new version of CP/M.

You must calculate the number of pages to be saved by dividing the amount of data by 256. You can use DDT to determine the size of your program. When you load a program into the TPA using DDT, DDT will tell you the size of the loaded data. Then, calculate the number of 256-byte pages that this represents.

For example, if you want to save the information from location 100H through 4FFH into the file NEWPGM.COM, you would use the command:
SAVE 4 NEWPGM.COM

You use the disk-id parameter when you want to save the information to a disk that is not the currently logged disk.

Examples:

SAVE 1 A:B
Save the contents of memory locations 100H through 1FFH to the file A:B.

SAVE 10 B:PGM.TST
Save the contents of memory locations 100H through AFFH to the file PGM.TST on disk B.

SAVE 5X
Save the contents of memory locations 100H through 5FFH to the file X on teh currently logged disk.

5.5.14 STAT

Format: STAT
or
STAT command

where:

command is a valid STAT command as described below.

Description:

You use a STAT command to display or change status information for a CP/M disk, file, group of files, device, or user number.

To display status information, you use one of these forms of the STAT command:

• STAT [disk-id:]

This shows the number of bytes remaining on disk disk-id. If you omit disk-id, STAT provides the in-
formation on the currently logged disk. The STAT message is (see Table 5.13 for the valid options):

\[ disk-id: \text{Option, Space: nnK} \]

- **STAT \[\text{disk-id:}]\text{DSK}**:  
  This shows the drive characteristics for disk \text{disk-id}. If you omit \text{disk-id}, STAT provides information related to the currently logged disk. The STAT information is:

  \begin{align*}
  \text{disk-id:} & \quad \text{Drive Characteristics} \\
  1088: & \quad 128 \text{ Byte Record Capacity} \\
  136: & \quad \text{Kilobyte Drive Capacity} \\
  64: & \quad 32 \text{ Byte Directory Entries} \\
  64: & \quad \text{Checked Directory Entries} \\
  128: & \quad \text{Records / Extent} \\
  8: & \quad \text{Records / Block} \\
  34: & \quad \text{Sectors / Track} \\
  2: & \quad \text{Reserved Tracks}
  \end{align*}

- **STAT \[\text{disk-id:}]\text{filename[.type]}**

  This shows the characteristics of the file(s) specified. You can use the wildcard (*) notation for the \text{filename} and/or \text{type} parameters. If you don't specify a \text{disk-id} parameter, STAT uses the currently logged disk.

  The STAT information for the specified file(s) is shown as:

  \begin{center}
  \begin{tabular}{llll}
  Recs & Bytes & Ext & Acc \\
  \hline
  nnn & nK & e & Options disk-id:filename.type \\
  \end{tabular}
  \end{center}

  ...for each file specified...

  Bytes Remaining on \text{disk-id: nnK}

where:

- \text{nnn} is the number of 128-byte records for the file.
- \text{nK} shows the file size in 1024-byte blocks.
- \text{e} shows the number of extents used for the file.
Options shows a valid STAT option from Table 5.13. disk-id:filename.type shows the filename.

If you specify a file which is not on the disk, STAT returns an error message:

FILE NOT FOUND

• STAT {DEV: | VAL: | USR:}

This shows the information for the CP/M devices (DEV:), STAT commands and external peripheral options (VAL:), or user numbers (USR:). This function refers to the I/O byte, which is not implemented and always returns the default device assignments.

<table>
<thead>
<tr>
<th>OPTION</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSK:</td>
<td>Show the characteristics of the specified drive.</td>
</tr>
<tr>
<td>DEV:</td>
<td>Show the characteristics of the CP/M system devices.</td>
</tr>
<tr>
<td>USR:</td>
<td>Show the files related to each USER number on the specified disk.</td>
</tr>
<tr>
<td>VAL:</td>
<td>Show the possible STAT commands and devices.</td>
</tr>
</tbody>
</table>

---

NOTE: The DEV- and VAL- options refer to the I/O byte, which is not implemented in the Commodore 64 BIOS.

To change status information, you use one of these forms of the STAT command (valid STAT attributes are shown in Table 5.14):

• STAT disk-id:=R/O

This changes the disk disk-id to a temporary read only mode (R/O).
- STAT \[\text{disk-id:}]\text{filename[.type]}=\#x\]
  where \(x\) is \{R/O \mid R/W \mid SYS \mid DIR\}

This changes the specified file(s) to read only (R/O), read/write (R/W), system (SYS), or nonsystem (DIR). You can use the wildcard (*) notation for the filename and/or type parameters. To change all your program files on disk A to read only, you enter the command:

\[
\text{STAT A:*\text{.COM}} \ #R/O
\]

**Table 5.4 STAT Command Attributes**

<table>
<thead>
<tr>
<th>ATTRIBUTE</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIR</td>
<td>Set the non-SYSTEM attribute for the file(s).</td>
</tr>
<tr>
<td>R/O</td>
<td>Set the file or disk to read only.</td>
</tr>
<tr>
<td>R/W</td>
<td>Set the file to read/write.</td>
</tr>
<tr>
<td>S</td>
<td>Show the size(s) of the file(s) based on the file last record number(s).</td>
</tr>
<tr>
<td>SYS</td>
<td>Set the SYSTEM attribute for the file(s).</td>
</tr>
</tbody>
</table>

Examples:

- \(\text{STAT *.*}\)  
  Show the statistical information for all files on the currently logged disk.

- \(\text{STAT A.B}\)  
  Show the statistical information for the file A.B on the currently logged disk.

- \(\text{STAT DSK:}\)  
  Show the statistical information for the currently logged disk.

- \(\text{STAT *\text{.COM}} \ #R/O\)  
  Set all files on the currently logged disk which have a type COM (CP/M program files) to read only.

- \(\text{STAT NEW.DTA} \ #R/W\)  
  Set the file NEW.DTA to read/write.
5.5.15 SUBMIT

Format:  SUBMIT [disk-id:]filename [parameters]

where:

  disk-id is an optional disk identifier.

  filename is the name of the file containing the CP/M commands. This file must be named filename.SUB.

  parameters are optional parameters passed to the SUBMIT commands.

Description:

You use a SUBMIT command to send a group of commands to CP/M for execution. SUBMIT makes your Commodore 64 operate in batch mode where, with a single command, you can execute any number of programs or utilities.

The file containing the commands must have a type SUB. This file can contain any CP/M commands. CP/M creates a file called $$$ SUB as a temporary work file when you execute a SUBMIT command.

________________________________________________________________________________

NOTE: All commands in a SUBMIT file must be in upper case.

________________________________________________________________________________

For example, you could have these commands in file DISK DTA.SUB:

  DIR
  STAT *. *
  ERA *.BAK
  STAT DSK:

To execute all four of these CP/M commands, you simply enter:

  SUBMIT DISKDTA <CR>
Remember, CP/M then executes the commands in the file in the order in which the commands appear in the file. SUBMIT processing only executes commands. It does not pass any information to the programs it executes. If you want to pass data to the programs, use the XSUB command.

You can chain from one .SUB file to another. Whenever a SUB file finds another SUBMIT command, the first file is stored and the second file becomes active. When the second file's commands are finished, the first .SUB file becomes active at the command following the SUBMIT command. For example, you could have these two files:

File A.SUB contains:

```
STAT DSK:
SUBMIT B
STAT DSK:
```

File B.SUB contains:

```
ERA *.BAK
DIR
```

When you enter the command:

```
SUBMIT A
```

the following commands are executed:

```
STAT DSK:
ERA *.BAK
DIR
STAT DSK:
```

You can also pass parameters to the .SUB file. The parameters are sequentially numbered in the file and have the form:

```
$\$n
```
where:
   $n$ starts at 1 and is incremented by 1.

The parameters can be any information required by the commands in your .SUB file. They can be filenames, disk id's, file types, or anything that you need. SUBMIT does a straight substitution of the parameter values for the parameter indicators ($n$) in the .SUB file before passing the commands to CP/M. The first parameter goes to all occurrences of $1$; the second to $2$, etc.

Suppose you want to check the status of your disk and then edit a file. You could have a file called DSKEDIT.SUB that contains this information:

```
STA $1:DSK:
ED $2.$3
STAT $1:$2.$3
```

Then, to check the status of Disk A and edit the file MY.DTA, you would use this submit command:

```
SUBMIT DSKEDIT A MY DTA
```

SUBMIT processing replaces the parameter indicators with the values in your SUBMIT command and the data in file. When passed to CP/M for processing, DSKEDIT.SUB looks like this:

```
STAT A:DSK:
ED MY.DTA
STAT A:MY.DTA
```

When you are using SUBMIT parameters, you can enter these special characters through the parameter string:

- To enter a $ as data, you must enter two consecutive $$$. This is transferred to the command line as a $. Thus, to enter the value "$XY" as a parameter, you must use $$$XY$.
- To enter a control character, use the up-arrow symbol (^) followed by the control character. To enter 
  
CTRL ·X, you would enter the character string ^X.
You can have a SUBMIT command as the last command in a .SUB file. This lets you chain from one .SUB command file to another.

Examples:

SUBMIT STARTUP   This executes the CP/M commands in the file called STARTUP.SUB.

SUBMIT NEW A B   This executes the CP/M commands in the file called NEW.SUB. The value “A” is passed to any $1 indicators in the file. The value “B” is passed to any $2 indicators.

5.5.16 SYSGEN

Format:  SYSGEN [disk-id:filename.type]

where:

disk-id is an optional disk identifier.

filename.type is the name of the file that will contain the new copy of the system.

Description:

You use a SYSGEN command to create a new copy of your CP/M operating system. The CP/M system is stored on special tracks called the system tracks (tracks 0 and 1). These tracks never appear in the file directory listing and you cannot read or write to these tracks as part of processing any normal program.

You need the system tracks on any disk from which you may do a warm or cold start. It’s a good idea to have a copy of the system on most disks that contain programs. Whenever you enter a \[CTRL].(^C), CP/M reloads part of its system tracks (the BEOS and CCP) in a warm start.

You use the SYSGEN command to copy these tracks from one disk to another or to create a new copy of the system after you have used a MOVCPM command.

You use a SYSGEN command in one of these three ways:
1. To copy your CP/M system from one disk to another. You do not make any changes to the system; you simply copy it.

2. You use MOVCPM to create a different sized version of CP/M and you use SYSGEN to copy it to a disk.

3. You use DDT to make special changes to your copy of CP/M and you use SYSGEN to write the system to a disk.

SYSGEN does not destroy any information currently on the user area of a disk. SYSGEN simply writes a new copy of the CP/M system on the disk.

If you specify a disk-td parameter, SYSGEN does not ask for the source drive but uses the value you selected for disk-td.

If you want to create a new copy of CP/M after using MOVCPM to create a new version, you follow this procedure. The text that you enter is shown in boldface. The messages from CP/M are shown in italics.

**SYSGEN <CR>**

*COMMODORE 64 SYSGEN VERSION 2.0*

*SOURCE DRIVE NAME*

*(OR RETURN TO SKIP) <CR>*

*DESTINATION DRIVE NAME*

*(OR RETURN TO SKIP) B<CR>*

*DESTINATION ON B, THEN TYPE RETURN <CR>*

*FUNCTION COMPLETE*

To copy a version of CP/M from one disk to another, follow the above procedure but supply the appropriate answers for the source and destination drives.

---

**NOTE:** If you SYSGEN onto your current system disk a version of CP/M that is a different size from the one you're running, you CANNOT warm start the system. The location of operating system components will not match and the CP/M will crash.
Example:
To copy the system tracks from your current disk to another disk, enter:

```
SYSGEN <CR>
```

and answer the questions that CP/M asks.

### 5.5.17 TYPE

**Format:**  
```
TYPE [disk-id:]filename.type
```

where:

- `disk-id` is an optional disk identifier.
- `filename.type` is the name of the file to be listed on your screen.

**Description:**
You use a `TYPE` command to list an ASCII format file on your screen. If you don't specify a `disk-id` value, CP/M uses the currently logged disk. You must specify a valid CP/M filename. `TYPE` does not accept the wildcard (*) notation.

You can use a `CTRL -P (^P)` before you enter your `TYPE` command and the listing will appear on your screen and on your printer. All commands and data continue to appear on both the screen and the printer until you enter another `^P`.

You can stop the `TYPE` listing by pressing any key. You can `temporarily stop` the listing by pressing a `CTRL -s (^S)`; you restart the listing by pressing any key.

Remember that `TYPE` displays the contents of the specified file, assuming that the file contains ASCII characters. If you `TYPE` a program file (.COM), you will see garbage on your screen. Be sure that you are listing a text file when you use `TYPE`.

**Examples:**

- `TYPE A:BILLS.LST` List the contents of the file on disk A called `BILLS.LST`. 

102  

**CP/M OPERATION**
**TYPE X**
List the contents of the file called X on the currently logged disk.

### 5.5.18 USER

**Format:** `USER [user-num]`

where:

*user-num* is a decimal integer between 0 and 15.

**Description:**

You use a `USER` command to display and change the current user number. CP/M assumes a default user number of zero (0).

Once you change the user number, you can access only those files associated with the new user number. You can always enter a user number 0 to return to the default setup.

To *display* the current user number enter:

```
USER <CR>
```

To *change* the current user number to 5 enter:

```
USER 5
```

You should not change the user number unless you want to protect certain files from use by those who do not know the associated user number. In a single-user CP/M system, it's generally unnecessary to change the user number.

**Examples:**

- `USER 2` Change the user number to 2.
- `USER` Display the current user number.
5.5.19 XSUB

Format: XSUB

Description:
You use an XSUB command when you want to enter more than commands in a .SUB file. XSUB is a subset of SUBMIT processing and CANNOT be entered as a response to the CP/M prompt. XSUB may appear only in a SUBMIT (.SUB) file. Read the description of the SUBMIT command for full details on how .SUB files are processed.

XSUB must be the first command in your .SUB file. You can enter parameters on an XSUB command in the same way as for a SUBMIT command.

XSUB allows you to enter data that would normally be entered through the keyboard for some programs. If you are using a program that accepts buffered console input (uses BDOS function 10), then the program will accept the answers from the XSUB file instead of waiting for you to enter data from the keyboard. Not all programs do this, but all the CP/M utilities and commands do accept data in this manner.

Example:
You want to submit a file that will run DDT and load the file you specify. Your file called DDTRUN.SUB contains:

```
XSUB
DDT
I$1.$2
R
```

You can submit this file and specify that the file WORDPROC.DTA be read into memory through DDT by entering:

```
SUBMIT DDTRUN WORDPROC DTA
```
This SUBMIT command accepts the DDT commands to read the file WORDPROC.DTA into memory by processing the information after the XSUB command.
CHAPTER 6

CP/M ON THE COMMODORE 64

- The Structure of CP/M
- The BOOT Programs
- The BIOS Programs
- CP/M Disk Organization
- The CP/M BDOS
- Calling a Z80 Program from the 6510
- Calling a 6510 Program from the Z80
- Program Execution under CP/M

- Z80 Schematic
- Commodore 64 Schematic
In this chapter, you will find technical information about implementing CP/M on your Commodore 64. You will need this information only if you intend to make changes or additions to CP/M as supplied with your Commodore 64 and its Z80 cartridge.

CP/M was one of the first microcomputer operating systems designed to run on machines of more than one manufacturer. It is written in Intel 8080 Assembler language. The Z80 add-on processor on your Commodore 64 executes a superset of the 8080 machine language. Any program written for the 8080 processor will run on the Z80, but the reverse may not be true.

When CP/M is running on your Commodore 64, the 6510 main processor and the Z80 add-on processor are alternately active. The two processors trade control of the computer according to what operations are required. Because device drivers already reside in your Commodore 64 operating system, all input and output is performed by the 6510. The Z80 runs only the CP/M operating system, its utilities, and applications.

In addition to the standard functions required by the CP/M operating system, you can access your own special purpose routines running in 6510 native mode. This is useful, for example, if you want to attach an instrument to the optional IEEE interface cartridge on your Commodore 64. You could then easily code a driver for the instrument and gain access to it through a well defined, and protected, interface.

6.1 THE STRUCTURE OF CP/M

The principal component of CP/M is the Basic Disk Operating System (BDOS). All requests for operating system services — disk input/output, printer output, screen output — are carried out through a set of standard calls to the BDOS.

---

**NOTE:** It is possible to call entry points in the CP/M BIOS directly. This technique is NOT recommended unless you are very sure of what you are doing. **WARNING:** Direct BIOS calls may be incompatible with future CP/M releases.

---

108 CP/M ON THE COMMODORE 64
A second major component of CP/M is the Console Command Processor (CCP). The CCP analyzes and interprets the commands that you enter from the keyboard, initiating whatever action you request. Of the resident CP/M system, the CCP occupies the lowest memory areas (see Figure 6.3).

Transient programs (those not a permanent part of the BDOS) are loaded into the Transient Program Area (TPA) and may, if they need the space, overlay the CCP when executing.

If a program executing in the TPA does overlay the CCP, the CCP must be reloaded when the transient program terminates. You will see this CCP reload operation (a "warm boot") as a line of asterisks appearing on your screen after a program has finished.

The final major component of CP/M is the Basic Input/Output System (BIOS). This has nothing to do with the BASIC language. The BIOS is the component of CP/M that allows CP/M to be run on a variety of machines. The BIOS forms a bridge between the BDOS and the individual characteristics of the machine that it runs on. Each machine has a specially tailored BIOS that supports the hardware and peripherals attached to it.

The CP/M BIOS is much like the CBM Kernal in your Commodore 64. Like the Kernal, the BIOS contains a set of standard routines that give you access to hardware functions.

Your Commodore 64 has a unique BIOS that provides easy access to the standard Commodore 64 peripherals, either serial or IEEE.

### 6.1.1 How CP/M Works on Your Commodore 64

Four specially tailored assembly language programs and the CP/M operating system are required to run CP/M on your Commodore 64. Two of the assembly language programs run under the 6510 microprocessor and two under the Z80 microprocessor:

- 6510 CP/M BOOT program (BOOT65)
- Z80 CP/M BOOT program (BOOT80)
• 6510 BIOS (BIOS65)
• Z80 BIOS (BIOS80)

The BOOT programs "bootstrap" CP/M. That is, they load it into memory, initialize some areas, and begin its execution. Once the BOOT programs have completed their tasks, they are no longer needed and the memory they occupied is used for other purposes.

CP/M comes from Digital Research as a core operating system. It needs an add-on software component called a BIOS (Basic Input/Output System). The BIOS contains a set of entry points that perform specific "primitive" tasks for CP/M, such as:

• Set the track number for the next read or write operation.
• Write a character to the printer.
• Read a character from the keyboard.

CP/M is not concerned with how these tasks are performed. All this work is taken care of in the custom BIOS written specifically to support a certain hardware environment. It is this BIOS that allows CP/M to run many different machines equipped with many different peripherals.

On your Commodore 64, the CP/M BIOS is in two parts. One part runs under the Z80 add-on processor (BIOS80) and the other under the 6510 Commodore 64 main processor (BIOS65). This arrangement allows the 6510 to serve as an input/output processor for the Z80, handling all disk, printer, keyboard, and screen input or output.

The 6510 part of the BIOS initiates execution of CP/M under the Z80 processor by transferring control to the Z80 BOOT program, which loads CP/M and BIOS80. Whenever a processor is switched on, it resumes execution at the instruction immediately following the instruction that switched it off. This means that when the Z80 returns control to the 6510, execution will resume within BIOS65.

When a CP/M program, running on the Z80, requests an input/output operation, the Z80 BIOS places a function code and any required parameter values at predetermined locations in memory. Remember, memory is shared between the two processors, which makes it very easy for them to pass data back and forth.
Once these parameter values are in place, BIOS80 switches the Z80 out and the 6510 in. The 6510 resumes execution in the 6510 portion of the BIOS. BIOS65 examines the function code passed to it by BIOS80 and initiates the indicated action.

Once the 6510 has completed the action, BIOS65 places return values and/or flag values into predetermined locations and switches control back to the Z80 processor.

Under the Z80 processor, execution resumes where it left off in BIOS80. BIOS80 examines the shared memory areas to determine the success or failure of the requested function and carries out any other action necessary to complete the function.

### 6.1.2 6510 Memory Use

Figure 6.1 shows the memory allocation as seen from the 6510 running in native mode. Figure 6.2 shows details on the BIOS65 memory area.

#### 6510 CP/M Memory Map

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$FFFF</td>
<td></td>
</tr>
<tr>
<td>$F000</td>
<td>6510 KERNAL ROM</td>
</tr>
<tr>
<td>$E000</td>
<td>6510 I/O SYSTEM</td>
</tr>
<tr>
<td>$D000</td>
<td>48K RAM AVAILABLE FOR Z80</td>
</tr>
<tr>
<td></td>
<td>RUNNING CP/M</td>
</tr>
<tr>
<td>$1000</td>
<td>BIOS65 AND SHARED DATA AREAS</td>
</tr>
<tr>
<td>$0800</td>
<td>0400 TO 07FF SCREEN RAM</td>
</tr>
<tr>
<td></td>
<td>0000 TO 03FF ZERO PAGE AND 6510 STACK</td>
</tr>
</tbody>
</table>
The addresses shown are for the 6510 microprocessor. For Z80 addresses, subtract $1000 hexadecimal from the addresses shown (see Section 6.1.3 for an explanation of Z80/6510 address conversion).

---

**NOTE:** If you add the IEEE interface cartridge to your Commodore 64 system, you can run only a 44K version of CP/M. The top 4K ($C000—$D000) of the CP/M 48K area is used to handle the IEEE interface cartridge.

---

### BIOS65 Memory Map

#### 6510 ADDRESS

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1000</td>
<td></td>
</tr>
<tr>
<td>$0F00</td>
<td></td>
</tr>
<tr>
<td>$0E00</td>
<td></td>
</tr>
<tr>
<td>$0D00</td>
<td>BIOS65</td>
</tr>
<tr>
<td>$0C00</td>
<td></td>
</tr>
<tr>
<td>$0B00</td>
<td></td>
</tr>
<tr>
<td>$0A00</td>
<td>SHARED DATA</td>
</tr>
<tr>
<td>$0900</td>
<td></td>
</tr>
<tr>
<td>$0800</td>
<td>DISK I/O BUFFER</td>
</tr>
</tbody>
</table>

The addresses shown are for the 6510 microprocessor. For Z80 addresses, add $F000 hexadecimal to the addresses shown (see Section 6.1.3 for an explanation of Z80/6510 address conversion).
6.1.3 Addresses under CP/M

You can see from the memory map in Figure 6.3 that the Z80 processor uses the memory between $1000 and $BFFF—a 48K byte area. CP/M, however, makes use of fixed areas in the zero page ($0000–$0100) of memory. This area is also required by the Commodore 64 operating system.

To avoid a conflict in the use of the zero page and to provide space for BIOS65, all Z80 addresses have $1000 added to them. Thus, the Z80 address $0000 becomes actual address $1000. Table 6.1 shows the mapping between Z80 addresses and actual memory addresses.

---

**NOTE:** If you are using the optional IEEE interface cartridge, you have only 44K bytes available for CP/M. The IEEE bus access routines require an additional 4K at the high end of the CP/M memory ($B000–$BFFF).

---

**Table 6.1 Z80 to 6510 Actual Address Mapping**

<table>
<thead>
<tr>
<th>Z80 ADDRESS</th>
<th>ACTUAL (6510) ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000–&gt;0FFF</td>
<td>1000–&gt;1FFF</td>
</tr>
<tr>
<td>1000–&gt;1FFF</td>
<td>2000–&gt;2FFF</td>
</tr>
<tr>
<td>2000–&gt;2FFF</td>
<td>3000–&gt;3FFF</td>
</tr>
<tr>
<td>3000–&gt;3FFF</td>
<td>4000–&gt;4FFF</td>
</tr>
<tr>
<td>4000–&gt;4FFF</td>
<td>5000–&gt;5FFF</td>
</tr>
<tr>
<td>5000–&gt;5FFF</td>
<td>6000–&gt;6FFF</td>
</tr>
<tr>
<td>6000–&gt;6FFF</td>
<td>7000–&gt;7FFF</td>
</tr>
<tr>
<td>7000–&gt;7FFF</td>
<td>8000–&gt;8FFF</td>
</tr>
<tr>
<td>8000–&gt;8FFF</td>
<td>9000–&gt;9FFF</td>
</tr>
<tr>
<td>9000–&gt;9FFF</td>
<td>A000–&gt;AFFF</td>
</tr>
<tr>
<td>A000–&gt;AFFF</td>
<td>B000–&gt;BFFF</td>
</tr>
<tr>
<td>B000–&gt;BFFF</td>
<td>C000–&gt;CFFF</td>
</tr>
<tr>
<td>C000–&gt;CFFF</td>
<td>D000–&gt;DFFF</td>
</tr>
<tr>
<td>D000–&gt;DFFF</td>
<td>E000–&gt;EFFF</td>
</tr>
<tr>
<td>E000–&gt;EFFF</td>
<td>F000–&gt;FFFF</td>
</tr>
<tr>
<td>F000–&gt;FFFF</td>
<td>0000–&gt;0FFF</td>
</tr>
</tbody>
</table>

---

**NOTE:** Notice that to access the 6510 low addresses, you reference the Z80 high addresses.
6.1.4 Z80 Memory Use

The amount of memory available to CP/M on your Commodore 64 depends on your hardware configuration. If you are using the standard *Commodore 64 serial disk drives and printer*, CP/M can occupy a maximum of 48K bytes. If you have acquired the *IEEE interface cartridge*, CP/M can occupy a maximum of 44K bytes. The IEEE interface cartridge consumes 4K at the high end of the CP/M address space (see Figure 6.1).

You can, of course, generate a CP/M system that is smaller than the maximum available space. You can do that if you need space for a routine that must run in Commodore 64 native mode (under the 6510 processor). You can, for example, generate a 40K CP/M version and have 8K (or 4K if you have the IEEE cartridge) available for your Commodore 64 native mode routine. Figure 6.3 shows a diagram of the Z80 address space.

---

**Z80 Memory Map**

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>44K</th>
<th>48K</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AFFF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$AA00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$9C06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$9400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$0000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*BIOS80*  
*BDOS*  
*CCP*  
*TPA*  
(44K — 33,792 bytes)  
(48K — 37,888 bytes)  
*ZERO PAGE*

Many microcomputer operating systems use the zero page of memory (addresses between $0000 and $0100) to hold important values. Both CP/M and your Commodore 64
operating system do this. Table 6.4 shows the contents of the CP/M Zero Page.

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0000 – $0003</td>
<td>Contains a jump instruction to the warm start entry point in the BIOS.</td>
</tr>
<tr>
<td>$0004</td>
<td>Contains the current default disk drive number (0=A and 1=B) in the low order 4 bits and the I/O byte in the high order 4 bits.</td>
</tr>
<tr>
<td>$0005 – $0007</td>
<td>Contains a jump instruction to the BDOS main entry point. The value stored in locations $0006 – $0007 is the lowest address required by CP/M. You also use this jump instruction (or the address) when you make direct BDOS calls.</td>
</tr>
<tr>
<td>$0038 – $003A</td>
<td>This is Restart Location 7 and is used by DDT for programmed breakpoints (an RST 7 instruction causes a call to this location).</td>
</tr>
<tr>
<td>$005C – $006C</td>
<td>This is the first default file control block for use by transient programs.</td>
</tr>
<tr>
<td>$006C – $007C</td>
<td>This is the second default file control block for use by transient programs.</td>
</tr>
</tbody>
</table>
Table 6.2 (Continued)

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$007D - $007F</td>
<td>This location contains the random record position for random file access via the first default file control block.</td>
</tr>
<tr>
<td>$0080 - $00FF</td>
<td>This is the default 128-byte disk input/output buffer. This area also receives the command line that you enter when your program is loaded by the CCP.</td>
</tr>
</tbody>
</table>

**NOTE:** The areas of the zero page not shown in this table are reserved for future use. You should not use any of these areas in programs you write unless you are sure of their use.

### 6.2 THE BOOT PROGRAMS

The BOOT programs—BOOT65 and BOOT80—are used to load CP/M from disk. Once they have completed this task, the memory they occupy is used for other purposes.

The BOOT65 program is in the file called "CP/M" that you LOAD and RUN to start execution of the CP/M operating system on your Commodore 64. You can find a listing of this program in Appendix E. The actual assembly language program source is available on one of your CP/M system diskettes.

You LOAD and RUN BOOT65 as you would any BASIC program on your Commodore 64. If you LIST it, you will see that it contains a single BASIC statement:

```
10 SYS (2036)
```

This statement transfers control to the actual BOOT65 code located at decimal address 2036.

The program then reads in the BIOS65 and BOOT80 pro-
grams and places them at the correct locations in memory. Finally, BOOT65 transfers control to the startup code in BIOS65.

The **BOOT80** program is a Z80 assembly language program that is the first program to execute when the Z80 processor is switched on. You can find a listing of this program in Appendix E. The actual assembly language program source is available on one of your CP/M system diskettes.

BOOT80 is loaded by the BOOT65 program at the Z80 reset address $0000 (6510 address $1000). When the Z80 is first turned on, it always begins execution at address $0000.

BOOT80 loads:

- Z80 BIOS (BIOS80)
- CP/M CCP (CP/M Command Processor)
- CP/M BDOS (Basic Disk Operating System)

When these programs are loaded, BOOT80 transfers control to the cold start entry point in BIOS80, thus beginning actual CP/M operating system execution.

### 6.3 THE BIOS PROGRAMS

The BIOS (Basic Input/Output System) is the specially tailored link between the CP/M operating system and the individual peripherals—printer, disk drives, screen—attached to your Commodore 64.

Each computer that runs CP/M has its own unique BIOS. On your Commodore 64 the BIOS is in two parts:

- BIOS65 executes under the 6510 main processor.
- BIOS80 executes under the Z80 add-on processor.

These two portions of the BIOS operate together to make your Commodore 64 peripherals available to CP/M.

Why are there two programs for the BIOS? Your Commodore 64 already has code in place to handle its peripherals. Thus more memory is made available for CP/M and your CP/M-based applications by simply providing a link to that existing code, rather than trying to re-implement the peripheral-handling code on the Z80.

In operation, BIOS80 is called from CP/M with a request
for an input/output operation. BIOS80 places required parameter values and a function flag in certain memory locations, then switches control from the Z80 back to the 6510 Commodore 64 main processor.

The 6510 resumes execution where it left off in BIOS65. BIOS65 examines the function code stored in memory to find out what it should do, carries out the task (usually an input/output request), places the result in a predetermined memory location, and switches the Z80 back on.

The Z80 resumes execution where it left off in BIOS80. BIOS80 retrieves the results passed to it from BIOS65 and returns the proper information to CP/M.

BIOS80 is called from the CP/M BDOS to perform the following functions:

- cold start boot
- warm start boot
- console (keyboard) status check
- get keyboard character (console input)
- write character to screen (console output)
- print a character (lister output)
- move disk head to the home position
- select disk
- set track to read/write
- set sector to read/write
- read disk sector
- write disk sector
- check printer status (lister status)
- sector translation

The punch and reader functions of the BIOS are meaningless on your Commodore 64. These are null routines in BIOS80.

Some of the functions listed above simply cause values to be placed in predefined memory locations. Others result in a transfer to the 6510 portion of the BIOS where the actual work is performed.

Before BIOS80 switches control back to the 6510, it places a function code at location $F900 ($0900 relative to the 6510). This code, which currently ranges from 0 to 9 and 255, tells BIOS65 what action is required. These function codes and their meanings are shown in Table 6.3.
### Table 6.3 BIOS80/BIOS65 Function Codes

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Read the specified sector</td>
</tr>
<tr>
<td>1</td>
<td>Write the specified sector</td>
</tr>
<tr>
<td>2</td>
<td>Get a character from the keyboard</td>
</tr>
<tr>
<td>3</td>
<td>Write a character to the screen</td>
</tr>
<tr>
<td>4</td>
<td>Check the printer status</td>
</tr>
<tr>
<td>5</td>
<td>Write a character to the printer</td>
</tr>
<tr>
<td>6</td>
<td>Disk format command</td>
</tr>
<tr>
<td>7</td>
<td>Jump to 6510 address $0E00</td>
</tr>
<tr>
<td>8</td>
<td>Jump to 6510 address $0F00</td>
</tr>
<tr>
<td>9</td>
<td>Jump indirect via a 6510 address stored at $F906</td>
</tr>
<tr>
<td>10–&gt;254</td>
<td>Reserved for future use</td>
</tr>
<tr>
<td>255</td>
<td>Execute a cold start reset on your Commodore 64</td>
</tr>
</tbody>
</table>

### Table 6.4 BIOS80/BIOS65 Communication Addresses

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z80</td>
<td>6510</td>
</tr>
<tr>
<td>$F900</td>
<td>$0900                     Command register: contains one of the function codes as shown in Table 6.2.</td>
</tr>
<tr>
<td>$F901</td>
<td>$0901                     Data register: used to pass data and error indicators between the two BIOS.</td>
</tr>
<tr>
<td>$F902</td>
<td>$0902                     Sector register: contains the current sector number for disk read and write requests.</td>
</tr>
<tr>
<td>$F903</td>
<td>$0903                     Track register: contains the current track number for disk read and write requests.</td>
</tr>
<tr>
<td>$F904</td>
<td>$0904                     Drive register: contains the disk drive number for disk read and write requests.</td>
</tr>
<tr>
<td>$F905</td>
<td>$0905                     Keyboard register: contains the last character read from the keyboard.</td>
</tr>
</tbody>
</table>
BIOS65 and BIOS80 communicate with each other through a series of contiguous memory locations as shown in Table 6.4.

### 6.4 CP/M DISK ORGANIZATION

Your Commodore 64 CP/M BIOS programs provide a completely compatible interface between your disks and the CP/M BDOS. All disk-related functions expected by the CP/M BDOS are available through your BIOS programs.

The organization of a CP/M disk is different from the organization of a standard Commodore 64 disk. The CP/M disk has somewhat less capacity than a Commodore 64 format disk.

A Commodore 64 CP/M disk is formatted as 35 tracks containing 17 256-byte sectors (0–16) where *track 1 is the outermost track* and track 35 is the innermost track. A Commodore 64 CP/M disk can hold a maximum of 136,000 characters of *user data*.

Notice that the full disk capacity (152,320 characters) is not available for user data storage.

Table 6.5 shows the allocation of tracks on your Commodore 64 CP/M format disk.

<table>
<thead>
<tr>
<th>TRACK</th>
<th>SECTOR</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>BOOT65 (Commodore 64 file &quot;CPM&quot;)</td>
</tr>
<tr>
<td>1</td>
<td>1–&gt;4</td>
<td>BIOS65</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>BOOT80</td>
</tr>
<tr>
<td>1</td>
<td>6–&gt;13</td>
<td>CP/M CCP (Command Processor)</td>
</tr>
<tr>
<td>1&amp;</td>
<td>14–&gt;16</td>
<td>CP/M BDOS</td>
</tr>
<tr>
<td>2</td>
<td>0–&gt;10</td>
<td>BIOS80</td>
</tr>
<tr>
<td>2</td>
<td>11–&gt;16</td>
<td>CP/M Disk Directory</td>
</tr>
<tr>
<td>3</td>
<td>0–&gt;7</td>
<td>CP/M Disk Space</td>
</tr>
<tr>
<td>3</td>
<td>8–&gt;16</td>
<td>CP/M Disk Space</td>
</tr>
<tr>
<td>4–&gt;17</td>
<td>0–&gt;16</td>
<td>CP/M Disk Space</td>
</tr>
<tr>
<td>18</td>
<td>0–&gt;16</td>
<td>Commodore 64 Directory</td>
</tr>
<tr>
<td>19–35</td>
<td>0–&gt;16</td>
<td>CP/M Disk Space</td>
</tr>
</tbody>
</table>
NOTE: The Commodore 64 Directory written on track 18 allows you to start CP/M from Commodore 64 running in native mode. This directory shows that only a single file—CPM—exists on the disk. The standard Commodore 64 Block Availability Map (BAM) indicates that the disk is completely full.

6.5 THE CP/M BDOS

The CP/M Basic Disk Operating System (BDOS) provides a standard interface between CP/M application programs and the hardware on which they run. All input/output and operating system service requests are routed through the BDOS. Because of this, you don't have to write device-specific code into your application program for every system that it might run on. The device-specific code for a particular system is written only once—in the CP/M BIOS.

The standard BDOS interface means that software can be written and run on any system able to support CP/M, as long as the software developer stays within the BDOS standard.

The 39 BDOS functions (numbered 0–37 and 40 decimal) perform tasks valuable in almost any application. For example, they

- Read a character from the keyboard.
- Write a character to the keyboard.
- Open a disk file.
- Print a string.
- Write to the printer.
- Delete a file.
- Create a file.

For a list of the BDOS functions, see Table 6.6.

You call the BDOS from Z80 Assembler or other languages through the BDOS jump vector at Z80 address $0005. This jump vector contains a single jump instruction:

\[ \text{JMP BDOS-ADDRESS} \]
The `bdos-address` varies with the size of the CP/M system you have generated. The JMP instruction itself is placed at location $0005$ when CP/M is loaded.

To use the BDOS functions, you code:

```
CALL 5
```

When the BDOS has completed the function, it returns control to the statement following the CALL statement.

---

**NOTE:** Bytes 6 and 7 of the BDOS jump vector contain the lowest address required by CP/M (stored as low byte/high byte). This means that your application program can use memory up to, but not including, this address.

---

BDOS functions are numbered. Some require that you pass to them the parameter values or the address of a parameter in certain registers. Some return an indicator or error code in a register.

When calling a BDOS function, you always load the **BDOS function code in register C**. If the function requires that you *pass it parameters*, you place:

- Single-byte parameters in register E.
- Double-byte parameters in register pair DE.

If the function *returns a value* to you, you find:

- Single-byte returns in register A.
- Double-byte returns in register pair HL.

---

**NOTE:** The BDOS does NOT preserve values stored in the Z80 registers. If you want to protect values stored in registers, you should push them onto the stack before you call the BDOS. You can then pop them off the stack on return from the BDOS call.

---

122 CP/M ON THE COMMODORE 64
6.5.1 Sample BDOS Function Call

As an example of a BDOS function call, we will use Function 1, the Console (keyboard) Input function. Function 1 returns in register A the last character entered from the keyboard. To use Function 1, you can write code like the following:

```
MVI C,1 ;LOAD FUNCTION 1 INTO REGISTER C
;      CALL 0005H ;CALL THE BDOS JUMP VECTOR
;      WHEN THE BDOS HAS A CHARACTER, IT RETURNS HERE
;      REGISTER A CONTAINS THE INPUT CHARACTER
;
STA KEYCHAR ;STORE REGISTER A IN KEYCHAR VARIABLE
```

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><strong>SYSTEM RESET</strong></td>
</tr>
<tr>
<td>INPUT: NONE</td>
<td></td>
</tr>
<tr>
<td>RETURN: NONE</td>
<td></td>
</tr>
<tr>
<td>Returns control to the CCP and resets CP/M as though you rebooted.</td>
<td></td>
</tr>
</tbody>
</table>

| 1 | **CONSOLE INPUT** |
| INPUT: NONE |
| RETURN: A ← character input |
| Reads a character from the keyboard. Examines the character to see if it is a CP/M control character. |
### Table 6.6 (Continued)

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Register C)</td>
<td></td>
</tr>
</tbody>
</table>

#### 2 CONSOLE OUTPUT

INPUT: \( E \leftarrow \text{character to display} \)
RETURN: NONE

Writes a character to the screen.

#### 3 READER INPUT

INPUT: NONE
RETURN: A \( \leftarrow \text{character read} \)

This function is not supported on your Commodore 64.

#### 4 PUNCH OUTPUT

INPUT: \( E \leftarrow \text{character to punch} \)
RETURN: NONE

This function is not supported on your Commodore 64.

#### 5 LIST OUTPUT

INPUT: \( E \leftarrow \text{character to print} \)
RETURN: NONE

Writes a character to your printer.
<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRECT CONSOLE I/O</td>
<td>INPUT: E ← character to display (output)</td>
</tr>
<tr>
<td></td>
<td>E ← OFFH (input)</td>
</tr>
<tr>
<td></td>
<td>RETURN: A ← character (input)</td>
</tr>
<tr>
<td></td>
<td>A ← status (output)</td>
</tr>
<tr>
<td></td>
<td>Performs raw console input (read from keyboard) and output (write to screen). Characters are transferred through the BDOS without being examined or changed.</td>
</tr>
<tr>
<td>GET I/O BYTE</td>
<td>INPUT: NONE</td>
</tr>
<tr>
<td></td>
<td>RETURN: A ← I/O byte</td>
</tr>
<tr>
<td></td>
<td>The I/O byte function is not supported on your Commodore 64.</td>
</tr>
<tr>
<td>SET I/O BYTE</td>
<td>INPUT: E ← new I/O byte</td>
</tr>
<tr>
<td></td>
<td>RETURN: NONE</td>
</tr>
<tr>
<td></td>
<td>The I/O byte function is not supported on your Commodore 64.</td>
</tr>
<tr>
<td>PRINT STRING</td>
<td>INPUT: DE ← string address</td>
</tr>
<tr>
<td></td>
<td>RETURN: NONE</td>
</tr>
<tr>
<td></td>
<td>Writes the character string to the screen. The string must terminate with a “$”.</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| 10 READ CONSOLE BUFFER         | INPUT: DE ← buffer address  
RETURN: characters in buffer |
|                               | Reads from the keyboard until a carriage return or CTL-M is entered or until the keyboard buffer overflows. |
| 11 GET CONSOLE STATUS          | INPUT: NONE  
RETURN: A ← console status |
|                               | Checks the keyboard status. A contains 0FFH if a character is ready: 00H if not. |
| 12 RETURN VERSION NUMBER       | INPUT: NONE  
RETURN: HL ← version number |
|                               | Returns the CP/M version number. |
| 13 RESET DISK SYSTEM           | INPUT: NONE  
RETURN: NONE |
<p>|                               | Resets the entire disk system to its initial state. |</p>
<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Register C)</td>
<td></td>
</tr>
</tbody>
</table>

### 14 SELECT DISK

**INPUT:** E ← disk number to select  
**RETURN:** NONE

Selects a disk (A=0 and B=1).

### 15 OPEN FILE

**INPUT:** DE ← address of FCB  
**RETURN:** A ← directory code

Opens a disk file for processing. Returns a 255 in A if the file could not be found.

### 16 CLOSE FILE

**INPUT:** DE ← address of FCB  
**RETURN:** A ← directory code

Closes a disk file. Returns a 255 in A if the file could not be found.

### 17 SEARCH FOR FIRST

**INPUT:** DE ← address of FCB  
**RETURN:** A ← directory code

Searches for the first file matching the name given in the FCB. Returns a 255 in A if no match was found.
<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| 18 SEARCH FOR NEXT | INPUT: NONE  
RETURN: A ← directory code  
Similar to Function 17, but begins search where 17 left off. Also returns a 255 in A if no match was found. |
| 19 DELETE FILE    | INPUT: DE ← address of FCB  
RETURN: A ← directory code  
Deletes a disk file. Returns a 255 in A if the file could not be found. |
| 20 READ SEQUENTIAL | INPUT: DE ← address of FCB  
RETURN: A ← directory code  
Reads the next 128-byte record into the memory pointed to by the current DMA address. Returns a 00H in A if the read succeeded; non-zero if end-of-file was encountered. |
| 21 WRITE SEQUENTIAL | INPUT: DE ← address of FCB  
RETURN: A ← directory code |
Table 6.6 (Continued)

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Register C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Writes the 128-byte record pointed to by the current DMA address. Returns a 00H in A if the write succeeded; a non-zero for a full disk.</td>
</tr>
</tbody>
</table>

22 MAKE FILE

INPUT: DE ← address of FCB
RETURN: A ← directory code

Creates the disk file named in the FCB. Returns a 255 in A if the create failed.

23 RENAME FILE

INPUT: DE ← address of FCB
RETURN: A ← directory code

Renames a disk file. The name of the file is in the first 16 bytes of the FCB, the new name is in the next 16 bytes. Returns a 255 in A if the rename fails.

24 RETURN LOGIN VECTOR

INPUT: NONE
RETURN: HL ← login vector

Returns the disk login vector. The least significant bit of L represents Disk A and the next Drive B. When set to 1, the drive is online.
<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| **25** RETURN CURRENT DISK | INPUT: NONE  
RETURN: A ← current disk number  
Returns the number of the currently logged disk (0=A and 1=B). |
| **26** SET DMA ADDRESS | INPUT: DE ← DMA address  
RETURN: NONE  
Sets the address of the 128-byte disk sector buffer. |
| **27** GET ADDR (ALLOC) | INPUT: NONE  
RETURN: HL ← ALLOC address  
Returns the address of the allocation vector of the current disk. |
| **28** WRITE PROTECT DISK | INPUT: NONE  
RETURN: NONE  
Protects the current disk from being written to. |
<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| **29 GET READ ONLY VECTOR** | INPUT: NONE  
RETURN: HL ← read only vector  
Returns a vector indicating which drives are temporarily write-protected. The least significant bit of L represents Disk A and the next Drive B. When set to 1, the drive is write-protected. |
| **30 SET FILE ATTRIBUTES** | INPUT: DE ← address of FCB  
RETURN: A ← directory code  
Sets read only and system file attributes. |
| **31 GET ADDR (DISK PARMS)** | INPUT: NONE  
RETURN: HL ← address of DPB  
Returns the address of the Disk Parameter Block. |
| **32 SET/GET USER CODE**  | INPUT: E ← user code (SET)  
E ← OFFH (GET)  
RETURN: A ← user code (GET) |
Table 6.6 (Continued)

<table>
<thead>
<tr>
<th>FUNCTION (Register C)</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns or sets the current user code (user number).</td>
<td></td>
</tr>
</tbody>
</table>

33 READ RANDOM

INPUT: DE ← address of FCB
RETURN: A ← return code

Performs a random record read on a disk file. Return codes are:

- 01 reading unwritten data
- 03 cannot close current extent
- 04 seek to unwritten extent
- 06 seek past end of disk

34 WRITE RANDOM

INPUT: DE ← address of FCB
RETURN: A ← return code

Performs a random record write to a disk file. Return codes are:

- 01 reading unwritten data
- 03 cannot close current extent
- 04 seek to unwritten extent
- 05 out of directory space
- 06 seek past end of disk
### Table 6.6 (Continued)

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Register C)</td>
<td></td>
</tr>
<tr>
<td><strong>35</strong> COMPUTE FILE SIZE</td>
<td></td>
</tr>
<tr>
<td>INPUT: DE ← address of FCB</td>
<td></td>
</tr>
<tr>
<td>RETURN: file size</td>
<td></td>
</tr>
<tr>
<td>Returns the size of the file, in records, to the random record field of the FCB.</td>
<td></td>
</tr>
<tr>
<td><strong>36</strong> SET RANDOM RECORD</td>
<td></td>
</tr>
<tr>
<td>INPUT: DE ← address of FCB</td>
<td></td>
</tr>
<tr>
<td>RETURN: NONE</td>
<td></td>
</tr>
<tr>
<td>Sets the random record number of a record that was read sequentially. The random record number is placed into the random record field of the FCB.</td>
<td></td>
</tr>
<tr>
<td><strong>37</strong> RESET DRIVE</td>
<td></td>
</tr>
<tr>
<td>INPUT: DE ← drive vector</td>
<td></td>
</tr>
<tr>
<td>RETURN: NONE</td>
<td></td>
</tr>
<tr>
<td>Resets the disk drives indicated in the drive vector. The least significant bit of L represents Disk A and the next Drive B. When set to 1, the drive is reset.</td>
<td></td>
</tr>
<tr>
<td><strong>38</strong> NOT USED</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>39</strong> NOT USED</td>
<td></td>
</tr>
</tbody>
</table>
Table 6.6 (continued)

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Register C)</td>
<td></td>
</tr>
</tbody>
</table>

40 WRITE RANDOM WITH ZERO FILL

INPUT: DE ← address of FCB
RETURN: A ← return code

Identical to WRITE RANDOM (Function 34), except that new blocks are zero-filled before data is moved into them.

6.6 CALLING A Z80 PROGRAM FROM THE 6510

You sometimes may want to call a Z80 routine from your Commodore 64 while it is running in native mode. You may, for example, want to take advantage of the Z80 register structure or its extended instruction set, which make some routines easier to write or more efficient to execute.

When you first switch on your Z80 processor, it will always begin execution at its reset address:

6510 ADDRESS $1000—Z80 ADDRESS $0000

To call a Z80 routine from the 6510, you must either:

- Load the routine at 6510 address $1000.
- Place a Z80 jump instruction at 6510 address $1001 that transfers control to the actual code location.

In BOTH cases, 6510 address $1000 (Z80 $0000) must contain a NOP instruction ($00). This is a requirement of the processor switching hardware. Of course, if you place a jump instruction at 6510 address $1001, you must load the actual Z80 routine elsewhere in memory.

On subsequent calls to the Z80, routine execution will resume at the instruction following the last instruction executed before the Z80 switched itself off. It does NOT resume execution at the reset address.
6.6.1 Some Examples

Suppose you load some Z80 code at 6510 address $1000. You can transfer control to that code by switching on the Z80 processor:

LDA #0 ;LOAD ZERO INTO A
STA $DE00 ;STORE ZERO IN THE MODE SWITCH LOCATION
NOP ;REQUIRED BY THE SWITCH HARDWARE

The first time this code is executed, the Z80 will start executing instructions at $0000 (6510 address $1000); that address must contain a NOP instruction. Subsequent executions of the code (without turning off your Commodore 64) will cause the Z80 to resume execution where it left off when it switched the 6510 back on.

Assume now that you have loaded your Z80 code at 6510 address $B000. This corresponds to a Z80 address of $A000. You can get to this routine by using code similar to the following:

LDA #$00 ;OPCODE FOR A NOP INSTRUCTION
STA $1000 ;MEET THE SWITCHING REQUIREMENT
LDA #$C3 ;Z80 JUMP INSTRUCTION OPCODE
STA $1001 ;FIRST BYTE OF JUMP INSTRUCTION
LDA #$00 ;LOW BYTE OF Z80 JUMP ADDRESS
STA $1002 ;NEXT BYTE OF JUMP INSTRUCTION
LDA #$A0 ;HIGH BYTE OF Z80 ADDRESS
STA $1003 ;LAST BYTE OF JUMP INSTRUCTION
LDA #0 ;LOAD ZERO INTO A
STA $DE00 ;STORE ZERO IN THE MODE SWITCH LOCATION
NOP ;REQUIRED BY THE SWITCH HARDWARE
Subsequent executions of this code (without turning off your Commodore 64) will cause the Z80 to resume execution where it left off when it switched the 6510 back on. You could thus use address $1000 for other purposes after calling the Z80 routine the first time.

You can return from your Z80 routine by using the code below:

```
MVI A,1 ;LOAD ONE INTO A
STA 0CE00H ;STORE ONE IN MODE SWITCH
            LOCATION
            ;TO TURN ON THE 6510
NOP ;REQUIRED BY THE HARDWARE
            AFTER A MODESW

;THE NEXT TIME IT IS SWITCHED ON, THE Z80 RESUMES
EXECUTION HERE
```

---

**NOTE:** You MUST follow the mode switching store instruction with a NOP instruction.

---

### 6.7 CALLING A 6510 PROGRAM FROM THE Z80

There may be times when you want the 6510, running in Commodore 64 native mode, to perform some special tasks for you.

For example, suppose you add the IEEE expansion cartridge to your Commodore 64 in order to attach an IEEE standard instrument. Instruments require special control commands that can be issued only by the 6510 main processor.

The 6510 portion of the BIOS (BIOS65) includes a facility for calling your own code. This facility is implemented through the BIOS function codes 7, 8, and 9.
• **BIOS function code 7** instructs BIOS65 to transfer control to:

   6510 ADDRESS $0E00 — Z80 ADDRESS $FE00

• **BIOS function code 8** instructs BIOS65 to transfer control to:

   6510 ADDRESS $0F00 — Z80 ADDRESS $FF00

• **BIOS function code 9** instructs BIOS65 to transfer control indirectly to the instruction whose address is stored at:

   6510 ADDRESS $0907 — Z80 ADDRESS $F907

The code that you load at these locations MUST end with a 6510 RTS instruction. This instruction returns control to BIOS65, which can then switch the Z80 processor back on.

As you see, function codes 7 and 8 always transfer control to the same location. If you use both functions 7 and 8, your programs cannot be larger than $100 bytes (256 decimal). If you use only function code 7, you can expand your program into the function code 8 space. This gives you a maximum program size of $200 bytes (512 decimal).

If you need more space than you can get under function codes 7 and 8, you can use function code 9. When you pass function code 9 to BIOS65, it transfers control to the address stored at 6510 location $0F07. This address can be anywhere in the 6510 address space.

---

**NOTE:** When you use BIOS function 9, the indirect address you store at Z80 address $FF07 (6510 address $0F07) MUST be a 6510 base address.

---

**6.7.1 Switching on the 6510**

If you are going to use a 6510 routine, you have to know how to switch on the 6510 processor. The two processors
cannot operate at the same time. When you switch one of them on, the other is automatically switched off.

Processor switching is controlled by storing a mode switch value in:

6510 ADDRESS $DE00—Z80 ADDRESS $CE00

The mode switch values are:

0 → activates the Z80 processor
1 → activates the 6510 processor

Suppose you load some 6510 code at 6510 address $0E00 that you wish to execute from a Z80 program. You can do that using code like the following:

```
MVI A,7 ;LOAD THE FUNCTION CODE INTO A
STA 0F900H ;STORE THE FUNCTION CODE IN COMMAND REGISTER

; PREPARE ANY OTHER PARAMETERS REQUIRED
; BY THE CODE YOU HAVE
; PLACED AT 6510 ADDRESS $0E00—Z80
; ADDRESS $FE00

; MVI A,1 ;LOAD ONE INTO A
STA 0CE00H ;STORE ONE IN MODE SWITCH LOCATION
; TO TURN ON THE 6510
NOP ;REQUIRED BY THE HARDWARE AFTER A MODESW

; AFTER COMPLETION OF THE 6510
ROUTINE, Z80 RESUMES EXECUTION HERE
```

From the example above, you can see that it's easy to call a 6510 routine from the Z80. The 6510 routine that you write does not have to switch control back to the Z80. The BIOS65 program takes care of the return to the Z80.

138 CP/M ON THE COMMODORE 64
NOTE: You MUST follow the mode-switching store instruction with a NOP instruction.

You must, of course, load your 6510 routine into the correct memory location before you transfer control to it. If you use BIOS function 9, you must also load the 6510 address of the code to be executed in indirect address location $F907 (Z80).

6.8 PROGRAM EXECUTION UNDER CP/M

Programs destined to execute under CP/M must be stored in a disk file and have a file name extension of .COM (see Chapter 5 for an explanation of CP/M file-naming conventions and details on executing programs). User programs running under CP/M are loaded into the Transient Program Area (TPA) for execution.

You execute a program under CP/M simply by entering its name (without the extension). The general form is:

```
[DISKID:]PROGRAM-FILENAME
```

where diskid is an optional disk identifier (A or B) and program-filename is the name of the file that contains your program. The program file MUST have the extension .COM.

Suppose, for example, that you have a program stored in a file named STARTREK.COM. To execute that program, you respond to the CP/M prompt (usually A>) with:

```
STARTREK
```

CP/M will then load the file STARTREK.COM into the TPA (Transient Program Area) and transfer control to it (at location $100). When STARTREK completes its execution, it returns to CP/M via a Z80 RET instruction or via a jump to location $0000. The return via a jump to location $0000 causes a warm start reboot of CP/M.
The following charts list which memory locations control placing characters on the screen, and the locations used to change individual character colors, as well as showing character color codes.

SCREEN MEMORY MAP
The actual values to POKE into a color memory location to change a character's color are:

Ø BLACK  8 ORANGE
1 WHITE  9 BROWN
2 RED  10 Light RED
3 CYAN  11 GRAY 1
4 PURPLE  12 GRAY 2
5 GREEN  13 Light GREEN
6 BLUE  14 Light BLUE
7 YELLOW  15 GRAY 3

For example, to change the color of a character located at the upper left-hand corner of the screen to red, type: POKE 55296,2.

COLOR MEMORY MAP

Columns

Rows
This bibliography lists a variety of currently available CP/M and Z80 books. Look at several books covering the topics that interest you before you make your selection.

Each author covers the topics from a different viewpoint. Find the book that you feel most comfortable with. Some people prefer a more technical discussion and should select a book with in-depth technical detail. Others like a less technical approach and should seek a book that is easy to understand.

You also can subscribe to a new magazine devoted exclusively to CP/M:

The User’s Guide to CP/M Systems and Software
Box 3050
Stanford, CA 94305

You may be interested in joining the CP/M User’s Group, which provides software written by members for their CP/M systems. Software is often available for only a copying charge. You can contact the CP/M User’s Group through:

CP/M User’s Group
c/o Lifeboat Associates
1651 Third Avenue
New York, NY 10028

B.1 CP/M Books

This list gives some of the most recent CP/M books in alphabetical order by title. It is by no means a list of all the CP/M books available today. The prices shown are subject to change.

CP/M Handbook With MP/M by Rodnay Zaks, SYBEX, paper, $14.95
This is a reference guide to CP/M, written in a readable style for beginners.

*CP/M Primer* by Stephen Murtha, Howard W. Sams, paper, $14.95

This book helps both the first-time microcomputer user and the experienced user who is just beginning to use CP/M.

*CP/M Word Processing* by Chris DeVoney, Que Corporation, paper, $16.50

This book covers the use of word processing packages developed to run under the CP/M operating system. It contains detailed evaluations of 17 popular CP/M word processing packages and tells how to decide which word processor best meets your needs.

*How to Get Started with CP/M* by Carl Townsend, Dilithium Press, paper, $13.95

This book describes the CP/M operating system in an easy, comfortable style. It eases the reader into understanding the details of this widely used microcomputer operating system.

*Osborne CP/M User Guide* by Thom Hogan, Osborne, paper, $12.99

One of the most complete and up-to-date CP/M books available. This book contains easy-to-understand descriptions of the CP/M operating system and commands. It also contains detailed technical information for more experienced users.

*Using CP/M* by Judi Fernandez and Ruth Ashley, John Wiley, paper, $12.95

This is a complete, detailed introduction to the use of CP/M, written in an easy-to-understand style.
This up-to-date computer resource for CP/M describes peripherals, software, and accessories for CP/M systems. It includes a bibliography and lists of user groups, magazines, supplies, and computer accessories.

B.2 Z80 Books

8080/Z80 Assembly Language by Alan Miller, John Wiley, paper, $10.95
A step-by-step guide to programming the 8080 and Z80 microprocessors. This book helps intermediate and advanced programmers to get even more out of their 8080/Z80.

Programming the Z80 by Rodnay Zaks, SYBEX, paper, $15.95
This book covers the Z80 from basic concepts through advanced programming techniques. Exercises are offered to measure reader comprehension along the way. The book’s topics range from hardware organizations to data structures.

Z80 and 8080 Assembly Language Programming by Kathe Spracklen, Hayden Book Co., paper, $9.70
This book covers programming techniques and gives complete instruction sets for the 8080 and Z80 microprocessors. Each chapter includes exercises and answers to help readers learn to use the Z80 and 8080 more efficiently.

Z80 Microcomputer Design Projects by William Barden, Howard W. Sams, paper, $13.95
This book gives a solid, in-depth look at the popular Z80 microprocessor. It provides a complete look at the internal architecture of the Z80.

**Z80 Microcomputer Handbook** by William Barden, Howard W. Sams, paper, $11.95

This book is designed to teach you about the Z80. There is extensive coverage of Z80 machine language and the Z80 assembler language.

**Z80 Microcomputer Programming and Interfacing, Books 1 and 2** by Elizabeth Nichols, Howard W. Sams, paper, Book 1 — $12.95, Book 2 — $12.95, Book 1 & 2 — $24.95

Book 1 introduces computers to readers who have no background in computer science. Book 2 assumes a familiarity with Book 1 and continues an in-depth discussion of the design and use of the popular Z80 microprocessor. Both volumes are written in a self-teaching format with exercises and answers.

**Z80 User's Manual** by Joseph Carr, Prentice-Hall, paper, $15.95

An all-in-one guide to the Z80. This book is useful for both beginning and advanced Z80 users. It includes in-depth technical details for the Z80.
APPENDIX C

CP/M COMMAND LIST

This appendix is a simple listing of CP/M commands. For details on these commands, see Chapter 5.

Load and execute a program:
   [disk-id:]filename <CR>

Change the currently logged disk:
   dsk-id:

Assemble a Z80 assembler program:
   ASM filename[.parms]

ASM error codes are given in Table 5.4.

Run the CP/M debugger:
   DDT [ [disk-id:]filename[.type] ]

DDT commands are given in Table 5.5.

Get a directory listing:
   DIR [disk-id:]filename.type

Dump a file in ASCII and hexadecimal format:
   DUMP [disk-id:]filename.type
Edit a file:
ED [disk-id:]filename[type] [[disk-id2:]
[filename2[type2]]]

ED control characters are given in Table 5.8.
ED commands are given in Table 5.9.

Erase a file:
ERA [disk-id:]filename.type

Create an executable module from ASM output:
LOAD [disk-id:]filename

Copy a new version of CP/M:
MOVCP [ { * | size } ] [ * ]

Copy a file or disk:
PIP destination=source[ command-parameters]

Table 5.10 gives PIP logical devices.
Table 5.11 gives special PIP devices.
Table 5.12 gives PIP command parameters.

Rename a file:
REN [disk-id: ]new-file=old-file

Save page-num 256-byte pages of memory beginning at the
start of the TPA (100 hexadecimal):
SAVE page-num [disk-id: ]filename[type]
Get disk and I/O device status information:
\texttt{STAT command}

Table 5.13 shows STAT command options.
Table 5.14 shows STAT command attributes.

Submit a file for batch execution:
\texttt{SUBMIT \([dlisk-id:]filename \[parameters]\)}

Generate a new CP/M system:
\texttt{SYSGEN \([dlisk-id:]filename.type\)}

Print a file to the screen:
\texttt{TYPE \([disk-id:]filename.type\)}

Change the user number:
\texttt{USER \([user-num]\)}

Include keyboard data in your SUBMIT file:
\texttt{XSUB}
APPENDIX D

ASCII, CHR $, AND HEXADECIMAL CHARACTER CODES

When running in native mode your Commodore 64 uses two sets of character codes:

- CHR$ Codes (see Appendix F of your Commodore 64 User's Guide).
- Screen Display Codes (see Appendix E of your Commodore 64 User's Guide).

CP/M employs another character code set called the ASCII Character Codes (shown in Table D.1 below).

---

NOTE: The CTRL-Shifted column of Table D.1 shows the values generated when you hold the CTRL key down and press the character key.

---

When you use the CONFIG utility to alter character code values, you must supply the ASCII hexadecimal value of the new character. Therefore, the character code values shown in Table D.1 are expressed in hexadecimal.

If you're not sure what a hexadecimal value is, don't worry. Look up the character in Table D.1 and use the value shown (including the letters).

Table D.1 ASCII Character Codes (Hexadecimal Values)

<table>
<thead>
<tr>
<th>CHARACTER</th>
<th>HEX VALUE</th>
<th>CTRL</th>
<th>SHIFTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN/STOP</td>
<td>03</td>
<td>03</td>
<td></td>
</tr>
<tr>
<td>INS/DEL</td>
<td>08</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>RETURN</td>
<td>0D</td>
<td>0D</td>
<td></td>
</tr>
<tr>
<td>CLR/HOME</td>
<td>1B</td>
<td>7F</td>
<td></td>
</tr>
<tr>
<td>LEFT/RIGHT</td>
<td>1C/1D</td>
<td>1D</td>
<td></td>
</tr>
<tr>
<td>UP/DOWN</td>
<td>1E/1F</td>
<td>1F</td>
<td></td>
</tr>
<tr>
<td>SPACE</td>
<td>20</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>CHARACTER</td>
<td>HEX VALUE</td>
<td>CTRL SHIFTED</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>22</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>23</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>24</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>25</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>%</td>
<td>26</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>&amp;</td>
<td>27</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>(</td>
<td>28</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>)</td>
<td>29</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>2A</td>
<td>2A</td>
<td>2A</td>
<td></td>
</tr>
<tr>
<td>2B</td>
<td>2B</td>
<td>2B</td>
<td></td>
</tr>
<tr>
<td>2C</td>
<td>7B</td>
<td>7B</td>
<td></td>
</tr>
<tr>
<td>2D</td>
<td>2D</td>
<td>2D</td>
<td></td>
</tr>
<tr>
<td>2E</td>
<td>7D</td>
<td>7D</td>
<td></td>
</tr>
<tr>
<td>2F</td>
<td>5C</td>
<td>5C</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>31</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>33</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>34</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>36</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>37</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>38</td>
<td>7B</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>39</td>
<td>7D</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>3A</td>
<td>7B</td>
<td></td>
</tr>
<tr>
<td>;</td>
<td>3B</td>
<td>7D</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>40</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>41</td>
<td>01</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>42</td>
<td>02</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>43</td>
<td>03</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>44</td>
<td>04</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>45</td>
<td>05</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>46</td>
<td>06</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>47</td>
<td>07</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>48</td>
<td>08</td>
<td></td>
</tr>
<tr>
<td>j</td>
<td>49</td>
<td>09</td>
<td></td>
</tr>
<tr>
<td>4A</td>
<td>0A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHARACTER</td>
<td>HEX VALUE</td>
<td>CTRL SHIFTED</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>4B</td>
<td>0B</td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>4C</td>
<td>0C</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>4D</td>
<td>0D</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>4E</td>
<td>0E</td>
<td></td>
</tr>
<tr>
<td>o</td>
<td>4F</td>
<td>0F</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>50</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>q</td>
<td>51</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>52</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>53</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>54</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>u</td>
<td>55</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>v</td>
<td>56</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>57</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>58</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>59</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>5A</td>
<td>1A</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>5B</td>
<td>1B</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>5C</td>
<td>1C</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>5D</td>
<td>1D</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>5E</td>
<td>1E</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>5F</td>
<td>1F</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>60</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>61</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>62</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>63</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>64</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>65</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>66</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>67</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>68</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>69</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>6A</td>
<td>2A</td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>6B</td>
<td>2B</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>6C</td>
<td>2C</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>6D</td>
<td>2D</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>6E</td>
<td>2E</td>
<td></td>
</tr>
<tr>
<td>U</td>
<td>6F</td>
<td>2F</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>70</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>71</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>72</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>73</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>CHARACTER</td>
<td>HEX VALUE</td>
<td>CTRL SHIFTED</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>74</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>U</td>
<td>75</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>76</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>77</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>78</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>79</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>7A</td>
<td>1A</td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>80</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>81</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>F3</td>
<td>82</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>F4</td>
<td>83</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>F5</td>
<td>84</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>F6</td>
<td>85</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>F7</td>
<td>86</td>
<td>87</td>
<td></td>
</tr>
<tr>
<td>F8</td>
<td>87</td>
<td>87</td>
<td></td>
</tr>
</tbody>
</table>
This appendix gives the source listings for the BIOS and BOO'T programs on the 6510 and the Z80.

**Xerox to Commodore 64 Receive Utility**

**COPYRIGHT © 1982 COMMODORE INTERNATIONAL**

```assembly
0100 = TPA      EQU 100H ;START ADDRESS OF PROGRAM
005C = FCB      EQU 005CH ;FILE CONTROL BLOCK
0080 = DMADDR   EQU 0080H ;DMA ADDRESS
000D = CR       EQU 0DH ;CARRIAGE RETURN
0006 = ACK      EQU 06H
0015 = NAK      EQU 15H
0000 = BOOT     EQU 0000H
0005 = BDOS     EQU 0005H
0E00 = SIO      EQU 0E00H
FF00 = MEM      EQU 0FF00H ;BUFFER MEMORY
0300 = PGM65    EQU 0300H
0080 = SIZE65   EQU 128

; SYNTAX FOR COMMAND IS
; RECEIVE FILENAME.EXT

0100 = ORG TPA

0100 31D802 = LXI SP,STACK ;SET UP LOCAL STACK

; CHECK FOR VALID FILENAME

0103 113802 = LXI D,NONAME ;NONAME MESSAGE
0106 3A5D00 = LDA FCB + 1
```
0109 FE20 CPI ','
010B CAE201 JZ DONE ;IF SPACE, NO NAME GIVEN

010E 115802 LXI D, BADNAM ;CHECK FOR AMBIGUOUS NAME
0111 215C00 LXI H, FCB
0114 3E3F MVI A, '?'
0116 0610 MVI B, 16 ;COUNTER

0118 BE QLOOP: CMP M ;IS IT '?'
0119 CAE201 JZ DONE ;IF SO, BAD NAME

011C 23 INX H
011D 05 DCR B
011E C21801 JNZ QLOOP ;DO 16 TIMES

0121 118000 LXI D, DMADDR
0124 CD1702 CALL SETDMA

; TRANSFER 6510 CODE TO $E00 (OFEOOH)

0127 0680 MVI B, SIZE65
0129 210003 LXI H, PGM65
012C 1100FE LXI D, OFEOOH

012F 78 MOV A, B
0130 A7 ANA A
0131 CA3C01 JZ SKIP
0134 7E LOADLP MOV A, M
0135 12 STAX D
0136 23 INX H
0137 13 INX D
0138 05 DCR B
0139 C23401 JNZ LOADLP ;GET READY BY OPENING FILES

013C 115C00 SKIP: LXI D, FCB
013F CD1D02 CALL DELETE
0142 115C00 LXI D, FCB
0145 CD2302 CALL MAKE

156 APPENDIX E
0148 117602  LXI  D,NODIR
0148  3C      INR  A           ;WAS 255 IF NO FILE SPACE
014C CAE201  JZ   DONE

014F 118000  LXI  D,DMADDR
0152 CD1702  CALL  SETDMA

; 0155 AF  READS:  XRA  A
0156 328702  STA  POINT

; 0159 3E06  GNEXT:  MVI  A,ACK      ;SEND INITIAL ACK
015B 32FFE E  GBLK.  STA  OFEFFH    ;/O LOCATION

; 015E 3E07  MVI  A,7
0160 3200F9  STA  0F900H
0163 3E01  MVI  A,1
0165 3200CE  STA  OCE00H
0168 00  NOP

; NEED TEST FOR ERROR

0169 3AFFFE  LDA  OFEFFH
016C A7  ANA  A
016D C2C401  JNZ  AGAIN

; 0170 118000  LXI  D,DMADDR
0173 3A8702  LDA  POINT
0176 B3  ORA  E
0177 5F  MOV  E,A
017B 2100FF  LXI  H,MEM
017B 7E  MOV  A,M
017C FE3A  CPI  ';'
017E C2C401  JNZ  AGAIN

; 0181 AF  XRA  A
0182 328602  STA  BADDAT
0185 CDE801  CALL  GYBTE
0188 A7  ANA  A
0189 CAD901  JZ  FINISH
018C FE20  CPI   32
018E C2C401 JNZ  AGAIN

; 0191 0E00 GETQ: MVI  C,0 ;CHECKSUM
0193 47  MOV  B,Z ;COUNTER

; 0194 C5  GQLP: PUSH  B
0195 CDE801 CALL  GBYTE

; 0198 12  STAX  D
0199 1C  INR  E
019A C1  POP  B
019B 81  ADD  C
019C 4F  MOV  C,A
019D 05  DCR  B
019E C29401 JNZ  GQLP

; 01A1 C5  PUSH  B
01A2 CDE01 CALL  CBYTE
01A5 C1  POP  B
01A6 81  ADD  C
01A7 C2C401 JNZ  AGAIN

; 01AA 3AB602 LDA  BADDAT
01AD B7  ORA  A
01AE C2C401 JNZ  AGAIN

; 01B1 3AB702 LDA  POINT
01B4 C620  ADI  32
01B6 32B702 STA  POINT
01B9 FE80  CPI  128
01BB C25901 JNZ  GNEXT

; 01BE CDC901 CALL  SWRITE
01C1 C35501 JMP  READS

; 01C4 3E15 AGAIN: MVI  A,NAK
01C6 C35801 JMP  GBLK
020D C9     RET
020E D630   NUMBER: SUI 0
             RET
0210 C9     ;
0211 3EFF   NOTHEX: MVI A, OFFH
0213 32B602 STA BADDAT
0216 C9     RET
0217 0E1A   SETDMA: MVI C, 26
0219 CD0500 CALL BDOS
021C C9     RET
021D 0E13   DELETE: MVI C, 19
021F CD0500 CALL BDOS
0222 C9     RET
0223 0E16   MAKE: MVI C, 22
0225 CD0500 CALL BDOS
0228 C9     RET
0229 0E15   WRITE: MVI C, 21
022B CD0500 CALL BDOS
022E C9     RET
022F 0E10   CLOSE: MVI C, 16
0231 CD0500 CALL BDOS
0234 C9     RET
0235 0E09   PRINT: MVI C, 9
0237 CD0500 CALL BDOS
023A C9     RET
023B 46494C454E NONAME: DB 'FILENAME MUST BE SPECIFIED', 0DH, 0DH, '$'
0258 414D424947 BADNAM: DB 'AMBIGUOUS FILES NOT ALLOWED', 0DH, 0DH, '$'
0276 4E4F204449  NODIR:   DB  'NO DIRECTORY SPACE AVAILABLE'
0292 0D0D24     DB  ODH,ODH, '$'
                 ;
0295 4449534820  DFULL:   DB  'DISK FULL'
029E 0D0D24     DB  ODH,ODH, '$'
                 ;
02A1 5452414E53  EOTRAN:  DB  'TRANSFER COMPLETE:',ODH,ODH, '$'
                 ;
0286        BADDAT:  DS   1
0287        POINT:   DS   1
0288        DS    32
02DB =      STACK   EQU  $

Commodore 64 Copy Utility 1.0

COPYRIGHT © 1982
COMMODORE INTERNATIONAL

0100        ORG  100H
            ;
            ;  EQUATES
            ;
F800 =      BUFFER   EQU  0F800H
F900 =      CMD      EQU  0F900H
F901 =      DATA     EQU  0F901H
F902 =      SECTOR   EQU  0F902H
F903 =      TRACK    EQU  0F903H
F904 =      DISKNO   EQU  0F904H
0001 =      OFF      EQU  1
CE00 =      MODESW   EQU  0CE00H
0000 =      VICRD    EQU  0
0001 =      VICWR    EQU  1
0006 =      VICFORM  EQU  6
0005 =      BDOS     EQU  0005H
0000 =      BOOT     EQU  0000H
000D =      CR       EQU  ODH; CARRIAGE RETURN
000A =      LF       EQU  OAH; LINE FEED
000C = CLS EQU 0CH ; CLEAR SCREEN

0100 31B06 START: LXI SP,STACK
0103 111403 LXI D,COPMSG
0106 CD0503 ; CALL PRINT ; PROGRAM NAME, ETC.
0109 CD0003 INIT04: CALL ~ CONIN

010C FE31 CPI '1'
010E CA2301 JZ FORMAT

0111 FE32 CPI '2'
0113 CAD701 JZ BACKUP

0116 FE33 CPI '3'
0118 CA7B01 JZ SYSTEM

0118 FE34 CPI '4'
011D CA0000 JZ BOOT

0120 C30901 JMP INIT04

0123 11A603 FORMAT LXI D,FMTMSG ; FORMAT A DISK
0126 CD0503 CALL PRINT

0129 CDDD02 CALL CRORS ; GET KEYBOARD INPUT
012C CA0001 JZ START ; IF RUN/STOP, GO TO MENU

012F 116104 LXI D,FMTIN ; FORMATTING MESSAGE
0132 CD0503 CALL PRINT

0135 3E06 MVI A,VICFMT
0137 CD0A03 CALL IO6510 ; SEND FORMAT COMMAND TO 6510

013A 3A01F9 LDA DATA ; CHECK FOR ERROR
013D A7 ANA A
013E C27501 JNZ FMTERR

0141 2100F8 LXI H,BUFFER ; FILL DISK BUFFER WITH ES's
0144 3EE MVI A,0E5H ; FOR DIRECTORY SECTORS
0146 77 FMT0: MOV M,A

162 APPENDIX E
0147 2C                        INR     L
0148 C24601                    JNZ     FMT0   ;DO THIS 256 TIMES

014B 3E03                      MVI     A,3
014D 3203F9                    STA     TRACK   ;DIRECTORY TRACK

0150 3E00                      MVI     A,0
0152 3204F9                    STA     DISKNO   ;FORCE DRIVE 0

0155 3E00                      MVI     A,0   ;INITIAL SECTOR

0157 3202F9  FMT1:            STA     SECTOR   ;SET SECTOR
015A 3E01                      MVI     A,VICWR   ;GET READY FOR WRITE
015C CD0A03                    CALL    IO6510   ;GO DO IT
015F 3A01F9                    LDA     DATA   ;A = 0 IF OK
0162 A7                        ANA     A
0163 C27501                    JNZ     FMTERR

0166 3A02F9                    LDA     SECTOR
0169 3C                        INR     A
016A FE08                      CPI     8   ;DO ONLY SECTORS 0-7
016C C25701                    JNZ     FMT1   ;LOOP UNTIL DONE

016F 118704                    LXI     D,FMTDON
0172 C37502                    JMP     DONE

0175 119A04  FMTERR:           LXI     D,FTERM
0178 C37502                    JMP     DONE

017B 11D304  SYSTEM:          LXI     D,SYSMSG   ;SYSTEM TRACKS ONLY
017E CD0503                    CALL    PRINT

0181 112905                    LXI     D,SRCMSG
0184 CD0503                    CALL    PRINT

0187 116905                    LXI     D,PRSMMSG
018A CD0503                    CALL    PRINT
018D CDD802                    CALL    CRORRS
$190 CA0001                    JZ     START   ;IF SPACEBAR, GO TO MENU

0193 CDEA02                    CALL    CRLF
; BEGINNING OF MEMORY SPACE ***

0199 3E01 MVI A,1
0198 CD8402 CALL RDTRK ;READ TRACK 1

019D 3E02 MVI A,2
01A0 CD8402 CALL RDTRK ;READ TRACK 2

01A3 3E12 MVI A,18
01A5 CD8402 CALL RDTRK ;READ TRACK 18

01A8 114905 LXI D,DSTMSG ;PRINT DESTINATION MESSAGE
01AB CD0503 CALL PRINT

01AE 110F06 LXI D,RTNMSG
01B1 CD0503 CALL PRINT

01B4 CD0003 SYS1 CALL CONIN
01B7 FE0D CPI CR ;WAIT FOR CARRIAGE RETURN
01B9 C2B401 JNZ SYS1

01BC CDEA02 CALL CRLF

01BF 216B06 LXI H,MEM ;SETUP FOR WRITE ***

01C2 3E01 MVI A,1
01C4 CDAE02 CALL WRTRK

01C7 3E02 MVI A,2
01C9 CDAE02 CALL WRTRK

01CC 3E12 MVI A,18
01CE CDAE02 CALL WRTRK

01D1 118E05 LXI D,SYSDON
01D4 C37502 JMP DONE

01D7 11AC05 BACKUP: LXI D,BAKMSG ;BACKUP DISK
01DA CD0503 CALL PRINT
01DD 116905  LXI   D,PRSMSG
01E0  CD6503  CALL  PRINT
01E3  CDD802  CALL  CRORRS
01E6  CA0001  JZ    START
01E9  CDEA02  CALL  CRLF
    
01EC  3E01   MVI   A,1  ;START WITH TRACK 1
01EE  3203F9  STA   TRACK
    
01F1  3E05   MVI   A,5  ;DO OUTER LOOP 5 TIMES
01F3  324A06  STA   OUTER
    
01F6  3A03F9  BKLP: LDA   TRACK
01F9  324806  STA   WTRACK ;SAVE FOR WRITE TRACK
    
01FC  3E07   MVI   A,7
01FE  324906  STA   INNER ;INNER LOOP COUNTER
    
0201  112905  LXI   D,SRCMSG
0204  CD0503  CALL  PRINT
    
0207  110F06  LXI   D,RTNMSG
020A  CD0503  CALL  PRINT
    
020D  CD0003  BKRD1: CALL  CONIN
0210  FE0D    CPI   CR
0212  C20D02  JNZ   BKRD1
    
0215  216806  LXI   H,MEM ;START OF AVAILABLE MEMORY
    
0218  3A03F9  BKRD: LDA   TRACK
0218  CD8402  CALL  RDTRK
021E  3A03F9  LDA   TRACK
0221  3C      INR   A
0222  3203F9  STA   TRACK
0225  3A4906  LDA   INNER
0228  3D      DCR   A
0229  324906  STA   INNER
022C  C21802  JNZ   BKRD
    
022F  3A4806  LDA   WTRACK
0232 3203F9 STA TRACK ;RESTORE TRACK POINTER
0235 3E07 MVI A, 7
0237 324906 STA INNER ;INNER COUNTER

023A 114905 LXI D, DSTMSG
023D CD0503 CALL PRINT
0240 110F06 LXI D, RTNMSG
0243 CD0503 CALL PRINT

0246 CD0003 BKWR1: CALL CONIN
0249 FE0D CPI ODH
024B C24602 JNZ BKWR1

024E 216806 LXI H, MEM ;START OF MEMORY AGAIN

0251 3A03F9 BKWR: LDA TRACK
0254 CDAAE02 CALL WRTRK
0257 3A03F9 LDA TRACK
025A 3C INR A
025B 3203F9 STA TRACK
025E 3A4906 LDA INNER
0261 3D DCR A
0262 324906 STA INNER
0265 C25102 JNZ BKWR

0268 214A06 LXI H, OUTER
026B 35 DCR M
026C C2F601 JNZ BKLP

026F 11FC05 LXI D, BAKDON
0272 C37502 JMP DONE

0275 CD0503 DONE: CALL PRINT ;PRINT DONE MESSAGE

0278 118804 LXI D, ANYKEY
027B CD0503 CALL PRINT
027E CD0003 CALL CONIN ;WAIT FOR ANY KEY
0281 C30001 JMP START

0284 3203F9 RDTRK: STA TACK ;A = TRACK ON ENTRY

166 APPENDIX E
0287 3E00 MVI A,0 ;START WITH SECTOR 0
0289 3202F9 RD1: STA SECTOR
028C 3E00 MVI A,VICRD ;READ SECTOR COMMAND
028E CD0A03 CALL IO6510 ;GO DO IT
0291 3A01F9 LDA DATA
0294 A7 ANA A
0295 C2FA02 JNZ RDERR ;READ ERROR IF <>0

0298 1100F8 LXI D,BUFFER
029B 1A RD2: LDAX D ;GET CHARACTER FROM BUFFER
029C 77 MOV M,A ;AND PUT IN MEMORY
029D 13 INX D
029E 23 INX H ;BUMP POINTERS
029F 7B MOV A,E ;DONE 256 YET?
02A0 A7 ANA A
02A1 C29B02 JNZ RD2 ;JUMP IF NO

02A4 3A02F9 LDA * SECTOR
02A7 3C INR A
02A8 FE11 CPI 17 ;17 = LAST SECTOR + 1
02AA C28902 JNZ RD1

02AD C9 RET

02AE 3203F9 WRTRK: STA TRACK ;A = TRACK ON ENTRY
02B1 3E00 MVI A,0

02B3 3202F9 WR1: STA SECTOR
02B6 1100F8 LXI D,BUFFER
02BA 7E WR2: MOV A,M
02BC 12 STAX D ;PUT CHAR IN BUFFER
02BB 23 INX H
02BC 13 INX D ;INCREMENT POINTERS
02BD 7B MOV A,E ;DONE 256 YET?
02BE A7 ANA A
02BF C28902 JNZ WR2 ;JUMP IF NO

02C2 3E01 MVI A,VICWR ;SECTOR WRITE COMMAND
02C4 CD0A03 CALL IO6510 ;GO DO IT

APPENDIX E 167
02C7 3A01F9  LDA DATA
02CA A7    ANA A
02CB C2F402 JNZ WRERR ;JUMP IF WRITE ERROR
02CE 3A02F9 LDA SECTOR
02D1 3C    INR A
02D2 FE11  CPI 17 ;17 = LAST SECTOR + 1
02D4 C2B302 JNZ WR1 ;KEEP READING
 ;
02D7 C9    RET
 ;
02D8 FE20  CR1. CPI 20H ;SPACEBAR?
02DA C8    RZ
 ;
02DB CD0003 CRORRS CALL CONIN
02DE FE0D  CPI CR ;CARRIAGE RETURN
02E0 C2DB02 JNZ CR1
 ;
02E3 A7    ANA A ;KILL ZERO FLAG
02E4 C9    RET
 ;
02E5 0E02  CONOUT: MVI C,2
02E7 C30500 JMP BDOS
 ;
02EA 1E0D  CRLF: MVI E.CR
02EC CDE502 CALL CONOUT
02EF 1E0A  MVI E.LF
02F1 C3E502 JMP CONOUT
 ;
02F4 111D06 WRERR: LXI D,WRMSG
02F7 C37502 JMP DONE
 ;
02FA 113D06 RDERR: LXI D,RDMSG
02FD C37502 JMP DONE
 ;
0300 0E01  CONIN: MVI C,1
0302 C30500 JMP BDOS
 ;
0305 0E09  PRINT: MVI C,9
0307 C30500 JMP BDOS
 ;
030A 3200F9 IO6510: STA CMD ;PUT A IN 6510 COMMAND REGISTER

168  APPENDIX E
030D 3E01 MVI A,OFF
030F 3200CE STA MODESW ;TURN OFF Z80
0312 00 NOP
0313 C9 RET

; TEXT AND MESSAGES:

0314 0C0A434F4D COPMSG: DB CLS,LF, 'COMMODORE 04 UTILITY 1 0'
0333 D0D0A0A DB CR,LF,LF
0336 2020312E20 DB 1. FORMAT DISK',CR,LF
0349 2020322E20 DB ' 2. BACKUP DISK',CR,LF
035C 2020332E20 DB ' 3. COPY SYSTEM TRACKS ONLY',CR,LF
037B 2020342E20 DB ' 4. EXIT',CR,LF,LF
038B 504C454153 DB 'PLEASE CHOOSE FUNCTION (1-4) $'

03A6 0C0A464F52 FMTMSG: DB CLS,LF, 'FORMAT DISK UTILITY',CR,LF,LF
03BE 494E495449 DB 'INITIALIZES DISK FOR CP/M',CR,LF
03D9 0A43415554 DB LF, 'CAUTION! FORMAT ERASES ALL DATA',CR,LF,LF
03FD 504C414345 DB 'PLACE DISK TO BE FORMATTED IN',CR,LF
041C 4452495645 DB 'DRIVE 0 AND PRESS ENTER',CR,LF,LF
0436 202020204F DB ' OR',CR,LF,LF
043F 5052455353 DB 'PRESS SPACEBAR TO RETURN TO MENU $'

0461 D0D0A0A464F FMTING: DB CR,LF,LF, 'FORMATTING DISK, PLEASE WAIT...'
0483 D0D0A0A24 DB CR,LF,LF, '$'

0487 464F524D41 FMTDON: DB 'FORMAT COMPLETE',CR,LF,LF, '$'

049A 492043414E FMTERM: DB 'I CANNOT FORMAT THIS DISK!',CR,LF,LF, '$'

048B 5052455353 ANYKEY: DB 'PRESS ANY KEY TO CONTINUE $'

04D3 0C0A535953 SYMSG: DB CLS,LF, 'SYSTEM TRACK COPY UTILITY',CR,LF,LF
04F1 434F504945 DB 'COPIES SYSTEM TRACKS FROM MASTER DISK',CR,LF
0518 544F20534C DB 'TO SLAVE DISK'.CR,LF,LF, '$'

0529 494E534552 SRCMSG: DB 'INSERT MASTER DISK IN DRIVE 0',CR,LF, '$_'
0549 494E534552 DSTMSG: DB 'INSERT SLAVE DISK IN DRIVE 0',CR,LF,'$'
0569 5052455353 PRSMN: DB 'PRESS RETURN (OR SPACEBAR FOR MENU) $'
05BE 5359535445 SYSDON: DB 'SYSTEM TRACK COPY COMPLETE',CR,LF,LF,'$'
05AC 0C0A444953 BAKMSG: DB CLS,LF,'DISK BACKUP UTILITY',CR,LF,LF
05CA 5448452045 DB 'THE ENTIRE MASTER DISK IS ',CR,LF
05E0 434F504945 DB 'COPIED TO THE SLAVE DISK',CR,LF,LF
05FB 24 DB '$'
05FC 4241434B55 BAKDON: DB 'BACKUP COMPLETE',CR,LF,LF,'$'
060F 5052455353 RTNMSG: DB 'PRESS RETURN $'
061D 0D0A0A4449 WRMSG: DB CR,LF,LF,'DISK WRITE ERROR',CR,LF,'$'
0633 0D0A0A4449 RDMSG: DB CR,LF,LF,'DISK READ ERROR',CR,LF,'$'

0648 WTRACK DS 1
0649 INNER DS 1
064A OUTER DS 1
064B DS 32
0668 = STACK QU $
066B = MEM EQU $

***

Z80 Bootstrap Routine for the Commodore 64

COPYRIGHT © 1982
COMMODORE INTERNATIONAL

This routine is loaded from Track 1, Sector 5 of the Commodore 64 CP/M disk by a routine in BIOS65.

The load address is 0000H (with respect to the Z80 CPU). When the Z80 is enabled this program loads the Z80 BIOS and CCP and BDOS into RAM and jumps to it.

3400 = CCP EQU 3400H
;CCP EQU 0000H ;FOR MAKING BOOT0.HEX
;CCP EQU 0100H ;FOR MAKING BOOT1.HEX
001C = NSECTS EQU 1CH
F903 = TRACK EQU 0F903H
F902 = SECTOR EQU 0F902H
F904 = DISKNO EQU 0F904H
FCFF = IOTYPE EQU 0FCFFH ;IO SETUP BYTE IN BIOS65
4A33 = KYBDMD EQU CCP + 1633H ;CAPS LOCK FLAG
0000 = VICRD EQU 0
F900 = CMD EQU 0F900H
0001 = OFF EQU 01H
CE00 = MODESW EQU 0CE00H
F901 = DATA EQU 0F901H
F800 = BUFFER EQU 0FB00H
4A00 = BOOT EQU CCP + 1600H

; ORG 0000H ;Z80 RESET LOCATION

; 0000 00 NOP ;NOP REQUIRED FOR HARDWARE
0001 110034 LXI D,CCEP ;START OF LOAD ADDRESS
0004 3E00 MVI A,0
0006 3204F9 STA DISKNO ;LOAD IN FROM DRIVE A
0009 2601 MVI H,1 ;READ BEGINNING TRK 1, SEC 6
000B 2E06 MVI L,6
000D 7C LOAD1 MOV A,H
000E 3203F9 STA TRACK
0011 7D MOV A,L
0012 3202F9 STA SECTOR
0015 3E00 MVI A,VICRD ;SECTOR READ COMMAND
0017 3200F9 STA CMD
001A 3E01 MVI A,OFF
001C 3200CE STA MODESW ;TURN OFF SELF
001F 00 NOP
0020 3A01F9 LDA DATA ;WAS TRANSFER OK?
0023 B7 ORA A
0024 C20D00 JNZ LOAD1 ;JUMP IF NO

; OUTPUT '*' TO SHOW LOADING

0027 3E2A MVI A,/*
0029 3201F9 STA DATA
002C 3E03 MVI A,3
002E 3200F9 STA CMD
0031 3E01 MVI A,OFF
CP/M Version 2.2 System Relocator — 2/80

CP/M Relocator Program, Included with the Module To Perform the Move from 900H to the Destination Address

COPYRIGHT © 1980
DIGITAL RESEARCH

Modified for Use on the Commodore 64

MODIFICATIONS COPYRIGHT © 1982
COMMODORE INTERNATIONAL
011E C21801  JNZ  QLOOP  ;DO 16 TIMES

0121 118000  LXI  D, DMADDR
0124 CD0002  CALL SETDMA

0127 3E07  MVI  A, 07H  ;1200 BAUD DATA
0129 D300  OUT  0

012B 3E18  MVI  A, 18H
012D D306  OUT  6
012F 210001  LXI  H, 0100H
0132 CD0602  CALL SETUP
0135 21C103  LXI  H, 03CLH
0138 CD0602  CALL SETUP
013B 214404  LXI  H, 0444H
013E CD0602  CALL SETUP
0141 216805  LXI  H, 0568H
0144 CD0602  CALL SETUP

0147 115C00  LXI  D, FCB
014A CD1302  CALL OPEN
014D 116002  LXI  D, NOFILE
0150 3C  INR  A  ;WAS 255 IF NO FILE
0151 CAA201  JZ  DONE

0154 CDCC01  WTACK:  CALL SIN  ;WAIT FOR INITIAL ACK
0157 FE06  CPI  ACK
0159 C25401  JNZ  WTACK

015C 3E00  RDNEXT:  MVI  A, 0
015E 32BF02  STA  POINT  ;QUARTER SECTOR POINTER

0161 115C00  LXI  D, FCB
0164 CD1902  CALL READ
0167 B7  ORA  A
0168 C2BB01  JNZ  EOF

0168 CDAB01  AGAIN:  CALL SEND  ;SEND 32 BYTES

016E CDCC01  WTANS:  CALL SIN
0171 FE15  CPI  NAK
0173 CA6B01 JZ AGAIN ;BAD CHECKSUM, SEND AGAIN
0176 FE06 CPI ACK
0178 C26E01 JNZ WTANS ;IF NOT ACK, KEEP WAITING
0178 3A8F02 LDA POINT ;POINT TO QUARTER
017E C620 ADI 32
0180 328F02 STA POINT
0183 FE80 CPI 128
0185 CA5C01 JZ RDNEXT ;IF 0, READ ANOTHER SECTOR
0188 C36B01 JMP AGAIN ;SEND NEXT QUARTER
0188 3E3A EOT: MVI A,':' ;OUTPUT START OF STRING
018D CDF001 CALL SOUT
0190 3E30 MVI A,'0'
0192 CDF001 CALL SOUT
0195 3E30 MVI A,'0'
0197 CDF001 CALL SOUT
019A 3E0D MVI A,CR
019C CDF001 CALL SOUT
019F 117A02 LXI D,EBTRAN
01A2 CD1F02 DONE: CALL PRINT
01A5 C30000 JMP BOOT
01A8 3E3A SEND: MVI A,':'
01AA CDF001 CALL SOUT
01AD 3E20 MVI A,32
01AF CDD901 CALL SHOUT ;NUMBER OF DATA BYTES
01B2 0E00 MVI C,0 ;CLEAR CHECKSUM
01B4 218000 LXI H,DMADDR
01B7 3A8F02 LDA POINT ;POINT TO SECTOR QUARTER
01BA B5 ORA L
01BB 6F MOV L,A ;OR DATA INTO LSB

186 APPENDIX E
01BC 79  SEND1:  MOV  A,C ;FORM CHECKSUM
01BD 86  ADD  M
01BE 4F  MOV  C,A
01BF 7E  MOV  A,M ;GET CHARACTER

01C0 E5  PUSH  H ;SAVE ADDRESS
01C1 CDD901 CALL  SHOUT ;OUTPUT HEX DIGITS
01C1 E1  POP  H

01C5 2C  INR  L ;NEXT BYTE
01C6 7D  MOV  A,L
01C7 E61F  ANI  1FH ;CHECK FOR MOD 32
01C9 C2BC01 JNZ  SEND1 ;DO 32 TIMES

01CC 79  MOV  A,C ;FIX CHECKSUM
01CD EEF F  XRI  OFFH
01CF 3C  INR  A
01D0 CDD901 CALL  SHOUT

01D3 3E0D  MVI  A,0DH
01D5 CDF001 CALL  SOUT
01D8 C9  RET

01D9 F5  SHOUT:  PUSH  PSW
01DA 0F  RRC
01DB 0F  RRC
01DC 0F  RRC
01DD 0F  RRC
01DE CDE201 CALL  SNOUT ;OUTPUT HIGH NIBBLE

01E1 F1  POP  PSW
01E2 E60F  SNOUT:  ANI  0FH ;MASK OFF BITS
01E4 FE0A  CPI  10
01E6 DAE001 JC  SNUM
01E9 C637  ADI  'A'-10
01EB C3F001 JMP  SOUT

01EE C630  SNUM:  ADI  '0'

01F0 F5  SOUT:  PUSH  PSW
01F1 D806  SOUT1:IN  06H ;XEROX CHANNEL A CONTROL

APPENDIX E  187
01F3  E604    ANI    04H
01F5  CAF101  JZ     SOUT1

01FB  F1      POP    PSW
01F9  D304    OUT    04H    ;XEROX CHANNEL A DATA
01FB  C9      RET

01FC  DB06    SIN:   IN     6
01FE  E601    ANI    01H
0200  CAFC01  JZ     SIN
0203  DB04    IN     4
0205  C9      RET

0206  7C      SETUP: MOV     A,H
0207  D306    OUT    6
0209  7D      MOV     A,1
020A  D306    OUT    6
020C  C9      RET

020D  0E1A    SETDMA: MVI    C,26
020F  CD0500  CALL    BDOS
0212  C9      RET

0213  0E0F    OPEN:   MVI    C,15
0215  CD0500  CALL    BDOS
0218  C9      RET

0219  0E14    READ:   MVI    C,20
0218  CD0500  CALL    BDOS
021E  C9      RET

021F  0E09    PRINT:  MVI    C,9
0221  CD0500  CALL    BDOS
0224  C9      RET

0225  46494C454E NONAME: DB 'FILENAME MUST BE SPECIFIED',0DH,0DH,'$'

0242  4144424947 BADNAM: DB 'AMBIGUOUS FILES NOT ALLOWED',0DH,0DH,'$'

0260  492043414E NOFILE: DB 'I CANNOT FIND THAT FILE',0DH,0DH,'$'

188  APPENDIX E
I/O Configuration Utility for Commodore 64

COPYRIGHT © 1982
COMMODORE INTERNATIONAL

FC00 = IOMEM EQU 0FC00H
F800 = BUFFER EQU 0F800H
FCFF = IOTYPE EQU 0FCFFH
FC10 = FNBASE EQU 0FC10H
FD00 = KYBASE EQU 0FD00H
0001 = VICWR EQU 1
F900 = CMD EQU 0F900H
F901 = DATA EQU 0F901H
F902 = SECTOR EQU F902H
F903 = TRACK EQU 0F903H
F904 = DISKNO EQU 0F904H
F905 = KYCHAR EQU 0F905H
0033 = KYBDM D EQU 33H
0001 = CRPOS EQU 1
F28D = SHFTST EQU 0F28DH
0063 = LASTKY EQU 63H
0066 = MSGPTR EQU 66H
0009 = CONINV EQU 09H
0001 = OFF EQU 01H
CE00 = MODESW EQU 0CE00H

0000 = BOOT EQU 0000H
0005 = BDOS EQU 0005H
000C = CLS EQU 0CH
000D = CR EQU 0DH
000A = LF EQU 0AH
0100 = ORG 100H
START:  LXI SP, STACK ; INITIALIZE STACK PTR
LXI D, IOMSG
CALL PRINT

; 0109 3AFFFC  LDA IOTYPE
010C E601  ANI 01H ; # OF DISKS
010E C631  ADI '1' ; FORM ASCII
0110 5F  MOV E, A
0111 CD7601  CALL CONOUT

; 0114 11C204  LXI D, PRTMSG
0117 CD7101  CALL PRINT

; 011A 11D604  LXI D, P1515
011D 3AFFFC  LDA IOTYPE
0120 E602  ANI 02H ; CHECK PRINTER TYPE

; 0122 CA2801  JZ ST1 ; 1515 IF = 0

; 0125 11DD04  LXI D, P4022 ; 4022 IF = 1

; 0128 CD7101  ST1: CALL PRINT

; 0128 11E404  LXI D, CAPMSG
012E CD7101  CALL PRINT

; 0131 11FB04  LXI D, ONMSG ; ASSUME ON
0134 3AFFFC  LDA IOTYPE
0137 E620  ANI 20H ; BIT 5

; 0139 CA3F01  JZ ST2
013C 110005  LXI D, OFFMSG

; 013F CD7101  ST2: CALL PRINT

; 0142 110605  LXI D, MENU
0145 CD7101  CALL PRINT

; 0148 CD7B01  ST3: CALL KEYIN
0148 FE31  CPI 'I'
014D CA9201  JZ CHGDRV

190 APPENDIX E
0150 FE32  CPI  '2'
0152 CA9D01 JZ  CHRPRT

0155 FE33  CPI  '3'
0157 CAB601 JZ  CHGCAP

015A FE34  CPI  '4'
015C CAC001 JZ  CHGFNC

015F FE35  CPI  '5'
0161 CACD02 JZ  CHGKEY

0164 FE36  CPI  '6'
0166 CA1A04 JZ  SAVDSK

0169 FE37  CPI  '7'
016B CA0000 JZ  BOOT

016E C34801 JMP  ST3    ; NOT A VALID RESPONSE

0171 0E09 PRINT:  MVI  C,9
0173 C30500 JMP  BDOS

0176 0E02 CONOUT: MVI  C,2
0178 C30500 JMP  BDOS

017B 1EFF KEYIN:  MVI  E,OFFH
017D 0E06 MVI  C,6
017F C30500 JMP  BDOS

0182 2A0100 CONIN:  LHLD  BOOT + 1
0185 2E09 MVI  L,CONINV
0187 E9 PCHL

0188 3200F9 IO6510:  STA  CMD
018B 3E01 MVI  A,OFF
018D 3200CE STA  MODESW
0190 00 NOP
0191 C9 RET

0192 3AFFFC CHGDRV: LDA IOTYPE
0195 EE01   XRI   01H
0197 32FFFC  STA   IOTYPE
019A C30001  JMP   START

;  
019D 21FFFC  CHGPRT: LXI   H,IOTYPE
01A0 7E   MOV   A,M
01A1 E602  ANI   02H
01A3 CAAD01  JZ   CHGP1

;  
01A6 7E   MOV   A,M ;GET IOTYPE
01A7 E6F1  ANI   0F1H ;CLEAR BITS FOR 1515 PRINTER
01A9 77   MOV   M,A
01AA C30001  JMP   START

;  
01AD 7E   CHGP1: MOV   A,M ;GET IOTYPE
01AE E6FB  ANI   0FBH ;CLEAR BIT 2
01B0 F60A  ORI   0AH ;SET BITS FOR 4022 PRINTER
01B2 77   MOV   M,A
01B3 C30001  JMP   START

;  
01B6 21FFFC  CHGCAP: LXI   H,IOTYPE
01B9 7E   MOV   A,M
01BA EE20  XRI   20H ;INVERT BIT
01BC 77   MOV   M,A
01BD C30001  JMP   START

;  
01C0 11707  CHGFNC: LXI   D,FNKMSG
01C3 CD7101  CALL   PRINT

;  
01C6 3E00   MVI   A,0
01C8 325F08  STA   KYMODE
01CB 11A007  FNNEXT: LXI   D,FM1
01CE CD7101  CALL   PRINT
01D1 3A5F08  LDA   KYMODE
01D4 C631  ADI   '1'
01D6 5F   MOV   E,A
01D7 CD7601  CALL   CONOUT
01DA 11A407  LXI   D,FM2
01DD CD7101  CALL   PRINT

;  
01E0 CDAB02  CALL   CALCAD

192   APPENDIX E
01E3  7E      FN2      MOV    A,M
01E4  23      INX    H
01E5  FE20     CPI    20H
01E7  DAF301   JC    CTRL

01EA  5F      MOV    E,A
01EB  E5      PUSH   H
01EC  CD7601   CALL   CONOUT
01EF  E1      POP    H
01F0  C3E301   JMP    FN2

01F3  F5      CTRL.  PUSH   PSW
01F4  1E22     MVI    E,""
01F6  CD7601   CALL   CONOUT
01F9  F1      POP    PSW
01FA  FE00     CPI    0
01FC  CA0502   JZ     CRLF

01FF  11A907   LXI    D,CRM
0202  CD7101   CALL   PRINT
0205  11AE07   CRLF:  LXI    CD,CRLF
0208  CD7101   CALL   PRINT

020B  215F08   LXI    H,KYMODE
020E  34      INR    M
020F  7E      MOV    A,M
0210  FE08-'   CPI    8
0212  C2CB01   JNZ    FFNEXT
0215  11B107   LXI    D,FNINST
0218  CD7101   CALL   PRINT

021B  CD7B01   ASKAGN. CALL   KEYIN
021E  D631     SUI    '1'
0220  DA1802   JC     ASKAGN

0223  FE08     CPI    8
0225  CA0001   JZ     START
0228  D21B02   JNC    ASKAGN

022B  325F08   STA    KYMODE
022E 111C08  LXI    D,FM3
0231 CD7101  CALL   PRINT

0234 11A007  LXI    D,FM1
0237 CD7101  CALL   PRINT

023A 3A5F08  LDA    KYMODE ;GET CURRENT FN #
023D C631   ADI    '1' ;FORM ASCII
023F 5F     MOV    E,A
0240 CD7601  CALL   CONOUT
0243 11A407  LXI    D,FM2
0246 CD7101  CALL   PRINT
0249 CDA802  CALL   CALCAD
024C 225D08  SHLD   KYADDR

024F 3E00   MVI    A,0
0251 326208  STA    NUMCHR

0254 CD7B01  IN_LOOP: CALL KEYIN
0257 FE0D   CPI    0DH
0259 CA8502  JZ     ITSCR

025C FE08   CPI    0BH
025E CA8902  JZ     ITSB5

0261 FE1A   CPI    1AH
0263 CA9102  JZ     ITSCZ

0266 FE20   CPI    20H
0268 DA5402  JC     IN_LOOP

026B FE80   CPI    80H
026D D25402  JNC    IN_LOOP

0270 47     MOV    B,A ;SAVE CHAR
0271 3A6208  LDA    NUMCHR
0274 FE0F   CPI    15 ;IF ALREADY 15 CHAR,
0276 D25402  JNC    IN_LOOP ; NO ROOM FOR 00H

0279 C5     PUSH   B
027A 58     MOV    E,B

194  APPENDIX E
027B CD7601 CALL CONOUT
027E C1 POP B

027F CD9902 CALL OUTPUT
0282 C35402 JMP INLOOP ;GO FOR MORE

0285 47 ITSCR: MOV B,A ;SAVE CHAR
0286 3A6208 LDA NUMCHR
0289 FEOF CPI 15 ;NO ROOM IF 15 CHAR
028B D25402 JNC INLOOP

028E CD9902 CALL OUTPUT

0291 0600 ITSCZ: MVI B,0
0293 CD9902 CALL OUTPUT
0296 C3C001 JMP CHGFNC

0299 2A5D08 OUTPUT. LHLD KYADDR
029C 3A6208 LDA NUMCHR
029F 3C INR A
02A0 326208 STA NUMCHR
02A3 3D DCR A
02A4 85 ADD L ;ADD IN OFFSET
02A5 6F MOV L,A
02A6 70 MOV M,B
02A7 C9 RET

02A8 2110FC CALCAD: LXI H,FNBASE
02AB 1600 MVI D,0
02AD 3A5F08 LDA KYMODE
02B0 17 RAL
02B1 17 RAL
02B2 17 RAL
02B3 17 RAL
02B4 E6F0 ANI OF0H
02B6 5F MOV E,A
02B7 19 DAD D
02B8 C9 RET

0289 3A6208 ITSBS: LDA NUMCHR
028C FE00 CPI 0
02BE CA5402 JZ INLOOP ;IF 0 JUST GO TO LOOP

02C1 3D DCR A
02C2 326208 STA NUMCHR
02C5 326208 STA NUMCHR
02C5 1E08 MVI E,08H ;BACKSPACE
02C7 CD7601 CALL CONOUT
02CA C35402 JMP INLOOP

02CD 114306 CHGKEY LXI D,KYINST
02D0 CD7101 CALL PRINT

02D3 112F07 CK0 LXI D,PRMSG
02D6 CD7101 CALL PRINT

02D9 CD8201 CALL CONIN
02DC 2A0100 LHLD BOOT + 1
02DF 2E33 MVI L,KYBDMD ;UNSHIFT = 0, CAPS = 1
02E1 46 MOV B,M
02E2 3ABDF2 LDA SHFTST ;GET MODIFIER STATUS
02E5 E601 ANI 01H ;IS SHIFT KEY DOWN?
02E7 CAEC02 JZ CK1 ;JUMP IF NO

02EA 0602 MVI B,2 ;SHIFT = 2
02EC 3ABDF2 CK1 LXI LDA SHFTST
02EF E604 ANI 04H ;IS THE CONTROL KEY DOWN?
02F1 CAF602 JZ CK2 ;JUMP IF NO

02F4 0603 MVI B,3 ;CONTROL = 3
02F6 2A0100 CK2. LHLD BOOT + 1
02F9 2E63 MVI L,LASTKY
02FB 7E MOV A,M
02FC 326008 STA KYCHK ;SAVE FOR EXIT TEST
02FF 87 ADD A ;*2
0300 87 ADD A ;*4
0301 80 ADD B ;ADD IN OFFSET
0302 2100FD LXI H,KYBASE
0305 85 ADD L
0306 6F MOV L,A ;HL NOW HAS ADDRESS OF KEY

196 APPENDIX E
SHLD KYADDR ; ADDRESS OF KEY
MOV A, B ; 8 IS THE MODE
STA KYMODE

LHLD BOOT + 1
MVI L, MSGPTR
MVI M, 0
INX H
MVI M, 0 ; DISABLE MESSAGE MODE IF ANY

LXI D, ISMSG
CALL PRINT

LHLD KYADDR
MOV A, M ; GET KEY CODE
CALL PHEX , AND PRINT IN HEX

LXI D, INMSG
CALL PRINT

LDA KYMODE
LXI D, UNSH ; UNSHIFT MODE IF 0
CPI 0
JZ PMODE

LXI D, CAPS
CPI 1
JZ PMODE ; CAPS MODE IF 1

LXI D, SHIFT
CPI 2
JZ PMODE ; SHIFT MODE IF 2

LXI D, CONT ; MUST BE CONTROL MODE
PMODE: CALL PRINT

LXI D, MODE
CALL PRINT

CALL GHEX
0355 C26303 JNZ ASGKEY
; 0358 3A6008 LDA KYCHK ;NO CHARACTERS, 2 CR'S?
035B FE01 CPI CRPOS ;IS IT CR KEY POSITION?
035D CA0001 JZ START ;RESTART IF 2 CR'S
;
0360 C3D302 JMP CK0 ;NEXT KEY
0363 2A5D08 ASGKEY. LHLD KYADDR
0366 77 MOV MA ;PUT NEW CHARACTER IN MEMORY
0367 C3D302 JMP CK0
; 036A F5 PHEX: PUSH PSW ;SAVE CHARACTER
036B 0F RRC
036C 0F RRC
036D 0F RRC
036E 0F RRC
036F CD7303 CALL HEX ;PRINT TOP NIBBLE
;
0372 F1 POP PSW ;PRINT LOWER NIBBLE
;
0373 E60F HEX: ANI 0FH ;4 BITS
0375 FE0A CPI 10 ;LETTER OR NUMBER?
0377 DA8003 JC NUMBER
;
037A C637 ADI 'A' -10 ;MAKE HEX LETTER
037C FF MOV E,A
037D C37601 JMP CONOUT
;
0380 C630 NUMBER. ADI '0' ;MAKE ASCII NUMBER
0382 5F MOV E,A
0383 C37601 JMP CONOUT
;
0386 3E00 GHEX: MVI A,0
0388 326208 STA NUMCHR
;
038B CD8201 GHQ: CALL CONIN
038E FE0D CPI 0DH
0390 C2A503 JNZ GH1
;
0393 3A6208 LDA NUMCHR

198 APPENDIX E
0396 FE00  CPI  0
0398 C8   RZ

0399 FE02  CPI  2
039B C28B03 JNZ GH0

039E 3EFF  MVI A,0FFH
03A0 A7   ANA A
03A1 3A6108 LDA HEXIN
03A4 C9   RET

03A5 FE08  GH1: CPI 08H
03A7 C2CA03 JNZ GH4  ; JUMP NOT BACKSPACE

03AA 3A6208 LDA NUMCHR
03AD FE00  CPI 0
03AF CA8B03 JZ GH0

03B2 3D   DCR A
03B3 326208 STA NUMCHR
03B6 3A6108 LDA HEXIN
03B9 0F   RRC
03BA 0F   RRC
03BB 0F   RRC
03BC 0F   RRC
03BD E60F  ANI 0FH
03BF 326108 STA HEXIN
03C2 1E08  MVI E 08H
03C4 CD7601 CALL CONOUT
03C7 C38B03 JMP GH0

03CA 47  GH4: MOV B,A
03CB 3A6208 LDA NUMCHR
03CE FE02  CPI 2
03D0 CA8B03 JZ GH0

03D3 78   MOV A,B
03D4 FE30  CPI '0'
03D6 DA8B03 JC GH0
03D9 FE3A  CPI '9' + 1
03DB DAFF03 JC GOTNUM

APPENDIX E 199
03DE FE41   CPI   'A'
03E0 DA8803  JC   GH0

03E3 FE47   CPI   'F' + 1
03E5 DAF203  JC   GOTLET

03E8 FE61   CPI   'A'
03EA DA8803  JC   GH0

03ED FE67   CPI   'F' + 1
03EF D28803  JNC  GH0

03F2 F5   GOTLET  PUSH  PSW
03F3 5F   MOV   E,A
03F4 CD7601  CALL  CONOUT
03F7 F1   POP   PSW
03FB E60F   ANI   OFH
03FA C609   ADI   9
03FC C30504  JMP  MAKNUM

03FF F5   GOTNUM  PUSH  PSW
0400 5F   MOV   E,A
0401 CD7601  CALL  CONOUT
0404 F1   POP   PSW

0405 E60F  MAKNUM:  ANI   OFH
0407 47   MOV   B,A
0408 3A6108  LDA  HEXIN
0408 87   ADD   A
040C 87   ADD   A
040D 87   ADD   A
040E 87   ADD   A
040F 80   ADD   B
0410 326108  STA  HEXIN

0413 216208  LXI   H, NUMCHR
0416 34   INR   M
0417 C38803  JMP  GH0
041A 2100FC SAVDSK LXI H, IOMEM
041D 3E03 MVI A, 3
041F 3202F9 STA SECTOR
0422 1100F8 SAV: LXI D, BUFFER
    ;
0425 7E SAV1: MOV A, M
0426 12 STAX D
0427 23 INX H
0428 13 INX D
0429 7D MOV A, L
042A A7 ANA A
042B C22504 JNZ SAV1 ; 256 TIMES
    ;
042E 3E00 MVI A, 0
0430 3204F9 STA DISKNO
    ;
0433 3C INR A
0434 3203F9 STA TRACK
    ;
0437 3E01 MVI A, VICWR
0439 CD8801 CALL IO6510
043C 3A01F9 LDA DATA
043F A7 ANA A
0440 C25204 JNZ WRERR
    ;
0443 3A02F9 LDA SECTOR
0446 3C INR A
0447 3202F9 STA SECTOR
044A FE05 CPI 5
044C C22204 JNZ SAV2 ; WRITE SECTORS 3 AND 4
    ;
044F C30001 JMP START
    ;
0452 111306 WRERR: LXI D, WERMSG
0455 CD7101 CALL PRINT
0458 CD8201 CALL CONIN
0458 C30001 JMP START
    ;
    ; MESSAGES
    ;
CLS,LF,'COMMODORE 64 I/O CONFIGURATION
UTILITY' CR,LF,LF

'THE CURRENT I/O ASSIGNMENTS
ARE:',CR,LF,LF

' NUMBER OF DRIVES. $'
CR,LF

' PRINTER TYPE: $'

'1515',CR,LF,'$

'4022',CR,LF,'$

' INITIAL CAPS MODE. $

'ON',CR,LF,'$

'OFF',CR,LF,'$

'DO YOU WISH TO',CR,LF,LF

'1. CHANGE NUMBER OF DISK DRIVES',CR,LF

'2. CHANGE PRINTER TYPE',CR,LF

'3. CHANGE INITIAL CAPS MODE',CR,LF

'4. CHANGE FUNCTION KEY
ASSIGNMENTS',CR,LF

'5. CHANGE KEY CODES',CR,LF

'6. SAVE CURRENT I/O SETUP ON DISK',CR,LF

'7. RETURN TO CP/M',CR,LF,LF

'PLEASE ENTER SELECTION (1-7) $'

'DISK WRITE ERROR',CR,LF

'PRESS ANY KEY TO CONTINUE $'

'HEXADECIMAL AFTER "CHANGE
TO" ',CR,LF,LF

'TO EXIT KEY CODE MODE. TYPE
"RETURN" ',CR,LF

'TWICE AFTER "PRESS KEY" ',CR,LF,LF

'TO KEEP CURRENT KEY CODE, TYPE',CR,LF
070E 2020202252 DB "RETURN" AFTER "CHANGE TO" ',CR,LF,LF
072E 24 DB '$'

072F 0D0A505245 PRMSG: DB CR,LF,'PRESS KEY $'
;
073C 0D49532024 ISMSG: DB CR,'IS $'
;
0741 20494E2024 INMSG: DB 'IN $'
0746 4341505320 CAPS DB 'CAPS $'
074E 534849465 SHIFT: DB 'SHIFT $'
0756 434F4E5452 CONT. DB 'CONTROL$'
075E 554E534849 UNSH: DB 'UNSHIFT$'
0766 204D4F4445 MODE. DB 'MODE — CHANGE TO $'
;
0779 0C0A544845 FNKMSG: DB CLS,LF,'THE FUNCTION KEY ASSIGNMENTS ARE ',CR,LF,LF
079F 24 DB '$'
07A0 20204624 FM1 DB 'F$'
;
07A4 3A20202224 FM2 DB ': "$'
;
07A9 3C43523E24 CRM DB '<CR>$'
07AE 0D0A24 CRLFVM DB CR,LF,'$'
;
07B1 0A454E5445 FNINST DB LF,'ENTER FUNCTION KEY NUMBER (1-8) ',CR,LF
07D3 2020544F20 DB 'TO CHANGE PRESET VALUES.',CR,LF,LF
07F0 454E544552 DB 'ENTER 9 TO LEAVE FUNCTION',CR,LF
0808 20204B4559 DB 'KEY UTILITY. $'
;
081C 0D0A0A5459 FM3 DB CR,LF,LF,'TYPE IN TEXT. USING "RETURN" ',CR,LF
083D 20204F5220 DB 'OR "CTRL-Z" AS TERMINATOR.',CR,LF,LF,'$'
;
085D KYADDR DS 2 ;KEYBOARD LOOKUP ADDRESS
085F KYSODE DS 1 ;KEYBOARD MODE
0860 KYCHK DS 1
0861 HEXIN DS 1
0862 NUMCHR DS 1
0863 DS 32
0883 = STACK EQU $
SYSGEN — System Generation Program 8/79

System Generation Program, Version for MDS

COPYRIGHT © DIGITAL RESEARCH

MODIFICATIONS COPYRIGHT © 1982
COMMODORE INTERNATIONAL

Modified for use on Commodore 64. The system sectors run linearly from Track 1 Sector to Track 2 Sector 16.

0022 = NSECTS EQU 34 ;NO. OF SECTORS PER TRACK
0002 = NTRKS EQU 2 ;LAST OS TRACK + 1
0003 = NDISKS EQU 3 ;NUMBER OF DISK DRIVES
0080 = SECSIZ EQU 128 ;SIZE OF EACH SECTOR
0007 = LOG2SECS EQU 7 ;LOG 2 SECSIZ
0001 = SKEW EQU 1 ;SECTOR SKEW FACTOR

005C = FCB EQU 005CH ;DEFAULT FCB LOCATION
007C = FCBCR EQU FCB + 32 ;CURRENT RECORD LOCATION
0100 = TPA EQU 0100H ;TRANSIENT PROGRAM AREA
0900 = LOADP EQU 900H ;LOAD POINT FOR SYSTEM DURING LOAD/STORE

0005 = BDOS EQU 5H ;DOS ENTRY POINT
0000 = BOOT EQU 0 ;JMP TO 'BOOT' TO REBOOT SYSTEM

0001 = CONI EQU 1 ;CONSOLE INPUT FUNCTION
0002 = CONO EQU 2 ;CONSOLE OUTPUT FUNCTION
000E = SELF EQU 14 ;SELECT DISK
000F = OPENF EQU 15 ;DISK OPEN FUNCTION
0014 = DREADF EQU 20 ;DISK READ FUNCTION

000A = MAXTRY EQU 10 ;MAXIMUM NUMBER OF RETRIES ON EACH READ/WRITE
00DD = CR EQU 0DH ;CARRIAGE RETURN
000A = LF EQU 0AH ;LINE FEED
0010 = STACKSIZE EQU 16 ;SIZE OF LOCAL STACK
0001 = WBOOT EQU 1

204 APPENDIX E
ADDRESS OF WARM BOOT (OTHER PATCH ENTRY
POINTS ARE COMPUTED RELATIVE
TO WBOOT)

0018 = SELDSK EQU 24 ,WBOOT + 24 FOR DISK SELECT
001B = SETTRK EQU 27 ,WBOOT + 27 FOR SET TRACK
FUNCTION
001E = SETSEC EQU ,,130 ,WBOOT + 30 FOR SET SECTOR
FUNCTION
0021 = SETDMA EQU 33 ,WBOOT + 33 FOR SET DMA
ADDRESS
0024 = READF EQU 36 ,WBOOT + 36 FOR READ
FUNCTION
0027 = WRITF EQU 39 ,WBOOT + 39 FOR WRITE
FUNCTION

0100 ORG TPA ;TRANSIENT PROGRAM AREA
0100 C32302 JMP START
0103 434F505952 DB 'COPYRIGHT @ 1978, DIGITAL RESEARCH '

, 0128 02 OST DB NTRKS ,OPERATING SYSTEM TRACKS
0129 22 SPT: DB NSECTS ,SECTORS PER TRACK (CAN BE
PATCHED)

; GETCHAR:
; READ CONSOLE CHARACTER TO REGISTER A
012A 0E01CD0500 MVI C,CONIT ' CALL BDOS'
; CONVERT TO UPPER CASE BEFORE RETURN
012F FE61D8 CPI 'A' OR 20H ! RC ;RETURN IF BELOW LOWER CASE A
0132 FE78 CPI ('Z' OR 20H) + 1
0134 D0 RNC ;RETURN IF ABOVE LOWER CASE Z
0135 E65FC9 ANI 5FH! RET

; PUTCHAR.
; WRITE CHARACTER FROM A TO CONSOLE
013B 5F0E02CD05 MOV E,A! MVI C,CONO! CALL BDOS! RET

, CRLF. ;SEND CARRIAGE RETURN, LINE FEED
013F 3E0D MVI A,CR
0141 CD3801 CALL PUTCHAR
0144 3E0A MVI A,LF

APPENDIX E 205
0146 CD3801 CALL  PUTCHAR
0149 C9  RET

;  
014A E5CD3F01E1 PUSH H! CALL CRLF! POP H  
;DROP THRU TO OUTMSG0

OUTMSG:
014F 7EB7C8 MOV A,M! ORA A! RZ
; MESSAGE NOT YET COMPLETED
0152 E5CD3801E1 PUSH H! CALL PUTCHAR! POP H! INX H
0158 C34F01 JMP  OUTMSG

;  
015B 4F2A010011 MOV C,A! LHLD WBOOT! LXI D,SELD! DAD D! PCHL
;  
TRK:  ;SET UP TRACK
0164 2A0100 LHLD  WBOOT  ;ADDRESS OF BOOT ENTRY
0167 111B00 LXI  D,SETRK  ;OFFSET FOR SETTRK ENTRY
016A 19  DAD  D
016B E9  PCHL  ;GONE TO SETTRK

;  
SEC:  ;SET UP SECTOR NUMBER
016C 2A0100 LHLD  WBOOT
016F 111E00 LXI  D,SECTSEC
0172 19  DAD  D
0173 E9  PCHL

;  
DMA:  ;SET DMA ADDRESS TO VALUE OF B,C
0174 2A0100 LHLD  WBOOT
0177 112100 LXI  D,SETDMA
017A 19  DAD  D
017B E9  PCHL

;  
READ:  ;PERFORM READ OPERATION
017C 2A0100 LHLD  WBOOT
017F 112400 LXI  D,READF
0182 19  DAD  D
0183 E9  PCHL

;  
WRITE:  ;PERFORM WRITE OPERATION
; DREAD: :DISK READ FUNCTION
018E 0E14 MVI C,DREADF
0190 C30500 JMP BDOS

; OPEN: ,FILE OPEN FUNCTION
0193 0E0FC30500 MVI C,OPENF ! JMP BDOS

; GETPUT
, GET OR PUT CP/M (RW = 0 FOR READ, 1 FOR WRITE)
, DISK IS ALREADY SELECTED

; CLEAR TRACK TO 00
0190 21B008 LXI H,LOADP-80H ;SET UP INITIAL DMADDR
019B 225204 SHLD

; RWTRK: ;READ OR WRITE NEXT TRACK
01AF 214F04 LXI H,TRACK
01B2 34 INR M ;TRACK = TRACK + 1
01B3 3A2801 LDA OST ;NUMBER OF OPERATING SYSTEM TRACKS
01B6 BE CMP M ; = TRACK NUMBER ?
01B7 CA2202 JZ ENDRW ;END OF READ OR WRITE

; OTHERWISE NOTDONE, GO TO NEXT TRACK
01BA 4E MOV C,M ;TRACK NUMBER
01BB CD6401 CALL TRK ;TO SET TRACK
01BE 3EFD MVI A,OFFH ;COUNTS 0, 1, ... 33
01C0 325004 STA SECTOR ,SECTOR INCREMENTED BEFORE READ OR WRITE

RWSEC: ;READ OR WRITE SECTOR
01C3 3A2901 LDA SPT ;SECTORS PER TRACK
01C6 215004 LXI H,SECTOR
01C9 34 INR M ;TO NEXT SECTOR
01CA BE CMP M ;A = 34 AND M = 012 . 33 (USUALLY)

01CB CAAF01 JZ RWTRK ;
01CE 2A5204 LHLDM DMADDR ;SET UP DMA FOR NEXT ADDR
01D1 118000 LXI D,80H ;SECTOR SIZE
01D4 19 DAD D ,DMADDR = DMADDR + 80H
01D5 225204 SHLD DMADDR

READ OR WRITE SECTOR TO OR FROM CURRENT DMA ADDR
01D8 215004 LXI H,SECTOR
01DB 4E MOV C,M ;VALUE TO C READY FOR SELECT
01DC CD6C01 CALL SEC ;SET UP SECTOR NUMBER
01DF 2A5204 LHLDM DMADDR ;BASE DMA ADDRESS FOR THIS TRACK

01E2 44 MOV B,H
01E3 4D MOV C,L ;TO BC FOR SEC CALL
01E4 CD7401 CALL DMA ,DMA ADDRESS SET FROM B,C DMA ADDRESS SET, CLEAR RETRY COUNT

01E7 AF XRA A
01EB 325404 STA RETRY ;SET TO ZERO RETRIES

TRYSEC: ;TRY TO READ OR WRITE CURRENT SECTOR
01EB 3A5404 LDA RETRY
01EE FE0A CPI MAXTRY ;TOO MANY RETRIES?
01F0 DA0702 JC TRYOK

PAST MAXTRIES, MESSAGE AND IGNORE
01F3 21C303 LXI H ERRMSG
01F6 CD4F01 CALL OUTMSG
01F9 CD2A01 CALL GETCHAR
01FC FE0D CPI CR
01FE C20E03 JNZ REBOOT
TYPED A CR, OK TO IGNORE

0201 CD3F01 CALL CRLF
0204 C3C301 JMP RWSEC

TRYOK:

OK TO TRY READ OR WRITE

0207 3C INR A
0208 325404 STA RETRY ,RETRY = RETRY + 1
0208 3A5104 LDA RW ;READ OR WRITE?
020E B7 ORA A
020F CA1802 JZ TRYREAD

MUST BE WRITE

0212 CD8401 CALL WRITE
0215 C31B02 JMP CHKRW ;CHECK FOR ERROR RETURNS

TRYREAD:

CALL READ

CHKRW:

ORIA A
021C CAC301 JZ RWSEC ;ZERO FLAG IF R/W OK

ERROR, RETRY OPERATION

021F C3EB01 JMP TRYSEC

ENDRW. ;END OF READ OR WRITE, RETURN TO CALLER

0222 C9 RET

START.

START.

0223 317504 LXI SP,STACK ;SET LOCAL STACK POINTER
0226 212003 LXI H,SIGNON
0229 CD4F01 CALL OUTMSG

CHECK FOR DEFAULT FILE LOAD INSTEAD OF GET

022C 3A5D00 LDA FCB + 1 ;BLANK IF NO FILE
022F FE20 CPI '
0231 CA8102 JZ GETSYS ;SKIP TO GET SYSTEM MESSAGE
                  IF BLANK
0234 115C00  LXI  D,FCB    ;TRY TO OPEN IT
0237 CD9301  CALL  OPEN    ;
023A 3C     INR  A       ;255 BECOMES 00
023B C24702  JNZ  RDK    ;OK TO READ IF NOT 255
  ; FILE NOT PRESENT, ERROR AND REBOOT
023E 212004  LXI  H,NOSTATE
0241 CD4A01  CALL  CMSG    ;
0244 C30E03  JMP  REBOOT    
  ; FILE PRESENT
  ; READ TO LOAD POINT
  ;
  RDK:
0247 AF     XRA  A        
0248 327C00  STA  FCBCR   ;CURRENT RECORD = 0
  ;
  ; PRE-READ AREA FROM TPA TO LOADP
  ;
024B 0E10   MOV  C,(LOADP-TPA)/SECSIZ
  ;
  ; PRE-READ FILE
  PRERD
024D C5     PUSH  B       ;SAVE COUNT
024E 115C00  LXI  D,FCB    ;INPUT FILE CONTROL COUNT
0251 CD8E01  CALL  DREAD    ;ASSUME SET TO DEFAULT BUFFER
0254 C1     POP  B        ;RESTORE COUNT
0255 B7     ORA  A       
0256 C27B02  JNZ  BADRD   ;CANNOT ENCOUNTER END-OF
  ; FILE
0259 0D     DCR  C       ;COUNT DOWN
025A C24D02  JNZ  PRERD   ;FOR ANOTHER SECTOR
  ;
  ; SECTORS SKIPPED AT BEGINNING OF FILE
  ;
025D 210009  LXI  H,LOADP
  RDINP:
0260 E5     PUSH  H       
0261 44     MOV  B,H     
0262 4D     MOV  C.L     ;READY FOR DMA
0263 CD7401  CALL  DMA    ;DMA ADDRESS SET
0266 115C00  LXI  D,FCB  ;READY FOR READ
0269 CD8E01  CALL  DREAD  
026C E1  POP  H  ;RECALL DMA ADDRESS
026D B7  ORA  A  ;00 IF READ OK
026E C2C702  JNZ  PUTSYS  ;ASSUME EOF IF NOT.

; MORE TO READ, CONTINUE
0271 118000  LXI  D,SECSIZ
0274 19  DAD  D  ;HL IS NEW LOAD ADDRESS
0275 C36002  JMP  RDINP

; BADRD:  ,EOF ENCOUNTERED IN INPUT FILE
0278 213704  LXI  H,BADFILE
027B CD4A01  CALL  CRMSG
027E C30E03  JMP  REBOOT

; GETSYS:
0281 212F03  LXI  H,ASKGET  ;GET SYSTEM?
0284 CD4A01  CALL  CRMSG
0287 CD2A01  CALL  GETCHAR
028A FE0D  CPI  CR
028C CAC702  JZ  PUTSYS  ;SKIP IF CR ONLY

; 028F D641  SUI  'A'  ;NORMALIZE DRIVE NUMBER
0291 FE03  CPI  NDISKS  ;VALID DRIVE?
0293 DA9C02  JC  GETC  ;SKIP TO GETC IF SO

; INVALID DRIVE NUMBER
0296 CD1903  CALL  BADDISK
0299 C38102  JMP  GETSYS  ;TO TRY AGAIN

; GETC
; SELECT DISK GIVEN BY REGISTER A
029C C641  ADI  'A'
029E 325F03  STA  GDISK  ;TO SET MESSAGE
02A1 D641  SUI  'A'
02A3 CD5B01  CALL  SEL  ;TO SELECT THE DRIVE

; GETSYS, SET RW TO READ AND GET THE SYSTEM
02A6 CD3F01  CALL  CRLF
02A9 215503  LXI  H,GETMSG
02AC CD4F01  CALL  OUTMSG

APPENDIX E  211
CALL GETCHAR
CPI CR
JNZ REBOOT
CALL CRLF
XRA A
STA RW
CALL GETPUT
LXI H, DONE
CALL OUTMSG
;
;
PUT SYSTEM
;
PUTSYS:
LXI H,ASKPUT
CALL CRMSG
CALL GETCHAR
CPI CR
JZ REBOOT
SUI 'A'
CALL NDISK5
JC PUTC
;
INVALID DRIVE NAME
CALL BADDDISK
JMP PUTSYS ,TO TRY AGAIN
;
PUTC.
;
SET DISK FROM REGISTER C
ADI 'A'
STA PDISK ;MESSAGE SET
SUI 'A'
CALL SEL ;SELECT DEST DRIVE
;
;
LXI H,PUTMSG
CALL CRMSG
CALL GETCHAR
CPI CR
JNZ REBOOT
CALL CRLF
;
LXI H,RW
0300 3601 MVI M,1
0302 CD9B01 CALL GETPUT ;TO PUT SYSTEM BACK ON DISKETTE
0305 21EA03 LXI HDONE
0308 CD4F01 CALL OUTMSG
030B C3C702 JMP PUTSYS ;FOR ANOTHER PUT OPERATION

REBOOT:
030E 3E00 MVI A,0
0310 CD5B01 CALL SEL
0313 CD3F01 CALL CRLF
0316 C30000 JMP BOOT

BADDISK:
0319 21FC03 LXI HQDISK
031C CD4A01 CALL CRMSG
031F C9 RET

DATA AREAS
MESSAGES
0320 5359534745 SIGNON- DB 'SYSGEN VER'
032B 322E30 DB VERS/0+'0',',',VERS MOD 10+'0'
032E 00 DB 0
032F 534F555243 ASKGET DB 'SOURCE DRIVE NAME'
0340 0D284F5220 DB 0DH,'(OR RETURN TO SKIP) ',0
0355 534F555243 GETMSG- DB 'SOURCE ON '
035F GDISK: DS 1 ;FILLED IN AT GET FUNCTION
0360 2C20544845 DB ' ', THEN TYPE RETURN',0
0373 4445535449 ASKPUT DB 'DESTINATION DRIVE NAME'
0389 0D284F5220 DB 0DH,'(OR RETURN TO REBOOT) ',0
03A0 4445535449 PUTMSGS. DB 'DESTINATION ON '
03AF PDISK: DS 1 ;FILLED IN AT PUT FUNCTION
03B0 2C20544845 DB ' ', THEN TYPE RETURN',0
03C3 5045524D41 ERRMSG: DB 'PERMANENT ERROR, TYPE RETURN TO IGNORE',0
03EA 46554E4354 DONE: DB 'FUNCTION COMPLETE',0
03FC 494E56414C QDISK: DB 'INVALID DRIVE NAME (USE A, B, OR C) ',0
0420 4E4F20534F NOFILE DB 'NO SOURCE FILE ON DISK',0
BADFILE:
Custom BIOS for CP/M 2.2 On Commodore 64

COPYRIGHT © 1982
COMMODORE INTERNATIONAL

This version has the following attributes:

1. Memory map set up for 52K RAM system with I/O and drivers by BOOT65
2. Disk tables and vectors included for 2 drives
3. The Intel I/O byte is not implemented
4. Punch and reader are null routines
5. Keyboard and message tables are part of BIOS65
6. A 20K to 48K byte CP/M environment can be supported on the Commodore 64 (44K with IEEE)
7. Virtual Drive B is supported for 1540
8. Drive B is not virtual on IEEE disk

0000 = BASE EQU 0000H ;BEGINNING OF ADDRESSABLE RAM

002C = MSIZE EQU 44 ;CP/M VERSION MEMORY SIZE IN KILOBYTES

"BIAS" IS ADDRESS OFFSET FROM 3400H FOR MEMORY SYSTEMS
THAN 20K (REFERRED TO AS "B" THROUGHOUT THE TEXT)

6000 = BIAS EQU (MSIZE-20) *1024

NOTE: TO CREATE MOVCPM, THE FOLLOWING CCP EQUATES ARE USED:

;CCP EQU 0000H ;FOR BIOSO.HEX
;CCP EQU 0100H ;FOR BIOS1.HEX

9400 = CCP EQU 3400H + BIAS ;BASE OF CCP
9C06 = BDOS EQU CCP + 806H BASE OF BDOS
AA00 = BIOS EQU CCP + 1600H BASE OF BIOS
0004 = CDISK EQU BASE + 0004H CURRENT DISK NUMBER 0 = A, ...
0003 = IOBYTE EQU BASE + 0003H INTEL I/O BYTE
0000 = TRANS EQU 0000H ;0 IMPLIES NO TRANSLATION
0005 = ENTRY EQU 0005H ;BDOS ENTRY VECTOR

Z80 INSTRUCTIONS

0018 = JR EQU 18H
0038 = JRC EQU 38H
0030 = JRNC EQU 30H
0028 = JRZ EQU 28H
0020 = JRNZ EQU 20H

THE FOLLOWING EQUATES DEFINE THE COMMON MEMORY FOR PASSING DATA TO AND FROM THE 6510
I/O ROUTINES

F800 = HSTBUF EQU 0F800H ;256 BYTE DISK BUFFER
F900 = CMD EQU 0F900H ;COMMAND REGISTER
F901 = DATA EQU 0F901H ;DATA REGISTER
F902 = SECTOR EQU 0F902H ;SECTOR REGISTER
F903 = TRACK EQU 0F903H ;TRACK REGISTER
F904 = DISKNO EQU 0F904H ;DRIVE NUMBER REGISTER
F905 = KYCHAR EQU 0F905H ;KEYBOARD CHARACTER REGISTER
FCFF = IO TYPE EQU 0FCFFH ,I/O CONFIGURATION BYTE

; THE Z80 SHUTS ITSELF OFF BY WRITING "OFF" TO THE
; LOCATION "MODESW"

0001 = OFF EQU 1
CE00 = MODESW EQU 0CE00H

; THE FOLLOWING ARE THE COMMANDS TO THE 6510 I/O
; ROUTINES

0000 = VICRD EQU 0 ,READ SPECIFIED SECTOR
0001 = VICWR EQU 1 ,WRITE SPECIFIED SECTOR
0002 = VICIN EQU 2 ,DO A KEYBOARD SCAN
0003 = VICOUT EQU 3 ,OUTPUT DATA TO SCREEG
0004 = VICPS T EQU 4 ,GET PRINTER STATUS
0005 = VICT PRT EQU 5 ,SEND CHARACTER TO PRINTER
0006 = VICFMT EQU 6 ,FORMAT DISK COMMAND
0007 = AUX1 EQU 7 ,JUMP TO $0E00 IN 6510 SPACE
0008 = AUX2 EQU 8 ,JUMP TO $0F00 IN 6510 SPACE
0009 = INDIR EQU 9 ,JUMP INDIRECT VIA 0F906

AA00 ORG BIOS ,ORIGIN OF THIS PROGRAM
0016 = NSECTS EQU ($-CCP)/256 ;WARM START SECTOR COUNT

; JUMP VECTOR FOR INDIVIDUAL SUBROUTINES
AA00 C36CAA JMP BOOT ,COLD START
AA03 C31DAB WBOOTE: JMP WBOOT ,WARM START
AA06 C39AAB JMP CONST ,CONSOLE STATUS
AA09 C3FEAB JMP CONIN ,CONSOLE CHARACTER IN
AA0C C376AC JMP CONOUT ,CONSOLE CHARACTER OUT
AA0F C3B1AC JMP LIST ,LIST CHARACTER OUT
AA12 C3FAAC JMP PUNCH ,PUNCH CHARACTER OUT
AA15 C3FDAC JMP READER ,READER CHARACTER OUT
AA18 C302AD JMP HOME ,MOVE HEAD TO HOME POSITION
AA1B C30CAD JMP SELDISK ,SELECT DISK
AA1E C320AD JMP SETTRK ,SET TRACK NUMBER
AA21 C326AD JMP SETSEC ,SET SECTOR NUMBER
AA24 C32BAD JMP SETDMA ,SET DMA ADDRESS
AA27 C334AD JMP READ ,READ RISK
<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA2A C347AD</td>
<td>JMP</td>
<td>WRITE</td>
<td>WRITE DISK</td>
</tr>
<tr>
<td>AA2D C3D1AC</td>
<td>JMP</td>
<td>LISTST</td>
<td>RETURN LIST STATUS</td>
</tr>
<tr>
<td>AA30 C331AD</td>
<td>JMP</td>
<td>SECTAN</td>
<td>SECTOR TRANSLATE</td>
</tr>
<tr>
<td>AA33 00</td>
<td>DB</td>
<td>00H</td>
<td>CAPS LOCK FLAG</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FIXED DATA TABLES FOR TWO DRIVES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DISK PARAMETER HEADER FOR DISK 00</td>
</tr>
<tr>
<td>AA34 00000000</td>
<td>DW</td>
<td>TRANS</td>
<td>TRANS, 0000H</td>
</tr>
<tr>
<td>AA38 00000000</td>
<td>DW</td>
<td>0000H</td>
<td>0000H, 0000H</td>
</tr>
<tr>
<td>AA3C F0AE54AA</td>
<td>DW</td>
<td>DIRBF</td>
<td>DIRBF, DPBLK</td>
</tr>
<tr>
<td>AA40 AEAF70AF</td>
<td>DW</td>
<td>CHK00</td>
<td>CHK00, ALLO0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DISK PARAMETER HEADER FOR DISK 01</td>
</tr>
<tr>
<td>AA44 00000000</td>
<td>DW</td>
<td>TRANS</td>
<td>TRANS, 0000H</td>
</tr>
<tr>
<td>AA48 00000000</td>
<td>DW</td>
<td>0000H</td>
<td>0000H, 0000H</td>
</tr>
<tr>
<td>AA4C F0AE54AA</td>
<td>DW</td>
<td>DIRBF</td>
<td>DIRBF, DPBLK</td>
</tr>
<tr>
<td>AA50 BEAF8FAF</td>
<td>DW</td>
<td>CHK01</td>
<td>CHK01, ALLO1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DISK PARAMETER BLOCK, COMMON TO ALL DISKS</td>
</tr>
<tr>
<td>AA54 2200</td>
<td>DW</td>
<td>34</td>
<td>SECTORS PER TRACK</td>
</tr>
<tr>
<td>AA56 03</td>
<td>DB</td>
<td>3</td>
<td>BLOCK SHIFT FACTOR</td>
</tr>
<tr>
<td>AA57 07</td>
<td>DB</td>
<td>7</td>
<td>BLOCK MASK</td>
</tr>
<tr>
<td>AA58 00</td>
<td>DB</td>
<td>0</td>
<td>NULL MASK</td>
</tr>
<tr>
<td>AA59 8700</td>
<td>DW</td>
<td>135</td>
<td>DISK SIZE-1</td>
</tr>
<tr>
<td>AA5B 3F00</td>
<td>DW</td>
<td>63</td>
<td>DIRECTORY MAX</td>
</tr>
<tr>
<td>AA5D C0</td>
<td>DB</td>
<td>192</td>
<td>ALLOC 0</td>
</tr>
<tr>
<td>AA5E 00</td>
<td>DB</td>
<td>0</td>
<td>ALLOC 1</td>
</tr>
<tr>
<td>AA5F 1000</td>
<td>DW</td>
<td>16</td>
<td>CHECK SIZE</td>
</tr>
<tr>
<td>AA61 0200</td>
<td>DW</td>
<td>2</td>
<td>TRACK OFFSET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>END OF FIXED TABLES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MEMORY INITIALIZED WHEN BIOS READ IN AT BOOT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TIME</td>
</tr>
<tr>
<td>AA63 40</td>
<td>DB</td>
<td>40H</td>
<td>VECTOR OF LAST KEY PRESSED</td>
</tr>
<tr>
<td>AA64 00</td>
<td>DB</td>
<td>00H</td>
<td>CAPS LOCK HOUSEKEEPING</td>
</tr>
<tr>
<td>AA65 00</td>
<td>DB</td>
<td>00H</td>
<td>CHARACTER AVAILABLE FLAG</td>
</tr>
<tr>
<td>AA66 0000</td>
<td>DW</td>
<td>0000H</td>
<td>MESSAGE POINTER</td>
</tr>
<tr>
<td>AA68 00FD</td>
<td>DW</td>
<td>0FD00H</td>
<td>KEYBOARD CODE TABLE</td>
</tr>
</tbody>
</table>

APPENDIX E 217
AA6A 00FC  MSGTBL:  DW  OFC00H  ;MESSAGE VECTOR TABLE

;  MISC. CONSOLE EQUATES

F28D  =  SHFTST  EQU  0F28DH  ;CONTROL,COMMODORE,SHIFT KEYS
FOCC  =  FLASH  EQU  0FOCCH  ;CURSOR FLASH ENABLE
FOCF  =  CURSOR  EQU  0FOCFH  ;CURSOR CHARACTER

;  INDIVIDUAL SUBROUTINES TO PERFORM EACH FUNCTION

BOOT:

AA6C 3E20  MVI  A, 20H  ;ASCII SPACE
AA6E 32CFF0  STA  CURSOR  ;SET UP CURSOR
AA71 AF  XRA  A  ;ZERO IN THE ACCUM
AA72 320300  STA  IOBYTE  ;CLEAR THE IOBYTE
AA75 320400  STA  CDISK  ;SELECT DISK ZERO
AA78 32EFAE  STA  CURDSK  ;CLEAR VIRTUAL DISK POINTER
AA7B 32E1AE  STA  HSTACT  ;HOST BUFFER INACTIVE
AA7E 32E3AE  STA  UNACNT  ;CLEAR UNALLOC COUNT
AA81 3EC3  MVI  A, 0C3H  ;C3 IS JUMP OPCODE
AA83 320000  STA  0 + BASE  ;FOR JUMP TO WBOOT
AA86 2103AA  LXI  H, WBOOTE  ;WBOOT ENTRY POINT
AA89 220100  SHLD  1 + BASE  ;SET ADDRESS FIELD

AA8C 320500  STA  5 + BASE  ;JUMP TO BDOS OPCODE
AA8F 21069C  LXI  H, BDOS  ;BDOS ENTRY POINT
AA92 220600  SHLD  6 + BASE  ;SET ADDRESS FIELD

AA95 018000  LXI  B, 80H + BASE  ;DEFAULT DMA ADDRESS
AA98 CD2BAD  CALL  SETDMA

AA9B 11A6AA  LXI  D, SIGNON  ;DE POINTS TO SIGNON MSG
AA9E 0E09  MVI  C, 9  ;PRINT STRING FUNCTION
AAA0 CD0500  CALL  ENTRY  ;GO TO BDOS
AAA3 C389AB  JMP  GOCPM1  ;GET READY FOR CCP

AAA6 0C0A  SIGNON:  DB  0CH, 0AH  ;CLEAR SCREEN
AAA8 2020202043  DB  ' COMMODORE 64 20K CP/M VERS 2 2'
AACC 0D0A0A  DB  0DH, 0AH, 0AH
AACF 2020436F70  DB  ' COPYRIGHT @ 1979, DIGITAL RESEARCH', 0DH, 0AH

218 APPENDIX E
WBOOT:

A1D 318000 LXI SP, 80H + BASE ;USE SPACE BELOW BUFFER FOR STACK
A20 0E00 MVI C, 0 ;SELECT DISK 0
A22 CD0CAD CALL SELDSK
A25 AF XRA A ;FORCE DRIVE A
A26 3204F9 STA DISKNO ;ABSOLUTELY, POSITIVELY
A29 CD79AE CALL CHGDSK ;IF NOT ALREADY SELECTED
A2C CD02AD CALL HOME ;GO TO TRACK 00
A2F 3E0D MVI A, 0DH ;CARRIAGE RETURN
A31 CDAAAC CALL COUT5 ;OUTPUT IT

A34 110094 LXI D, CCP ;START OF LOAD
A37 0616 MVI B, NSECTS
A39 2601 MVI H, 1 ;TRACK NUMBER
A3B 2E06 MVI L, 6 ;SECTOR NUMBER
A3D 7C LOAD1: MOV A, H
A3E 3203F9 STA TRACK
A41 7D MOV A, L
A42 3202F9 STA SECTOR
A45 3E00 MVI A, VICRD ;DISK READ COMMAND
A47 CD90AB CALL IO6510

A4A 3A01F9 LDA DATA
A4D B7 ORA A
A4E 20ED J1: DB JRNZ, (LOAD1-J1-2) AND OFFH
A50 E5 PUSH H
A51 C5 PUSH B
A52 010001 LXI B, 256
A55 2100F8 LXI H, HSTBUF ;DISK BUFFER
A58 ED DB 0EDH ;LDIR INSTRUCTION
A59 B0 DB 0BDH
A5A 0E2A MVI C, '*' ;SHOW IT'S LOADING
A5C CD76AC CALL CONOUT
A5F C1 POP B
A60 E1 POP H
A61 05 DCR B ;DECREMENT SECTOR COUNT
AB62 280B J2 DB JRZ,GOCPM-J2-2
AB64 2C INR L ;NEXT SECTOR
AB65 7D MOV A,L
AB66 FE11 CPI 17
AB68 38D3 J3: DB JRC, (LOAD1-J3-2) AND 0FFH
AB6A 24 INR H
AB6B 2E00 MVI L,0
AB6D 18CE J4: DB JR, (LOAD1-J4-2) AND 0FFH
; END OF LOAD OPERATION, SET PARAMETERS AND GO TO CP/M
GOCPM:

AB6F 3EC3 MVI A,0C3H ,C3 IS A JMP INSTRUCTION
AB71 320000 STA 0+BASE ,FOR JMP TO WBOOT
AB74 2103AA LXI H,WBOOTE ,WBOOT ENTRY POINT
AB77 220100 SHLD 1+BASE ;SET ADDRESS FIELD FOR JMP AT 0

AB7A 320500 STA 5+BASE ;FOR JMP TO BDOS
AB7D 21069C LXI H,BDOS ;BDOS ENTRY POINT
AB80 220600 SHLD 6+BASE ;ADDRESS FIELD OF JUMP AT 5 TO BDOS

AB83 018000 LXI B,80H+BASE ;DEFAULT DMA ADDRESS IS 80H
AB86 CD2BAD CALL SETDMA

AB89 3A0400 GOCPM1: LDA CDISK ;GET CURRENT DISK NUMBER
AB8C 4F MOV C,A ;SEND TO THE CCP
AB8D C30094 JMP CCP ;GO TO CP/M FOR FURTHER PROCESSING

; MAIN ROUTINE TO TRANSFER EXECUTION TO 6510

AB90 3200F9 I06510: STA CMD ;PUT A IN 6510 COMMAND REGISTER
AB93 3E01 MVI A,OFF
AB95 3200CE STA MODESW ;TURN OFF Z80
AB98 00 NOP ;REQUIRED BY HARDWARE
AB99 C9 RET

220 APPENDIX E
CONST :CONSOLE STATUS, RETURN 0FFH IF CHARACTER READY,
00H IF NOT

AB9A 2A66AA LHLHMSGPTR ,MESSAGE MODE?
AB9D 7C MOV A,H
AB9E B5 ORAL
AB9F 3EFF MVI A,OFFH ,DATA READY FLAG
ABA1 C0 RNZ ,RETURN IF MSGPTR<>0

ABA2 3A65AA LDA CSTAT ,ALREADY A CHAR?
ABA5 A7 ANAA
ABA6 C0 RNZ ,YES IF NOT 0

ABA7 3E02 MVI A,VICIN ,CHECK KEYBOARD COMMAND
ABA9 CD90AB CALL IO6510

ABAC 3A8DF2 LDA SHFTST ,GET STATUS OF CONTROL KEYS
ABAF E602 ANI 02H ,CHECK FOR COMMODORE KEY
ABB1 2810 J5. DB JRZ,CONST0-J5-2 ;JUMP IF NOT Pressed

ABB3 3A64AA LDA TOGGLE ,IS THIS AN UPSTROKE?
ABB6 A7 ANAA
ABB7 200A J6. DB JRNZ,CONST0-J6-2 ,NO WAITING TO
RELEASE

ABB9 3A33AA LDA KYBMDM ,GET CAPS MODE FLAG
ABBC EE01 XRIO1H ,TOGGLE MODE BIT
ABBE 3233AA STA KYBMDM
ABC1 3E01 MVI A.1
ABC3 3264AA CONST0 STA TOGGLE

ABC6 3A05F9 LDA KYCHAR ,GET SCANNED DATA
ABC9 FE3A CPI 3AH ,BAD CONTROL DATA
ACB8 280A J7. DB JRZ,CONST1-J7-2

ABCD FE3D CPI 3DH ,BAD CONTROL DATA
ABCF 2806 J8. DB JRZ,CONST1-J8-2

ABD1 2163AA LXI H, LASTKY ,COMPARE WITH PREVIOUS
ABD4 BE CMP M ,SCAN DATA
ABD5 2005 J9. DB JRNZ,CONST2-J9-2 ,IF DIFFERENT, NEW KEY

APPENDIX E 221
ABD7 AF  CONST1: XRA A ;DATA NOT READY FLAG
ABD8 3265AA  STA CSTAT ;SAVE FOR LATER
ABDB C9  ;RET

ABDC F5  CONST2: PUSH PSW
ABDD 01F401  LXI B,500
ABE0 0B  CONST3: DCX B ;DELAY FOR KEYBOUNCE
ABE1 79  MOV A,C
ABE2 B0  , ORA B
ABE3 20FB  J10: DB JRNZ,(CONST3-J10-2) AND 0FFH

ABE5 3E02  MVI A,VICIN ;GET CHARACTER AGAIN
ABE7 CD90AB  CALL IO6510

ABEA F1  POP PSW
ABEB 2105F9  LXI H,KYCHAR
ABEE BE  CMP M
ABEF 20E6  J11: DB JRNZ,(CONST1-J11-2) AND 0FFH ;IF<>0, BOUNCING

ABF1 3263AA  STA LASTKY ;UPDATE LAST KEY
ABF4 FE40  CPI 40H ;IF 40H, NO KEY PRESSED
ABF6 28DF  J12: DB IRZ,(CONST1-J12-2) AND 0FFH

ABF8 3EFF  MVI A,OFFH ;DATA READY FLAG
ABFA 3265AA  STA CSTAT ;SAVE FOR LATER
ABFD C9  ;RET

CONIN: ;CONSOLE CHARACTER INTO REGISTER A
ABFE 3E00  MVI A,0 ;TURN ON CURSOR
AC00 32CCF0  STA FLASH

AC03 2A66AA  LHLD MSGPTR ;ARE WE IN MESSAGE MODE?
AC06 7C  MOV A,H
AC07 B5  ORA L
AC08 2044  J13: DB JRNZ,CONIN5-J13-2

AC0A CD9AAB  CONIN1. CALL CONST ;CHECK CONSOLE STATUS
AC0D B7  ORA A
AC0E 28FA  J14: DB JRZ,(CONIN1-J14-2) AND 0FFH ;UNTIL NEW

222 APPENDIX E
AC10 AF XRA A
AC11 3265AA STA CSTAT ;CLEAR CSTAT
AC14 3A33AA CONIN2 LDA KYBDMD ;UNSHIFT = 0, CAPS = 1
AC17 47 MOV B,A
AC18 3A8DF2 LDA SHFTST ;GET MODIFIER STATUS
AC1B E601 ANI 01H ;IS A SHIFT KEY DOWN?
AC1D 2802 J15 DB JRZ,CONIN3-J15-2 ;JUMP IF NO

AC1F 0602 MVI B,2 ;SHF1T = 2
AC21 3A8DF2 CONIN3 LDA SHFTST ;GET MODIFIER STATUS
AC24 E604 ANI 04H ;IS THE CONTROL KEY DOWN?
AC26 2802 J16 DB JRZ,CONIN4-J16-2 ;JUMP IF NO

AC28 0603 MVI B,3 ;CONTROL = 3
AC2A 3A63AA CONIN4 LDA LASTKY ;GET KEY POSITION
AC2D 87 ADD A ;*2
AC2E 87 ADD A ;*4
AC2F 80 ADD B ;ADD IN OFFSET
AC30 2A68AA LHLD TBLPTR ;GET BEGINNING OF KEYTL
AC33 85 ADD L ;VECTOR INTO TABLE
AC34 6F MOV L,A
AC35 3E00 MVI A,0
AC37 8C ADC H
AC38 67 MOV H,A
AC39 7E MOV A,M ;GET CHARACTER FROM TABLE
AC3A FE80 CPI 80H ;MESSAGE IF >7FH
AC3C 3820 J17 DB JRZ,CONIN7-J17-2 ;JUMP IF ASCII CHAR

AC3E 2A6AAA LHLD MSGTB1 ;GET BEGINNING OF MVTBL
AC41 E67F ANI 7FH ;STRIP OF MESSAGE BIT
AC43 87 ADD A ;*2
AC44 85 ADD L ;VECTOR INTO TABLE
AC45 6F MOV L,A
AC46 3E00 MVI A,0
AC48 8C ADC H
AC49 67 MOV H,A
AC4A 7E MOV A,M ;LOW ORDER BYTE
AC4B 23 INX H
AC4C 66 MOV H,M ;HIGH ORDER BYTE
AC4D 6F  MOV   L,A
AC4E 46  CONINS: MOV   B,M     ;GET CHARACTER
AC4F 23  INX   H     ;CHECK NEXT CHARACTER
AC50 7E  MOV   A,M
AC51 A7  ANA   A
AC52 2003 J18:  DB    JRNZ,CONINS6-J18-2   ;IF 0, B HAS LAST CHAR
               
AC54 210000 LXI   H,0000H  ;END OF MESSAGE MODE
AC57 2266AA CONINS6 SHLD  MSGPTR  ;SAVE MESSAGE POINTER
AC5A 78  MOV   A,B   ;CHECK CHARACTER
AC5B A7  ANA   A     ;MAYBE 1ST IS 0
AC5C 28AC J19:  DB    JRZ,(CONINS1-J19-2) ND 0FFH  ;IF<>0, NOT CHAR
               
AC5E F5  CONINS7: PUSH  PSW     ;SAVE CHARACTER
AC5F 3E01  MVI   A,1
AC61 32CCF0 STA   FLASH    ;TURN OFF CURSOR
AC64 2AD1F0 LHLDA  0F0D1H
AC67 3AD3F0 LDA   0F0D3H
AC6A 85  ADD   L
AC6B 6F  MOV   L,A
AC6C 3EF0  MVI   A,0F0H
AC6E 8C  ADC   H
AC6F 67  MOV   H,A
AC70 7E  MOV   A,M
AC71 E67F  ANI   07FH
AC73 77  MOV   M,A
AC74 F1  POP   PSW     ;GET CHARACTER
AC75 C9  RET    ;DONE

CONOUT: ;CONSOLE CHARACTER OUTPUT FROM REGISTER C
AC76 3AFFFC LDA   IOTYPE     ;GET CONFIGURATION BYTE
AC79 E601 ANI   10H       ;BIT 4 = 1 TO IGNORE FILTER
AC7B 79  MOV   A,C    ;GET TO ACCUMULATOR
AC7C 202C J20:  DB    JRNZ,COUT5-J20-2   ;PRINT AS RECEIVED
               
AC7E CDDAAC CALL  SWAP     ;EXCHANGE UPPER AND LOWER CASE
AC81 FEOC CPI   OCH     ;ASCII CLEAR SCREEN?
AC83 2004 J21:  DB    JRNZ,COUT1-J21-2   ;JUMP IF NO
               
AC85 3E93  MVI   A,93H    ;COMMODORE CLEAR SCREEN CMD
AC87 1821  J22:  DB  JR,COUT5-J22-2

AC89 FE08  COUT1:  CPI  08H  ;ASCII BACKSPACE?
AC8B 2004  J23:  DB  JRNZ,COUT2-J23-2  ;JUMP IF NO

AC8D 3E14  MVI  A,14H  ;COMMODORE BACKSPACE CMD
AC8F 1819  J24:  DB  JR,COUT5-J24-2

AC91 FE0A  COUT2:  CPI  0AH  ;LINE FEED?

AC95 3E11  MVI  A,17  ;COMMODORE LINE FEED
AC97 1811  J26:  DB  JR,COUT5-J26-2

AC99 FE0D  COUT3:  CPI  0DH  ;CARRIAGE RETURN?

AC9D CDAAAC  CALL  COUT5
ACAO 3E91  MVI  A,145  ;UP 1 LINE TO NEGATE AUTO LF

ACAA 3201F9  COUT4:  CPI  20H  ;RETURN IF UNDECODED
ACAE 5D8  RC  CONTROL CHAR

ACAF 2006  CPI  80H  ;RETURN IF NOT ASCII
ACAF 2006  RNC  CHARACTER

ACAA 3201F9  COUT5:  STA  DATA  ;PUT DATA IN CHARACTER
ACAD 3E03  STA  REGISTER
ACAF 181D  MVI  A,VICOUT  ;SCREEN OUTPUT COMMAND
ACAF 181D  J29:  DB  JR,LIST3-J29-2

LIST:  ;LIST CHARACTER FROM REGISTER C
ACB1 3AFFFC  LDA  IOTYPE  ;WHAT KIND OF PRINTER?
ACB4 E604  ANI  04H  ;0 IF 1515, 1 IF 4022
ACB6 79  MOV  A,C  ;CHARACTER TO REGISTER A
ACB7 2010  J30:  DB  JRNZ,LIST2-J30-2  ;JUMP IF NO SWAP

ACB9 3AFFFC  LDA  IOTYPE  ;WHICH TYPE OF SWAP?
ACBE 79                MOV    A,C     ;GET CHARACTER
ACBF 2005              DB      JRNZ, LIST1-J31-2

; ACC1 CDDAC            CALL    SWAP    ;SWAP UPPER AND LOWER CASE
ACC4 1803              J32:    DB      JR, LIST2-J32-2

; ACC6 CDEDAC            LIST1:   CALL    SWAP2    ;4022 SWAP ROUTINE
ACC9 3201F9            LIST2:   STA      DATA    ;PUT DATA IN REGISTER
ACCC 3E05              MVI      A, VICPRT ;ASSUME 1540
ACCE C390AB            LIST3:   JMP      IO6510 

, LISTST:   ;RETURN LIST STATUS (0 IF NOT READY, 1 IF READY)
ACD1 3E04              MVI      A, VICPST  ;PRINTER STATUS COMMAND
ACD3 CD90AB            CALL     IO6510
ACD6 3A01F9            LDA      DATA    ;DATA IS STATUS
ACD9 C9                RET

, SWAP-                ;SWAP UPPER AND LOWER CASE FOR COMMODORE-64
ACDA FE41              CPI      41H     ;LESS THAN UC 'A'?
ACDC D8                RC       ;RETURN IF SO

, ACDD FE5B             CPI      5BH     ;UC LETTER?
ACDF 3809              J33:    DB      JRC, SWAP1-J33-2 ;JUMP IF SO

, ACE1 FE61             CPI      61H     ;LESS THAT LC 'A'
ACE3 D8                RC       ;RETURN IF SO

, ACE4 FE7B             CPI      7BH     ;LC LETTER?
ACE6 D0                RNC      ;RETURN IF NO

, ACE7 E65F             ANI      5FH     ;TURN OFF BIT 5
ACE9 C9                RET

; ACEA F620              SWAP1.   ORI      20H     ;TURN ON BIT 5
ACEC C9                RET

; ACED FE41              SWAP2:   CPI      41H     ;CY IF LESS THAN UC 'A'
ACEF D8                RC
ACFo FE60               CPI      60H     ;CY IF 40H < A < 60H
ACF2 3003              J34:    DB      JRNC, SWAP3-J34-2

226 APPENDIX E
ACF4 F680
ACF6 C9

; ORI 80H
RET

ACF7 E65F
ACF9 C9

; SWAP3: ANI 5FH
RET

; PUNCH: ;PUNCH CHARACTER FROM REGISTER C
ACFA 79
ACFB 00
ACFC C9

; MOV A,C ,CHARACTER TO REGISTER A
NOP
RET ;NULL SUBROUTINE

; READER: ;READ CHARACTER INTO REGISTER A FROM READER
DEVICE
ACFD 3E1A
ACFF E67F
AD01 C9

; MVI A,1AH ;ENTER END OF FILE FOR NOW
(REPLACE LATER)
ANI 7FH ;REMEMBER TO STRIP PARITY BIT
RET

;* * * * * * * * * * * * * * * *
; *
;* CP/M TO HOST DISK CONSTANTS *
; *
;* * * * * * * * * * * * * * * *

0400 = BLKSIZ EQU 1024 ;CP/M ALLOCATION SIZE
0100 = HSTSIZ EQU 256 ;HOST DISK SECTOR SIZE
0011 = HSTSPT EQU 17 ;HOST DISK SECTORS/TRK
0002 = HSTBLK EQU HSTSIZ/128 ;CP/M SECTS/HOST BUFF
0022 = CPMSPT EQU HSTBLK * HSTSPT ;CP/M SECTORS/TRACK
0001 = SECMSK EQU HSTBLK-1 ;SECTOR MASK
0001 = SECSHF EQU 1 ;LOG2(HSTBLK)

;* * * * * * * * * * * * * * * *
; *
;* BDOS CONSTANTS ON ENTRY TO WRITE *
; *
;* * * * * * * * * * * * * * * *

0000 = WRALL EQU 0 ;WRITE TO ALLOCATED
0001 = WRDIR EQU 1 ;WRITE TO DIRECTORY
0002 = WRUAL EQU 2 ;WRITE TO UNALLOCATED

APPENDIX E 227
HOME THE SELECTED DISK

HOME:
AD02 3AE2AE LDA HSTWRT ,CHECK FOR PENDING WRITE
AD05 87 ORA A
AD06 2003 J35. DB JRNZ, HOMED-J35-2
AD08 32E1AE STA HSTACT ,CLEAR HOST ACTIVE FLAG

HOMED:
AD0B C9 RET

; SELDSK:
;SELECT DISK
AD0C 210000 LXI H,0000H ,ERROR RETURN CODE
AD0F 79 MOV A,C ;SELECTED DISK NUMBER
AD10 32DBAE STA SEKDSK ;SEEK DISK NUMBER
AD13 FE02 CPI 2 ;MUST BE 0-1
AD15 D0 RNC ;NO CARRY IF 2,3,
AD16 6F MOV L,A ;DISK NUMBER TO HL
AD17 29 DAD H ,MULTIPLY BY 16
AD18 29 DAD H
AD19 29 DAD H
AD1A 29 DAD H
AD1B 1134AA LXI D,DPBASE ;BASE OF PARM BLOCK
AD1E 19 DAD D ;HL=.DPB(CURDSK)
AD1F C9 RET

; SETTRK;
,SET TRACK GIVEN BY REGISTERS BC
AD20 60 MOV H,B
AD21 69 MOV L,C
AD22 22D9AE SHLD SEKTRK ;TRACK TO SEEK
AD25 C9 RET

; SETSEC;
;SET SECTOR GIVEN BY REGISTER C
AD26 79 MOV A,C
AD27 32DBAE STA SEKSEC ,SECTOR TO SEEK
AD2A C9 RET

; SETDMA:
;SET DMA ADDRESS GIVEN BY BC

22B APPENDIX E
AD2B 60     MOV    H,B  
AD2C 69     MOV    L,C  
AD2D 22ECAE SHLD   DMAADR      
AD30 C9     RET  

; SECTRAN.    
;TRANSLATE SECTOR NUMBER BC  
AD31 60     MOV    H,B  
AD32 69     MOV    L,C  
AD33 C9     RET  

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
;*   
;* THE READ ENTRY POINT TAKES THE PLACE OF  
;* THE PREVIOUS BIOS DEFINITION FOR READ.  
;* 
;* 
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
READ:  
,READ THE SELECTED CP/M SECTOR  
AD34 AF     XRA    A  
AD35 32E3AE STA    UNACNT  
AD38 3E01   MVI    A,1  
AD3A 32EAEE STA    READOP  
           ;READ OPERATION  
AD3D 32E9AE STA    RSFLAG  
           ;MUST READ DATA  
AD40 3E02   MVI    A,WRUAL  
AD42 32EBAE STA    WRTYPE  
           ;TREAT AS UNALLOC  
AD45 1864   J36: DB    JR,RWOPER-J36 -2  
           ;TO PERFORM THE READ  

;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
;*   
;* THE WRITE ENTRY POINT TAKES THE PLACE OF  
;* THE PREVIOUS BIOS DEFINITION FOR WRITE.  
;* 
;* 
;* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
WRITE:  
;WRITE THE SELECTED CP/M SECTOR  
AD47 AF     XRA    A  
           ;0 TO ACCUMULATOR  
AD48 32EAAE STA    READOP  
           ;NOT A READ OPERATION  
AD4B 79     MOV    A,C  
           ;WRITE TYPE IN C  
AD4C 32EBAE STA    WRTYPE  
AD4F FF02   CPI    WRUAL  
           ;WRITE UNALLOCATED?
J37:  DB JRNZ, CHKUNA-J37-2 ; CHECK FOR UNALLOC
      ; WRITE TO UNALLOCATED, SET PARAMETERS
AD53 3E08   MVI A, BLKSIZ/128; NEXT UNALLOC RECS
AD55 32E3AE  STA UNACNT
AD58 3ADB8AE  LDA SEKDSK ; DISK TO SEEK
AD5B 32E4AE  STA UNADSK ; UNADSK = SEKDSK
AD5E 2AD9AE  LHLDE SEKTRK
AD61 22E5AE  SHLD UNATRK ; UNATRK = SECTRK
AD64 3ADB8AE  LDA SEKSEC
AD67 32E7AE  STA UNASEC ; UNASEC = SEKSEC

; CHKUNA.

; CHECK FOR WRITE TO UNALLOCATED SECTOR
AD6A 3AE3AE  LDA UNACNT ; ANY UNALLOC REMAIN?
AD6D B7    ORA A
AD6E 2833  J38:  DB JRZ, ALLOC-J38-2 ; SKIP IF NOT

; MORE UNALLOCATED RECORDS REMAIN
AD70 3D    DCR A ; UNACNT = UNACNT-1
AD71 32E3AE  STA UNACNT
AD74 3ADB8AE  LDA SEKDSK ; SAME DISK?
AD77 21E4AE  LXI H, UNADSK
AD7A BE   CMP M ; SEKDSK = UNADSK?
AD7B 2026  J39:  DB JRNZ, ALLOC-J39-2 ; SKIP IF NOT

; DISKS ARE THE SAME
AD7D 21E5AE  LXI H, UNATRK
AD80 CD40AE  CALL TRKCMP ; SEKTRK = UNATRK?
AD83 201E  J40:  DB JRNZ, ALLOC-J40-2 ; SKIP IF NOT

; TRACKS ARE THE SAME
AD85 3ADB8AE  LDA SEKSEC ; SAME SECTOR?
AD88 21E7AE  LXI H, UNASEC
AD8B BE   CMP M ; SEKSEC = UNASEC?
AD8C 2015  J41:  DB JRNZ, ALLOC-J41-2 ; SKIP IF NOT

; MATCH, MOVE TO NEXT SECTOR FOR FUTURE REF
AD98 34    INR M ; UNASEC = UNASEC + 1
AD8F 7E    MOV A, M ; END OF TRACK?
AD90 FE22 CPI CPMSPT ;COUNT CP/M SECTORS
AD92 3809 J42; DB JRC, NOOVF-J42-2 ;SKIP IF NO OVERFLOW ;
AD94 3600 MVI M, 0 ;UNASEC = 0
AD96 2AE5AE LHLD UNATRK
AD99 23 INX H
AD9A 22E5AE SHLD UNATRK ;UNATRK = UNATRK + 1 ;
NOOVF;
;MATCH FOUND, MARK AS UNNECESSARY READ
AD9D AF XRA A ;0 TO ACCUMULATOR
AD9E 32E9AE STA RSFLAG ;RSFLAG = 0
ADA1 1808 J43; DB JR, RWOPER-J43-2 ;TO PERFORM THE WRITE ;
ALLOC;
;NOT AN UNALLOCATED RECORD, REQUIRES PRE-READ
ADA3 AF XRA A ;0 TO ACCUM
ADA4 32E3AE STA UNACNT ;UNACNT = 0
ADA7 3C INR A ;1 TO ACCUM
ADA8 32E9AE STA RSFLAG ;RSFLAG = 1 ;
;
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*;
* COMMON CODE FOR READ AND WRITE FOLLOWS *
*
*
* * * * * * * * * * * * * * * * * * * * * * * * * * *
RWOPER:
;ENTER HERE TO PERFORM THE READ/WRITE
ADAB AF XRA A ;ZERO TO ACCUM
ADAC 32E8AE STA ERFLAG ;NOERRORS (YET)
ADAF 3ADB8AE LDA SEKSEC ;COMPUTE HOST SECTOR
ADB2 B7 ORA A ;CARRY = 0
ADB3 1F RAR ;SHIFT RIGHT
ADB4 32E0AE STA SEKHST ;HOST SECTOR TO SEEK ;
;
ACTIVE HOST SECTOR?
ADB7 21E1AE LXI H, HSTACT ;HOST ACTIVE FLAG
ADBA 7E MOV A, M
ADBB 3601 MVI M, 1 ;ALWAYS BECOMES 1
ADBD B7 ORA A ;WAS IT ALREADY?
ADBE 2821 J44:  DB JRZ, FILHST-J44-2 ; FILL HOST IF NOT
    ;
    ; HOST BUFFER ACTIVE, SAME AS SEEK BUFFER?
ADCO 3AD8AE  LDA SEKSDK
ADC3 21DCAE  LXI H, HSTDSK ; SAME DISK?
ADC6 BE     CMP M ; SEKSDK = HSTDSK?
ADC7 2011 J45:  DB JRNZ, NOMTCH-J45-2
    ;
    ; SAME DISK, SAME TRACK?
ADC9 21DDAE  LXI H, HSTTRK
ADCC CD40AE  CALL TRKCMP ; SEKTRK = HSTTRK?
ADCF 2009 J46.  DB JRNZ, NOMTCH-J46-2
    ;
    ; SAME DISK, SAME TRACK, SAME BUFFER?
ADD1 3AE0AE  LDA SEKHST
ADD4 21DFAE  LXI H, HSTSEC ; SEKHST = HSTSEC?
ADD7 BE     CMP M
ADD8 2824 J47.  DB JRZ, MATCH-J47-2 ; SKIP IF MATCH
    ;
    ; NOMTCH.
    ; PROPER DISK, BUT NOT CORRECT SECTOR
ADDA 3AE2AE  LDA HSTWRT ; HOST WRITTEN?
ADDD B7     ORA A
ADDE C44CAE  CNZ WRHST ; CLEAR HOST BUFF
    ;
    ; FILHST:
    ; MAY HAVE TO FILL THE HOST BUFFER
ADE1 3AD8AE  LDA SEKSDK
ADE4 32DCAE  STA HSTDSK
ADE7 2AD9AE  LHLD SEKTRK
ADEA 22DDAE  SHLD HSTTRK
ADED 3AE0AE  LDA SEKHST
ADF0 32DFAE  STA HSTSEC
ADF3 3AE9AE  LDA RSFLAG ; NEED TO READ?
ADF6 B7     ORA A
ADF7 C49DAE  CNZ RDHST ; YES, IN 1
\DFA AF     XRA A ; 0 TO ACCUM
ADF8 32E2AE  STA HSTWRT ; NO PENDING WRITE
    ;
    ; MATCH:
    ; COPY DATA TO OR FROM BUFFER

232 APPENDIX E
LDA SEKSEC ;MASK BUFFER NUMBER
ANI SECMSK ;LEAST SIGNIF BITS
MOV L,A ;READY TO SHIFT
MVI H,0 ;DOUBLE COUNT
DAD H ;SHIFT LEFT 7
DAD H
DAD H
DAD H
DAD H
DAD H
DAD H
DAD H
DAD H
DAD H
DAD H
LXI D,HSTBUF
DAD D ;HL = HOST ADDRESS
XCHG ;NOW IN DE
LHLD DMAADR ;GET/PUT CP/M DATA
MVI C,128 ;LENGTH OF MOVE
LDA READOP ;WHICH WAY?
ORA A
J48: DB JRNZ,RWMOVE-J48-2 ;SKIP IF READNZ

WRITE OPERATION, MARK AND SWITCH DIRECTION

MVI A,1
STA HSTWRT ;HSTWRT = 1
XCHG ;SOURCE/DEST SWAP

RWMOVE-

C INITIALLY 128, DE IS SOURCE, HL IS DEST
LDA X D ;SOURCE CHARACTER
INX D
MOV M,A ;TO DEST
INX H
DCR C ;LOOP 128 TIMES
J49: DB JRNZ,(RWMOVE-J49-2) AND 0FFH

DATA HAS BEEN MOVED TO/FROM HOST BUFFER

LDA WRTYPE ;WRITE TYPE
CPI WRDIR ;TO DIRECTORY?
LDA ERFLAG ;IN CASE OF ERRORS
RNZ ;NO FURTHER PROCESSING

CLEAR HOST BUFFER FOR DIRECTORY WRITE
ORA A ;ERRORS?
RNZ ;SKIP IF SO
XRA A ;0 TO ACCUM
STA HSTWRT ;BUFFER WRITTEN
CALL WRHST
LDA ERFLAG
RET

;UTILITY SUBROUTINE FOR 16-BIT COMPARE
TRKCMP:

;HL = UNATRK OR HSTTRK, COMPARE WITH SEKTRK

XCHG
LXI H, SEKTRK
LDAX D ;LOW BYTE COMPARE
CMP M ;SAME?
RNZ ;RETURN IF NOT EQUAL

INX D
INX H
LDAX D
CMP M ;SETS FLAGS
RET

;WRHST PERFORMS THE PHYSICAL WRITE TO
;THE HOST DISK, RDHST READS THE PHYSICAL
;DISK.

;WRHST:
;HSTDSK = HOST DISK #; HSTTRK = HOST TRACK #;
;HSTSEC = HOST SECT #. WRITE “HSTSZ” BYTES
;FROM HSTBUF AND RETURN ERROR FLAG IN ERFLAG.
;RETURN ERFLAG NON-ZERO IF ERROR

MVI A, VICWR ;LOAD DISK WRITE COMMAND
WRHST0: STA RW ;PUT COMMAND IN REGISTER
AE51 3ADCAE  LDA  HSTSDK  ;GET HOST DISK NUMBER
AE54 3204F9  STA  DISKNO  ; AND PUT IN COMMON AREA
AE57 CD79AE  CALL  CHGDSK  ;CORRECT VIRTUAL DISK?
AE5A 3ADDAE  WRHST2: LDA  HSTRK  ;GET HOST TRACK NUMBER
AE5D 3C     INR  A    ;ADD 1 FOR VIC OFFSET
AE5E FE12   CPI  18  ;WE WANT TO SKIP TRACK 18
AE60 3801   J50: DB  JRC,WRHST3-J50-2  ;CARRY IF TRACK<18
AE62 3C     INR  A
AE63 3203F9  WRHST3: STA  TRACK  ;PUT IN COMMON AREA
AE66 3ADFAE  LDA  HSTSEC  ;GET HOST SECTOR NUMBER
AE69 3202F9  STA  SECTOR  ;PUT IN COMMON AREA
AE6C 3AEEAE  LDA  RW    ;GET DISK COMMAND
AE6F CD90AB  CALL  IO6510
AE72 3A01F9  LDA  DATA  ;GET DISK STATUS
AE75 32E8AE  STA  ERFAG  ; AND STORE IN ERFAG
AE78 C9     RET

AE79 67     CHGDSK: MOV  H,A  ;SAVE DISK NUMBER
AE7A 3AFFFC  LDA  IOTYPE  ;BIT 0 = 0 FOR VIRTUAL
AE7D E601   ANI  01
AE7F C0     RNZ  ,NOT ZERO IF 2 DRIVES
AE80 3204F9  STA  DISKNO  ;FORCE DRIVE A
AE83 7C     MOV  A,H  ;RESTORE DISK NUMBER

AE84 21EFAE  LXI  H,CURDSK  ;IS THIS OUR CURRENT DISK?
AE87 BE     CMP  M
AE88 C8     RZ   ;RETURN IF OK

AE89 77     MOV  M,A  ;SET UP NEW DISK
AE8A C641   ADI  'A'  ;FORM ASCII DRIVE LETTER
AE8B 32AFAE  STA  DSKMNT  ;PUT IN MESSAGE

AE8F 21A1AE  LXI  H,MNTMSG  ;INSERT DISK MESSAGE
AE92 CDCCAЕ CALL  PMSG  ;GO PRINT IT
AE95 CDFEAB  CHGDS1: CALL  CONIN  .WAIT FOR RETURN
AE98 FE0D   CPI  0DH
AE9A 20F9   J51: DB  JRNZ,(CHGDS1-J51-2) AND 0FFH
AE9C C9     RET

RDHST;

;HSTSDK = HOST DISK #, HSTRK = HOST TRACK #,
;HSTSEC = HOST SECTOR#. READ "HSHSZ" BYTES
; INTO HSTBUF AND RETURN ERROR FLAG IN EFLAGS.
AE9D 3E00  MVI  A, VICRD  ; DISK READ COMMAND
AE9F 18AD  J52  DB  JR,(WRHST0-J52-2) AND OFFH  ; REST LIKE WRITE

AEA1 0D0A496E73  MNTMSG: DB  0DH, 0AH, 'INSERT DISK'
AEA9 F1  DSKMNT: DB  'A'
AEB0 20696E746F  DB  ' INTO DRIVE 0, PRESS RETURN'
AECB 00  DB  00H

AEEC 7E  PMSG:  MOV  A, M
AEDD A7  ANA  A
AECC C8  RZ
AECF E5  PUSH H
AED0 4F  MOV  C, A
AED1 CD76AC  CALL  CONOUT
AED4 E1  POP  H
AED5 23  INX  H
AED6 18F4  J53:  DB  JR,(PMSG-J53-2) AND OFFH

;  ;  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
;  ;  *  *
;  ;  UNINITIALIZED RAM DATA AREAS
;  ;  *  *
;  ;  *
;  ;  *  *  *  *  *  *  *  *  *  *  *  *  *  *

AED8  SEKDSK: DS  1 , SEEK DISK NUMBER
AED9  SEKTRK: DS  2 , SEEK TRACK NUMBER
AEDB  SEKSEC: DS  1 , SEEK SECTOR NUMBER

AEDC  HSTDSK: DS  1 , HOST DISK NUMBER
AEDD  HSTTRK: DS  2 , HOST TRACK NUMBER
AEDF  HSTSEC: DS  1 , HOST SECTOR NUMBER

AEE0  SEKHST: DS  1 , SEEK SHR SEC SHF
AEE1  HSTACT: DS  1 , HOST ACTIVE FLAG
AEE2  HSTWRT: DS  1 , HOST WRITTEN FLAG

AEE3  UNACNT: DS  1 , UNALLOC REC CNT
AEE4  UNADSK: DS  1 , LAST UNALLOC DISK
AEE5  UNATRK: DS  2 , LAST UNALLOC TRACK
AEE7 UNASEC: DS 1 ,LAST UNALLOC SECTOR

AEEB ERFLAG: DS 1 ,ERROR REPORTING
AAEE9 R$FLAG: DS 1 ,READ SECTOR FLAG
AEEA READOP DS 1 ; 1 IF READ OPERATION
AEEB WRTYPE: DS 1 ;WRITE OPERATION TYPE
AEEC DMAADR: DS 2 ;LAST DMA ADDRESS
AEEE RW: DS 1 ,TEMPORARY COMMAND REGISTER
AEEF CURDSK: DS 1 ;VIRTUAL DISK POINTER

, SCRATCH RAM AREA FOR BDOS USE

AEF0 = BEGDAT EQU $ ;BEGINNING OF DATA AREA
AEF0 DIRBF: DS 128 ;SCRATCH DIRECTORY AREA
AF70 ALLO0: DS 31 ;ALLOCATION VECTOR 0
AFBF ALLO1: DS 31 ;ALLOCATION VECTOR 1
AFAE CHK00: DS 16 ;CHECK VECTOR 0
AFBE CHK01: DS 16 ;CHECK VECTOR 1

AFCE = ENDDAT EQU $ ;END OF DATA AREA
00DE = DATSIZ EQU $-BEGDAT ;SIZE OF DATA AREA
AFCE END
The Commodore Z80 microprocessor and CP/M® operating system let you turn your Commodore 64 into a dual processor home microcomputer.

CP/M® lets you use more than 15,000 CP/M® application programs. CP/M® software includes widely used business applications such as financial reporting and analysis, investment planning, word processing, farm and restaurant management, data base, exotic language compilers, and much, much more.

The Commodore 64 CP/M® Operating System User's Guide tells you how to use the Z80 cartridge and the CP/M® operating system. This manual gives you detailed information on how to bring up CP/M® on your system. We also give you a detailed reference section with descriptions of all the CP/M® commands and utility programs.

For the beginner, this manual offers simple, step-by-step instructions with all the information you need to use CP/M® on your Commodore 64.

For the advanced user, this manual provides detailed information on the technical workings of CP/M® on your Commodore 64 and the engineering details of your Z80 cartridge.

This manual is written in an easy-to-read style and is designed to help you get the most out of the Z80 microprocessor and the CP/M® operating system.